



SQL Anywhere® Server SQL-Referenzhandbuch

Version 16.0

Februar 2013

Version 16.0
Februar 2013

© 2013 SAP AG oder ein SAP-Konzernunternehmen. Alle Rechte vorbehalten.

Sie können diese Dokumentation (ganz oder teilweise) unter folgenden Bedingungen benutzen, reproduzieren und verteilen: 1) Sie müssen diese und alle anderen Urheberrechtsvermerke auf allen Kopien oder Auszügen der Dokumentation wiedergeben. 2) Sie dürfen die Dokumentation nicht verändern. 3) Sie dürfen nichts tun, aus dem abgeleitet werden könnte, dass Sie oder jemand anderer als SAP Verfasser oder Quelle der Dokumentation ist. Die hier enthaltenen Informationen können jederzeit ohne vorherigen Hinweis geändert werden.

Einige Softwareprodukte, die von der SAP AG oder einem ihrer Vertriebspartner vermarktet werden, enthalten Softwarekomponenten anderer Softwareanbieter. Die nationalen Produktspezifikationen können unterschiedlich sein.

Diese Dokumentationen werden von der SAP AG und ihren Tochtergesellschaften ("SAP Group") lediglich zu Informationszwecken bereitgestellt, ohne dass eine Gewährleistung oder eine Garantie irgendeiner Art gegeben wird. Die SAP Group übernimmt keine Verantwortung im Hinblick auf Fehler oder Auslassungen in den Dokumentationen. Die einzigen Garantien für Produkte und Dienstleistungen der SAP Group sind diejenigen, die in den mit den Produkten und Dienstleistungen eventuell gelieferten ausdrücklichen Garantieerklärungen enthalten sind. Keine der hier enthaltenen Informationen kann als Gewährung einer weitergehenden Garantie betrachtet werden.

SAP und weitere erwähnte SAP-Produkte und -Dienstleistungen sowie die entsprechenden Logos sind Marken oder eingetragene Marken der SAP AG in Deutschland und anderen Ländern. Weitere Hinweise finden Sie unter <http://www.sap.com/corporate-en/legal/copyright/index.epx#trademark>.

Inhalt

Über diese Dokumentation	v
SQL-Sprachelemente	1
Schlüsselwörter	1
Bezeichner	4
Zeichenfolgen	6
Konstanten	6
Operatoren	9
Ausdrücke	22
Suchbedingungen	42
Spezialwerte	70
Variablen	85
Kommentare	92
Benannte Parameter	93
SQL-Datentypen	95
Zeichendatentypen	95
Numerische Datentypen	103
Währungsdantypen	113
Bit-Array-Datentypen	114
Datentypen für Datum und Uhrzeit	116
Binärdatentypen	133
Zusammengesetzte Datentypen	136
Räumliche Datentypen	138
Domänen	138
Datentypvergleiche	140
Datentypkonvertierungen	146
SQL-Funktionen	153
Funktionstypen	153
Funktionen	166

SQL-Anweisungen	445
Allgemeine Elemente der SQL-Syntax	445
Syntaxkonventionen	446
Indikatoren der Anweisungsanwendbarkeit	448
SQL-Anweisungen	449
 Tabellen	 1129
Systemtabellen	1129
Diagnoseprotokollierungstabellen	1144
Andere Tabellen	1158
 Systemprozeduren	 1161
Anzeigen von Details über Systemprozeduren und Funktionen	1161
Webdienst-Systemprozeduren	1162
Systemprozeduren für Rollen und Privilegien	1162
MAPI- und SMTP-Systemprozeduren	1162
Systemprozeduren für Verzeichnisse und Dateien	1164
Systemprozeduren für gesicherte Funktionen	1165
System- und Katalogprozeduren von Adaptive Server Enterprise	1166
Alphabetische Liste der Systemprozeduren	1168
 Ansichten	 1437
Systemansichten	1437
Konsolidierte Ansichten	1514
Kompatibilitätsansichten	1534
Ansichten für Transact-SQL-Kompatibilität	1546
 Index	 1547

Über diese Dokumentation

Dieses Handbuch bietet Referenzinformationen zu Systemprozeduren und zum Katalog (Systemtabellen und Systemansichten). Es enthält auch Erklärungen der SQL Anywhere-Implementierung der SQL-Sprache (Suchbedingungen, Syntax, Datentypen und Funktionen).

SQL-Sprachelemente

Schlüsselwörter

Jede SQL-Anweisung enthält ein oder mehrere Schlüsselwörter. SQL beachtet die Groß- und Kleinschreibung bei Schlüsselwörtern nicht, in dieser Dokumentation werden Schlüsselwörter dennoch durchweg in Großbuchstaben dargestellt.

In der folgenden Anweisung zum Beispiel sind SELECT und FROM Schlüsselwörter:

```
SELECT *  
FROM Employees;
```

Die folgenden Anweisungen sind mit der obigen äquivalent:

```
Select *  
From Employees;  
select * from Employees;  
sELECT * FRoM Employees;
```

Manche Schlüsselwörter können nur als Bezeichner verwendet werden, wenn sie in Anführungszeichen gesetzt werden. Diese Schlüsselwörter heißen reservierte Wörter. Andere Schlüsselwörter, wie z.B. DBA, erfordern keine Anführungszeichen und sind keine reservierte Wörter.

Reservierte Wörter

Manche Schlüsselwörter in SQL sind **reservierte Wörter**. Wenn Sie ein reserviertes Wort in einer SQL-Anweisung als Bezeichner verwenden möchten, müssen Sie es in Anführungszeichen, eckige Klammern oder invertierte Hochkommata einschließen. Viele in SQL-Anweisungen erscheinende Schlüsselwörter sind reservierte Wörter. Verwenden Sie beispielsweise die folgende Syntax, um den Inhalt einer Tabelle namens SELECT abzurufen:

```
SELECT *  
FROM "SELECT"
```

Für SQL-Schlüsselwörter wird die Groß- und Kleinschreibung nicht berücksichtigt. Die folgenden Wörter können groß, klein oder gemischt geschrieben werden. Alle Zeichenfolgen, die sich nur durch die Großschreibung eines der folgenden Wörter unterscheiden, sind reservierte Wörter.

Sie können Schlüsselwörter auch mithilfe der Option non_keywords deaktivieren.

Die Option reserved_keywords aktiviert einzelne Schlüsselwörter, die standardmäßig deaktiviert sind.

Wenn Sie Embedded SQL verwenden, können Sie mit der Datenbank-Bibliotheksfunktion sql_needs_quotes ermitteln, ob eine Zeichenfolge Anführungszeichen erfordert. Eine Zeichenfolge erfordert Anführungszeichen, wenn sie ein reserviertes Wort ist oder ein Zeichen enthält, das normalerweise in einem Bezeichner nicht zulässig ist.

Sie können mithilfe der sa_reserved_words-Systemprozedur eine Liste der reservierten Wörter abrufen.

Die reservierten SQL-Schlüsselwörter in SQL Anywhere lauten wie folgt:

add	all	alter	and
any	array	as	asc
attach	backup	begin	between
bigint	binary	bit	bottom
break	by	call	capability
cascade	case	cast	char
char_convert	character	check	checkpoint
close	comment	commit	compressed
conflict	connect	constraint	contains
continue	convert	create	cross
cube	current	current_timestamp	current_user
cursor	date	datetimeoffset	dbspace
deallocate	dec	decimal	declare
default	delete	deleting	desc
detach	distinct	do	double
drop	dynamic	else	elseif
encrypted	end	endif	escape
except	exception	exec	execute
existing	exists	externlogin	fetch
first	float	for	force
foreign	forward	from	full
goto	grant	group	having
holdlock	identified	if	in
index	inner	inout	insensitive

insert	inserting	install	instead
int	integer	integrated	intersect
into	is	isolation	join
json	kerberos	key	lateral
left	like	limit	lock
login	long	match	membership
merge	message	mode	modify
natural	nchar	new	no
noholdlock	not	notify	null
numeric	nvarchar	of	off
on	open	openstring	openxml
option	options	or	order
others	out	outer	over
passthrough	precision	prepare	primary
print	privileges	proc	procedure
publication	raiserror	readtext	real
reference	references	refresh	release
remote	remove	rename	reorganize
resource	restore	restrict	return
revoke	right	rollback	rollup
row	rowtype	save	savepoint
scroll	select	sensitive	session
set	setuser	share	smallint
some	spatial	sqlcode	sqlstate
start	stop	subtrans	subtransaction

synchronize	table	temporary	then
time	timestamp	tinyint	to
top	tran	treat	trigger
truncate	tsequal	unbounded	union
unique	uniqueidentifier	unknown	unnest
unsigned	update	updating	user
using	validate	values	varbinary
varbit	varchar	variable	varray
varying	view	wait	waitfor
when	where	while	window
with	within	work	writetext
xml			

Siehe auch

- „Bezeichner“ auf Seite 4
- „sql_needs_quotes-Funktion“ [[SQL Anywhere Server - Programmierung](#)]
- „non_keywords-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „reserved_keywords-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sa_reserved_words-Systemprozedur“ auf Seite 1298

Bezeichner

Bezeichner sind die Namen von Objekten in der Datenbank – z.B. Benutzer-IDs, Tabellen und Spalten.

Bemerkungen

Bezeichner haben eine maximale Länge von 128 Byte. Bezeichner müssen in Anführungszeichen, eckige Klammern oder invertierte Hochkommata (`...`) gesetzt werden, wenn eine der folgenden Bedingungen zutrifft:

- Der Bezeichner enthält Leerstellen.
- Das erste Zeichen des Bezeichners ist kein alphabetisches Zeichen.
- Der Bezeichner ist ein reserviertes Wort.
- Der Bezeichner enthält nicht-alphabetische Zeichen und Ziffern.

Alphabetische Zeichen sind Buchstaben, der Unterstrich (_), das At-Zeichen (@, auch "Klammeraffe" genannt), die Raute (#) und das Dollarzeichen (\$). Die Kollationssequenz der Datenbank bestimmt, welche Zeichen als alphabetische Zeichen gelten und welche als Ziffern.

Die folgenden Zeichen sind keine zulässigen Bezeichner:

- Anführungszeichen
- Steuerzeichen (alle Zeichen unter 0x20)
- Backslashes
- Eckige Klammern
- Invertierte Hochkommata

Hinweis

Wenn Sie eine Datenbank aus einer früheren Version als 16.0 neu laden möchten, entfernen Sie eckige Klammern oder invertierte Hochkommata in Bezeichnern. Andernfalls schlägt das Neuladen fehl.

Falls die Datenbankoption `quoted_identifier` auf OFF gesetzt wurde, werden doppelte Anführungszeichen als Begrenzungszeichen für SQL-Zeichenfolgen verwendet. Sie können daher nicht als Bezeichner verwendet werden. Unabhängig von der Einstellung von `quoted_identifier` können Sie Bezeichner jedoch in eckige Klammern oder Back Quotes setzen. Die Option `quoted_identifier` ist für Open Client- und jConnect-Verbindungen standardmäßig auf OFF gesetzt, andernfalls ist der Standardwert ON.

- Benutzer-IDs dürfen Folgendes nicht:
 - Sie dürfen nicht mit Leerstellen, Apostrophen oder Anführungszeichen beginnen.
 - Sie dürfen nicht mit Leerstellen enden.
 - Sie dürfen keine Semikola enthalten.
- Kennwörter berücksichtigen die Groß- und Kleinschreibung. Im Übrigen gilt Folgendes:
 - Sie dürfen nicht mit Leerstellen, Apostrophen oder Anführungszeichen beginnen.
 - Sie dürfen nicht mit Leerstellen enden.
 - Sie dürfen keine Semikola enthalten.
 - länger als 255 Byte sein

Standards und Kompatibilität

- **SQL/2008** Die Möglichkeit zum Erstellen von Bezeichnern mit bis zu 128 Zeichen ist die optionale SQL-Sprachenfunktion F391 des SQL/2008-Standards.

Siehe auch

- „Reservierte Wörter“ auf Seite 1
- „quoted_identifier-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiele

Dies sind gültige Bezeichner:

- Surname
- "Clientname"
- `Clientname`
- [Surname]
- SomeBigName

Zeichenfolgen

Eine Zeichenfolge ist eine Sequenz von Zeichen mit einer Größe von bis zu 2 GB. Eine Zeichenfolge kann in SQL folgendermaßen auftreten:

- Als ein **Zeichenfolgenliteral**. Ein Zeichenfolgenliteral ist eine Zeichenfolge, die in Apostrophe eingeschlossen ist. Ein Zeichenfolgenliteral stellt einen bestimmten, konstanten Wert dar und kann Escapesequenzen für Sonderzeichen enthalten, die nicht problemlos als Zeichen eingegeben werden können.
- Als der Wert einer Spalte oder Variablen mit einem CHAR- oder NCHAR-Datentyp.
- Als Ergebnis der Ausführung eines Ausdrucks.

Die Länge einer Zeichenfolge kann auf zwei Arten gemessen werden:

- **Bytelänge** Die Bytelänge ist die Anzahl der Bytes in der Zeichenfolge.
- **Zeichenlänge** Die Zeichenlänge ist die Anzahl der Zeichen in der Zeichenfolge und basiert auf dem verwendeten Zeichensatz.

Bei Einbyte-Zeichensätzen wie cp1252 sind die Byte- und die Zeichenlänge gleich. Bei Mehrbyte-Zeichensätzen ist die Bytelänge einer Zeichenfolge größer oder gleich ihrer Zeichenlänge.

Siehe auch

- [„Zeichenfolgenlitterale“ auf Seite 8](#)

Konstanten

Dieser Abschnitt beschreibt binäre Literale und Zeichenfolgenlitterale.

Binäre Literale

Ein binäres Literal ist eine Sequenz von hexadezimalen Zeichen, die aus den Ziffern 0-9 und den Buchstaben A-F in Groß- und Kleinschreibung bestehen. Wenn Sie Binärdaten als Literale eingeben, muss den Daten ein 0x (eine Null, gefolgt von einem x) vorangestellt werden, und rechts von diesem

Präfix muss eine gerade Anzahl von Ziffern sein. Beispiel: Das hexadezimale Äquivalent von 39 ist 0027 und wird als 0x0027 ausgedrückt.

Die hexadezimalen Konstanten in der Form 0x12345678 werden als Binärzeichenfolgen behandelt. Eine unbegrenzte Anzahl von Ziffern kann nach 0x eingefügt werden.

Ein binäres Literal wird auch als "binäre Konstante" bezeichnet. In SQL Anywhere ist der bevorzugte Ausdruck "binäres Literal".

Konvertierung in und aus hexadezimalen Werten

Sie können die Funktionen CAST, CONVERT, HEXTOINT und INTTOHEX verwenden, um eine Binärzeichenfolge in eine Ganzzahl zu konvertieren. Die Funktionen CAST und CONVERT konvertieren hexadezimale Konstanten in TINYINT, 32-Bit-Ganzzahlen mit oder ohne Vorzeichen, 64-Bit-Ganzzahlen mit oder ohne Vorzeichen, NUMERIC, etc. Die Funktion HEXTOINT konvertiert nur eine hexadezimale Konstante in eine 32-Bit-Ganzzahl mit Vorzeichen.

Der von der CAST-Funktion zurückgegebene Wert darf 8 Ziffern nicht überschreiten. Werte über 8 Ziffern geben einen Fehler zurück. Bei Werten mit weniger als 8 Ziffern werden links Nullen eingefügt. Die folgende Anweisung gibt beispielsweise den Wert -2,147,483,647 zurück:

```
SELECT CAST ( 0x0080000001 AS INT );
```

Das folgende Argument gibt einen Fehler zurück, weil der 10-stellige Wert nicht als 32-Bit-Ganzzahl mit Vorzeichen dargestellt werden kann:

```
SELECT CAST ( 0xff80000001 AS INT );
```

Der von der HEXTOINT-Funktion zurückgegebene Wert kann 8 Ziffern übersteigen, wenn der Wert als 32-Bit-Ganzzahl mit Vorzeichen dargestellt werden kann. Die HEXTOINT-Funktion akzeptiert Zeichenfolgenlitterale oder Variablen, die ausschließlich aus Ziffern und den groß- oder kleingeschriebenen Buchstaben A-F mit oder ohne 0x-Präfix bestehen. Der hexadezimale Wert ist eine negative Ganzzahl, wenn die 8. Ziffer von rechts die Ziffer 8 oder 9 oder einer der Buchstaben A bis F ist oder die davor stehenden Stellen ein oder mehrere F oder f sind.

Die folgenden Argumente geben den Wert --2,147,483,647 zurück:

```
SELECT HEXTOINT( '0xFF80000001' );
```

```
SELECT HEXTOINT( '0x80000001' );
```

```
SELECT HEXTOINT ( '0xFFFFFFFFFFFFFFFF80000001' );
```

Das folgende Argument gibt einen Fehler zurück, weil das Argument einen positiven Ganzzahlwert darstellt, der nicht als 32-Bit-Ganzzahl mit Vorzeichen dargestellt werden kann:

```
SELECT HEXTOINT( '0x0080000001' );
```

Siehe auch

- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186
- „CONVERT-Funktion [Datentypkonvertierung]“ auf Seite 200
- „HEXTOTEXT-Funktion [Datentypkonvertierung]“ auf Seite 275
- „INTTOHEX-Funktion [Datentypkonvertierung]“ auf Seite 292

Zeichenfolgenlitterale

Ein Zeichenfolgenliteral ist eine Folge von Zeichen, die in Apostrophe eingeschlossen ist. Beispiel: 'Hello world' ist ein Zeichenfolgenliteral vom Typ CHAR. Die Bytelänge ist 11, und die Zeichenlänge ist ebenfalls 11.

Ein Zeichenfolgenliteral wird auch Zeichenfolgenkonstante, Literal-Zeichenfolge oder einfach Zeichenfolge genannt. In SQL Anywhere ist der bevorzugte Ausdruck Zeichenfolgenliteral.

Sie können ein NCHAR-Zeichenfolgenliteral angeben, indem dem Wert in Anführungszeichen das Präfix N vorangestellt wird. Beispiel: N'Hello world' ist ein Zeichenfolgenliteral vom Typ NCHAR. Die Bytelänge ist 11, und die Zeichenlänge ist ebenfalls 11. Die Bytes in einem NCHAR-Zeichenfolgenliteral werden anhand des CHAR-Zeichensatzes der Datenbank interpretiert und in NCHAR konvertiert. Die Syntax N'string' ist eine abgekürzte Form für CAST('string' AS NCHAR).

Escapesequenzen

Manchmal müssen Sie Zeichen in Zeichenfolgenlitterale einfügen, die nicht normal eingegeben werden können. Beispiele dafür sind Steuerzeichen (wie z.B. eine Zeilenendmarke), Apostrophe (die ansonsten das Ende eines Zeichenfolgenliterals markieren) und hexadezimale Bytewerte. Zu diesem Zweck verwenden Sie eine Escapesequenz.

Die folgenden Beispiele zeigen, wie Sie Escapesequenzen in Zeichenfolgenlitteralen verwenden.

- Ein Apostroph wird verwendet, um den Anfang und das Ende eines Zeichenfolgenliterals zu markieren. Daher muss ein Apostroph in einer Zeichenfolge mit einem weiteren Apostroph als Escapezeichen versehen werden, und zwar folgendermaßen: 'John ' 's database '
- Ein Backslash gefolgt von einem beliebigen Zeichen außer n, x, X oder \ wird als zwei separate Zeichen interpretiert. Zum Beispiel werden mit \q ein Backslash und der Buchstabe q eingefügt.

Hexadezimale Escapesequenzen können für jeden Zeichen- oder Binärwert verwendet werden. Eine hexadezimale Escapesequenz ist ein Backslash, gefolgt von einem "x", gefolgt von zwei hexadezimalen Zahlen. Der hexadezimale Wert wird als Zeichen im CHAR-Zeichensatz für CHAR- und NCHAR-Zeichenfolgenlitterale interpretiert. Der Wert \x09 muss als \\x09 kodiert werden, wenn Sie den Wert nicht als einzelnes Tabulatorzeichen speichern möchten, aber \xyy würde als \xyy gespeichert. Das folgende Beispiel, in Code Page 1252, stellt die Zahlen 1, 2 und 3 dar, gefolgt vom Euro-Währungssymbol: '123\x80'.
- Maskieren Sie einen Backslash folgendermaßen mit einem weiteren Backslash: 'c:\\november'. Bei Pfaden können Sie auch den Schrägstrich (/) statt des Backslash verwenden: 'c:/november'.

- Eine Zeilenendmarke wird durch einen Backslash gefolgt von n (\n) dargestellt: 'First line:\nSecond line:'

Sie können dieselben Zeichen und Escapesequenzen mit NCHAR-Zeichenfolgenliteralen wie mit CHAR-Zeichenfolgenliteralen verwenden.

Wenn Sie Unicode-Zeichen verwenden möchten, die nicht direkt in das Zeichenfolgenliteral eingegeben werden können, verwenden Sie die UNISTR-Funktion.

Siehe auch

- [„UNISTR-Funktion \[Zeichenfolge\]“ auf Seite 421](#)

Operatoren

In diesem Abschnitt werden arithmetische Operatoren sowie Zeichenfolgen-, Array- und Bit-für-Bit-Operatoren beschrieben.

Es wird der normale Vorrang von mathematischen Operationen angewendet. Ausdrücke in Klammern werden zuerst ausgewertet, dann Multiplikation und Division vor Addition und Subtraktion. Zeichenfolgenverkettung erfolgt nach Addition und Subtraktion.

Siehe auch

- [„Vorrang der Operatoren“ auf Seite 21](#)
- [„Suchbedingungen“ auf Seite 42](#)

Vergleichsoperatoren

Dies ist die Syntax für Vergleiche:

expression comparison-operator expression

Dabei gilt: *comparison-operator* kann sein:

Operator	Beschreibung
=	Gleich
>	Größer als
<	Kleiner als
>=	Größer als oder gleich
<=	Kleiner als oder gleich
!=	Ungleich

Operator	Beschreibung
<>	Ungleich
!>	Nicht größer als
!<	Nicht kleiner als

Groß-/Kleinschreibung

Standardmäßig werden die SQL Anywhere-Datenbanken so erstellt, dass die Groß- und Kleinschreibung nicht berücksichtigt wird. Vergleiche werden unter derselben Berücksichtigung der Groß- und Kleinschreibung ausgeführt, wie die der Datenbank, mit der sie arbeiten. Sie können die Berücksichtigung von Groß- und Kleinschreibung der SQL Anywhere-Datenbanken mit der Option -c beim Erstellen der Datenbank definieren.

Die Berücksichtigung von Groß- und Kleinschreibung wird bei der Erstellung der Datenbank eingerichtet.

Hinweis

Zeichenfolgenvergleiche berücksichtigen die *Groß- und Kleinschreibung nicht*, außer bei der Erstellung der Datenbank wurde die Berücksichtigung der Groß- und Kleinschreibung aktiviert.

Nachgestellte Leerzeichen

Das Verhalten von SQL Anywhere beim Vergleichen von Zeichenfolgen wird bei der Erstellung der Datenbank eingerichtet.

Siehe auch

- „Groß-/Kleinschreibung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- Bei Vergleichen nachgestellte Leerzeichen ignorieren [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Logische Operatoren

Suchbedingungen können mit den Operatoren AND oder OR kombiniert werden. Sie können sie auch negieren, indem Sie den NOT-Operator verwenden, oder testen, ob ein Ausdruck mit TRUE, FALSE oder UNKNOWN zurückgegeben wird, indem Sie den IS-Operator verwenden.

- **AND-Operator** Der AND-Operator wird wie folgt zwischen Suchbedingungen gesetzt:

...WHERE condition1 AND condition2

Wenn Sie AND verwenden, ist die kombinierte Bedingung TRUE, falls beide Bedingungen TRUE sind, FALSE, wenn eine Bedingung FALSE ist, und sonst UNKNOWN.

- **OR-Operator** Der OR-Operator wird wie folgt zwischen Suchbedingungen gesetzt:

...WHERE condition1 OR condition2

Wenn Sie OR verwenden, ist die kombinierte Bedingung TRUE, wenn eine Bedingung TRUE ist; FALSE, wenn beide Bedingungen FALSE sind, und sonst UNKNOWN.

- **NOT-Operator** Der NOT-Operator wird wie folgt vor eine Bedingung gesetzt, um sie zu negieren:

...**WHERE NOT** *condition*

Die NOT-Bedingung ist TRUE, falls *condition* FALSE ist; FALSE, falls *condition* TRUE ist; und UNKNOWN, falls *condition* UNKNOWN ist.

- **IS-Operator** Der IS-Operator wird zwischen einen Ausdruck und den wahren Wert gesetzt, den Sie testen möchten. Dies ist die Syntax für den IS-Operator:

expression **IS [NOT]** *truth-value*

Die IS-Bedingung ist TRUE, wenn *expression* den angegebenen *truth-value* ergibt, der TRUE, FALSE UNKNOWN oder NULL sein muss. Andernfalls ist der Wert FALSE.

Beispiel: $5 * 3 = 15$ IS TRUE prüft, ob der Ausdruck $5 * 3 = 15$ TRUE ist.

Siehe auch

- „Drei-Werte-Logik“ auf Seite 67

Arithmetische Operatoren

Ausdruck + Ausdruck Addition. Falls ein Ausdruck NULL, ist das Ergebnis NULL.

Ausdruck - Ausdruck Subtraktion. Falls ein Ausdruck NULL, ist das Ergebnis NULL.

-Ausdruck Negation. Falls der Ausdruck NULL ist, ist das Ergebnis NULL.

Ausdruck * Ausdruck Multiplikation. Falls ein Ausdruck NULL ist, ist das Ergebnis NULL.

Ausdruck / Ausdruck Division. Falls ein Ausdruck NULL ist oder falls der zweite Ausdruck 0 ist, ist das Ergebnis NULL.

Ausdruck % Ausdruck Modulo ergibt den ganzzahligen Rest nach einer Division, die zwei ganze Zahlen betrifft. Beispiel: $21 \% 11 = 10$, da 21 geteilt durch 11 den Ganzzahlwert 1 mit einem Rest von 10 ergibt.

Standards und Kompatibilität

- **SQL/2008** Die Verwendung von % als Modulooperator ist eine Erweiterung des Herstellers.

Zeichenfolgenoperatoren

Ausdruck || Ausdruck Zeichenfolgenverkettung (zwei Senkrechtstriche). Falls eine Zeichenfolge NULL ist, wird sie bei der Verkettung als leere Zeichenfolge behandelt.

Ausdruck + Ausdruck Alternative Zeichenfolgenverkettung. Wenn Sie den Verkettungsoperator "+" verwenden, sollten Sie die Operanden explizit auf Zeichendatentypen setzen und sich nicht auf die implizite Datenkonvertierung verlassen.

Die folgende Abfrage ergibt zum Beispiel den ganzzahligen Wert 579:

```
SELECT 123 + 456;
```

Die folgende Abfrage ergibt dagegen die Zeichenfolge 123456:

```
SELECT '123' + '456';
```

Mit den Funktionen CAST oder CONVERT können Sie Datentypen ausdrücklich konvertieren.

Standards und Kompatibilität

- **SQL/2008** Der Operator "||" ist der Operator für die Verkettung von Zeichenfolgen in SQL/2008. Im SQL-Standard gilt jedoch: Wenn einer der beiden Operanden von "||" NULL ist, das Ergebnis der Verkettung ebenfalls NULL. Bei SQL Anywhere wird NULL vom Operator "||" als leere Zeichenfolge behandelt.

OPENXML-Operator

Erzeugt eine Ergebnismenge aus einem XML-Dokument.

Syntax 1

```
OPENXML(  
  xml-data  
  , xpath  
  [, flags  
  [, namespaces ] ]  
)  
WITH ( column-name column-type  
  [ xpath ] [ , ... ]  
)
```

Syntax 2

```
OPENXML( { USING FILE | USING VALUE }  
  xml-data  
  , xpath  
  [, flags  
  [, namespaces ] ]  
)  
WITH ( column-name column-type  
  [ xpath ] [ , ... ]  
)  
[ OPTION ( scan-option ) ]  
[ AS ] correlation-name  
  
scan-option :  
ENCODING encoding  
| BYTE ORDER MARK { ON | OFF }
```

Argumente

- **WITH-Klausel** Gibt das Schema der Ergebnismenge an und wie der Wert für jede einzelne Spalte in der Ergebnismenge gefunden wird. *xpath*-Argumente in der WITH-Klausel werden entsprechend den Übereinstimmungen mit *xpath* im zweiten Argument zugeordnet. Wenn ein WITH-Klausel-Ausdruck mehr als einen Knoten findet, wird nur der erste Knoten in der Dokumentordnung verwendet. Falls es sich bei dem Knoten nicht um einen Textknoten handelt, wird das Ergebnis gefunden, indem alle untergeordneten Elemente des Textknotens angehängt werden. Wenn ein WITH-Klausel-Ausdruck mit keinem der Knoten übereinstimmt, ist die Spalte für die betreffende Zeile NULL.

Die *xpath*-Argumente in der WITH-Klausel können Literalzeichenfolgen oder Variablen sein.

Die Syntax der OPENXML WITH-Klausel ähnelt der Syntax zum Auswählen aus einer gespeicherten Prozedur.

- **USING FILE | USING VALUE** Verwenden Sie die USING FILE-Klausel, um Daten aus einer Datei zu laden.

Hinweis

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Verwenden Sie die USING VALUE-Klausel, um Daten aus einem Ausdruck des Typs CHAR, NCHAR, BINARY oder LONG BINARY bzw. einer BLOB-Zeichenfolge zu laden.

- **xml-data** Die XML-Daten, auf denen die Ergebnismenge basiert. Dies kann ein beliebiger Zeichenfolgenausdruck sein, wie etwa eine Konstante, eine Variable oder eine Spalte.

Die *xml-data* werden direkt in der NCHAR-Kodierung syntaktisch analysiert, wenn die Ausgabe NCHAR-Spalten enthält. Die Argumente *xpath* und *namespaces* werden ebenfalls in die NCHAR-Kodierung konvertiert und syntaktisch analysiert.

- **xpath** Eine Zeichenfolge, die eine XPath-Abfrage enthält. Mit XPath können Sie Muster angeben, die die Struktur des abzufragenden XML-Dokumentes beschreiben. Das in diesem Argument enthaltene XPath-Muster wählt die Knoten aus dem XML-Dokument. Jeder Knoten, der mit der XPath-Abfrage im zweiten *xpath*-Argument übereinstimmt, erzeugt eine Zeile in der Tabelle.

Meta-Eigenschaften können nur in *xpath*-Argumenten der WITH-Klausel angegeben werden. Auf eine Meta-Eigenschaft wird innerhalb einer XPath-Abfrage so zugegriffen, als handele es sich um ein Attribut. Wenn *namespaces* nicht angegeben wird, bindet das System standardmäßig das Präfix "mp" an den Uniform Resource Identifier (URI) `urn:ianywhere-com:sa-xpath-metaprop`. Wenn *namespaces* angegeben wird, muss dieser URI an "mp" oder an ein anderes Präfix gebunden werden, damit auf die Meta-Eigenschaften der Abfrage zugegriffen werden kann. Meta-Eigenschaften berücksichtigen die Groß- und Kleinschreibung. Die OPENXML-Anweisung unterstützt die folgenden Meta-Eigenschaften:

- **@mp:id** Gibt eine ID für einen Knoten zurück, der innerhalb des XML-Dokumentes eindeutig ist. Die ID für einen bestimmten Knoten in einem bestimmten Dokument kann sich ändern, falls der Datenbankserver neu gestartet wird. Der Wert dieser Meta-Eigenschaft steigt mit der Dokumentordnung.
 - **@mp:localname** Gibt den lokalen Teil des Knotennamens zurück oder NULL, falls der Knoten keinen Namen trägt.
 - **@mp:prefix** Gibt den Präfixteil des Knotennamens zurück oder NULL, falls der Knoten keinen Namen trägt oder falls der Name kein Präfix hat.
 - **@mp:namespaceuri** Gibt den URI des Namespaces zurück, dem der Knoten angehört, bzw. NULL, falls der Knoten sich in keinem Namespace befindet.
 - **@mp:xmltext** Gibt eine Unterstruktur des XML-Dokuments in XML-Form zurück. Wenn Sie z.B. einen internen Knoten suchen, können Sie diese Meta-Eigenschaft dafür benutzen, eine XML-Zeichenfolge zurückzugeben, und nicht verkettete Werte der untergeordneten Textknoten.
- **flags** Gibt die Zuordnung an, die zwischen den XML-Daten und der Ergebnismenge verwendet werden sollte, wenn in der WITH-Klausel keine XPath-Abfrage angegeben wird. Wenn der Parameter *flags* nicht angegeben wird, ist das Standardverhalten, in der Ergebnismenge Attribute und Spalten zuzuordnen. Der Parameter *flags* kann einen der folgenden Werte annehmen:

Wert	Beschreibung
1	XML-Attribute werden in der Ergebnismenge Spalten zugeordnet (Standardeinstellung).
2	XML-Elemente werden in der Ergebnismenge Spalten zugeordnet.

- **namespace-declaration** Ein XML-Dokument. Die bekannten Namespaces ("in-scope namespaces") für die Abfrage werden aus dem Wurzelement des Dokumentes entnommen. Wenn Namespaces angegeben werden, müssen Sie das Argument *flags* einbeziehen, auch wenn alle *xpath*-Argumente angegeben werden.
- **column-name** Name der Spalte in der Ergebnismenge
 - **column-type** Datentyp der Spalte in der Ergebnismenge. Der Datentyp muss mit den aus dem XML-Dokument ausgewählten Werten kompatibel sein.
 - **OPTION-Klausel** Verwenden Sie die OPTION-Klausel, um Optionen für die syntaktische Analyse der Eingabedatei anzugeben, wie Escapezeichen, Begrenzer, Kodierung, usw.
 - **ENCODING-Klausel** Mit der ENCODING-Klausel können Sie die Kodierung angeben, mit der die Datei gelesen werden soll.

Wenn die ENCODING-Klausel nicht angegeben ist, erfolgt das Laden der Werte im Zeichensatz der Datenbank (db_charset), wenn die Werte vom Typ CHAR oder BINARY sind, und im NCHAR-Datenbankzeichensatz (nchar_charset), wenn die Werte vom Typ NCHAR sind.

- **BYTE ORDER MARK-Klausel** Verwenden Sie die BYTE ORDER MARK-Klausel, um anzugeben, ob eine Byte Order Mark (BOM) in der Kodierung enthalten ist. Standardmäßig ist diese Option ON, sodass der Server eine Byte Order Mark (BOM) am Beginn der Daten suchen und interpretieren kann. Wenn BYTE ORDER MARK OFF ist, sucht der Server nicht nach einer BOM.

Sie müssen die BYTE ORDER MARK-Klausel angeben, wenn die Eingabedaten kodiert sind.

Wenn die ENCODING-Klausel angegeben ist:

- Wenn die BYTE ORDER MARK-Option ON ist und Sie eine UTF-16-Kodierung mit Endian wie UTF-16BE oder UTF-16LE angeben, sucht der Datenbankserver nach einer BOM am Beginn der Daten. Wenn eine BOM vorhanden ist, wird sie benutzt, um den Endian der Daten zu prüfen. Wenn Sie den falschen Endian angeben, wird ein Fehler zurückgegeben.
- Wenn die BYTE ORDER MARK-Option ON ist und Sie eine UTF-16-Kodierung ohne expliziten Endian angegeben haben, sucht der Datenbankserver nach einer BOM am Beginn der Daten. Wenn eine BOM vorhanden ist, wird sie benutzt, um den Endian der Daten zu bestimmen. Sonst wird der Endian des Betriebssystems angenommen.
- Wenn die BYTE ORDER MARK-Option ON ist und Sie eine UTF-8-Kodierung angegeben haben, sucht der Datenbankserver nach einer BOM am Beginn der Daten. Wenn eine BOM vorhanden ist, wird sie ignoriert.

Wenn die ENCODING-Klausel nicht angegeben ist:

- Wenn Sie keine ENCODING-Klausel angeben und die BYTE ORDER MARK-Option ON ist, sucht der Server nach einer BOM am Beginn der Eingabedaten. Wenn es eine BOM findet, wird die Quellkodierung basierend auf der Kodierung der BOM (UTF-16BE, UTF-16LE oder UTF-8) automatisch gewählt und die BOM wird nicht als Teil der zu ladenden Daten angesehen.
- Wenn Sie keine ENCODING-Klausel angeben und die BYTE ORDER MARK-Option OFF ist oder keine BOM am Beginn der Eingabedaten gefunden wird, verwendet Interactive SQL die CHAR-Kodierung.

Bemerkungen

Der OPENXML-Operator analysiert die *xml-data* syntaktisch und modelliert das Ergebnis als Struktur. Die Struktur enthält für jedes einzelne Element, Attribut, für jeden Textknoten bzw. für jedes sonstige XML-Konstrukt einen separaten Knoten. Die für den OPENXML-Operator angegebenen XPath-Abfragen werden verwendet, um Knoten aus der Struktur auszuwählen, und die ausgewählten Knoten werden anschließend der Ergebnismenge zugeordnet.

Der vom OPENXML-Operator verwendete XML-Parser führt keine Validierung durch und liest weder die externe DTD-Teilmenge noch externe Parameterentitäten.

Wenn für einen Spaltenausdruck mehrere Übereinstimmungen vorhanden sind, wird die erste Übereinstimmung in der Dokumentordnung verwendet (die Ordnung des ursprünglichen XML-Dokuments, bevor es syntaktisch analysiert wurde). NULL wird zurückgegeben, wenn es keine

übereinstimmenden Knoten gibt. Wenn ein interner Knoten ausgewählt wird, enthält das Ergebnis alle untergeordneten Textknoten des internen Knotens in verketteter Form.

Für Spalten vom Typ BINARY, LONG BINARY, IMAGE und VARBINARY wird das Format Base64-kodiert angenommen und sie werden automatisch dekodiert. Wenn Sie XML mit der FOR XML-Klausel generieren, werden diese Typen Base64-kodiert und können mit dem OPENXML-Operator dekodiert werden.

Der OPENXML-Operator unterstützt eine Teilmenge der XPath-Syntax, und zwar wie folgt:

- Die Achsen "child", "self", "attribute", "descendant", "descendant-or-self" und "parent" werden vollständig unterstützt.
- Sowohl abgekürzte als auch nicht abgekürzte Syntax kann für alle unterstützten Funktionen verwendet werden. Beispiel: 'a' ist gleichwertig mit 'child::a' und '..' ist gleichwertig mit 'parent::node()'.
- Namenstests können Platzhalter verwenden. Beispiel: 'a/*b'.
- Die folgenden "Kind"-Tests werden unterstützt: node(), text(), processing-instruction() und comment().
- Qualifizierer der Form *expr1*[*expr2*] und *expr1*[*expr2*="string"] können verwendet werden, wobei *expr2* ein beliebiger unterstützter XPath-Ausdruck ist. Ein Qualifizierer ergibt TRUE, wenn *expr2* mit einem oder mehreren Knoten übereinstimmt. Beispiel: 'a[b]' findet a-Knoten, die zumindest ein b-Kind-Objekt (Child) haben, und a[b="I"] findet a-Knoten, die zumindest ein b-Kind-Objekt (Child) mit einem Textwert von I haben.

Privilegien

Wenn die USING FILE-Klausel angegeben ist, müssen Sie das READ FILE-Systemprivileg haben. Andernfalls sind keine Privilegien erforderlich.

Siehe auch

- „FOR XML und Binärdaten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- XPath-Ausdrücke verwenden [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „SQL-Datentypen“ auf Seite 95
- „Importieren von XML mit dem OPENXML-Operator“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Unterstützte Zeichensätze“ [*SQL Anywhere Server - Datenbankadministration*]
- „FROM-Klausel“ auf Seite 863
- XPath-Abfragesprache: <http://www.w3.org/TR/xpath>.

Beispiel

Die folgende Abfrage generiert eine Ergebnismenge aus dem XML-Dokument, das als erstes Argument des OPENXML-Operators angegeben wurde:

```
SELECT * FROM OPENXML( '<products>
    <ProductType ID="301">Tee Shirt</ProductType>
    <ProductType ID="401">Baseball Cap</ProductType>
</products>',
```

```

        '/products/ProductType' )
WITH ( ProductName LONG VARCHAR 'text()', ProductID CHAR(3) '@ID');

```

Diese Abfrage erzeugt das folgende Ergebnis:

ProductName	ProductID
Tee Shirt	301
Baseball Cap	401

Im folgenden Beispiel enthält das erste <ProductType>-Element eine Entität. Wenn Sie eine Abfrage ausführen, wird dieser Knoten syntaktisch als ein Element mit vier Child-Objekten analysiert: Tee, &, Sweater und Set. Sie können einen Punkt (.) verwenden, um die Child-Objekte in der Ergebnismenge zu verketteten.

```

SELECT * FROM OPENXML( '
        <ProductType ID="301">Tee &amp; Sweater Set</ProductType>
        <ProductType ID="401">Baseball Cap</ProductType>
    </products>',
        '/products/ProductType' )
WITH ( ProductName LONG VARCHAR '.', ProductID CHAR(3) '@ID');

```

Diese Abfrage erzeugt das folgende Ergebnis:

ProductName	ProductID
Tee Shirt & Sweater Set	301
Baseball Cap	401

Die folgende Abfrage verwendet ein Gleichheitsprädikat, um eine Ergebnismenge aus dem gelieferten XML-Dokument zu generieren.

```

SELECT * FROM OPENXML( '<EmployeeDirectory>
    <Employee>
        <column name="EmployeeID">105</column>
        <column name="GivenName">Matthew</column>
        <column name="Surname">Cobb</column>
        <column name="Street">7 Pleasant Street</column>
        <column name="City">Grimsby</column>
        <column name="State">UT</column>
        <column name="PostalCode">02154</column>
        <column name="Phone">6175553840</column>
    </Employee>
    <Employee>
        <column name="EmployeeID">148</column>
        <column name="GivenName">Julie</column>
        <column name="Surname">Jordan</column>
        <column name="Street">1244 Great Plain Avenue</column>
        <column name="City">Woodbridge</column>
        <column name="State">AZ</column>
        <column name="PostalCode">01890</column>
        <column name="Phone">6175557835</column>
    </Employee>
</EmployeeDirectory>'
WITH ( EmployeeID CHAR(5) '@EmployeeID',
        GivenName LONG VARCHAR '@GivenName',
        Surname LONG VARCHAR '@Surname',
        Street LONG VARCHAR '@Street',
        City LONG VARCHAR '@City',
        State LONG VARCHAR '@State',
        PostalCode LONG VARCHAR '@PostalCode',
        Phone LONG VARCHAR '@Phone' )

```

```
<column name="EmployeeID">160</column>
<column name="GivenName">Robert</column>
<column name="Surname">Breault</column>
<column name="Street">358 Cherry Street</column>
<column name="City">Milton</column>
<column name="State">PA</column>
<column name="PostalCode">02186</column>
<column name="Phone">6175553099</column>
</Employee>
<Employee>
  <column name="EmployeeID">243</column>
  <column name="GivenName">Natasha</column>
  <column name="Surname">Shishov</column>
  <column name="Street">151 Milk Street</column>
  <column name="City">Grimsby</column>
  <column name="State">UT</column>
  <column name="PostalCode">02154</column>
  <column name="Phone">6175552755</column>
</Employee>
</EmployeeDirectory>', '/EmployeeDirectory/Employee')
WITH ( EmployeeID INT 'column[@name="EmployeeID"]',
      GivenName CHAR(20) 'column[@name="GivenName"]',
      Surname CHAR(20) 'column[@name="Surname"]',
      PhoneNumber CHAR(10) 'column[@name="Phone"]');
```

Diese Abfrage erzeugt die folgende Ergebnismenge:

EmployeeID	GivenName	Surname	PhoneNumber
105	Matthew	Cobb	6175553840
148	Julie	Jordan	6175557835
160	Robert	Breault	6175553099
243	Natasha	Shishov	6175552755

Die folgende Abfrage verwendet den XPath-Ausdruck @attribute, um eine Ergebnismenge zu generieren:

```
SELECT * FROM OPENXML( '<Employee
  EmployeeID="105"
  GivenName="Matthew"
  Surname="Cobb"
  Street="7 Pleasant Street"
  City="Grimsby"
  State="UT"
  PostalCode="02154"
  Phone="6175553840"
/>', '/Employee' )
WITH ( EmployeeID INT '@EmployeeID',
      GivenName CHAR(20) '@GivenName',
      Surname CHAR(20) '@Surname',
      PhoneNumber CHAR(10) '@Phone');
```

Die folgende Abfrage verarbeitet ein XML-Dokument wie jenes in der obenstehenden Abfrage, nur dass ein XML-Namespace verwendet wird. Es veranschaulicht die Verwendung von Platzhaltern im Namenstest bei der XPath-Abfrage und generiert dieselbe Ergebnismenge wie die obenstehende Abfrage.

```

SELECT * FROM OPENXML( '<Employee  xmlns="http://www.iAnywhere.com/
EmployeeDemo"
    EmployeeID="105"
    GivenName="Matthew"
    Surname="Cobb"
    Street="7 Pleasant Street"
    City="Grimsby"
    State="UT"
    PostalCode="02154"
    Phone="6175553840"
/>', '/*:Employee' )

WITH ( EmployeeID INT '@EmployeeID',
      GivenName CHAR(20) '@GivenName',
      Surname CHAR(20) '@Surname',
      PhoneNumber CHAR(10) '@Phone' );

```

Als Alternative können Sie eine Namespace-Deklaration angeben:

```

SELECT * FROM OPENXML( '<Employee  xmlns="http://www.iAnywhere.com/
EmployeeDemo"
    EmployeeID="105"
    GivenName="Matthew"
    Surname="Cobb"
    Street="7 Pleasant Street"
    City="Grimsby"
    State="UT"
    PostalCode="02154"
    Phone="6175553840"
/>', '/prefix:Employee', 1, '<r xmlns:prefix="http://www.iAnywhere.com/
EmployeeDemo"/>' )
WITH ( EmployeeID INT '@EmployeeID',
      GivenName CHAR(20) '@GivenName',
      Surname CHAR(20) '@Surname',
      PhoneNumber CHAR(10) '@Phone' );

```

Array-Operatoren

Ausdruck || Ausdruck Array-Verkettung (zwei Senkrechtstriche). Wenn eines der Arrays NULL ist, wird es bei der Verkettung als Array mit Null-Länge behandelt.

Standards und Kompatibilität

- **SQL/2008** Der Operator "||" ist der Verkettungsoperator in SQL/2008. Im SQL-Standard gilt jedoch: Wenn einer der beiden Operanden von "||" NULL ist, das Ergebnis der Verkettung ebenfalls NULL. Bei SQL Anywhere wird NULL vom Operator "||" als Array mit Null-Länge behandelt.

Array-Operator UNNEST

Erstellt aus den angegebenen Array-Ausdrücken eine abgeleitete Tabelle, die eine Zeile pro Array-Element liefert.

Syntax

```

UNNEST ( array-expression [, ...] )
[ WITH ORDINALITY ]

```

Argumente

- **array-expression** Ein Array, aus dem eine Tabellenspalte abgeleitet werden soll.
- **WITH ORDINALITY** Die WITH ORDINALITY-Klausel ermöglicht es der Anwendung, das ursprüngliche Array-Element erneut aufzurufen, aus dem der jeweilige Wert abgerufen wurde. Für gültige mit UNNEST abgeleitete Tabellen müssen Namen für alle resultierenden Ausdrücke angegeben werden (mithilfe der AS-Klausel). Die Reihenfolge der aus UNNEST resultierenden Zeilen ist nicht gewährleistet. Benutzer können die gewünschte Reihenfolge mit einer ORDER BY-Klausel erzwingen.

Bemerkungen

Wenn die Array-Ausdrücke unterschiedliche Kardinalitäten haben, werden die fehlenden Ausgabeausdrücke aus den kürzeren Arrays auf NULL gesetzt. Wenn die WITH ORDINALITY-Klausel angegeben wird, enthält die Ergebnismenge eine Ganzzahlspalte mit der Kardinalzahl des Array-Elements, das die Zeile darstellt. Die neue Spalte wird als letzte Spalte an die mit UNNEST abgeleitete Tabelle angehängt.

Privilegien

Keine

Siehe auch

- [„Zusammengesetzte Datentypen“ auf Seite 136](#)
- [„ARRAY-Konstruktor \[zusammengesetzt\]“ auf Seite 168](#)
- [„Vergleiche von zusammengesetzten Typen“ auf Seite 145](#)

Beispiel

Das folgende Beispiel veranschaulicht, wie der UNNEST-Operator auf zwei Arrays mit unterschiedlichen Kardinalitäten angewendet wird:

```
SELECT * FROM UNNEST( ARRAY(2,3,4), ARRAY(4,5,6) ) WITH ORDINALITY ASs
      DT(X,Y,Z);
```

Die SQL-Anweisung gibt das folgende Ergebnis zurück:

X	Y	Z
2	4	1
3	5	2
4	6	3

Bit-Operatoren

Die folgenden Operatoren können in SQL Anywhere für Bitdatentypen, Ganzzahldatentypen (einschließlich aller Varianten wie etwa BIT, TINYINT, SMALLINT usw.), Binärwerte und Bit-Array-Datentypen verwendet werden.

Operator	Beschreibung
&	Bit-Operator AND
	Bit-Operator OR
^	Bit-Operator Exklusiv-OR
~	Bit-Operator NOT

Die Bit-Operatoren &, | und ~ sind mit den logischen Operatoren AND, OR und NOT nicht austauschbar.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Bit-für-Bit-Operatoren sowie die Datentypen BIT VARYING und BIT wurden im SQL/1999-Standard als SQL-Sprachenfunktion F511 unterstützt. Diese Funktion wurde aus dem SQL/2003-Standard entfernt.

Beispiel

Die folgende Anweisung wählt Zeilen aus, in denen die richtigen Bits gesetzt sind. Wenn beispielsweise der Wert von **options** gleich 0x1001 ist, wird die betreffende Zeile einbezogen.

```
SELECT *
FROM tableA
WHERE ( options & 0x0101 ) <> 0;
```

Join-Operatoren

Hinweis

Die Unterstützung für die Transact-SQL-Outer-Join-Operatoren "*"=" und "=*" ist veraltet. Wenn Sie Outer-Join-Operatoren von Transact-SQL verwenden möchten, müssen Sie die `tsql_outer_joins`-Datenbankoption auf ON setzen.

SQL Anywhere unterstützt zwei weitere Vergleichsoperatoren, *= und =*. Dies sind die Outer-Join-Operatoren von Transact-SQL. Wenn einer dieser Operatoren in einem Vergleichsprädikat verwendet wird, bedeutet dies einen impliziten LEFT oder RIGHT OUTER JOIN.

Siehe auch

- „`tsql_outer_joins`-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Transact-SQL-Outer-Joins (*= oder =*)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Vorrang der Operatoren

Der Vorrang bei Operatoren in Ausdrücken wird in der untenstehenden Liste dargestellt. Die Operatoren am Anfang der Liste werden vor denen weiter unten berücksichtigt.

1. unäre Operatoren (Operatoren, die einen einzelnen Operanden benötigen)
2. `&`, `|`, `^`, `~`
3. `*`, `/`, `%`
4. `+`, `-`
5. `||`
6. **not**
7. **and**
8. **or**

Wenn Sie mehr als einen Operator in einem Ausdruck verwenden, wird empfohlen, dass Sie die Reihenfolge mithilfe von Klammern explizit festlegen.

Ausdrücke

Ein Ausdruck ist eine Anweisung, die in Rückgabewerten ausgewertet werden kann.

Syntax

expression:
case-expression
constant
[correlation-name.]column-name
- expression
expression operator expression
(expression)
function-name (expression, ...)
if-expression
special value
(subquery)
variable-name
sequence-expression

case-expression :
CASE *expression*
WHEN *expression*
THEN *expression,...*
[ELSE *expression* **]**
END

alternative form of case-expression :
CASE
WHEN *search-condition*
THEN *expression, ...*
[ELSE *expression* **]**
END

constant :
integer | number | string | host-variable

special-value :
CURRENT { **DATE** | **TIME** | **TIMESTAMP** }
NULL
SQLCODE
SQLSTATE
USER

if-expression :
IF *condition*
THEN *expression*
[**ELSE** *expression*]
ENDIF

sequence-expression :
sequence-name. [**CURRVAL** | **NEXTVAL**]
FROM *table-name*

operator:
{ **+** | **-** | ***** | **/** | **||** | **%** }

Bemerkungen

Ausdrücke werden an vielen verschiedenen Stellen verwendet.

Ausdrücke werden aus verschiedenen Elementen zusammengesetzt. Diese sind in den Abschnitten zu Funktionen und Variablen beschrieben.

Sie müssen mit der Datenbank verbunden sein, um Ausdrücke auswerten zu können.

Nebenwirkungen

Keine.

Siehe auch

- „Konstanten in Ausdrücken“ auf Seite 24
- „Spezialwerte“ auf Seite 70
- „Spaltennamen in Ausdrücken“ auf Seite 24
- „SQL-Funktionen“ auf Seite 153
- „Unterabfragen in Ausdrücken“ auf Seite 24
- „Suchbedingungen“ auf Seite 42
- „SQL-Datentypen“ auf Seite 95
- „Variablen“ auf Seite 85
- „CASE-Ausdrücke“ auf Seite 25

Standards und Kompatibilität

- Sie finden die Beschreibungen zu jeder Ausdrucksklasse in den folgenden Abschnitten.

Konstanten in Ausdrücken

Konstanten sind Zahlen oder Zeichenfolgen. Zeichenfolgenkonstante werden in Apostrophe ('einfache Anführungszeichen') eingeschlossen. Ein Apostroph innerhalb einer Zeichenfolge wird durch zwei aufeinanderfolgende Apostrophe dargestellt.

Spaltennamen in Ausdrücken

Ein Spaltenname ist ein Bezeichner, dem ein optionaler Korrelationsname vorangestellt ist. Ein Korrelationsname ist normalerweise ein Tabellename.

Falls ein Spaltenname andere Zeichen als Buchstaben, Ziffern und den Unterstrich enthält, muss er von Anführungszeichen (""") umgeben werden. Dies sind zum Beispiel zulässige Spaltennamen:

- Employees.Name
- address
- "date hired"
- "salary"."date paid"

Siehe auch

- [„Bezeichner“ auf Seite 4](#)
- [„FROM-Klausel“ auf Seite 863](#)

Unterabfragen in Ausdrücken

Eine Unterabfrage ist eine SELECT-Anweisung, die in einer anderen SELECT-, INSERT-, UPDATE- oder DELETE-Anweisung bzw. einer anderen Unterabfrage verschachtelt ist.

Wenn eine Unterabfrage keine übereinstimmenden Zeilen findet, wird sie mit NULL ausgewertet.

Die SELECT-Anweisung muss in Klammern gesetzt werden und darf nur ein einziges Auswahlstenelement enthalten. Wenn sie als ein Ausdruck verwendet wird, darf die Unterabfrage üblicherweise nur einen Wert zurückgeben.

Eine Unterabfrage kann überall dort verwendet werden, wo ein Spaltenname zulässig ist. Eine Unterabfrage kann zum Beispiel in der Auswahlliste einer anderen SELECT-Anweisung verwendet werden.

Siehe auch

- [„Unterabfragen in Suchbedingungen“ auf Seite 44](#)

IF-Ausdrücke

Dies ist die Syntax des IF-Ausdrucks:

```
IF condition  
THEN expression1
```

```
[ ELSE expression2 ]
{ ENDIF | END IF }
```

Dieser Ausdruck gibt Folgendes zurück:

- Wenn *condition* den Wert TRUE hat, gibt der IF-Ausdruck *expression1* zurück.
- Wenn *condition* den Wert FALSE hat, gibt der IF-Ausdruck *expression2* zurück.
- Wenn *condition* den Wert FALSE hat und kein *expression2* vorhanden ist, gibt der IF-Ausdruck NULL zurück.
- Wenn *condition* den Wert UNKNOWN hat, gibt der IF-Ausdruck NULL zurück.

expression1 wird nur ausgewertet, wenn *condition* den Wert TRUE hat. Ebenso wird *expression2* nur ausgewertet, wenn *condition* den Wert FALSE hat. Sowohl *expression1* als auch *expression2* sind beliebige Ausdrücke. *condition* ist eine beliebige gültige Suchbedingung.

Hinweis

Die IF-Anweisung unterscheidet sich vom IF-Ausdruck.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Der SQL/2008-Standard definiert die NULLIF-, COALESCE- und CASE-Ausdrücke, die als Ersatz für einen IF-Ausdruck verwendet werden können.

Siehe auch

- „IF-Anweisung“ auf Seite 906
- „Suchbedingungen“ auf Seite 42
- „NULL-Spezialwert“ auf Seite 76

CASE-Ausdrücke

Der CASE-Ausdruck ermöglicht bedingte SQL-Ausdrücke. CASE-Ausdrücke können überall dort eingesetzt werden, wo Ausdrücke verwendet werden können.

Dies ist die Syntax des CASE-Ausdrucks:

```
CASE expression-1
WHEN expression-2
THEN expression-3, ...
[ ELSE expression-4 ]
{ END | END CASE }
```

Falls der Ausdruck, der der CASE-Klausel folgt, gleich dem Ausdruck ist, der der WHEN-Klausel folgt, wird der Ausdruck zurückgegeben, der der THEN-Anweisung folgt. Andernfalls wird der Ausdruck, der der ELSE-Anweisung folgt, zurückgegeben, falls sie vorhanden ist.

Der CASE-Ausdruck gibt NULL zurück, wenn die ELSE-Klausel nicht vorhanden ist und *expression-1* zu keinem der Werte *expression-2...expression-n* passt.

Der folgende Code verwendet zum Beispiel eine CASE-Anweisung als zweite Klausel in einer SELECT-Anweisung.

```
SELECT ID,
  ( CASE Name
    WHEN 'Tee Shirt' then 'Shirt'
    WHEN 'Sweatshirt' then 'Shirt'
    WHEN 'Baseball Cap' then 'Hat'
    ELSE 'Unknown'
  END ) as Type
FROM GROUPO.Products;
```

Dies ist eine alternative Syntax:

```
CASE
WHEN search-condition
THEN expression-1, ...
[ ELSE expression-2 ]
END [ CASE ]
```

Falls die Suchbedingung erfüllt wird, die der WHEN-Klausel folgt, wird der Ausdruck zurückgegeben, der der THEN-Anweisung folgt. Andernfalls wird der Ausdruck, der der ELSE-Anweisung folgt, zurückgegeben, falls sie vorhanden ist.

Die folgende Anweisung verwendet zum Beispiel einen CASE-Ausdruck als dritte Klausel einer SELECT-Anweisung, um eine Zeichenfolge einer Suchbedingung zuzuordnen.

```
SELECT ID, Name,
  ( CASE
    WHEN Name='Tee Shirt' then 'Sale'
    WHEN Quantity >= 50 then 'Big Sale'
    ELSE 'Regular price'
  END ) as Type
FROM GROUPO.Products;
```

NULLIF-Funktion für abgekürzte CASE-Ausdrücke

Die NULLIF-Funktion ermöglicht es, einige CASE-Klauseln in Kurzform zu schreiben. Dies ist die Syntax für NULLIF:

```
NULLIF ( expression-1, expression-2 )
```

NULLIF vergleicht die Werte der zwei Ausdrücke. Wenn der erste Ausdruck gleich dem zweiten Ausdruck ist, gibt NULLIF den Wert NULL zurück. Wenn der erste Ausdruck nicht gleich dem zweiten Ausdruck ist, gibt NULLIF den ersten Ausdruck zurück.

Hinweis

Verwechseln Sie die Syntax des CASE-Ausdrucks nicht mit der der CASE-Klausel.

Standards und Kompatibilität

- **SQL/2008** Der CASE-Ausdruck ist eine Kernfunktion des SQL/2008-Standards. Der Standard lässt zu, dass jeder von der Anweisung referenzierte Ausdruck zu jedem beliebigen Zeitpunkt während der Ausführung ausgewertet werden kann. Bei SQL Anywhere werden Ausdrücke ausgewertet, wenn auch die einzelnen WHEN-Klausel in der syntaktischen Reihenfolge ausgewertet werden, mit Ausnahme von konstanten Werten, die zum Zeitpunkt der Kompilierung ermittelt werden können.

Die Unterstützung für END CASE mit CASE-Ausdrücken, zusätzlich zu END, ist eine Erweiterung des Herstellers. Der SQL/2008-Standard definiert END für die Verwendung mit CASE-Ausdrücken und END CASE für die Verwendung mit CASE-Klauseln.

Siehe auch

- „CASE-Anweisung“ auf Seite 566

Überblick über reguläre Ausdrücke

Ein **regulärer Ausdruck** ist eine Sequenz von Zeichen, Platzhalterzeichen oder Operatoren, die ein Muster für das Durchsuchen einer Zeichenfolge festlegt. SQL Anywhere unterstützt reguläre Ausdrücke als Teil der Suchbedingungen REGEXP oder SIMILAR TO in der WHERE-Klausel einer SELECT-Anweisung oder als Argument für die REGEXP_SUBSTR-Funktion. Die LIKE-Suchbedingung unterstützt keine regulären Ausdrücke. Allerdings ähneln einige der Platzhalterzeichen und Operatoren, die Sie bei LIKE verwenden können, den Platzhalterzeichen und Operatoren der regulären Ausdrücke.

Die nachstehende SELECT-Anweisung benutzt einen regulären Ausdruck `((K|C[^\h])%)`, um in der Tabelle Contacts nach Kontakten zu suchen, deren Nachname mit K oder C, nicht aber mit Ch beginnt, und diese zurückzugeben:

```
SELECT Surname, GivenName
FROM GROUPO.Contacts
WHERE Surname SIMILAR TO '(K|C[^\h])%';
```

Ein regulärer Ausdruck kann zusätzliche Syntax enthalten, um Gruppierungen, Quantifizierungen, Assertierungen und Alternierungen festzulegen, wie unten angegeben.

- **Gruppierung** Bei der Gruppierung können Teile eines regulären Ausdrucks in Gruppen zusammengefasst werden, um zusätzliche Übereinstimmungskriterien zu liefern. Beispiel: `'(abc){2}'` steht für `abcabc`.

Sie können die Gruppierung auch verwenden, um die Reihenfolge zu steuern, in der die Teile des Ausdrucks ausgewertet werden. Beispiel: `'ab(cdc d)'` sucht erst nach einem Vorkommen von "cdc d" und ermittelt dann, ob vor dem "cdc d" die Zeichenfolge "ab" steht.

- **Quantifizierung** Mit der Quantifizierung können Sie steuern, wie oft der voranstehende Teil des Ausdrucks vorkommen kann. Beispiel: Ein Fragezeichen (?) ist ein Quantifizierer, der null oder ein Auftreten des vorhergehenden Zeichens angibt. Daher gilt: `'honou?r'` stimmt sowohl mit `honor` als auch mit `honour` überein.
- **Assertierung** Normalerweise bringt die Suche nach einem Muster eben dieses Muster als Ergebnis. Mit Assertierungen können Sie das Vorhandensein eines Musters prüfen, ohne dass dieses Muster in das Suchergebnis aufgenommen wird. Beispiel: `'SQL(?= Anywhere)'` findet SQL nur, wenn danach eine Leerstelle und Anywhere kommt.
- **Alternierung** Bei der Alternierung können Sie alternative Suchmuster angeben, wenn das erste Suchmuster nicht gefunden wird. Alternative Muster werden von links nach rechts ausgewertet und die Suche bricht bei der ersten Fundstelle ab. Beispiel: `'col(o|ou)r'` sucht nach einem Vorkommen von "color". Wenn dies nicht gefunden wird, geht die Suche nach "colour" weiter.

Siehe auch

- „Syntax für reguläre Ausdrücke“ auf Seite 28
- „Suchbedingungen“ auf Seite 42
- „Die Suchbedingungen LIKE, REGEXP und SIMILAR TO“ auf Seite 48
- „REGEXP-Suchbedingung“ auf Seite 55
- „SIMILAR TO-Suchbedingung“ auf Seite 56
- „REGEXP_SUBSTR-Funktion [Zeichenfolge]“ auf Seite 349

Syntax für reguläre Ausdrücke

Reguläre Ausdrücke werden mit den Suchbedingungen SIMILAR TO und REGEXP sowie mit der Funktion REGEXP_SUBSTR unterstützt. Für SIMILAR TO stimmt die Syntax für reguläre Ausdrücke mit dem ANSI/ISO SQL-Standard überein. Für REGEXP und REGEXP_SUBSTR stimmen die Syntax und die Unterstützung mit Perl 5 überein.

Reguläre Ausdrücke werden von REGEXP und SIMILAR TO verwendet, um eine *Zeichenfolge* zu suchen, während reguläre Ausdrücke von REGEXP_SUBSTR verwendet werden, um eine *Teilzeichenfolge* zu suchen. Um das Teilzeichenfolgenverhalten für REGEXP und SIMILAR TO zu erreichen, können Sie auf beiden Seiten des gesuchten Musters Platzhalterzeichen einfügen. Beispiel: REGEXP '.*car.*' findet car, carwash und vicar. Sie können Ihre Abfrage auch so neu schreiben, dass sie die REGEXP_SUBSTR-Funktion verwendet.

Die Suche mit regulären Ausdrücken in SIMILAR TO unterscheidet nicht zwischen Groß- und Kleinschreibung. REGEXP und REGEXP_SUBSTR werden durch die Berücksichtigung von Groß- und Kleinschreibung oder von Akzenten in der Datenbank nicht beeinflusst.

Reguläre Ausdrücke: Metazeichen

Metazeichen sind Symbole oder Zeichen, die in einem regulären Ausdruck eine besondere Bedeutung haben.

Die Verarbeitung von Metazeichen kann unterschiedlich sein:

- Je nachdem, ob der reguläre Ausdruck mit den Suchbedingungen SIMILAR TO oder REGEXP bzw. der Funktion REGEXP_SUBSTR verwendet wird.
- Abhängig davon, ob das Metazeichen sich in einer Zeichenklasse des regulären Ausdrucks befindet.

Bevor Sie fortfahren, sollten Sie sich die Definition einer **Zeichenklasse** vergegenwärtigen. Eine Zeichenklasse ist eine Zeichenmenge in eckigen Klammern, mit denen eine Übereinstimmung in einer Zeichenfolge gesucht wird. In der Syntax SIMILAR TO 'ab[1-9]' ist [1-9] beispielsweise eine Zeichenklasse und passt zu einer Ziffer im Bereich 1 bis 9. Die Verarbeitung von Metazeichen in einem regulären Ausdruck kann verschieden sein, je nachdem, ob das Metazeichen in einer Zeichenklasse enthalten ist oder nicht. Insbesondere werden die meisten Metazeichen als reguläre Zeichen behandelt, wenn sie in einer Zeichenklasse enthalten sind.

Nur bei SIMILAR TO müssen die Metazeichen *, ?, +, _, |, (,), { in einer Zeichenklasse mit einem Escapezeichen versehen sein.

Um ein literales Minuszeichen (-), ein Einschaltungszeichen (^) oder eine rechte eckige Klammer (]) in eine Zeichenklasse einzubeziehen, müssen Sie davor ein Escapezeichen setzen.

Eine Liste der unterstützten Metazeichen für reguläre Ausdrücke finden Sie nachstehend. Fast alle Metazeichen werden von SIMILAR TO, REGEXP und REGEXP_SUBSTR auf identische Weise behandelt:

Zeichen	Zusätzliche Informationen
[und]	<p>Geöffnete und geschlossene eckige Klammern werden benutzt um eine Zeichenklasse anzugeben. Eine Zeichenklasse ist eine Teilmenge von Zeichen, die als Suchmuster dient.</p> <p>Mit Ausnahme des Bindestrichs (-) und des Einschaltungszeichens (^) haben Metazeichen und Quantifizierer (wie * und {m}), die in einer Zeichenklasse angegeben werden, keine besondere Bedeutung und werden als normale Zeichen behandelt.</p> <p>SQL Anywhere unterstützt auch Teilzeichenklassen wie die POSIX-Zeichenklassen.</p>
*	Das Sternchen kann benutzt werden, um in einem Suchmuster ein Zeichen 0 oder mehr Mal darzustellen. Beispiel: REGEXP '.*abc' sucht eine Zeichenfolge, die mit abc endet und mit einem beliebigen Präfix beginnt. Daher werden aabc, xyzabc und abc gefunden, nicht aber bc und abcc.
?	Das Fragezeichen kann benutzt werden, um in einem Suchmuster ein Zeichen 0 oder 1 Mal darzustellen. Beispiel: 'colou?r' findet color und colour.
+	Das Pluszeichen kann benutzt werden, um in einem Suchmuster ein Zeichen 1 oder mehr Mal darzustellen. Beispiel: 'bre+' findet bre und bree, aber nicht br.
-	Ein Bindestrich kann in einer Zeichenklasse benutzt werden, um einen Bereich darzustellen. Beispiel: REGEXP '[a-e]' findet a, b, c, d und e.
%	<p>Das Prozentzeichen kann mit SIMILAR TO verwendet werden, um eine beliebige Anzahl von Zeichen darzustellen.</p> <p>Das Prozentzeichen wird für REGEXP und REGEXP_SUBSTR nicht als Metazeichen interpretiert. Wenn es angegeben wird, findet es ein Prozentzeichen (%).</p>
_ (Unterstrich)	<p>Der Unterstrich kann mit SIMILAR TO verwendet werden, um ein einzelnes Zeichen zu finden.</p> <p>Der Unterstrich wird für REGEXP und REGEXP_SUBSTR nicht als Metazeichen interpretiert. Wenn er angegeben wird, findet er einen Unterstrich (_).</p>
	Der Senkrechtstrich wird verwendet, um alternative Suchmuster anzugeben. In einer Zeichenfolge von Suchmustern, die durch einen Senkrechtstrich getrennt sind, wird der Senkrechtstrich als OR interpretiert und die Suche hält beim ersten Treffer beginnend beim äußerst linken Suchmuster an. Daher müssen Sie die Muster in absteigender Prioritätsreihenfolge eingeben. Sie können eine unbegrenzte Anzahl alternativer Suchmuster eingeben.

Zeichen	Zusätzliche Informationen
(und)	Geöffnete und geschlossene runde Klammern sind Metazeichen, wenn sie zum Gruppieren von Teilen eines regulären Ausdrucks verwendet werden. Beispiel: <code>(ab)*</code> findet null oder mehr Wiederholungen von ab. Wie bei mathematischen Ausdrücken können Sie die Gruppierung verwenden, um die Reihenfolge zu bestimmen, in der die Teile eines regulären Ausdrucks ausgewertet werden.
{ und }	<p>Geöffnete und geschlossene geschweifte Klammern sind Metazeichen, wenn Sie zur Angabe von Quantifizierern verwendet werden. Quantifizierer geben an, wie oft sich ein Suchmuster wiederholen muss, um ein Treffer zu sein. Beispiel:</p> <ul style="list-style-type: none"> • {m} Findet ein Zeichen genau <i>m</i> Mal. Beispiel: <code>'519-[0-9]{3}-[0-9]{4}'</code> findet eine Telefonnummer mit der Vorwahl 519 (wenn die Daten so formatiert sind wie in der Syntax vorgegeben). • {m,} Findet ein Zeichen mindestens <i>m</i> Mal. Beispiel: <code>'[0-9]{5,}'</code> findet jede Zeichenfolge mit fünf oder mehr Ziffern. • {m,n} Findet ein Zeichen mindestens <i>m</i> Mal, aber nicht mehr als <i>n</i> Mal. Beispiel: <code>SIMILAR TO '_{5,10}'</code> findet jede Zeichenfolge mit 5 bis 10 Zeichen.
\	Der Backslash wird als Escapezeichen für Metazeichen benutzt. Er kann auch als Escapezeichen für Zeichen benutzt werden, die keine Metazeichen sind.
^	<p>Wenn sich bei REGEXP und REGEXP_SUBSTR ein Einschaltungszeichen außerhalb einer Zeichenklasse befindet, findet das Einschaltungszeichen den Beginn einer Zeichenfolge. Beispiel: <code>'^[hc]at'</code> findet hat und cat, aber nur am Beginn der Zeichenfolge.</p> <p>Wenn es innerhalb einer Zeichenklasse verwendet wird, gilt das nachstehende Verhalten:</p> <ul style="list-style-type: none"> • REGEXP und REGEXP_SUBSTR Wenn das Einschaltungszeichen das erste Zeichen in einer Zeichenklasse ist, findet es alle Zeichen außer denen in der Zeichenklasse. Beispiel: <code>REGEXP '[^abc]'</code> findet jedes Zeichen außer a, b oder c. <p>Wenn das Einschaltungszeichen nicht das erste Zeichen innerhalb der eckigen Klammern ist, findet es Einschaltungszeichen. Beispiel: <code>REGEXP_SUBSTR '[a-e^c]'</code> findet a, b, c, d, e und ^.</p> <ul style="list-style-type: none"> • SIMILAR TO Bei SIMILAR TO wird das Einschaltungszeichen als Subtraktionsoperator verarbeitet. Beispiel: <code>SIMILAR TO '[a-e^c]'</code> findet a, b, d und e.
\$	Wenn das Dollarzeichen mit REGEXP und REGEXP_SUBSTR verwendet wird, findet es das Ende einer Zeichenfolge. Beispiel: <code>REGEXP 'cat\$'</code> findet cat, aber nicht catfish.

Zeichen	Zusätzliche Informationen
.	Wenn das Punktzeichen mit REGEXP und REGEXP_SUBSTR verwendet wird, findet es jedes einzelne Zeichen. Beispiel: REGEXP 'a.cd' findet jede Zeichenfolge mit vier Zeichen, die mit a beginnt und mit cd endet. Wenn es mit SIMILAR TO verwendet wird, findet es einen Punkt (.).
:	Der Doppelpunkt wird in einer Zeichenklasse verwendet, um eine Teilzeichenklasse zu finden. Beispiel: '[[:alnum:]]'.

Reguläre Ausdrücke: Spezielle Teilzeichenklassen

Teilzeichenklassen sind spezielle Zeichenklassen, die in eine größere Zeichenklasse eingebettet sind. Zusätzlich zu benutzerdefinierten Zeichenklassen, bei denen Sie die Zeichenmenge definieren, mit denen eine Übereinstimmung gesucht wird (z.B. beschränkt [abxq4] die Menge der übereinstimmenden Zeichen auf a, b, x, q und 4), unterstützt SQL Anywhere auch Teilzeichenklassen wie etwa die meisten POSIX-Zeichenklassen. Beispiel: [[:alpha:]] stellt die Menge aller Groß- und Kleinbuchstaben dar.

Die Suchbedingung REGEXP und die Funktion REGEXP_SUBSTR unterstützen alle Syntaxkonventionen in der nachstehenden Tabelle, die Suchbedingung SIMILAR TO hingegen nicht. Konventionen, die von SIMILAR TO unterstützt werden, sind durch ein "J" in der Spalte SIMILAR TO gekennzeichnet.

In REGEXP und bei der Verwendung der REGEXP_SUBSTR-Funktion können Teilzeichenklassen mit einem Einschaltungszeichen negiert werden. Beispiel: [[^alpha:]] findet die Menge aller Zeichen mit Ausnahme der Buchstaben.

Teilzeichenklasse	Zusätzliche Informationen	SIMILAR TO
[:alpha:]	Findet alphabetische Zeichen in Groß- und Kleinschreibung in der aktuellen Kollation. Beispiel: '[0-9]{3}[:alpha:]{2}' findet drei Ziffern, gefolgt von zwei Buchstaben.	J
[:alnum:]	Findet Ziffern sowie alphabetische Zeichen in Groß- und Kleinschreibung in der aktuellen Kollation. Beispiel: '[:alnum:]{2}+' findet eine Zeichenfolge mit mindestens einem Buchstaben und Ziffern.	J
[:digit:]	Findet Ziffern in der aktuellen Kollation. Beispiel: '[:digit:]-]+' findet eine Zeichenfolge mit mindestens einer Ziffer oder Bindestrichen. Und '[^[:digit:]-]+' findet eine Zeichenfolge mit mindestens einem Zeichen, das weder eine Ziffer noch ein Bindestrich ist.	J

Teilzeichenklasse	Zusätzliche Informationen	SIMILAR TO
[:lower:]	Findet alphabetische Zeichen in Kleinschreibung in der aktuellen Kollation. Beispiel: ' [:lower:] ' findet A nicht, weil A ein Großbuchstabe ist.	J
[:space:]	Findet ein einzelnes Leerzeichen (' '). Beispiel: Die folgende Anweisung sucht in Contacts.City nach einer Stadt mit einem Namen aus zwei Wörtern: <pre>SELECT City FROM GROUPO.Contacts WHERE City REGEXP '.*[:space:]*';</pre>	J
[:upper:]	Findet alphabetische Zeichen in Großschreibung in der aktuellen Kollation. Beispiel: ' [:upper:]ab] ' findet einen beliebigen Großbuchstaben, a oder b.	J
[:whitespace:]	Findet eine Leerstelle wie ein Leerzeichen, Tabulatorzeichen, Zeilenvorschubzeichen und Wagenrücklaufzeichen.	J
[:ascii:]	Findet jedes 7-Bit-ASCII-Zeichen (Ordinalwert zwischen 0 und 127).	
[:blank:]	Findet ein Leerzeichen oder einen horizontalen Tabulator. [:blank:] ist gleichwertig zu [\t].	
[:cntrl:]	Findet ASCII-Zeichen mit einem Ordinalwert unter 32 oder Zeichenwert 127 (Steuerzeichen). Steuerzeichen sind Neue Zeile, Formularvorschub, Rückschritt, etc.	
[:graph:]	Findet druckbare Zeichen. [:graph:] ist gleichwertig zu [:alnum:] [:punct:]].	
[:print:]	Findet druckbare Zeichen und Leerstellen. [:print:] ist gleichwertig zu [:graph:] [:white-space:]].	
[:punct:]	Findet eines der folgenden Zeichen: !"#\$%&'()*+,-./:;<=>? @[\]^_`{ }~. In der Teilzeichenklasse [:punct:] sind möglicherweise bestimmte Nicht-ASCII-Satzzeichen, die in der aktuellen Kollation verfügbar sind, nicht enthalten.	

Teilzeichenklasse	Zusätzliche Informationen	SIMILAR TO
[:word:]	Findet alphabetische Zeichen, Ziffern oder Unterstriche in der aktuellen Kollation. [[:word:]] ist gleichwertig zu [[:alnum:] _].	
[:xdigit:]	Findet ein Zeichen, das in der Zeichenklasse [0-9A-Fa-f] ist.	

Reguläre Ausdrücke: Andere unterstützte Syntaxkonventionen

Die folgenden Syntaxkonventionen werden von der Suchbedingung REGEXP und der Funktion REGEXP_SUBSTR unterstützt und setzen voraus, dass der Backslash das Escapezeichen ist. *Diese Konventionen werden vom Suchausdruck SIMILAR TO nicht unterstützt.*

Syntax des regulären Ausdrucks	Name und Bedeutung
\0xxx	Findet das Zeichen, dessen Wert \0xxx ist, wobei xxx eine Abfolge von Oktalzeichen und 0 die Ziffer Null ist. Beispiel: \0134 findet einen Backslash.
\a	Findet das Klingelzeichen.
\A	Wenn es außerhalb des Zeichensatzes verwendet wird, findet es den Beginn einer Zeichenfolge. Entspricht der Verwendung von ^ außerhalb eines Zeichensatzes.
\b	Findet ein Rückschittzeichen.
\B	Findet ein Backslash-Zeichen (\).
\c X	Findet ein benanntes Steuerzeichen. Beispiel: \cZ für Strg-Z.

Syntax des regulären Ausdrucks	Name und Bedeutung
\d	<p>Findet eine Ziffer in der aktuellen Kollation. Beispiel: Die nachstehende Anweisung sucht in Contacts.Phone alle Telefonnummern, die mit 00 enden:</p> <pre>SELECT Surname, Surname, City, Phone FROM GROUPO.Contacts WHERE Phone REGEXP '\d{8}00';</pre> <p>\d kann sowohl innerhalb als auch außerhalb von Zeichenklassen verwendet werden und ist gleichwertig mit <code>[[:digit:]]</code>.</p>
\D	<p>Findet alles, was keine Ziffer ist. Dies ist das Gegenteil von \d.</p> <p>\D kann sowohl innerhalb als auch außerhalb von Zeichenklassen verwendet werden und ist gleichwertig mit <code>[^[:digit:]]</code>.</p> <p>Bei der Verwendung von Kurzzeichen innerhalb von eckigen Klammern ist Vorsicht geboten. <code>[\D\S]</code> ist nicht dasselbe wie <code>^[^d\s]</code>. Der zweite Ausdruck findet jedes Zeichen, das keine Ziffer oder Leerstelle ist. Also findet er x, aber nicht 8. Der erste Ausdruck hingegen findet jedes Zeichen, das entweder keine Ziffer oder kein Leerzeichen ist. Da eine Ziffer kein Leerzeichen ist, und ein Leerzeichen keine Ziffer, findet <code>[\D\S]</code> jedes Zeichen: Ziffern, Leerzeichen oder andere.</p>
\e	Findet das Escapezeichen.
\E	Beendet die Verarbeitung von Metazeichen als Nicht-Metazeichen, eingeleitet durch \Q.
\f	Findet ein Formularvorschub-Zeichen.
\n	Findet eine Zeilenendmarke.
\Q	Behandelt alle Metazeichen als Nicht-Metazeichen, bis \E angetroffen wird. Beispiel: \Q[\$\E entspricht <code>[\ \$</code> .
\r	Findet ein Wagenrücklauf-Zeichen (CR).

Syntax des regulären Ausdrucks	Name und Bedeutung
\s	<p>Findet eine Leerstelle oder ein als Leerzeichen behandeltes Zeichen. Beispiel: Die folgende Anweisung gibt alle Produktnamen aus Products.ProductName zurück, die mindestens eine Leerstelle im Namen haben:</p> <pre>SELECT Name FROM GROUPO.Products WHERE Name REGEXP '.*\s.*'</pre> <p>\s kann sowohl innerhalb als auch außerhalb von Zeichenklassen verwendet werden und ist [[: whitespace:]] gleichwertig.</p>
\S	<p>Findet ein Zeichen, das kein Leerzeichen ist. Dies ist das Gegenteil von \s und [^[: whitespace:]] gleichwertig.</p> <p>\S kann sowohl innerhalb als auch außerhalb von Zeichenklassen verwendet werden.</p> <p>Bei der Verwendung von Kurzzeichen innerhalb von eckigen Klammern ist Vorsicht geboten. [\D\S] ist nicht dasselbe wie [^\d\s]. Der zweite Ausdruck findet jedes Zeichen, das keine Ziffer oder Leerstelle ist. Also findet er x, aber nicht 8. Der erste Ausdruck hingegen findet jedes Zeichen, das entweder keine Ziffer oder kein Leerzeichen ist. Da eine Ziffer kein Leerzeichen ist, und ein Leerzeichen keine Ziffer, findet [\D\S] jedes Zeichen: Ziffern, Leerzeichen oder andere.</p>
\t	Findet einen horizontalen Tabulator.
\v	Findet einen vertikalen Tabulator.

Syntax des regulären Ausdrucks	Name und Bedeutung
<code>\w</code>	<p>Findet ein alphabetisches Zeichen, eine Ziffer oder einen Unterstrich in der aktuellen Kollation. Beispiel: Die folgende Anweisung gibt alle Nachnamen aus Contacts.Surname zurück, die genau sieben alphanumerische Zeichen lang sind:</p> <pre>SELECT Surname FROM GROUPO.Contacts WHERE Surname REGEXP '\w{7}';</pre> <p><code>\w</code> kann sowohl innerhalb als auch außerhalb von Zeichenklassen verwendet werden.</p> <p>Entspricht <code>[[:alnum:]]</code>.</p>
<code>\W</code>	<p>Findet alle Zeichen außer alphabetischen Zeichen, Ziffern oder Unterstrichen in der aktuellen Kollation. Dies ist das Gegenteil von <code>\w</code> und <code>[^[:alnum:]]</code> gleichwertig.</p> <p>Dieser reguläre Ausdruck kann sowohl innerhalb als auch außerhalb von Zeichenklassen verwendet werden.</p>
<code>\xhh</code>	<p>Findet das Zeichen, dessen Wert <code>0xhh</code> ist, wobei <code>hh</code> maximal zwei hexadezimale Zeichen sein können. Beispiel: <code>\x2D</code> entspricht dem Bindestrich.</p> <p>Entspricht <code>\x{hh}</code>.</p>
<code>\x{hhh}</code>	<p>Findet das Zeichen, dessen Wert <code>0xhhh</code> ist, wobei <code>hhh</code> maximal drei hexadezimale Zeichen sein können.</p>
<code>\z</code> und <code>\Z</code>	<p>Findet die Position (nicht das Zeichen) am Ende der Zeichenfolge.</p> <p>Entspricht <code>\$</code>.</p>

Reguläre Ausdrücke: Assertierung

Mit Assertierungen testen Sie, ob eine Bedingung zutrifft, und bestimmen die Position in der Zeichenfolge, an der die Übereinstimmungssuche beginnt. Assertierungen geben keine Zeichen zurück, da das Assertierungsmuster in der Fundstelle nicht enthalten ist. Assertierungen werden von der Suchbedingung REGEXP und der REGEXP_SUBSTR-Funktion unterstützt. Diese Konventionen werden vom Suchausdruck SIMILAR TO nicht unterstützt.

Die Assertierungen "Lookahead" und "Lookbehind" können bei REGEXP_SUBSTR nützlich sein, wenn Sie versuchen, eine Zeichenfolge aufzuteilen. So können Sie beispielsweise die Liste der Straßen (ohne Hausnummern) in der Address-Spalte der Customers-Tabelle zurückgeben, indem Sie folgende Anweisung ausführen:

```
SELECT REGEXP_SUBSTR( Street, '(?<=^\S+\s+).*\$' )
FROM GrouPO.Customers;
```

Ein anderes Beispiel: Sie können einen regulären Ausdruck verwenden, um zu prüfen, ob ein Kennwort bestimmten Regeln entspricht. Verwenden Sie eine Assertierung mit Nullbreite in der folgenden Form:

```
IF password REGEXP '(?=.*[[:digit:]])(?=.*[[:alpha:]].*[[:alpha:]])[:word:]{4,12}' THEN
    MESSAGE 'Password conforms' TO CLIENT;
ELSE
    MESSAGE 'Password does not conform' TO CLIENT;
END IF
```

Das Kennwort ist gültig, wenn folgende Bedingungen zutreffen:

- *password* hat mindestens eine Stelle (positive Assertierung der Nullbreite mit `[[:digit:]]`)
- *password* hat mindestens zwei Buchstaben (positive Assertierung der Nullbreite mit `[[:alpha:]].*[[:alpha:]]`)
- *password* enthält nur alphanumerische Zeichen oder Unterstriche (`[[:word:]]`)
- *Kennwort* ist mindestens 4 Zeichen und höchstens 12 Zeichen lang (`{4,12}`)

Die nachstehende Tabelle enthält die von SQL Anywhere unterstützten Assertierungen:

Syntax	Bedeutung
<code>(?= pattern)</code>	<p>Positive Lookahead-Assertierung mit Nullbreite Prüft, ob auf die aktuelle Position in der Zeichenfolge unmittelbar ein <i>pattern</i> folgt, ohne dass <i>pattern</i> Teil der Vergleichszeichenfolge wird. 'A(?=B)' prüft ein A, dem ein B folgt, ohne dass B Teil des Suchmusters wird.</p> <p>Beispiel: <code>SELECT REGEXP_SUBSTR('in new york city', 'new(=?\syork)')</code>; gibt die Teilzeichenfolge "new" zurück, da unmittelbar darauf "york" folgt (beachten Sie das Leerzeichen vor york).</p>
<code>(?! pattern)</code>	<p>Negative Lookahead-Assertierung mit Nullbreite Prüft, ob auf die aktuelle Position in der Zeichenfolge <i>nicht</i> unmittelbar ein <i>pattern</i> folgt, ohne dass <i>pattern</i> Teil der Vergleichszeichenfolge wird. Daher gilt: 'A(?!B)' findet ein A, auf das nicht ein B folgt.</p> <p>Beispiel: <code>SELECT REGEXP_SUBSTR('new jersey', 'new(?!\syork)')</code>; gibt die Teilzeichenfolge "new" zurück.</p>

Syntax	Bedeutung
(?<= <i>pattern</i>)	<p>Positive Lookbehind-Assertierung mit Nullbreite Prüft, ob vor der aktuellen Position in der Zeichenfolge unmittelbar ein <i>pattern</i> steht, ohne dass <i>pattern</i> Teil der Vergleichszeichenfolge wird. Daher gilt: '(?<=A)B' findet ein B, vor dem unmittelbar ein A steht, ohne dass A Bestandteil des Suchmusters wird.</p> <p>Beispiel: <code>SELECT REGEXP_SUBSTR('new york', '(?<=new\s)york');</code> gibt die Teilzeichenfolge "york" zurück.</p>
(?<! <i>pattern</i>)	<p>Negative Lookbehind-Assertierung mit Nullbreite Prüft, ob vor der aktuellen Position in der Zeichenfolge <i>nicht</i> unmittelbar ein <i>pattern</i> steht, ohne dass <i>pattern</i> Teil der Vergleichszeichenfolge wird.</p> <p>Beispiel: <code>SELECT REGEXP_SUBSTR('about york', '(?<!new\s)york');</code> gibt die Teilzeichenfolge "york" zurück.</p>
(?> <i>pattern</i>)	<p>Possessiver lokaler Unterausdruck Findet nur das größte Präfix der restlichen Zeichenfolge, das mit <i>pattern</i> übereinstimmt</p> <p>Beispiel: In 'aa' REGEXP '(?>a*)a', '(?>a*)' wird das aa gefunden (und verbraucht), nie aber nur das vorstehende a. Daher wird 'aa' REGEXP '(?>a*)a' mit FALSE ausgewertet.</p>
(?: <i>pattern</i>)	<p>Nicht-erfassender Block Diese Funktionalität entspricht einfach dem <i>pattern</i> und wird aus Kompatibilitätsgründen bereitgestellt.</p> <p>Beispiel: In 'bb' REGEXP '(?:b*)b', '(?:b*)' wird bb gefunden (und verbraucht). Hingegen wird, anders als beim possessiven lokalen Unterausdruck, das letzte b in bb aufgegeben, damit die gesamte Übereinstimmung erfolgreich gefunden wird (das heißt, dass die Übereinstimmung mit b außerhalb des nicht-erfassenden Blocks gefunden werden kann).</p> <p>Analog findet 'a(?:bc b)c' abcc und abc. Beim Finden der Übereinstimmung mit abc wird eine Rückverfolgung des letzten c in bc vorgenommen, damit das c außerhalb der Gruppe benutzt werden kann, um die Suche erfolgreich durchzuführen.</p>
(?# <i>text</i>)	Wird für Kommentare benutzt. Der Inhalt von <i>text</i> wird ignoriert.

Siehe auch

- „Beispiele für reguläre Ausdrücke“ auf Seite 39
- „REGEXP_SUBSTR-Funktion [Zeichenfolge]“ auf Seite 349
- Unterschiede im Zeichenvergleich von LIKE, REGEXP und SIMILAR TO auf Seite 49
- Reguläre Ausdrücke: Metazeichen auf Seite 28
- Reguläre Ausdrücke: Spezielle Teilzeichenklassen auf Seite 31

Beispiele für reguläre Ausdrücke

Die folgende Tabelle zeigt Beispiele für die Verwendung regulärer Ausdrücke. Alle Beispiele funktionieren für REGEXP und einige auch für SIMILAR TO, wie in der Spalte "Beispiel" angegeben. Die Ergebnisse hängen von der Suchbedingung ab, die Sie für die Suche verwenden. Ergebnisse, die mit SIMILAR TO funktionieren, können je nach Berücksichtigung der Groß- und Klein- sowie Akzentschreibung voneinander abweichen.

Beachten Sie, dass Backslashes verdoppelt werden müssen, wenn die Beispiele in Literalzeichenfolgen (z.B. ' . + @ . + \ \ . . + ') verwendet werden.

Beispiel	Fundstellen
Kreditkartennummern (nur REGEXP): Visa: <code>4[0-9]{3}\s[0-9]{4}\s[0-9]{4}\s[0-9]{4}</code> MasterCard: <code>5[0-9]{3}\s[0-9]{4}\s[0-9]{4}\s[0-9]{4}</code> American Express: <code>37[0-9]{2}\s[0-9]{4}\s[0-9]{4}\s[0-9]{4}</code> Discover: <code>6011\s[0-9]{4}\s[0-9]{4}\s[0-9]{4}</code>	Gefunden (Visa): 4123 6453 2222 1746 Nicht gefunden (Visa): 3124 5675 4400 4567, 4123-6453-2222-1746 Bei MasterCard wird eine Menge von 16 Ziffern, beginnend mit 5, gefunden, wobei zwischen jeder Teilmenge von vier Ziffern eine Leerstelle steht. American Express und Discover sind ähnlich, müssen aber mit 37 bzw. 6011 beginnen.
Datumsangaben (REGEXP und SIMILAR TO): <code>([0-2][0-9] 30 31)/([01-9] 1[0-2])/[0-9]{4}</code>	Gefunden: 31/04/1999, 15/12/4567 Nicht gefunden: 31/4/1999, 31/4/99, 1999/04/19, 42/67/25456
Absolute Pfade in Windows (nur REGEXP): <code>([A-Za-z]: \\ \/)[[:alnum:][:whitespace:]]*"#\$%&'()+, -. \ ; = @ \ [\] ^ _ ` { } ~ .] *</code>	Fundstellen: \\server\share\file Nicht gefunden: \directory\directory2, /directory2
E-Mail-Adressen (nur REGEXP): <code>[[:word:]]\[-.]+\@[[:word:]]\[-.]+\.[[:alpha:]]{2,3}</code>	Gefunden: abc.123@def456.com, _123@abc.ca Nicht gefunden: abc@dummy, ab*cd@efg.hijkl

Beispiel	Fundstellen
E-Mail-Adressen (nur REGEXP): <code>.+@.+\. .+</code>	Gefunden: *@qrstuv@wxyz.12345.com, __1234^%@@abc.def.ghijkl Nicht gefunden: abc.123.*&ca, ^%abcdefg123
Hexadezimale HTML-Farbcodes (REGEXP und SIMILAR TO): <code>[A-F0-9]{6}</code>	Gefunden: AB1234, CCCCCC, 12AF3B Nicht gefunden: 123G45, 12-44-CC
Hexadezimale HTML-Farbcodes (nur REGEXP): <code>[A-F0-9]{2}\s[A-F0-9]{2}\s[A-F0-9]{2}</code>	Gefunden: AB 11 00, CC 12 D3 Nicht gefunden: SS AB CD, AA BB CC DD, 1223AB
IP-Adressen (nur REGEXP): <code>((2(5[0-5] [0-4][0-9]) 1([0-9][0-9]) ([1-9][0-9]) [0-9])\.){3}(2(5[0-5] [0-4][0-9]) 1([0-9][0-9]) ([1-9][0-9]) [0-9]) ([0-9]))</code>	Gefunden: 10.25.101.216 Nicht gefunden: 0.0.0, 256.89.457.02
Java-Kommentare (nur REGEXP): <code>/*.**/ //[^\n]*</code>	Findet Java-Kommentare, die zwischen /* und */ gesetzt sind, oder einzeilige Kommentare, denen // vorangestellt ist. Nicht gefunden: a=1
Geldwerte (nur REGEXP): <code>(\+ -)?\\$[0-9]*\.[0-9]{2}</code>	Gefunden: \$1.00, -\$97.65 Nicht gefunden: \$1, 1.00\$, \$-75.17
Positive, negative Zahlen und Dezimalzahlen (nur REGEXP): <code>(\+ -)?[0-9]+(\.[0-9]+)?</code>	Gefunden: +41, -412, 2, 7968412, 41, +41.1, -3.141592653 Nicht gefunden: ++41, 41.1.19, --97.14
Kennwörter (REGEXP und SIMILAR TO): <code>[[:alnum:]]{4,10}</code>	Gefunden: abcd, 1234, A1b2C3d4, 1a2B3 Nicht gefunden: abc, *ab12, abcdefghijkl
Kennwörter (nur REGEXP): <code>[a-zA-Z]\w{3,7}</code>	Gefunden: AB_cd, A1_b2c3, a123_ Nicht gefunden: *&^g, abc, 1bcd

Beispiel	Fundstellen
Telefonnummern (REGEXP und SIMILAR TO): <pre> ([2-9][0-9]{2}-[2-9][0-9]{2}-[0-9]{4}) ([2-9][0-9]{2}\s[2-9][0-9]{2}\s[0-9]{4}) </pre>	Gefunden: 519-883-6898, 519 888 6898 Nicht gefunden: 888 6898, 5198886898, 519 883-6898
Texte (nur REGEXP): <pre> [A-Z0-9].*(\. \\? !) </pre>	Gefunden: Hello, how are you? Nicht gefunden: i am fine
Texte (nur REGEXP): <pre> [:upper:]0-9].*[:upper:] </pre>	Gefunden: Hello, how are you? Nicht gefunden: i am fine
Sozialversicherungsnummern (REGEXP und SIMILAR TO): <pre> [0-9]{3}-[0-9]{2}-[0-9]{4} </pre>	Gefunden: 123-45-6789 Nicht gefunden: 123 45 6789, 123456789, 1234-56-7891
URLs (nur REGEXP): <pre> (http://)?www\.[a-zA-Z0-9]+\.[a-zA-Z]{2,3} </pre>	Gefunden: http://www.sample.com, www.sample.com Nicht gefunden: http://sample.com, http://www.sample.comm

Siehe auch

- „Syntax für reguläre Ausdrücke“ auf Seite 28
- „Die Suchbedingungen LIKE, REGEXP und SIMILAR TO“ auf Seite 48

Kompatibilität von Ausdrücken

SQL Anywhere setzt die SQL/2008-Konvention ein. Das bedeutet, dass in Apostrophe (einfache Anführungszeichen) eingeschlossene Zeichenfolgen Konstantenausdrücke sind und in doppelte Anführungszeichen eingeschlossene Zeichenfolgen begrenzte Bezeichner (Namen für Datenbankobjekte).

Die Option "quoted_identifier"

SQL Anywhere enthält die Option `quoted_identifier`, die eine Änderung der Interpretation von begrenzten Zeichenfolgen zulässt. Standardmäßig ist die Option `quoted_identifier` in SQL Anywhere auf ON gesetzt.

Sie können reservierte SQL-Wörter nicht als Bezeichner verwenden, wenn die Option `quoted_identifier` auf OFF gesetzt ist.

Einstellen der Option

Die folgende Anweisung ändert die Einstellung der Option `quoted_identifier` auf ON:

```
SET quoted_identifier On;
```

Die folgende Anweisung ändert die Einstellung der Option `quoted_identifier` auf OFF:

```
SET quoted_identifier Off;
```

Kompatible Interpretation von begrenzten Zeichenfolgen

In SQL Anywhere können Sie entweder die SQL/2008-Konvention oder die standardmäßige Transact-SQL-Konvention verwenden, solange die Option `quoted_identifier` in jedem DBMS auf denselben Wert gesetzt ist.

Beispiele

Wenn Sie mit auf ON gesetzter `quoted_identifier`-Option (der standardmäßigen Einstellung) arbeiten wollen, sind die folgenden Anweisungen für das Schlüsselwort **user** für beide DBMS zulässig:

```
CREATE TABLE "user" ( coll char(5) )
go
INSERT "user" ( coll )
VALUES ( 'abcde' )
go
```

Wenn Sie mit auf OFF gesetzter `quoted_identifier`-Option arbeiten wollen, ist die folgenden Anweisung für beide DBMS zulässig: Im folgenden Beispiel ist "Chin" eine Zeichenfolge und kein Bezeichner.

```
SELECT *
FROM GROUPO.Employees
WHERE Surname = "Chin"
go
```

Siehe auch

- „`quoted_identifier`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Reservierte Wörter“ auf Seite 1

Suchbedingungen

Eine Suchbedingung ist das Kriterium, das für eine WHERE-Klausel, eine HAVING-Klausel, eine CHECK-Klausel, eine ON-Formulierung in einem Join oder einen IF-Ausdruck angegeben wird. Eine Suchbedingung wird auch als Prädikat bezeichnet.

Syntax

```
search-condition :
expression comparison-operator expression
| expression comparison-operator { [ ANY | SOME ] | ALL } ( subquery )
| expression IS [ NOT ] DISTINCT FROM expression
| expression IS [ NOT ] NULL
| expression [ NOT ] BETWEEN expression AND expression
| expression [ NOT ] LIKE pattern [ ESCAPE expression ]
| expression [ NOT ] SIMILAR TO pattern [ ESCAPE escape-expression ]
```

```

| expression [ NOT ] REGEXP pattern [ ESCAPE escape-expression ]
| expression [ NOT ] IN ( expression , ... )
| ( query-expression )
| NOT search-condition
| CONTAINS (column-name [... ] , query-string )
| EXISTS ( query-expression )
| search-condition [ { AND | OR } search-condition ] [ ... ]
| ( search-condition )
| ( search-condition , estimate )
| search-condition IS [ NOT ] { TRUE | FALSE | UNKNOWN }
| expression IS [ NOT ] OF ( [ ONLY ] type-name , ... )
| trigger-operation

```

comparison-operator :

```

=
>
<
>=
<=
<>
!=
!<
!>

```

trigger-operation :

```

INSERTING
| DELETING
| UPDATING [ ( column-name-string ) ]
| UPDATE( column-name )

```

Parameter

- ALL-Suchbedingung
- ANY- und SOME-Suchbedingungen
- IS [NOT] DISTINCT FROM-Suchbedingung
- BETWEEN-Suchbedingung
- CONTAINS-Suchbedingung
- EXISTS-Suchbedingung
- LIKE-Suchbedingung
- SIMILAR TO-Suchbedingung
- REGEXP-Suchbedingung
- IS OF *type-expression* und IS NOT OF *type-expression*
Dieses Typprädikat wurde zur Unterstützung von räumlichen Geometrien hinzugefügt, aber es kann auch für alle bestehenden Datentypen verwendet werden.

Bemerkungen

Suchbedingungen werden entweder verwendet, um eine Teilmenge der Zeilen einer Tabelle auszuwählen, oder in einer Steueranweisung wie z.B. einer IF-Anweisung, um die Ablaufsteuerung zu bestimmen.

In SQL wird jede Bedingung als TRUE, FALSE oder UNKNOWN ausgewertet. Dies wird Drei-Werte-Logik genannt. Das Ergebnis eines Vergleichs ist UNKNOWN, falls einer der beiden verglichenen Werte NULL ist.

Zeilen erfüllen eine Suchbedingung nur dann, wenn das Ergebnis der Bedingung TRUE ist. Zeilen, für die die Bedingung UNKNOWN oder FALSE ist, erfüllen die Suchbedingung nicht.

Unterabfragen bilden eine wichtige Klasse von Ausdrücken, die in vielen Suchbedingungen verwendet wird.

Die Suchbedingungen LIKE, SIMILAR TO und REGEXP sind einander sehr ähnlich.

Voraussetzungen

Verbindung mit der Datenbank ist erforderlich

Nebenwirkungen

Keine.

Siehe auch

- „Ausdrücke“ auf Seite 22
- „ALL-Suchbedingung“ auf Seite 45
- „ANY- und SOME-Suchbedingungen“ auf Seite 46
- „BETWEEN-Suchbedingung“ auf Seite 48
- „CONTAINS-Suchbedingung“ auf Seite 59
- „EXISTS-Suchbedingung“ auf Seite 66
- „LIKE-Suchbedingung“ auf Seite 51
- „SIMILAR TO-Suchbedingung“ auf Seite 56
- „REGEXP-Suchbedingung“ auf Seite 55
- „Syntax räumlicher Datentypen“ [*SQL Anywhere Server - Unterstützung für räumliche Daten*]
- „Drei-Werte-Logik“ auf Seite 67
- „NULL-Spezialwert“ auf Seite 76
- „Unterabfragen in Suchbedingungen“ auf Seite 44
- „Die Suchbedingungen LIKE, REGEXP und SIMILAR TO“ auf Seite 48

Unterabfragen in Suchbedingungen

Unterabfragen, die genau eine Spalte und entweder Null oder eine Zeile zurückgeben, können in jeder SQL-Anweisung verwendet werden, in der ein Spaltenname benutzt werden kann, auch in der Mitte eines Ausdrucks.

Ausdrücke können zum Beispiel in Vergleichsbedingungen mit Unterabfragen verglichen werden, solange die Unterabfrage nicht mehr als eine Zeile zurückgibt. Wenn die Unterabfrage (die genau eine

Spalte haben muss) eine Zeile zurückgibt, wird der Wert dieser Zeile mit dem Ausdruck verglichen. Wenn eine Unterabfrage keine Zeilen zurückgibt, ist der Wert der Unterabfrage NULL.

Unterabfragen, die genau eine Spalte und beliebig viele Zeilen zurückgeben, können in IN-, ANY- und ALL- und SOME-Suchbedingungen verwendet werden. Unterabfragen, die beliebig viele Spalten und Zeilen zurückgeben, können in EXISTS-Suchbedingungen verwendet werden. Diese Suchbedingungen werden in den folgenden Abschnitten besprochen.

Standards und Kompatibilität

- **SQL/2008** Die Verwendung einer skalaren Unterabfrage als beliebiger Ausdruck ist eine Kernfunktion des SQL/2008-Standards.

Siehe auch

- „[Vergleichsoperatoren](#)“ auf Seite 9

ALL-Suchbedingung

Syntax

expression comparison-operator **ALL** (*subquery*)

comparison-operator:

```
=
>
<
>=
<=
<>
!=
<
>
```

Bemerkungen

Bei einer ALL-Suchbedingung wird die Suchbedingung mit TRUE ausgewertet, wenn der Wert der Unterabfragen-Ergebnismenge eine leere Menge ist. Andernfalls ergibt die Suchbedingung TRUE, FALSE oder UNKNOWN, abhängig vom Wert von *expression* und der von der Unterabfrage zurückgegebenen Ergebnismenge, und zwar folgendermaßen:

Ausdruckswert	Die von der Unterabfrage zurückgegebene Ergebnismenge enthält zumindest einen NULL-Wert	Die von der Unterabfrage zurückgegebene Ergebnismenge enthält keine NULL-Werte
NULL	UNKNOWN	UNKNOWN

Ausdruckswert	Die von der Unterabfrage zurückgegebene Ergebnismenge enthält zumindest einen NULL-Wert	Die von der Unterabfrage zurückgegebene Ergebnismenge enthält keine NULL-Werte
Nicht-NULL	Wenn es mindestens einen Wert in der Ergebnismenge der Unterabfrage gibt, bei der der Vergleich mit dem Ausdruckswert FALSE ergibt, wird die Suchbedingung als FALSE bewertet. Ansonsten ergibt die Suchbedingung UNKNOWN.	Wenn es mindestens einen Wert in der Ergebnismenge der Unterabfrage gibt, bei der der Vergleich mit dem Ausdruckswert FALSE ergibt, wird die Suchbedingung als FALSE bewertet. Ansonsten ergibt die Suchbedingung TRUE.

Standards und Kompatibilität

- **SQL/2008** Kernfunktion.

ANY- und SOME-Suchbedingungen

Syntax

expression comparison-operator { ANY | SOME }(subquery)

comparison-operator:

=
>
<
>=
<=
<>
!=
<<
>>

Bemerkungen

Die Schlüsselwörter ANY und SOME sind synonym.

Bei einer ANY- oder SOME-Suchbedingung wird die Suchbedingung mit FALSE ausgewertet, wenn die Ergebnismenge der Unterabfrage eine leere Menge ist. Andernfalls ergibt die Suchbedingung TRUE, FALSE oder UNKNOWN, abhängig vom Wert von *expression* und der von der Unterabfrage zurückgegebenen Ergebnismenge, und zwar folgendermaßen:

Ausdruckswert	Die von der Unterabfrage zurückgegebene Ergebnismenge enthält zumindest einen NULL-Wert	Die von der Unterabfrage zurückgegebene Ergebnismenge enthält keine NULL-Werte
NULL	UNKNOWN	UNKNOWN

Ausdruckswert	Die von der Unterabfrage zurückgegebene Ergebnismenge enthält zumindest einen NULL-Wert	Die von der Unterabfrage zurückgegebene Ergebnismenge enthält keine NULL-Werte
Nicht-NULL	Wenn es mindestens einen Wert in der Ergebnismenge der Unterabfrage gibt, bei der der Vergleich mit dem Ausdruckswert TRUE ergibt, wird die Suchbedingung als TRUE bewertet. Ansonsten ergibt die Suchbedingung UNKNOWN.	Wenn es mindestens einen Wert in der Ergebnismenge der Unterabfrage gibt, bei der der Vergleich mit dem Ausdruckswert TRUE ergibt, wird die Suchbedingung als TRUE bewertet. Ansonsten ergibt die Suchbedingung FALSE.

Eine Suchbedingung ANY oder SOME mit Gleichheitsoperator ist TRUE, wenn *expression* gleich einem Wert im Ergebnis der Unterabfrage ist, und FALSE, wenn der Ausdruck nicht NULL und nicht gleich einem Wert im Ergebnis der Unterabfrage ist bzw. die Ergebnismenge keine NULL-Werte enthält.

Hinweis

Die Syntax = ANY oder = SOME ist mit der Verwendung des IN-Schlüsselworts äquivalent.

Standards und Kompatibilität

- **SQL/2008** Kernfunktion.

Suchbedingungen IS DISTINCT FROM und IS NOT DISTINCT FROM

Syntax

expression1 IS [NOT] DISTINCT FROM *expression2*

Bemerkungen

Die Suchbedingungen IS DISTINCT FROM und IS NOT DISTINCT FROM sind als Suchargument nutzbar und werden mit TRUE oder FALSE ausgewertet.

Die IS NOT DISTINCT FROM-Suchbedingung wird mit TRUE ausgewertet, wenn *expression1* gleich *expression2* ist oder wenn beide Ausdrücke NULL sind. Dies ist gleichbedeutend mit einer Kombination aus zwei Suchbedingungen, wie folgt:

```
expression1 = expression2 OR ( expression1 IS NULL AND expression2 IS NULL )
```

Die IS DISTINCT FROM-Syntax kehrt die Bedeutung um. Das heißt, IS DISTINCT FROM wird mit TRUE ausgewertet, wenn *expression1* nicht gleich *expression2* ist und mindestens einer der Ausdrücke nicht NULL ist. Dieser entspricht Folgendem:

```
NOT( expression1 = expression2 OR ( expression1 IS NULL AND expression2 IS NULL ) )
```

Standards und Kompatibilität

- **SQL/2008** Das Prädikat IS [NOT] DISTINCT FROM ist im SQL/2008-Standard definiert. Das Prädikat IS DISTINCT FROM ist Funktion T151, "DISTINCT-Prädikat", des SQL/2008-Standards. Das Prädikat IS NOT DISTINCT FROM ist Funktion T152, "DISTINCT-Prädikat mit Negation", des SQL/2008-Standards.

Siehe auch

- „Abfrageprädikate“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

BETWEEN-Suchbedingung

Syntax

expression [**NOT**] **BETWEEN** *start-expression* **AND** *end-expression*

Bemerkungen

Die BETWEEN-Suchbedingung kann als TRUE, FALSE oder UNKNOWN gewertet werden. Ohne das Schlüsselwort NOT wird die Suchbedingung als TRUE ausgewertet, wenn *expression* zwischen *start-expression* und *end-expression* liegt. Das Schlüsselwort NOT kehrt die Bedeutung der Suchbedingung um, lässt UNKNOWN aber unverändert.

Die BETWEEN-Suchbedingung ist gleichbedeutend mit einer Kombination aus zwei Ungleichheiten.

[**NOT**] (*expression* >= *start-expression* **AND** *expression* <= *end-expression*)

Standards und Kompatibilität

- **SQL/2008** Kernfunktion.

Die Suchbedingungen LIKE, REGEXP und SIMILAR TO

Die Suchbedingungen REGEXP, LIKE und SIMILAR TO sind einander insofern ähnlich, als sie die Übereinstimmung eines Musters mit einer Zeichenfolge suchen. Außerdem versuchen alle drei, die Übereinstimmung mit einer kompletten Zeichenfolge zu finden, nicht mit einer Teilzeichenfolge in der Zeichenfolge.

Die Basissyntax ist bei allen drei Suchbedingungen ähnlich:

expression search-condition pattern

Unterschiede in der *pattern*-Definition von LIKE, REGEXP und SIMILAR TO

- REGEXP unterstützt eine Obermenge der Syntax von regulären Ausdrücken, die von SIMILAR TO unterstützt wird. Zur Kompatibilität mit anderen Produkten unterstützt die Suchbedingung REGEXP darüber hinaus einige Syntaxerweiterungen. Außerdem haben REGEXP und SIMILAR TO verschiedene Standard-Escapezeichen und die Zeichen Unterstrich (_), Prozent (%) und Einschaltungszeichen (^) werden unterschiedlich verarbeitet. Das REGEXP-Verhalten ist stark auf Perl 5 abgestimmt (außer dort, wo die Perl-Syntax und Operatoren nicht unterstützt werden).

- Die LIKE-Syntax für *pattern* ist einfach und unterstützt eine kleine Menge von Platzhalterzeichen, nicht aber die komplette Syntax für reguläre Ausdrücke.
- Die Syntax von SIMILAR TO für *pattern* ermöglicht eine geradlinige Suche nach Musterübereinstimmungen mit regulären Ausdrücken, die im ANSI/ISO SQL-Standard definiert sind.

Unterschiede im Zeichenvergleich von LIKE, REGEXP und SIMILAR TO

Das Verhalten von REGEXP bei Vergleichen unterscheidet sich von LIKE und SIMILAR TO. Für REGEXP-Vergleiche benutzt der Datenbankserver Codepunktwerte im **Zeichensatz der Datenbank**, um Suchmuster zu vergleichen. Dies entspricht anderen Implementierungen von regulären Ausdrücken wie Perl.

Bei LIKE und SIMILAR TO benutzt der Datenbankserver die Äquivalenz und die Sortierreihenfolge in der **Kollation der Datenbank**, um Suchmuster zu vergleichen. Dies entspricht der Art, wie die Datenbank Vergleichsoperatoren wie > und = handhabt.

Der Unterschied der Zeichenvergleichsmethoden bedeutet, dass Ergebnisse des Suchmusters und der Bereichsauswertung für REGEXP und LIKE/SIMILAR ebenfalls voneinander abweichen.

- **Unterschiede bei der Suche nach Übereinstimmungen** Da REGEXP Codepunktwerte verwendet, wird ein Literal in einem Suchmuster nur als übereinstimmend ausgewertet, wenn es ein identisches Zeichen ist. Die Suche nach Übereinstimmungen wird daher bei REGEXP durch Besonderheiten, wie die Berücksichtigung von Groß-, Klein- und Akzentschreibung oder die Kollation in der Datenbank, nicht beeinflusst. Beispiel: Für 'a' wird daher nie 'A' als Übereinstimmung gefunden.

Da LIKE und SIMILAR TO die Kollation der Datenbank verwenden, werden die Ergebnisse durch die Berücksichtigung der Groß-, Klein- und Akzentschreibung beeinflusst. Wenn die Kollation der Datenbank beispielsweise die Groß-, Klein und Akzentschreibung nicht berücksichtigt, wird diese auch bei Suchvorgängen nicht berücksichtigt. 'A' kann daher für das Suchmuster 'a' gefunden werden.

- **Unterschiede bei der Auswertung von Bereichen** Da REGEXP Codepunkte für die Auswertung von Bereichen benutzt, gilt ein Zeichen als im Bereich enthalten, wenn sein Codepunkt dem Start- oder Endzeichen des Bereichs entspricht bzw. dazwischen liegt. Beispiel: Der Vergleich `x REGEXP '[A-C]'` für das einzelne Zeichen `x` entspricht `CAST(x AS BINARY) >= CAST('A' AS BINARY) AND CAST(x AS BINARY) <= CAST('C' AS BINARY)`.

Da LIKE und SIMILAR TO die Sortierreihenfolge der Kollation für die Auswertung des Bereichs verwenden, wird ein Zeichen als im Bereich enthalten angesehen, wenn seine Position in der Kollation dem Start- oder Endzeichen des Bereichs entspricht oder dazwischen liegt. Beispiel: Der Vergleich `x SIMILAR TO '[A-C]'` (wobei gilt: `x` ist ein einzelnes Zeichen) entspricht `x >= 'A' AND x <= 'C'` und die Vergleichsoperatoren werden unter Verwendung der Sortierreihenfolge der Kollation ausgewertet.

Die folgende Tabelle zeigt die Menge der Zeichen, die im Bereich `'[A-C]'` enthalten sind, wenn er durch LIKE, SIMILAR TO und REGEXP ausgewertet wird. Beide Datenbanken verwenden die 1252LATIN1-Kollation, aber die erste Datenbank berücksichtigt die Groß- und Kleinschreibung nicht, die zweite hingegen schon.

	LIKE/SIMILAR TO '[A-C]'	REGEXP '[A-C]'
<i>demo.db</i> (keine Berücksichtigung von Groß- und Kleinschreibung)	A,B,C,a,b,c, ^a ,À,Á,Â,Ã,Ä,Å,Æ,Ç,à,á,â,ã,ä,å,æ,ç	A, B, C
<i>charsensitive.db</i> (Berücksichtigung von Groß- und Kleinschreibung)	A,B,C,b,c,À,Á,Â,Ã,Ä,Å,Æ,Ç,ç	A, B, C

In den Ergebnissen ist Folgendes zu beobachten:

- LIKE und SIMILAR TO enthalten Akzentbuchstaben im Bereich.
- LIKE und SIMILAR TO enthalten unterschiedliche Zeichen je nach der Berücksichtigung der Groß- und Kleinschreibung durch die Datenbank. Insbesondere enthalten sie Kleinbuchstaben des Bereichs, die Sie möglicherweise beim Durchsuchen einer Datenbank, die die Groß- und Kleinschreibung berücksichtigt, nicht erwartet haben.

Ebenso können bei einer Datenbank mit Berücksichtigung der Groß- und Kleinschreibung einige in den Bereich einbezogene Zeichen inkonsistent erscheinen. Zum Beispiel umfasst SIMILAR TO '[A-C]' in einer Datenbank mit Berücksichtigung von Groß- und Kleinschreibung A, b, B, C, C nicht aber a, weil a in der Sortierreihenfolge vor dem Großbuchstaben A steht.

- REGEXP gibt nur A, B, C zurück, ungeachtet der Berücksichtigung der Groß- und Kleinschreibung. Wenn der Bereich Kleinbuchstaben enthalten soll, müssen Sie sie in die Bereichsdefinition einfügen. Beispiel: REGEXP '[a-cA-C]'.
- Die REGEXP-Zeichenmenge ändert sich nicht, ungeachtet der Berücksichtigung der Groß- und Kleinschreibung durch die Datenbank.

Auch wenn Ihre Datenbank eine andere Kollation verwendet oder andere Einstellungen für die Berücksichtigung der Groß-, Klein- und Akzentschreibung hat als die oben beschriebenen Datenbanken, können Sie einen ähnlichen Test durchführen, um zu ermitteln, was von LIKE, SIMILAR TO, oder REGEXP zurückgegeben wird, indem Sie sich mit der Datenbank verbinden und eine der folgenden Anweisungen ausführen:

```
SELECT CHAR( row_num ) FROM RowGenerator WHERE CHAR( row_num ) LIKE '[A-C]';
SELECT CHAR( row_num ) FROM RowGenerator WHERE CHAR( row_num ) REGEXP '[A-C]';
SELECT CHAR( row_num ) FROM RowGenerator WHERE CHAR( row_num ) SIMILAR TO '[A-C]';
```

Siehe auch

- „Überblick über reguläre Ausdrücke“ auf Seite 27
- „Syntax für reguläre Ausdrücke“ auf Seite 28
- „Beispiele für reguläre Ausdrücke“ auf Seite 39

LIKE-Suchbedingung

Syntax

Dies ist die Syntax für die LIKE-Suchbedingung:

```
expression [ NOT ] LIKE pattern [ ESCAPE escape-character ]
```

Parameter

- **expression** Die zu durchsuchende Zeichenfolge.
- **pattern** Das Muster, nach dem in *expression* gesucht werden soll.
- **Escapezeichen** Das Zeichen, das als Escapezeichen für Sonderzeichen wie Unterstriche oder Prozentzeichen benutzt werden soll.

Bemerkungen

Die LIKE-Suchbedingung versucht, eine Übereinstimmung zwischen *expression* und *pattern* zu finden. Sie ergibt TRUE, FALSE oder UNKNOWN.

Die Suchbedingung wird mit TRUE ausgewertet, wenn *expression* mit *pattern* übereinstimmt (wenn nicht NOT angegeben wurde). Falls entweder *expression* oder *pattern* der NULL-Wert ist, wird diese Suchbedingung mit UNKNOWN ausgewertet. Das Schlüsselwort NOT kehrt die Bedeutung der Suchbedingung um, lässt UNKNOWN aber unverändert.

expression wird als CHAR- oder NCHAR-Zeichenfolge interpretiert. Der gesamte Inhalt von *expression* wird für die Suche nach Übereinstimmungen verwendet. Auch *pattern* wird als CHAR- oder NCHAR-Zeichenfolge interpretiert und kann eine beliebige Anzahl von Platzhalterzeichen enthalten, die in der folgenden Tabelle aufgelistet sind:

Platzhalterzeichen	Gefunden
_ (Unterstrich)	Ein Zeichen. Beispiel: a_ finden ab und ac, nicht aber a.
% (Prozent)	Eine Zeichenfolge mit null oder mehr Zeichen. Beispiel: bl% findet bl und bla.
[]	Ein einzelnes Zeichen im angegebenen Bereich oder der angegebenen Menge. Beispiel: T[oi]m findet Tom oder Tim.
[^]	Ein einzelnes Zeichen, das <i>nicht</i> im angegebenen Bereich oder der angegebenen Menge enthalten ist. Beispiel: M[^c] findet Mb und Md, aber nicht Mc.

Alle anderen Zeichen müssen genau übereinstimmen.

Beispiel: Die folgende Suchbedingung ergibt TRUE für eine Zeile, in der der Name mit dem Buchstaben a beginnt und den Buchstaben b als zweitletztes Zeichen hat.

```
... name LIKE 'a%b_'
```

Wenn ein *escape-character* angegeben wurde, muss es mit einem Ein-Byte-CHAR- oder NCHAR-Zeichen ausgewertet werden. Das Escapezeichen kann vor einem Prozent-, einem Unterstrich-Zeichen,

einer linken eckigen Klammer oder einem anderen Sonderzeichen in *pattern* stehen, um zu verhindern, dass das Sonderzeichen eine spezielle Bedeutung hat. Wenn ein Escapezeichen gesetzt wird, werden Prozentzeichen und Unterstriche als Zeichen gefunden.

Alle Muster bis zu 126 Byte werden unterstützt. Muster mit mehr als 126 Byte, die keine Platzhalterzeichen enthalten, werden nicht unterstützt. Die Anzahl der Byte, die für die Darstellung des Musters verwendet werden, hängt davon ab, ob das Muster CHAR oder NCHAR ist.

Unterschiedliche Methoden für die Verwendung der LIKE-Suchbedingung

Suche nach	Beispiel	Zusätzliche Informationen
Eine Zeichenmenge	LIKE 'sm[iy]th'	Die Suche einer Menge von Zeichen wird festgelegt, indem die Zeichen innerhalb von eckigen Klammern aufgelistet werden. In diesem Beispiel findet die Suchbedingung <i>smith</i> und <i>smyth</i> .
Ein Zeichenbereich	LIKE '[a-r]ough'	<p>Die Suche nach einem Bereich von Zeichen wird festgelegt, indem die Bereichsgrenzen innerhalb von eckigen Klammern durch einen Bindestrich getrennt angegeben werden. In diesem Beispiel findet die Suchbedingung <i>bough</i> und <i>rough</i>, aber nicht <i>tough</i>.</p> <p>Der Bereich der Zeichen [a-z] wird als "größer als oder gleich mit a und kleiner als oder gleich mit z" interpretiert, wobei die Vorgänge "größer als" und "kleiner als" innerhalb der Kollation der Datenbank ausgeführt werden.</p> <p>Die untere Bereichsgrenze muss vor der oberen Bereichsgrenze stehen. Beispiel: [z-a] findet nichts, weil kein Zeichen zum Bereich [z-a] gehört.</p>
Bereiche und Mengen kombiniert	... LIKE '[a-rt]ough'	<p>Sie können Bereiche und Mengen innerhalb eckiger Klammern kombinieren. In diesem Beispiel findet ... LIKE '[a-rt]ough' <i>bough</i>, <i>rough</i> und <i>tough</i>.</p> <p>Das Muster "[a-rt]" wird als "Genau ein Zeichen, das entweder im Bereich von a bis inklusive r liegt oder t ist" interpretiert.</p>
Ein Zeichen außerhalb des Bereichs	... LIKE '[^a-r]ough'	<p>Mit dem Einschaltungszeichen (^) wird ein Bereich von Zeichen festgelegt, der von einer Suche ausgeschlossen ist. In diesem Beispiel findet LIKE '[^a-r]ough' die Zeichenfolge <i>tough</i>, aber nicht <i>rough</i> oder <i>bough</i>.</p> <p>Das Einschaltungszeichen negiert den Rest des Inhalts der eckigen Klammern. Der Klammerausdruck [^a-rt] wird als "Genau ein Zeichen, das nicht im Bereich von a bis inklusive r liegt oder nicht t ist" interpretiert.</p>

Suche nach	Beispiel	Zusätzliche Informationen
Suchmuster mit nachgestellten Leerzeichen	'90 ', '90[]' und '90_'	Wenn Ihr Suchmuster nachgestellte Leerzeichen enthält, bringt der Datenbankserver das Suchmuster nur mit Werten in Übereinstimmung, die Leerzeichen enthalten – es wird nicht mit Leerzeichen aufgefüllt. Die Suchmuster "90 ", "90[]" und "90_" stimmen z.B. mit dem Ausdruck "90 " überein, nicht aber mit "90", auch wenn der geprüfte Wert in einer CHAR- oder VARCHAR-Spalte steht, die drei oder mehr Zeichen breit ist.

Spezialfälle von Bereichen und Mengen

Jedes einzelne Zeichen in eckigen Klammern bedeutet genau dieses Zeichen. Beispiel: [a] entspricht nur dem Zeichen a. [^] entspricht nur dem Einschaltungszeichen, [%] entspricht nur dem Prozentzeichen (das Prozentzeichen steht in diesem Fall nicht als Platzhalter) und [_] entspricht nur dem Unterstrich-Zeichen. [[] entspricht ebenso nur dem Zeichen [.

Andere Spezialfälle:

- Das Muster [a -] entspricht dem Zeichen a oder -.
- Das Muster [] findet nie eine Entsprechung und gibt nie Zeilen zurück.
- Die Muster [oder [abp-q geben Syntaxfehler zurück, weil eine geschlossene eckige Klammer fehlt.
- Platzhalterzeichen können nicht innerhalb von eckigen Klammern verwendet werden. Das Muster [a %b] findet a, % oder b.
- Sie können das Einschaltungszeichen nur als erstes Zeichen in der eckigen Klammer zum Negieren von Bereichen verwenden. Das Muster [a^b] findet a, ^ oder b.

Berücksichtigung von Groß- und Kleinschreibung und Vorgehensweise beim Vergleichen

Wenn die Kollation der Datenbank Groß-/Kleinschreibung berücksichtigt, tut die Suchbedingung dies ebenfalls. Wenn Sie eine Suche ohne Berücksichtigung der Groß- und Kleinschreibung mit einer Kollation durchführen möchten, in der die Groß- und Kleinschreibung berücksichtigt wird, müssen Sie Groß- und Kleinbuchstaben eingeben. Die folgende Suchbedingung ergibt zum Beispiel für die Zeichenfolgen Bough, rough und TOUGH den Wert TRUE.

```
LIKE '[a-zA-Z][oO][uU][gG][hH]'
```

Vergleiche werden Zeichen für Zeichen ausgeführt, im Gegensatz zum Gleichheitsoperator (=) und zu anderen Operatoren, bei denen der Vergleich Zeichenfolge für Zeichenfolge erfolgt. Wenn beispielsweise ein Vergleich in einer UCA-Kollation ausgeführt wird (CHAR oder NCHAR mit Kollationseinstellung UCA) ist 'Æ' = 'ÆE' TRUE, aber 'Æ' LIKE 'ÆE' ist FALSE.

Bei einem Zeichen-für-Zeichen-Vergleich muss jedes einzelne Zeichen im gesuchten Ausdruck mit einem einzelnen Zeichen (entsprechend der Zeichenäquivalenz der Kollation) oder einem Platzhalterzeichen im LIKE-Ausdruck übereinstimmen.

Unterstützung nationaler Zeichen (National Characters, NCHAR)

LIKE-Suchbedingungen können verwendet werden, um CHAR- und NCHAR-Zeichenfolgen zu vergleichen. In diesem Fall wird die Zeichensatzkonvertierung so durchgeführt, dass der Vergleich mit einem gemeinsamen Datentyp erfolgt. Danach wird ein Zeichen-für-Zeichen-Vergleich durchgeführt.

Sie können *expression* oder *pattern* als NCHAR-Zeichenfolgenliteral angeben, indem Sie den in Anführungszeichen gesetzten Wert mit dem Präfix N versehen (zum Beispiel *expression* LIKE N'*pattern*'). Sie können auch die CAST-Funktion verwenden, um das Muster in CHAR oder NCHAR umzuwandeln (z.B. *expression* LIKE CAST(*pattern* AS datatype)).

Mit Leerzeichen aufgefüllte Datenbanken

Die Semantik eines LIKE-Musters ändert sich nicht, wenn die Datenbank mit Leerzeichen aufgefüllt ist, da die Suche nach Übereinstimmungen von *expression* mit *pattern* einen Zeichen-für-Zeichen-Vergleich von links nach rechts erfordert. Während der Auswertung erfolgt keine zusätzliche Auffüllung mit Leerzeichen für den Wert von *expression* oder *pattern*. Der Ausdruck *a1* findet daher das Muster *a1*, aber nicht die Muster '*a1*' (*a1*, mit Leerstelle danach) oder *a1_*.

Standards und Kompatibilität

- **SQL/2008** Die Suchbedingung LIKE ist eine Kernfunktion des SQL/2008-Standards. Es gibt jedoch feine Unterschiede im Verhalten gegenüber dem Standard, weil SQL Anywhere Kollationen ohne Berücksichtigung von Groß- und Kleinschreibung und das Auffüllen mit Leerzeichen unterstützt.

SQL Anywhere unterstützt die optionale SQL-Sprachenfunktion F281, die zulässt, dass das Muster und Escapeausdrücke beliebige Ausdrücke sein können, die zum Ausführungszeitpunkt ausgewertet werden. Funktion F281 lässt auch zu, dass *expression* ein komplexerer Ausdruck ist als eine einfache Spaltenreferenz.

Die Verwendung von Zeichenbereichen und Zeichenmengen in eckigen Klammern [] ist eine Erweiterung des Herstellers.

SQL Anywhere unterstützt SQL/2008-Funktion T042, die zulässt, dass LIKE-Suchbedingungen Zeichenfolgenausdrücke referenzieren, die LONG VARCHAR-Werte sind.

LIKE-Suchbedingungen, die NCHAR-Zeichenfolgenausdrücke oder Muster angeben, gehören zur optionalen SQL-Sprachenfunktion F421 des ANSI SQL/2008-Standards.

Siehe auch

- „Vergleiche zwischen CHAR und NCHAR“ auf Seite 141
- „Zeichenfolgenlitterale“ auf Seite 8
- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186
- Unterschiede im Zeichenvergleich von LIKE, REGEXP und SIMILAR TO auf Seite 49
- „Die Suchbedingungen LIKE, REGEXP und SIMILAR TO“ auf Seite 48
- „Die WHERE-Klausel: Zeilen angeben“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „REGEXP-Suchbedingung“ auf Seite 55
- „SIMILAR TO-Suchbedingung“ auf Seite 56

REGEXP-Suchbedingung

Sucht nach Übereinstimmungen eines Musters mit einer Zeichenfolge.

Syntax

expression [**NOT**] **REGEXP** *pattern* [**ESCAPE** *escape-expression*]

Parameter

- **expression** Die zu durchsuchende Zeichenfolge.
- **pattern** Der reguläre Ausdruck, nach dem in *expression* gesucht werden soll.
- **escape-expression** Das für die Übereinstimmung zu verwendende Escapezeichen. Das Standardzeichen ist der Backslash (\).

Bemerkungen

Die REGEXP-Suchbedingung findet eine komplette Zeichenfolge, keine Teilzeichenfolge. Um die Übereinstimmung einer Teilzeichenfolge mit der Zeichenfolge zu finden, setzen Sie die Zeichenfolge zwischen Platzhalterzeichen, die für den Rest der Zeichenfolge stehen (. *pattern. *). Beispiel: `SELECT ... WHERE Description REGEXP 'car'` findet nur car, nicht aber sportscar. Hingegen findet `SELECT ... WHERE Description REGEXP '.*car'` car, sportscar und jede Zeichenfolge, die mit car endet. Als Alternative können Sie Ihre Abfrage auch neu schreiben und die REGEXP_SUBSTR-Funktion verwenden, die speziell für die Suche nach einer Teilzeichenfolge innerhalb einer Zeichenfolge entwickelt wurde.

Wenn Übereinstimmungen mit einer Teilzeichenklasse gesucht werden, müssen Sie die äußeren eckigen Klammern und die eckigen Klammern für die Teilzeichenklasse eingeben. Beispiel: `expression REGEXP '[[:digit:]]'`.

Datenbankkollation und Suche nach Übereinstimmungen

REGEXP findet ein Literal in einem Suchmuster nur, wenn es genau dasselbe Zeichen darstellt (also wenn es denselben Codepunktwert hat). Bereiche in Zeichenklassen (z.B. '[A-F]') finden nur Zeichen mit Codepunktwerten, die mindestens gleich dem Codepunktwert des ersten Zeichens des Bereichs (A) und maximal gleich dem Codepunktwert des zweiten Zeichens des Bereichs (F) sind.

Vergleiche werden Zeichen für Zeichen ausgeführt, im Gegensatz zum Gleichheitsoperator (=) und zu anderen Operatoren, bei denen der Vergleich Zeichenfolge für Zeichenfolge erfolgt. Wenn beispielsweise ein Vergleich in einer UCA-Kollation ausgeführt wird (CHAR oder NCHAR mit Kollationseinstellung UCA) ist 'Æ' = 'AE' TRUE, aber 'Æ' REGEXP 'AE' ist FALSE.

Unterstützung nationaler Zeichen (National Characters, NCHAR)

REGEXP-Suchbedingungen können verwendet werden, um CHAR- und NCHAR-Zeichenfolgen zu vergleichen. In diesem Fall wird die Zeichensatzkonvertierung so durchgeführt, dass der Vergleich mit einem gemeinsamen Datentyp erfolgt. Danach wird ein Codepunkt-für-Codepunkt-Vergleich durchgeführt.

Sie können *expression* oder *pattern* als NCHAR-Zeichenfolgenliteral angeben, indem Sie den in Anführungszeichen gesetzten Wert mit dem Präfix N versehen (zum Beispiel `expression REGEXP N'pattern'`). Sie können auch die CAST-Funktion verwenden, um das Muster in CHAR oder NCHAR umzuwandeln (z.B. `expression REGEXP CAST(pattern AS datatype)`).

Siehe auch

- „Vergleiche zwischen CHAR und NCHAR“ auf Seite 141
- „Zeichenfolgenlitterale“ auf Seite 8
- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186
- „Überblick über reguläre Ausdrücke“ auf Seite 27
- „SIMILAR TO-Suchbedingung“ auf Seite 56
- „LIKE-Suchbedingung“ auf Seite 51
- „REGEXP_SUBSTR-Funktion [Zeichenfolge]“ auf Seite 349
- „Die Suchbedingungen LIKE, REGEXP und SIMILAR TO“ auf Seite 48
- Reguläre Ausdrücke: Spezielle Teilzeichenklassen auf Seite 31

Standards und Kompatibilität

- **SQL/2008** Die Suchbedingung REGEXP ist eine Erweiterung des Herstellers, ist aber ungefähr kompatibel mit der LIKE_REGEX-Suchbedingung des SQL/2008-Standards (SQL-Sprachenfunktion F841).

SQL Anywhere unterstützt ANSI SQL/2008-Funktion F281, die zulässt, dass das Muster und Escapeausdrücke beliebige Ausdrücke sein können, die zum Ausführungszeitpunkt ausgewertet werden. Funktion F281 lässt auch zu, dass *expression* ein komplexerer Ausdruck ist als eine einfache Spaltenreferenz.

SQL Anywhere unterstützt die ANSI-SQL/2008-Funktion T042, die zulässt, dass LIKE-Suchbedingungen Zeichenfolgenausdrücke referenzieren, die LONG VARCHAR-Werte sind.

REGEXP-Suchbedingungen, die NCHAR-Zeichenfolgenausdrücke oder Muster angeben, gehören zu Funktion F421 des ANSI SQL/2008-Standards.

SIMILAR TO-Suchbedingung

Sucht nach Übereinstimmungen eines Musters mit einer Zeichenfolge.

Syntax

`expression [NOT] SIMILAR TO pattern [ESCAPE escape-expression]`

Parameter

- **expression** Der zu suchende Ausdruck.
- **pattern** Der reguläre Ausdruck, nach dem in *expression* gesucht werden soll.
- **escape-expression** Das für die Übereinstimmung zu verwendende Escapezeichen. Das Standard-Escapezeichen ist das Nullzeichen, das in einem Zeichenfolgenliteral als `'\x00'` angegeben werden kann.

Syntax des regulären Ausdrucks	Bedeutung
<code>\x</code>	Findet alles, was gleich <i>x</i> ist, wobei als Escapezeichen das Backslash-Zeichen (\) angenommen wird. Beispiel: <code>\[</code> steht für '['.
<code>x</code>	Jedes Zeichen (außer einem Metazeichen) findet sich selbst. Beispiel: A findet 'A'.

Bemerkungen

Um eine Übereinstimmung von einer Teilzeichenfolge mit der Zeichenfolge zu finden, verwenden Sie das Prozentzeichen als Platzhalterzeichen (*%expression*). Beispiel: `SELECT ... WHERE Description SIMILAR TO 'car'` findet nur car, nicht aber sportscar. Hingegen findet `SELECT ... WHERE Description SIMILAR TO '%car'` car, sportscar, und jede Zeichenfolge, die mit car endet.

Wenn Übereinstimmungen mit einer Teilzeichenklasse gesucht werden, müssen Sie die äußeren eckigen Klammern und die eckigen Klammern für die Teilzeichenklasse eingeben. Beispiel: `expression SIMILAR TO [[:digit:]]`.

Vergleiche werden Zeichen für Zeichen ausgeführt, im Gegensatz zum Gleichheitsoperator (=) und zu anderen Operatoren, bei denen der Vergleich Zeichenfolge für Zeichenfolge erfolgt. Wenn beispielsweise ein Vergleich in einer UCA-Kollation ausgeführt wird (CHAR oder NCHAR mit Kollationseinstellung UCA) ist `'Æ' = 'AE'` TRUE, aber `'Æ' SIMILAR TO 'AE'` ist FALSE.

Bei einem Zeichen-für-Zeichen-Vergleich muss jedes einzelne Zeichen im gesuchten Ausdruck mit einem einzelnen Zeichen oder einem Platzhalterzeichen im SIMILAR TO-Muster übereinstimmen.

Datenbankkollation und Suche nach Übereinstimmungen

SIMILAR TO verwendet die Kollation, um die Gleichwertigkeit von Zeichen zu ermitteln und Zeichenklassenbereiche auszuwerten. Beispiel: Wenn die Datenbank die Groß-, Klein- und Akzentschreibung nicht berücksichtigt, wird diese auch bei Suchvorgängen nicht berücksichtigt. Bereiche werden auch anhand der Sortierreihenfolge der Kollation ausgewertet.

Unterstützung nationaler Zeichen (National Characters, NCHAR)

SIMILAR TO-Suchbedingungen können verwendet werden, um CHAR- und NCHAR-Zeichenfolgen zu vergleichen. In diesem Fall wird die Zeichensatzkonvertierung so durchgeführt, dass der Vergleich mit einem gemeinsamen Datentyp erfolgt. Danach wird ein Zeichen-für-Zeichen-Vergleich durchgeführt.

Sie können *expression* oder *pattern* als NCHAR-Zeichenfolgenliteral angeben, indem Sie den in Anführungszeichen gesetzten Wert mit dem Präfix N versehen (zum Beispiel `expression SIMILAR TO N'pattern'`). Sie können auch die CAST-Funktion verwenden, um das Muster in CHAR oder NCHAR umzuwandeln (z.B. `expression SIMILAR TO CAST(pattern AS datatype)`).

Siehe auch

- „Überblick über reguläre Ausdrücke“ auf Seite 27
- „Vergleiche zwischen CHAR und NCHAR“ auf Seite 141
- „Zeichenfolgenlitterale“ auf Seite 8
- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186
- „REGEXP-Suchbedingung“ auf Seite 55
- „LIKE-Suchbedingung“ auf Seite 51
- „REGEXP_SUBSTR-Funktion [Zeichenfolge]“ auf Seite 349
- Reguläre Ausdrücke: Spezielle Teilzeichenklassen auf Seite 31
- „Die Suchbedingungen LIKE, REGEXP und SIMILAR TO“ auf Seite 48

Standards und Kompatibilität

- **SQL/2008** Das Prädikat SIMILAR TO ist die optionale SQL-Sprachenfunktion T141 des SQL/2008-Standards.

IN-Suchbedingung

Syntax

expression [**NOT**] **IN** { (*query-expression*) | (*expression-list*) }

Bemerkungen

Eine IN-Suchbedingung vergleicht *expression* mit einer Gruppe von Werten, die von *query-expression* zurückgegeben werden, oder einer Gruppe von Werten, die in *expression-list* angegeben sind. Ohne das Schlüsselwort NOT wird die IN-Suchbedingung nach den folgenden Regeln bewertet:

- TRUE, wenn *expression* nicht NULL ist und gleich mindestens einem der Werte ist.
- UNKNOWN, wenn *expression* NULL und die Werteliste nicht leer ist oder wenn mindestens einer der Werte NULL ist und *expression* nicht gleich einem der anderen Werte ist.
- FALSE, wenn *expression* NULL ist und *query-expression* keine Werte zurückgibt; oder wenn *expression* nicht NULL ist, keiner der Werte NULL ist und *expression* gleich keinem der Werte ist.

Das Schlüsselwort NOT vertauscht TRUE und FALSE.

Die Suchbedingung *expression* **IN** (*expression-list*) entspricht *expression* = **ANY** (*expression-list*).

Die Suchbedingung *expression* **NOT IN** (*expression-list*) entspricht *expression* \neq **ALL** (*expression-list*).

Die Ausdrücke in einer *expression-list* können ein Literal, eine Variable, eine Hostvariable oder ein Abfrageausdruck sein, deren Ergebnis eine einzige Zeile und eine einzige Spalte ist.

Standards und Kompatibilität

- **SQL/2008** Kernfunktion.

CONTAINS-Suchbedingung

Syntax

CONTAINS (*column-name* [...], *contains-query-string*)

contains-query-string :
simple-expression
 | *or-expression*

simple-expression :
primary-expression
 | *and-expression*

or-expression :
simple-expression { **OR** | | } *contains-query-string*

primary-expression :
basic-expression
 | **FUZZY** " *fuzzy-expression* "
 | *and-not-expression*

and-expression :
primary-expression [**AND** | &] *simple-expression*

and-not-expression :
primary-expression [**AND** | &] { **NOT** | - } *basic-expression*

basic-expression :
term
 | *phrase*
 | (*contains-query-string*)
 | *near-expression*
 | *before-expression*

fuzzy-expression :
term
 | *fuzzy-expression term*

term :
simple-term
 | *prefix-term*

prefix-term :
*simple-term**

phrase :
 " *phrase-string* "

near-expression :
term **NEAR** [[*min-distance*], *max-distance*] *term*
 | *term* { **NEAR** | ~ } *term*

before-expressions :
term **BEFORE** [[*min-distance*] *max-distance*] *term*
 | *term* **BEFORE** *term*

phrase-string :
term
| *phrase-string term*

simple-term: Eine Zeichenfolge, die durch Leerzeichen und Sonderzeichen unterbrochen wird und einen einzelnen, indizierten Begriff (Wort) darstellt, nach dem gesucht werden soll.

distance: Eine positive Ganzzahl

Parameter

- **and-expression** Gibt an, dass sowohl *primary-expression* als auch *simple-expression* im Textindex gefunden werden müssen.

Standardmäßig gilt: Wenn zwischen Begriffen oder Ausdrücken kein Operator steht, wird *and-expression* angenommen. 'a b' wird beispielsweise als 'a AND b' interpretiert.

Ein kaufmännisches Und (&) kann anstelle von AND verwendet werden und ohne Zwischenraum an beiden Seiten an die Ausdrücke oder Begriffe gesetzt werden (Beispiel: 'a &b').

- **and-not-expression** Gibt an, dass *primary-expression* im Textindex vorhanden sein muss, aber der *basic-expression* im Textindex *nicht* enthalten sein darf. Dies wird auch als **Negation** bezeichnet.

Wenn Sie einen Bindestrich für die Negation verwenden, muss links vor dem Bindestrich ein Leerzeichen gesetzt werden, rechts nach dem Bindestrich hingegen muss der Begriff ohne Leerzeichen unmittelbar folgen, da sonst der Bindestrich nicht als Negationszeichen angesehen wird. Beispiel: 'a -b' ist gleich 'a AND NOT b', während bei 'a - b' der Bindestrich ignoriert wird und die Zeichenfolge 'a AND b' entspricht. 'a-b' entspricht der Phrase '"a b" '.

- **or-expression** Gibt an, dass entweder *simple-expression* oder *contains-query-string* im Textindex vorhanden sein muss. 'a|b' wird beispielsweise als 'a OR b' interpretiert.
- **fuzzy-expression** Sucht nach Begriffen, die dem angegebenen Begriff ähnlich sind. Fuzzy-Suchen werden nur in NGRAM-Textindizes unterstützt.
- **near-expression** Sucht nach Begriffen, die nahe beieinander liegen. Dies wird auch als **Nachbarschaftssuche** bezeichnet. 'b NEAR[5] c' sucht beispielsweise nach Vorkommen von b und c, die maximal fünf Begriffe voneinander entfernt sind. Die Reihenfolge der Begriffe ist nicht wichtig. 'b NEAR c' entspricht 'c NEAR b'.

Wenn die maximale Entfernung nicht angegeben ist, beträgt die Standardentfernung 10. Wenn die Mindestentfernung nicht angegeben ist, beträgt der Standardwert 1.

Die Abfrage 'apple NEAR[2, 10] tree' findet Übereinstimmungen in den folgenden Dokumenten:

```
"apple grows on the tree"  
"apple and tree"  
"tree and apple"
```

Die Abfrage findet jedoch keine Übereinstimmungen in den folgenden Dokumenten:

```
"apple tree"
"tree apple"
```

Sie können statt NEAR auch eine Tilde (~) eingeben. Die Verwendung einer Tilde ist gleichwertig mit dem Angeben von NEAR ohne Entfernung, sodass standardmäßig von höchstens 10 Begriffen und mindestens 1 Begriff ausgegangen wird. Sie können keine Höchst- oder Mindestentfernung angeben, wenn Sie eine Tilde verwenden. Der Wert beträgt dann immer 10 Begriffe.

NEAR-Ausdrücke können nicht verkettet werden (z.B. ist 'a NEAR[1] b NEAR[1] c' ungültig).

- **before-expression** Verwenden Sie *before-expression*, um nach einem Begriff suchen, der vor einen anderen Begriff steht. Dies wird auch als **Nachbarschaftssuche** bezeichnet. Die Argumente müssen im übereinstimmenden Text in derselben Reihenfolge gefunden werden, wie sie in der CONTAINS-Abfragezeichenfolge angegeben sind. 'apple BEFORE[2, 10] tree' findet beispielsweise Übereinstimmungen in den folgenden Dokumenten:

```
"apple grows on the tree"
"apple and tree"
```

Die Abfrage findet jedoch keine Übereinstimmungen in den folgenden Dokumenten:

```
"tree and apple"
"apple tree"
"tree apple"
```

Die folgenden Abfragen sind gleichwertig:

```
'apple BEFORE tree'
'apple BEFORE[10] tree'
'apple BEFORE[1, 10] tree'
```

Wenn die maximale Entfernung nicht angegeben ist, beträgt die Standardentfernung 10. Wenn die Mindestentfernung nicht angegeben ist, beträgt der Standardwert 1.

- **prefix-term** Sucht nach Begriffen, die mit dem angegebenen Präfix beginnen. Beispiel: 'datab*' sucht nach einem Begriff, der mit datab beginnt. Dies wird auch als **Präfixsuche** bezeichnet. In einer Präfixsuche werden Übereinstimmungen in dem Abschnitt des Begriffs gesucht, der sich links vom Sternchen befindet.

Bemerkungen

Die CONTAINS-Suchbedingung übernimmt eine Spaltenliste und *contains-query-string* als Argumente. Sie kann überall dort verwendet werden, wo eine Suchbedingung (auch als Prädikat bezeichnet) angegeben werden kann, und gibt TRUE oder FALSE zurück. *contains-query-string* muss eine Konstantenzeichenfolge oder eine Variable sein, mit einem Wert, der zum Zeitpunkt der Abfrage bekannt ist. *contains-query-string* darf weder NULL oder eine leere Zeichenfolge sein noch 300 gültige Begriffe überschreiten. Ein gültiger Begriff ist ein Begriff, der innerhalb der zulässigen Begriffslänge liegt und nicht in die Stoppliste aufgenommen wurde. Ein Fehler wird zurückgegeben, wenn *contains-query-string* 300 gültige Begriffe überschreitet.

Wenn die Textkonfigurationseinstellungen bewirken, dass alle Begriffe in *contains-query-string* ausgeschlossen werden, lautet das Ergebnis der CONTAINS-Suchbedingung FALSE.

Weitere Hinweise zur Interpretation von *contains-query-string* finden Sie unter „[Beispiel-Textkonfigurationsobjekte](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Wenn mehrere Spalten angegeben werden, müssen sie sich alle auf eine einzige Basistabelle beziehen. Ein Textindex kann sich nicht über mehrere Basistabellen erstrecken. Diese Basistabelle kann direkt in der FROM-Klausel referenziert werden oder sie kann in einer Ansicht oder einer abgeleiteten Tabelle verwendet werden.

Die folgenden Warnungen gelten für die Verwendung von nicht-alphanumerischen Zeichen in Abfragezeichenfolgen:

- Ein Stern in der Mitte eines Begriffs bewirkt die Rückgabe eines Fehlers.
- Nicht alphanumerische Zeichen (einschließlich Sonderzeichen) dürfen nicht in *fuzzy-expression* verwendet werden, weil sie als Leerzeichen behandelt werden und als Begriffssegmentierer fungieren.
- Wenn möglich, sollten nicht-alphanumerische Zeichen, die keine Sonderzeichen sind, in einer Abfragezeichenfolge nicht verwendet werden. Jedes nicht-alphanumerische Zeichen, das kein Sonderzeichen ist, bewirkt, dass der Begriff, in dem es vorkommt, als Phrase behandelt wird, wobei der Begriff an der Stelle des Zeichens segmentiert wird. 'things we've done' wird beispielsweise als 'things "we ve" done' interpretiert.

Innerhalb von Phrasen ist der Stern das einzige Sonderzeichen, das weiterhin als Sonderzeichen interpretiert wird. Alle anderen Sonderzeichen in Phrasen werden als Leerzeichen interpretiert und als Begriffssegmentierer verwendet.

Die Interpretation von *contains-query-string* erfolgt in zwei Hauptschritten:

- **Schritt 1: Interpretation von Operatoren und Vorrang** Während dieses Schrittes werden Schlüsselwörter als Operatoren interpretiert und Vorrangregeln angewendet.
- **Schritt 2: Anwendung von Einstellungen für das Textkonfigurationsobjekt** Während dieses Schrittes werden die Einstellungen für Textkonfigurationsobjekte auf die Begriffe angewendet. Beispiel: Bei einem NGRAM-Textindex werden Begriffe in ihre n-gram-Darstellung segmentiert. Während dieses Schrittes werden die Abfragebegriffe, die die Einstellungen für die Begriffslänge überschreiten oder in der Stoppliste enthalten sind, entfernt.

Operatorvorrang in einer CONTAINS-Suchbedingung

Während der Auswertung der Abfrage werden die Ausdrücke gemäß der nachstehenden Vorrangfolge ausgewertet:

1. FUZZY, NEAR
2. AND NOT
3. AND
4. OR

Behandlung von BEFORE als Schlüsselwort

SQL Anywhere unterstützt derzeit nicht das BEFORE-Schlüsselwort als Operator. Wenn Sie zum Beispiel `"CONTAINS(column-name, 'a before b')"` angeben, wird ein Fehler zurückgegeben. Erstellen Sie Ihre Abfrage stattdessen mit dem NEAR-Schlüsselwort.

Sie *können* nach dem Wort **before** suchen, sofern es Teil einer Phrasenabfrage ist. z. B.
`CONTAINS(column-name, '"a before b"')`. Dies sucht nach der Phrase "a before b".

Zulässige Syntax für den Sternchen-Platzhalter (*)

Das Sternchen wird für die **Präfixsuche** verwendet. Ein Sternchen muss entweder am Ende der Abfragezeichenfolge stehen oder es muss ein Leerzeichen, ein kaufmännisches Und, ein Senkrechtstrich, eine geschlossene eckige Klammer oder ein schließendes Anführungszeichen darauf folgen. In allen anderen Fällen wird ein Fehler gemeldet.

In der folgenden Tabelle werden die zulässigen Verwendungsmöglichkeiten für einen Sternchen-Platzhalter angegeben:

Abfragezeichenfolge	Entspricht	Interpretiert als
<code>'th*'</code>		Suche nach allen Begriffen, die mit th beginnen.
<code>'th*&best'</code>	<code>'th* AND best'</code> und <code>'th* best'</code>	Suche nach einem Begriff, der mit th beginnt, und nach dem Begriff best.
<code>'th* best'</code>	<code>'th* OR best'</code>	Suche entweder nach einem Begriff, der mit th beginnt, oder nach dem Begriff "best".
<code>'very&(best th*)'</code>	<code>'very AND (best OR th*)'</code>	Suche nach dem Begriff "very" und dem Begriff "best" oder einem beliebigen Begriff, der mit th beginnt.
<code>'"fast auto*"'</code>		Suche nach dem Begriff "fast", auf den unmittelbar ein Begriff folgt, der mit auto beginnt.
<code>'"auto* price"'</code>		Suche nach einem Begriff, der mit auto beginnt und auf den unmittelbar der Begriff "price" folgt.

Hinweis

Die Interpretation der Abfragezeichenfolgen mit Sternchen kann je nach den Einstellungen für das Textkonfigurationsobjekt unterschiedlich ausfallen.

Zulässige Syntax für den Bindestrich (-)

Der Bindestrich kann für die **Negation** eines Begriffs oder Ausdrucks verwendet werden und entspricht NOT. Ob ein Bindestrich als Negation interpretiert wird, hängt von seiner Stellung in der Abfragezeichenfolge ab. Wenn beispielsweise ein Bindestrich direkt vor einem Begriff oder Ausdruck

steht, wird er als Negation interpretiert. Wenn ein Bindestrich in einen Begriff eingebettet ist, wird er als Bindestrich interpretiert.

Vor einem als Negation verwendeten Bindestrich muss ein Leerzeichen stehen und danach muss unmittelbar ein Ausdruck folgen.

Ein Bindestrich in einer Phrase oder einem Annähernden_Ausdruck wird als Leerzeichen interpretiert und als Begriffssegmentierer verwendet.

Die nachstehende Tabelle zeigt die zulässige Syntax für einen Bindestrich:

Abfragezeichenfolge	Entspricht:	Interpretiert als:
'the -best'	'the AND NOT best', 'the AND -best', 'the & -best', 'the NOT best'	Suche nach dem Begriff "the", aber nicht nach dem Begriff "best".
'the -(very best)'	'the AND NOT (very AND best)'	Suche nach dem Begriff "the", aber nicht nach den Begriffen "very" und "best".
'the -"very best"'	'the AND NOT "very best"'	Suche nach dem Begriff "the", aber nicht nach der Phrase "very best".
'alpha- numerics'	'"alpha numerics"'	Suche nach dem Begriff "alpha", auf den unmittelbar der Begriff "numerics" folgt.
'wild - west'	'wild west', und 'wild AND west'	Suche den Begriff "wild" und den Begriff "west".

Zulässige Syntax für Sonderzeichen

Die nachstehende Tabelle zeigt die zulässige Syntax für alle Sonderzeichen mit Ausnahme von Sternchen und Bindestrich.

Diese Zeichen werden nicht als Sonderzeichen angesehen, wenn sie in einer Phrase vorkommen, und daher entfernt.

Hinweis

Dieselben Einschränkungen hinsichtlich der Angabe von Zeichenfolgenliteralen gelten auch für die Abfragezeichenfolge. So müssen beispielsweise die Apostrophe mit Escapezeichen versehen werden, usw.

Zeichen oder Syntax	Beispiele zur Verwendung und Bemerkungen
Kaufmännisches Und (&)	<p>Das kaufmännische Und entspricht AND und kann folgendermaßen angegeben werden:</p> <ul style="list-style-type: none"> • 'a & b' • 'a &b' • 'a&b' • 'a& b'
Senkrechtstrich ()	<p>Der Senkrechtstrich entspricht OR und kann folgendermaßen angegeben werden:</p> <ul style="list-style-type: none"> • 'a b' • 'a b' • 'a b' • 'a b'
Anführungszeichen (")	<p>Anführungszeichen werden verwendet, um eine Folge von Begriffen abzugrenzen, in der die Reihenfolge und der relative Abstand wichtig sind. In der Abfragezeichenfolge 'learn "full text search"' ist beispielsweise full text search eine Phrase. In diesem Beispiel kann "learn" vor oder nach der Phrase kommen oder in einer anderen Spalte vorhanden sein (wenn der Textindex in mehr als einer Spalte erstellt wurde), aber die exakte Phrase muss in einer Spalte gefunden werden.</p>
Klammern ()	<p>Klammern werden verwendet, um die Reihenfolge der Auswertung von Ausdrücken anzugeben, wenn sie von der Standardreihenfolge abweicht. Beispiel: 'a AND (b c)' wird als a und b oder c interpretiert.</p>
Tilde (~)	<p>Die Tilde ist gleichwertig mit NEAR[10]. Die Abfragezeichenfolge 'full~text' entspricht 'full NEAR text' und wird interpretiert als der Begriff "full" in einem maximal zehn Begriffe betragenden Abstand vom Begriff "text".</p> <p>Sie können mit der Tilde keine Entfernung angeben.</p>

Zeichen oder Syntax	Beispiele zur Verwendung und Bemerkungen
Eckige Klammern []	Eckige Klammern werden in Verbindung mit dem NEAR-Schlüsselwort verwendet, um <i>distance</i> anzugeben. Bei jeder anderen Verwendung von eckigen Klammern wird ein Fehler zurückgegeben.

Standards und Kompatibilität

- **SQL/2008** Das Prädikat CONTAINS ist eine Erweiterung des Herstellers.

Siehe auch

- Zulässige Syntax für Sonderzeichen auf Seite 64
- „Nachbarschaftssuche“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Ansichtensuche“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- Operatorvorrang in einer CONTAINS-Suchbedingung auf Seite 62
- „Präfixsuche“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- Zulässige Syntax für den Bindestrich (-) auf Seite 63
- Zulässige Syntax für den Sternchen-Platzhalter (*) auf Seite 63
- „Zeichenfolgenlitterale“ auf Seite 8
- „Volltextsuche“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „FROM-Klausel“ auf Seite 863
- „sa_char_terms-Systemprozedur“ auf Seite 1171
- „sa_nchar_terms-Systemprozedur“ auf Seite 1277

EXISTS-Suchbedingung

Syntax

EXISTS (*subquery*)

Bemerkungen

Die EXISTS-Suchbedingung ist TRUE, falls das Ergebnis der Unterabfrage mindestens eine Zeile enthält und FALSE, falls das Ergebnis der Unterabfrage keine Zeilen enthält. Die EXISTS-Suchbedingung kann nicht UNKNOWN sein.

Standards und Kompatibilität

- **SQL/2008** Kernfunktion.

IS NULL- und IS NOT NULL-EXISTS-Suchbedingungen

Syntax

expression IS [NOT] NULL

Bemerkungen

Ohne das Schlüsselwort NOT ist die IS NULL-Suchbedingung TRUE, falls der Ausdruck NULL ist, und sonst FALSE. Das Schlüsselwort NOT kehrt die Bedeutung der Suchbedingung um.

Standards und Kompatibilität

- **SQL/2008** Kernfunktion.

Wahrwert-Suchbedingungen

Syntax

IS [NOT] *truth-value*

Bemerkungen

Ohne das Schlüsselwort NOT ist die Suchbedingungen TRUE, wenn *condition* mit dem angegebenen *truth-value* ausgewertet wird, der entweder TRUE, FALSE oder UNKNOWN sein muss. Andernfalls ist der Wert FALSE. Das Schlüsselwort NOT kehrt die Bedeutung der Suchbedingung um, lässt UNKNOWN aber unverändert.

Standards und Kompatibilität

- **SQL/2008** Wahrwert-Suchbedingungen enthalten die optionale SQL-Sprachenfunktion F571 des SQL/2008-Standards.

Drei-Werte-Logik

Die folgenden Tabellen zeigen, wie die logischen SQL-Operatoren AND, OR, NOT und IS in der Drei-Werte-Logik arbeiten.

AND-Operator

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR-Operator

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

NOT-Operator

TRUE	FALSE	UNKNOWN
FALSE	TRUE	UNKNOWN

IS-Operator

IS	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	FALSE
UNKNOWN	FALSE	FALSE	TRUE

Standards und Kompatibilität

- **SQL/2008** Kernfunktion. Wahrwert-Tests, z.B. IS UNKNOWN, enthalten die SQL-Sprachenfunktion F571.

Siehe auch

- „NULL-Spezialwert“ [auf Seite 76](#)

Explizite Selektivitätsschätzungen

SQL Anywhere benutzt Statistiken, um die wirksamste Strategie für das Ausführen der einzelnen Anweisungen festzulegen. SQL Anywhere stellt diese Statistiken automatisch zusammen und aktualisiert sie. Diese Statistiken werden in der Datenbank in der Systemtabelle ISYSCOLSTAT permanent gespeichert. Statistiken, die während der Prozessverarbeitung einer Anweisung erstellt werden, stehen bei der Suche nach effizienten Verfahren für die Ausführung nachfolgender Anweisungen zur Verfügung.

Es kommt gelegentlich vor, dass die Statistiken ungenau werden oder dass relevante Statistiken nicht verfügbar sind. Dieser Fall tritt üblicherweise dann ein, wenn eine große Menge von Daten hinzugefügt, aktualisiert oder gelöscht wurde und dann nur wenige Abfragen ausgeführt wurden. In dieser Situation müssen Sie eventuell eine CREATE STATISTICS-Anweisung ausführen.

Wenn es Probleme mit einem bestimmten Ausführungsplan gibt, können Sie Optimierer-Hints verwenden, damit ein bestimmter Index verwendet wird.

Unter ungewöhnlichen Umständen allerdings können sich diese Maßnahmen als nicht effektiv herausstellen. In diesen Fällen können Sie manchmal die Performance steigern, indem Sie Selektivitätsschätzungen explizit angeben.

Für jede Tabelle in einem möglichen Ausführungsplan muss der Optimierer die Anzahl der Zeilen schätzen, die Teil der Ergebnismenge sein können. Wenn Sie wissen, dass eine Bedingung eine Erfolgsrate hat, die von der Schätzung des Optimierers abweicht, können Sie in der Suchbedingung eine eigene Schätzung übergeben.

Die Schätzung ist ein Prozentsatz. Sie kann eine positive Ganzzahl sein oder Bruchwerte enthalten.

Vorsicht

Versuchen Sie, wo immer möglich, explizite Schätzungen in Anweisungen zu vermeiden, die weiterhin genutzt werden. Wenn sich die Daten ändern, verliert die explizite Schätzung ihre Gültigkeit, wodurch der Optimierer möglicherweise ungeeignete Pläne auswählt. Wenn Sie explizite Selektivitätsschätzungen verwenden, achten Sie darauf, dass die Zahl genau ist. Verwenden Sie beispielsweise nicht die Werte zwischen 0% und 100%, um die Verwendung eines Indexes zu erzwingen.

Sie können Benutzerschätzungen ausschließen, indem Sie die Datenbankoption `user_estimates` auf `OFF` setzen. Der Standardwert für `user_estimates` ist "Override-Magic", was bedeutet, dass auf benutzerdefinierte Selektivitätsschätzungen nur dann zurückgegriffen wird, wenn der Optimierer einen als "MAGIC" bezeichneten Standardwert für die Selektivität einer Bedingung verwenden würde. Der Optimierer benutzt MAGIC-Werte als letzte Möglichkeit, wenn er die Selektivität eines Prädikats nicht genau vorhersagen kann.

Beispiele

In der folgenden Abfrage gibt die Schätzung an, dass ein Prozent der ShipDate-Werte später als 2001/06/30 sind:

```
SELECT ShipDate
FROM   GROUPO.SalesOrderItems
WHERE  ( ShipDate > '2001/06/30', 1 )
ORDER BY ShipDate DESC;
```

In den folgenden Abfragen wird geschätzt, dass ein halbes Prozent der Zeilen die Bedingung erfüllt:

```
SELECT *
FROM   GROUPO.Customers c, GROUPO.SalesOrders o
WHERE  (c.ID = o.CustomerID, 0.5);
```

Bruchwerte ermöglichen genauere Benutzerschätzungen für Joins und große Tabellen.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Siehe auch

- „CREATE STATISTICS-Anweisung“ auf Seite 729
- „FROM-Klausel“ auf Seite 863
- „user_estimates-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Optimiererschätzungen und -statistiken“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Spezialwerte

Spezialwerte können in Ausdrücken und als Spalten-Standardwerte bei der Erstellung von Tabellen verwendet werden.

Einige Spezialwerte können abgefragt werden und andere können nur als Standardwerte für Spalten benutzt werden. Zum Beispiel können **LAST USER**, **TIMESTAMP** und **UTC TIMESTAMP** nur als Standardwerte benutzt werden.

CURRENT DATABASE-Spezialwert

CURRENT DATABASE gibt den Namen der aktuellen Datenbank zurück.

Datentyp

Zeichenfolge

Siehe auch

- „Ausdrücke“ auf Seite 22

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

CURRENT DATE-Spezialwert

CURRENT DATE gibt das aktuelle Jahr, Monat und Tag zurück.

Datentyp

DATE

Siehe auch

- „DATE-Datentyp“ auf Seite 123
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Datentyp“ auf Seite 124
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „Ausdrücke“ auf Seite 22
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Im ANSI SQL/2008-Standard heißt das spezielle Register, in dem das aktuelle Datum definiert wird, CURRENT_DATE. SQL Anywhere unterstützt CURRENT_DATE nicht.

CURRENT PUBLISHER-Spezialwert

CURRENT PUBLISHER gibt eine Zeichenfolge zurück, die die Benutzer-ID des Publikationseigentümers der Datenbank für SQL Remote-Replikationen enthält.

Datentyp

Zeichenfolge

Bemerkungen

Der Publikationseigentümer wird mit der PUBLIC.db_publisher-Option festgelegt oder mithilfe der Anweisungen GRANT PUBLISH und REVOKE PUBLISH.

CURRENT PUBLISHER kann als Standardwert in Spalten mit Zeichendatentypen verwendet werden.

Siehe auch

- „db_publisher-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „GRANT PUBLISH-Anweisung [SQL Remote]“ auf Seite 894
- „REVOKE PUBLISH-Anweisung [SQL Remote]“ auf Seite 1007
- „Anzeigen des Publikationseigentümers“ [*SQL Remote*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

CURRENT REMOTE USER-Spezialwert

Falls die aktuelle Verbindung zur Empfangsphase von SQL Remote gehört, gibt CURRENT REMOTE USER die Benutzer ID des entfernten Benutzers zurück, der die zurzeit auf diese Verbindung

angewendeten Nachrichten erstellt hat. In allen anderen Fällen ist CURRENT REMOTE USER ein NULL-Wert.

Datentyp

Zeichenfolge

Bemerkungen

Der Spezialwert CURRENT REMOTE USER wird durch die Empfangsphase von SQL Remote eingestellt, wenn Nachrichten in der Datenbank übernommen werden. Der Spezialwert CURRENT REMOTE USER ist besonders nützlich, um in Triggern zu ermitteln, ob die übernommenen Vorgänge von der Empfangsphase von SQL Remote übernommen werden und, wenn dies der Fall ist, welcher entfernte Benutzer diese Vorgänge erstellt hat.

Siehe auch

- [Den CURRENT REMOTE USER-Spezialwert verwenden \[SQL Remote\]](#)
- [-t, Option, SQL Remote-Nachrichtenagent-Dienstprogramm \(dbremote\) \[SQL Remote\]](#)

Standards und Kompatibilität

Erweiterung des Herstellers.

CURRENT TIME-Spezialwert

CURRENT TIME gibt Stunde, Minute, Sekunde und Sekundenbruchteil der aktuellen Zeit zurück.

Datentyp

TIME

Bemerkungen

Der Sekundenbruchteil wird mit 6 Dezimalstellen gespeichert. Die Genauigkeit der aktuellen Uhrzeit ist durch die Genauigkeit der Systemuhr begrenzt.

Siehe auch

- „Ausdrücke“ auf Seite 22
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75
- „DATE-Datentyp“ auf Seite 123
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Datentyp“ auf Seite 124
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Datentyp“ auf Seite 129
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Im ANSI SQL/2008-Standard heißt das spezielle Register, in dem die aktuelle Uhrzeit definiert wird, CURRENT_TIME. SQL Anywhere unterstützt CURRENT_TIME nicht.

CURRENT TIMESTAMP-Spezialwert

CURRENT TIMESTAMP kombiniert CURRENT DATE und CURRENT TIME zu einem TIMESTAMP-Wert, der Jahr, Monat, Tag, Stunde, Minute, Sekunde und Sekundenbruchteil enthält.

Datentyp

TIMESTAMP

Bemerkungen

Der Sekundenbruchteil wird mit 6 Dezimalstellen gespeichert. Die Genauigkeit der aktuellen Uhrzeit ist durch die Genauigkeit der Systemuhr begrenzt.

Im Unterschied zu DEFAULT TIMESTAMP enthalten mit DEFAULT CURRENT TIMESTAMP deklarierte Spalten nicht notwendigerweise eindeutige Werte. Wenn Eindeutigkeit erforderlich ist, sollten Sie stattdessen DEFAULT TIMESTAMP verwenden.

Die von CURRENT TIMESTAMP zurückgegebenen Werte ähneln den von den Funktionen GETDATE und NOW zurückgegebenen Werten.

CURRENT_TIMESTAMP ist mit CURRENT TIMESTAMP äquivalent.

Hinweis

Der Hauptunterschied zwischen `DEFAULT CURRENT TIMESTAMP` und `DEFAULT TIMESTAMP` liegt darin, dass `DEFAULT CURRENT TIMESTAMP` nur bei `INSERT` gesetzt wird, während `DEFAULT TIMESTAMP` bei `INSERT` und bei `UPDATE` gesetzt wird.

Siehe auch

- „Ausdrücke“ auf Seite 22
- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75
- „DATE-Datentyp“ auf Seite 123
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Datentyp“ auf Seite 124
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Datentyp“ auf Seite 129
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Im SQL/2008-Standard heißt das spezielle Register, in dem der aktuelle Zeitstempel definiert wird, `CURRENT_TIMESTAMP`.

CURRENT USER-Spezialwert

`CURRENT USER` gibt eine Zeichenfolge zurück, die die Benutzer-ID der aktuellen Verbindung enthält.

Datentyp

Zeichenfolge

Bemerkungen

`CURRENT USER` kann als Standardwert in Spalten mit Zeichendatentypen verwendet werden.

Bei `UPDATE` werden Spalten mit `CURRENT USER`-Standardwert nicht geändert, wenn nicht explizit aktualisiert wird. Der `LAST USER`-Standardwert wird für die Protokollierung der Aktualisierungen von Benutzern verwendet.

`CURRENT_USER` ist mit `CURRENT USER` äquivalent.

Siehe auch

- „Ausdrücke“ auf Seite 22
- „LAST USER-Spezialwert“ auf Seite 76
- „USER-Spezialwert“ auf Seite 83

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Im SQL/2008-Standard heißt das spezielle Register, in dem der aktuelle Benutzer definiert wird, CURRENT_USER.

CURRENT UTC TIMESTAMP-Spezialwert

CURRENT UTC TIMESTAMP gibt die UTC (Coordinated Universal Time) mit Jahr, Monat, Tag, Stunde, Minute, Sekunde, Sekundenbruchteil und die Zeitzone.

Datentyp

TIMESTAMP WITH TIME ZONE

Bemerkungen

Mit dieser Funktion können Daten mit einer konsistenten Zeitreferenz ohne Rücksicht auf die Zeitzone am Ort der Dateneingabe eingegeben werden.

Siehe auch

- „Ausdrücke“ auf Seite 22
- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „DATE-Datentyp“ auf Seite 123
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Datentyp“ auf Seite 124
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Datentyp“ auf Seite 129
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Der TIMESTAMP WITH TIME ZONE-Datentyp ist die optionale SQL-Sprachenfunktion F411 im SQL/2008-Standard.

LAST USER-Spezialwert

LAST USER ist die Benutzer-ID des Benutzers, der die Zeile zuletzt geändert hat.

Datentyp

String

Bemerkungen

LAST USER kann als Standardwert in Spalten mit Zeichendatentypen verwendet werden.

Diese Konstante hat bei INSERT dieselbe Wirkung wie CURRENT USER.

Wenn Sie eine Spalte mit einem Standardwert LAST USER bei einer UPDATE-Anweisung nicht ausdrücklich ändern, wird sie auf den Namen des aktuellen Benutzers abgeändert.

Durch Kombinieren mit DEFAULT TIMESTAMP kann ein Standardwert von LAST USER dazu verwendet werden, sowohl den Benutzer als auch Datum und Zeit zu protokollieren (in getrennten Spalten), wenn eine Zeile zuletzt geändert wurde.

Siehe auch

- „CREATE TABLE-Anweisung“ auf Seite 737
- „CURRENT USER-Spezialwert“ auf Seite 74
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „USER-Spezialwert“ auf Seite 83

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

NULL-Spezialwert

NULL gibt einen Wert an, der unbekannt oder nicht anwendbar ist.

Syntax

NULL

Bemerkungen

NULL ist ein Spezialwert, der sich von jedem zulässigen Wert bei einem beliebigen Datentyp unterscheidet. NULL ist jedoch in jedem Datentyp ein zulässiger Wert. Mit NULL werden fehlende oder nicht anwendbare Informationen repräsentiert. Beachten Sie, dass dies zwei getrennte und unterschiedliche Fälle für NULL sind:

Situation	Beschreibung
Fehlend	Das Feld hat einen Wert, dieser Wert ist aber nicht bekannt.
Nicht anwendbar	Das Feld gilt nicht für diese spezielle Zeile.

Mit SQL können Spalten mit der Einschränkung NOT NULL erzeugt werden. Diese speziellen Spalten dürfen nicht NULL enthalten.

NULL führt das Konzept der Drei-Werte-Logik in SQL ein. Ein Vergleich von NULL mit einem Wert (einschließlich NULL) mithilfe eines Vergleichsoperators ist UNKNOWN. Die einzige Suchbedingung, die TRUE zurückgibt, ist das Prädikat IS NULL. In SQL werden nur dann Zeilen ausgewählt, wenn die Suchbedingung in der WHERE-Klausel mit TRUE ausgewertet wird. Mit UNKNOWN oder FALSE ausgewertete Zeilen werden nicht ausgewählt.

Die Spaltenraumnutzung für NULL-Werte ist 1 Bit pro Spalte und der Speicherplatz wird mit 8-Bit-Multiplen zugewiesen. Die NULL-Bit-Nutzung wird basierend auf der Anzahl der Spalten in der Tabelle fixiert, die NULL-Werte zulassen.

Mit der IS [NOT] *truth-value*-Klausel, in der der *truth-value* entweder TRUE, FALSE oder UNKNOWN ist, können Zeilen ausgewählt werden, die NULL enthalten.

In den folgenden Beispielen enthält die Salary-Spalte NULL.

Bedingung	Wahrwert	Ausgewählt?
Salary = NULL	UNKNOWN	NO
Salary <> NULL	UNKNOWN	NO
NOT (Salary = NULL)	UNKNOWN	NO
NOT (Salary <> NULL)	UNKNOWN	NO
Salary = 1000	UNKNOWN	NO
Salary IS NULL	TRUE	YES
Salary IS NOT NULL	FALSE	NO
Salary = <i>expression</i> IS UNKNOWN	TRUE	YES

Dieselben Regeln werden beim Vergleich von Spalten aus zwei verschiedenen Tabellen angewendet. Deshalb werden beim Verbinden von zwei Tabellen keine Zeilen ausgewählt, in denen eine der verglichenen Spalten den Wert NULL enthält.

NULL hat außerdem eine interessante Eigenschaft, wenn der Wert in numerischen Ausdrücken verwendet wird. Das Ergebnis jedes numerischen Ausdrucks, in dem NULL vorkommt, ist NULL. Wenn NULL zu einer Zahl hinzugefügt wird, ist das Ergebnis NULL und keine Zahl. Wenn NULL als 0 behandelt werden soll, verwenden Sie die Funktion **ISNULL(*expression*, 0)**.

Viele gebräuchliche Fehler beim Formulieren von SQL-Abfragen werden durch das Verhalten von NULL verursacht. Sie müssen diese Problembereiche sorgfältig vermeiden.

Mengenoperatoren und DISTINCT-Klausel

In SQL liefern Vergleiche mit NULL innerhalb von Suchbedingungen das Ergebnis UNKNOWN. Bei der Feststellung, ob zwei Zeilen Duplikate voneinander sind, behandelt SQL jedoch NULL als gleichbedeutend mit NULL. Diese Semantik gilt für die Mengenoperatoren (UNION, INTERSECT, EXCEPT) sowie für GROUP BY, PARTITION innerhalb einer WINDOW-Klausel und SELECT DISTINCT.

Wenn z.B. eine Spalte "Redundant" in einer Tabelle T1 für jede Zeile NULL enthält, würde die folgende Anweisung nur eine Zeile zurückgeben:

```
SELECT DISTINCT redundant FROM T1;
```

Voraussetzungen

Verbindung mit der Datenbank ist erforderlich

Nebenwirkungen

Keine.

Standards und Kompatibilität

- **SQL/2008** Kernfunktion.
- **Transact-SQL** Unter bestimmten Umständen behandelt Adaptive Server Enterprise Vergleiche mit NULL-Werten anders. Wenn ein *expression* unter Verwendung von Gleichheit und Ungleichheit mit einer Variablen oder einem NULL-Literal verglichen wird und es sich bei dem *expression* um einen einfachen Ausdruck handelt, der sich auf die Spalte einer Basistabelle oder Ansicht bezieht, wird der Vergleich mithilfe einer zweiwertigen Logik durchgeführt, wobei NULL = NULL das Ergebnis TRUE liefert statt UNKNOWN. Die folgende Liste enthält die möglichen Vergleiche mit dieser Semantik und die dazugehörigen SQL/2008-Entsprechungen:

Transact-SQL-Vergleich	SQL/2008-Entsprechung
<i>expression</i> = NULL	<i>expression</i> IS NULL
<i>expression</i> != NULL	NOT (<i>expression</i> IS NULL)
<i>expression</i> = <i>variable</i>	<i>expression</i> = <i>variable</i> IS TRUE OR (<i>expression</i> IS NULL AND <i>variable</i> IS NULL)
<i>expression</i> != <i>variable</i>	<i>expression</i> != <i>variable</i> IS TRUE AND (NOT <i>expression</i> IS NULL OR NOT <i>variable</i> IS NULL)

SQL Anywhere implementiert diese Semantik zum Angleichen des Verhaltens von Adaptive Server Enterprise, wenn die Option ansinull auf OFF gesetzt ist. Die Option ansinull ist für Open Client- und jConnect-Verbindungen standardmäßig auf OFF gesetzt. Um die SQL/2008-Semantik zu gewährleisten, können Sie entweder die Option ansinull auf ON zurücksetzen oder statt eines Gleichheitsvergleichs ein IS [NOT] NULL-Prädikat verwenden.

Eindeutige Indizes in SQL Anywhere können Zeilen mit NULL enthalten, die aber sonst identisch sind. Adaptive Server Enterprise lässt solche Einträge in eindeutigen Indizes nicht zu.

Wenn Sie jConnect verwenden, steuert die Option `tds_empty_string_is_null`, ob leere Zeichenfolgen als NULL-Zeichenfolgen zurückgegeben werden oder als Zeichenfolgen, die nur ein Leerzeichen enthalten.

Siehe auch

- „`tds_empty_string_is_null`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Ausdrücke“ auf Seite 22
- „`ansinull`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „`tds_empty_string_is_null`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Suchbedingungen“ auf Seite 42

Beispiel

Die folgende INSERT-Anweisung fügt eine NULL in die Spalte "date_returned" (Rückgabedatum) der Tabelle "Borrowed_book" (Geliehene Bücher) ein.

```
INSERT INTO Borrowed_book ( date_borrowed, date_returned, book )
VALUES ( CURRENT DATE, NULL, '1234' );
```

SQLCODE-Spezialwert

SQLCODE zeigt die Disposition der zuletzt ausgeführten SQL-Anweisung an.

Datentyp

INTEGER mit Vorzeichen

Bemerkungen

Der Datenbankserver setzt einen SQLSTATE und SQLCODE für jede von ihm durchgeführte SQL-Anweisung. SQLCODEs sind produktspezifisch (z.B. hat MobiLink seine eigenen SQLCODEs) und können verwendet werden, um zusätzliche Informationen über den SQLSTATE zu erhalten. Beispiel: Positive Werte außer 100 geben produktspezifische *warning*-Bedingungen an. Negative Werte geben produktspezifische *exception*-Bedingungen an. Der Wert 100 gibt "no data" (keine Daten) an (zum Beispiel am Ende einer von einem Cursor abgerufenen Ergebnismenge).

SQLSTATE und SQLCODE sind insofern aufeinander bezogen, als jeder SQLCODE einem SQLSTATE entspricht und jeder SQLSTATE einem oder mehreren SQLCODEs entsprechen kann.

Um eine Fehlerbedingung in Verbindung mit einem SQLCODE zurückzugeben, können Sie die Funktion `ERRORMSG` verwenden.

Hinweis

SQLSTATE ist der bevorzugte Statusindikator für das Ergebnis einer SQL-Anweisung.

Siehe auch

- „SQLSTATE-Spezialwert“ auf Seite 80
- „SQL Anywhere-Fehlermeldungen - sortiert nach SQLCODE“ [*Fehlermeldungen*]
- „Ausdrücke“ auf Seite 22
- „ERRORMSG-Funktion [Verschiedene]“ auf Seite 251

Standards und Kompatibilität

- **SQL/2008** SQLCODE wurde im ANSI SQL/1992-Standard nicht mehr empfohlen und aus SQL/1999 komplett entfernt. SQLCODE-Werte werden in SQL Anywhere weiter unterstützt, um die Abwärtskompatibilität mit Anwendungen zu gewährleisten. SQLSTATE ist der bevorzugte Statusindikator.

SQLSTATE-Spezialwert

SQLSTATE gibt an, ob die zuletzt ausgeführte SQL-Anweisung einen Erfolg, einen Fehler oder eine Warnung bewirkte.

Datentyp

String

Bemerkungen

Der Datenbankserver setzt einen SQLSTATE und SQLCODE für jede von ihm durchgeführte SQL-Anweisung. Ein SQLSTATE ist eine Zeichenfolge, die anzeigt, ob die zuletzt ausgeführte SQL-Anweisung einen Erfolg, eine Warnung oder einen Fehler bewirkte.

Jeder SQLSTATE stellt Fehler dar, die auf allen Plattformen auftreten können, und enthält in der Regel Formulierungen, die nicht produktspezifisch sind. Das Format eines SQLSTATE-Werts ist ein Klassenwert mit zwei Zeichen, dem ein Unterklassenwert mit drei Zeichen folgt. Die Richtlinien für die Konformität von SQLSTATE in Bezug auf Klassen und Unterklassenwerte sind im ISO/ANSI SQL-Standard ausgeführt.

SQL Anywhere hält sich an die ISO/ANSI SQLSTATE-Konventionen mit den folgenden Hinzufügungen und Ausnahmen:

Klasse und Unterklasse	Bedingung
01WCx	Warnungen zur Zeichensatzkonvertierung
38xxx	Externe Funktionsausnahmebedingung
42Xxx	Syntaxfehler: Ausdrücke
42Rxx	Syntaxfehler: referenzielle Integrität (z.B. Versuch der Erstellung eines zweiten Primärschlüssels)

Klasse und Unterklasse	Bedingung
42Wxx	Syntaxfehler: Generisch
42Uxx	Syntaxfehler: mehrfach vorhandene, undefinierte oder zweideutige Objektreferenz
42Zxx	Zugriffsverletzung
54Wxx	Produktgrenze überschritten
55Wxx	Objekt ist nicht im erforderlichen Zustand für einen erfolgreichen Vorgang
57xxx	Ressource nicht verfügbar oder Eingriff des Bedieners
5Rxxx	SQL Remote-Fehler
WBxxx	Onlinesicherungsfehler
WLxxx	Interne Datenbankfehler
WPxxx	Fehler in Prozeduren, Variablen, etc.
WLxxx	Fehler beim Laden bzw. Entladen
WWxxx	Diverse SQL Anywhere-spezifische Fehler-/Warnmeldungen (einschließlich Systemstörungen)
WOxxx	Funktionsbedingte Fehler beim Ferndatenzugriff
WJxxx	JCS- und JDBC-bezogene Fehler
WCxxx	Zeichenübersetzungsfehler
WXxxx	XML-bezogene Fehler
WTxxx	Textbezogene Fehler

Die Klasse für den erfolgreichen Abschluss ist '00xxx' (Beispiel: '00000').

SQLSTATE und SQLCODE sind insofern aufeinander bezogen, als jeder SQLCODE einem SQLSTATE entspricht und jeder SQLSTATE einem oder mehreren SQLCODEs entsprechen kann.

Um eine Fehlerbedingung in Verbindung mit einem SQLSTATE zurückzugeben, können Sie die Funktion ERRORMSG verwenden.

Siehe auch

- „[ERRORMSG-Funktion \[Verschiedene\]](#)“ auf Seite 251
- „[SQLCODE-Spezialwert](#)“ auf Seite 79
- „[SQL Anywhere-Fehlermeldungen - sortiert nach SQLSTATE](#)“ [[Fehlermeldungen](#)]
- „[Ausdrücke](#)“ auf Seite 22

Standards und Kompatibilität

- **SQL/2008** SQLSTATE-Klassen (die ersten beiden Zeichen), die mit den Werten '0'-'4' und 'A'-'H' beginnen, sind im ANSI-Standard definiert. Andere Klassen werden in der Implementierung definiert. Ebenso gilt: Unterklassen, die mit den Werten '0'-'4' und 'A'-'H' beginnen, sind im ANSI-Standard definiert. Unterklassenwerte außerhalb dieser Bereiche werden in der Implementierung definiert.

TIMESTAMP-Spezialwert

Der TIMESTAMP-Standardwert wird verwendet, um aufzuzeichnen, mit welchen lokalen Werten für Datum und Uhrzeit eine Zeile in einer Tabelle zuletzt geändert wurde.

Datentyp

TIMESTAMP

Bemerkungen

Der Sekundenbruchteil wird mit 6 Dezimalstellen gespeichert. Die Genauigkeit der aktuellen Uhrzeit ist durch die Genauigkeit der Systemuhr begrenzt.

Ist eine Spalte mit DEFAULT TIMESTAMP deklariert, dann wird bei Einfügungen ein Standardwert bereitgestellt, und der Wert wird bei jeder Aktualisierung der Zeile mit Datum und Uhrzeit aktualisiert.

Spalten, die mit DEFAULT TIMESTAMP deklariert wurden, enthalten eindeutige Werte. Damit können Anwendungen fast gleichzeitige Aktualisierungen derselben Zeile ermitteln. Wenn der aktuelle TIMESTAMP-Wert mit dem letzten Wert übereinstimmt, wird er durch den Wert der Option default_timestamp_increment hochgezählt.

In SQL Anywhere können Sie timestamp-Werte automatisch kürzen, indem Sie die Option default_timestamp_increment verwenden. Dies ist hilfreich, wenn Sie die Kompatibilität mit anderen Datenbankprogrammen sicherstellen möchten, die timestamp-Werte weniger genau erfassen.

Die globale Variable @@dbts gibt einen TIMESTAMP-Wert zurück, der den zuletzt für eine Spalte mit DEFAULT TIMESTAMP generierten Wert repräsentiert.

Hinweis

Der Hauptunterschied zwischen DEFAULT TIMESTAMP und DEFAULT CURRENT TIMESTAMP liegt darin, dass DEFAULT CURRENT TIMESTAMP nur bei INSERT gesetzt wird, während DEFAULT TIMESTAMP bei INSERT und bei UPDATE gesetzt wird.

Siehe auch

- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75
- „DATE-Datentyp“ auf Seite 123
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Datentyp“ auf Seite 124
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „default_timestamp_increment-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Datentyp“ auf Seite 129
- „truncate_timestamp_values-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

USER-Spezialwert

USER enthält die Benutzer-ID der aktuellen Verbindung.

Datentyp

Zeichenfolge

Bemerkungen

USER kann als Standardwert in Spalten mit Zeichendatentypen verwendet werden. Dies ist gleichwertig mit CURRENT USER.

Bei UPDATE werden Spalten mit USER-Standardwert nicht geändert, wenn nicht explizit aktualisiert wird. Der LAST USER-Standardwert wird stattdessen für die Protokollierung der Aktualisierungen von Benutzern verwendet.

Siehe auch

- „Ausdrücke“ auf Seite 22
- „CURRENT USER-Spezialwert“ auf Seite 74
- „LAST USER-Spezialwert“ auf Seite 76

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

UTC TIMESTAMP-Spezialwert

Der UTC TIMESTAMP-Standardwert wird verwendet, um aufzuzeichnen, mit welcher Coordinated Universal Time (UTC) eine Zeile in einer Tabelle zuletzt geändert wurde.

Datentyp

TIMESTAMP WITH TIME ZONE

Bemerkungen

Der Sekundenbruchteil wird mit 6 Dezimalstellen gespeichert. Die Genauigkeit der aktuellen Uhrzeit ist durch die Genauigkeit der Systemuhr begrenzt.

Ist eine Spalte mit DEFAULT UTC TIMESTAMP deklariert, dann wird bei Einfügungen ein Standardwert bereitgestellt, und der Wert wird bei jeder Aktualisierung der Zeile mit UTC Datum und Uhrzeit aktualisiert.

Spalten, die mit DEFAULT UTC TIMESTAMP deklariert wurden, enthalten eindeutige Werte. Damit können Anwendungen fast gleichzeitige Aktualisierungen derselben Zeile ermitteln. Wenn der aktuelle UTC TIMESTAMP-Wert mit dem letzten Wert übereinstimmt, wird er durch den Wert der Option `default_timestamp_increment` hochgezählt.

In SQL Anywhere können Sie UTC-Zeitstempelwerte automatisch kürzen, indem Sie die Option `default_timestamp_increment` verwenden. Dies ist hilfreich, wenn Sie die Kompatibilität mit anderen Datenbankprogrammen sicherstellen möchten, die timestamp-Werte weniger genau erfassen.

Hinweis

DEFAULT UTC TIMESTAMP wird sowohl bei INSERT als auch bei UPDATE gesetzt und DEFAULT CURRENT UTC TIMESTAMP bei INSERT.

Siehe auch

- „Ausdrücke“ auf Seite 22
- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75
- „DATE-Datentyp“ auf Seite 123
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Datentyp“ auf Seite 124
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „default_timestamp_increment-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Datentyp“ auf Seite 129
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „TIMESTAMP WITH TIME ZONE-Datentyp“ auf Seite 131
- „truncate_timestamp_values-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Variablen

SQL Anywhere unterstützt drei Stufen von Variablen:

- **Lokale Variablen** Diese werden innerhalb einer zusammengesetzten Anweisung in einer Prozedur oder einem Batch mithilfe der DECLARE-Anweisung definiert. Sie bestehen nur innerhalb der zusammengesetzten Anweisung.
- **Verbindungsebenen-Variablen** Diese werden mithilfe einer CREATE VARIABLE-Anweisung definiert. Sie gehören zur aktuellen Verbindung und verschwinden, wenn Sie die Verbindung zur Datenbank trennen oder wenn Sie die DROP VARIABLE-Anweisung verwenden.
- **Globale Variablen** Dies sind systemeigene Variablen, die systemeigene Werte haben. Alle globalen Variablen haben Namen, die mit zwei "@"-Zeichen beginnen. Die globale Variable @@version hat zum Beispiel einen Wert, der die aktuelle Versionsnummer des Datenbankservers beinhaltet. Benutzer können keine globalen Variablen definieren.

Lokale Variablen und Variablen auf Verbindungsebene werden vom Benutzer definiert und können in Prozeduren oder Batches von SQL-Anweisungen zum Festhalten von Informationen verwendet werden. Globale Variablen sind systemeigene Variablen, die systemeigene Werte liefern.

Siehe auch

- „TIMESTAMP-Datentyp“ auf Seite 129
- „CREATE VARIABLE-Anweisung“ auf Seite 771

Standards und Kompatibilität

- **SQL/2008** In gespeicherten SQL-Prozeduren deklarierte Variablen oder Funktionen mit der DECLARE-Anweisung werden im ANSI SQL/2008-Standard unterstützt (SQL-Sprachenfunktion P002, "Verarbeitungsvollständigkeit"). CREATE VARIABLE, DROP VARIABLE und globale Variablen sind Erweiterungen des Herstellers.

Lokale Variablen

Lokale Variablen werden mithilfe der DECLARE-Anweisung definiert, die nur innerhalb einer zusammengesetzten Anweisung (das heißt umgeben von den Schlüsselwörtern BEGIN und END) verwendet werden kann. Für jede DECLARE-Anweisung kann in SQL Anywhere nur eine Variable deklariert werden.

Wenn das DECLARE innerhalb einer zusammengesetzten Anweisung ausgeführt wird, ist der Bereich auf die zusammengesetzte Anweisung begrenzt.

Die Variable wird anfangs auf NULL gesetzt. Der Wert der Variablen kann mithilfe der SET-Anweisung gesetzt oder mithilfe einer SELECT-Anweisung mit einer INTO-Klausel zugewiesen werden.

Dies ist die Syntax der DECLARE-Anweisung:

```
DECLARE variable-name data-type
```

Lokale Variablen können als Argumente an Prozeduren übergeben werden, solange die Prozedur innerhalb der zusammengesetzten Anweisung aufgerufen wird.

Beispiele

Der nachstehende Batch veranschaulicht die Verwendung von lokalen Variablen:

```
BEGIN
  DECLARE local_var INT;
  SET local_var = 10;
  MESSAGE 'local_var = ', local_var TO CLIENT;
END
```

Wenn dieser Batch über Interactive SQL ausgeführt wird, erscheint die Meldung `local_var = 10` im Interactive SQL-Fensterausschnitt **Ergebnisse** auf der Registerkarte **Meldungen**.

Die Variable `local_var` besteht nicht außerhalb der zusammengesetzten Anweisung, in der sie deklariert ist. Der folgende Batch ist ungültig und ergibt einen Fehler.

```
-- This batch is invalid.
BEGIN
  DECLARE local_var INT;
  SET local_var = 10;
END;
MESSAGE 'local_var = ', local_var TO CLIENT;
```

Das folgende Beispiel zeigt die Verwendung von SELECT mit einer INTO-Klausel, um den Wert einer lokalen Variablen zu setzen:

```
BEGIN
  DECLARE local_var INT;
  SELECT 10 INTO local_var;
  MESSAGE 'local_var = ', local_var TO CLIENT;
END
```

Wenn dieser Batch über Interactive SQL ausgeführt wird, erscheint die Meldung `local_var = 10` im Interactive SQL-Fensterausschnitt **Ergebnisse** auf der Registerkarte **Meldungen**.

Standards und Kompatibilität

- **SQL/2008** Die DECLARE-Anweisung wird im ANSI SQL/2008-Standard unterstützt (SQL-Sprachenfunktion P002, "Verarbeitungsvollständigkeit").

Siehe auch

- „Spalte '%1' nicht gefunden“ [[Fehlermeldungen](#)]
- „Variable in Transact-SQL-Prozeduren“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Verbindungsebenen-Variablen

Variablen auf Verbindungsebene werden mithilfe der CREATE VARIABLE-Anweisung deklariert. Variablen auf Verbindungsebene können als Parameter an Prozeduren übergeben werden.

Dies ist die Syntax für die CREATE VARIABLE-Anweisung:

CREATE VARIABLE *variable-name data-type*

Wenn eine Variable erstellt wird, ist sie anfangs auf NULL gesetzt. Der Wert von Variablen auf Verbindungsebene kann wie der von lokalen Variablen mithilfe der SET-Anweisung oder einer SELECT-Anweisung mit einer INTO-Klausel gesetzt werden.

Variablen auf Verbindungsebene bestehen so lange, bis die Verbindung getrennt oder die Variable mithilfe der DROP VARIABLE-Anweisung explizit gelöscht wird. Die folgende Anweisung löscht die Variable `con_var`:

```
DROP VARIABLE con_var;
```

Beispiel

Der folgende Batch von SQL-Anweisungen zeigt die Verwendung von Verbindungsebenen-Variablen:

```
CREATE VARIABLE con_var INT;
SET con_var = 10;
MESSAGE 'con_var = ', con_var TO CLIENT;
```

Wenn dieser Batch über Interactive SQL ausgeführt wird, erscheint die Meldung `con_var = 10` im Interactive SQL-Fensterausschnitt **Ergebnisse** auf der Registerkarte **Meldungen**.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Globale Variablen

Globale Variablen haben vom Datenbankserver gesetzte Werte. Die globale Variable @@version hat zum Beispiel einen Wert, der die aktuelle Versionsnummer des Datenbankservers beinhaltet.

Globale Variablen werden durch zwei ihrem Namen vorangestellte @-Zeichen von lokalen Variablen und Variablen auf Verbindungsebene unterschieden. @@error und @@rowcount z.B. sind globale Variablen. Benutzer können keine globalen Variablen erstellen und die Werte von globalen Variablen nicht direkt aktualisieren.

Einige globale Variablen, wie zum Beispiel @@identity, enthalten verbindungsspezifische Informationen und haben deshalb verbindungsspezifische Werte. Andere Variablen, wie zum Beispiel @@connections, haben Werte, die für alle Verbindungen gelten.

Globale Variablen und Spezialwerte

Die Spezialwerte (zum Beispiel CURRENT DATE, CURRENT TIME, USER und SQLSTATE) sind den globalen Variablen ähnlich.

Die folgende Anweisung ruft einen Wert der globalen Variablen "version" ab.

```
SELECT @@version;
```

In Prozeduren und Triggern können globale Variablen in eine Variablenliste selektiert werden. Die folgende Prozedur gibt die Serverversionsnummer im Parameter *ver* zurück.

```
CREATE PROCEDURE VersionProc ( OUT ver VARCHAR(100) )
BEGIN
    SELECT @@version
    INTO ver;
END;
```

In Embedded SQL können globale Variablen in eine Hostvariablenliste selektiert werden.

Unterstützte globale Variablen

Variablenname	Bedeutung
@@char_convert	0 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@@client_csid	-1 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@@client_cname	NULL (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@@connections	Die Anzahl der Logins, seitdem der Server zuletzt gestartet wurde.
@@cpu_busy	0 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)

Variablenname	Bedeutung
@@dbts	Ein Wert vom Typ TIMESTAMP, der den letzten generierten Wert darstellt, der bei allen mit DEFAULT TIMESTAMP definierten Spalten verwendet wurde
@@error	<p>Ein Transact-SQL-Fehler, der den Erfolg oder Fehlschlag der zuletzt durchgeführten Anweisung prüft. Wenn die vorherige Anweisung erfolgreich war, wird 0 zurückgegeben. Wenn die vorherige Anweisung nicht erfolgreich war, wird die letzte Fehlernummer zurückgegeben, die das System erstellt hat. Weitere Hinweise und Beschreibungen der Werte, die von @@error zurückgegeben werden, finden Sie unter „Fehlerbehandlung in Transact-SQL-Prozeduren“ [SQL Anywhere Server - SQL-Benutzerhandbuch].</p> <p>Eine Anweisung wie z.B. <code>if @@error != 0 return</code> veranlasst ein Beenden, falls ein Fehler auftritt. Jede Anweisung setzt " @@error", einschließlich PRINT-Anweisungen oder IF-Tests, zurück. Deshalb muss die Statusprüfung unmittelbar auf die Anweisung folgen, deren Erfolg Sie überprüfen möchten.</p>
@@fetch_status	<p>Enthält Statusinformationen aus der letzten Abruf-Anweisung. Diese Funktion entspricht @@sqlstatus, außer dass sie unterschiedliche Werte zurückgibt. Sie dient der Kompatibilität mit Microsoft SQL Server. @@fetch_status kann die folgenden Werte enthalten:</p> <ul style="list-style-type: none"> • 0 Die Abruf-Anweisung wurde erfolgreich abgeschlossen. • -1 Die Abruf-Anweisung ergab einen Fehler. • -2 Es gibt keine weiteren Daten in der Ergebnismenge.
@@identity	Der zuletzt in eine beliebige IDENTITY- oder DEFAULT AUTOINCREMENT-Spalte durch eine INSERT oder SELECT INTO-Anweisung eingefügte Wert.
@@idle	0 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@@io_busy	0 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@@isolation	Die aktuelle Isolationsstufe der Verbindung. " @@isolation" nimmt den Wert der aktiven Stufe an.
@@langid	Gibt eine eindeutige Sprach-ID für die in der aktuellen Verbindung verwendete Sprache zurück
@@language	Name der von der Verbindung verwendeten Sprache

Variablenname	Bedeutung
@ @max_connections	Für den Personal Server ist dies die maximale Anzahl von gleichzeitigen Verbindungen, die mit dem Server hergestellt werden können. Der Wert beträgt 10. Für den Netzwerkserver ist dies die maximale Anzahl aktiver Clients (und nicht Datenbankverbindungen, da jeder Client mehrere Verbindungen unterstützen kann).
@ @maxcharlen	Maximale Länge eines Zeichens im CHAR-Zeichensatz, in Byte.
@ @ncharsize	Maximale Länge eines Zeichens im NCHAR-Zeichensatz, in Byte.
@ @nestlevel	-1 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@ @pack_received	0 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@ @pack_sent	0 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@ @packet_errors	0 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@ @procid	Die Kennung der gerade ausgeführten gespeicherten Prozedur.
@ @rowcount	<p>Die Anzahl der von der letzten Anweisung betroffenen Zeilen. Der Wert von "@ @rowcount" sollte unmittelbar nach jeder Anweisung überprüft werden.</p> <p>Die INSERT-, UPDATE- und DELETE-Anweisungen setzen "@ @rowcount" auf die Anzahl der betroffenen Zeilen.</p> <p>Bei Cursor repräsentiert "@ @rowcount" die Gesamtanzahl der von der Cursorergebnismenge an den Client zurückgegebenen Zeilen bis zur letzten Abrufanforderung.</p> <p>"@ @rowcount" wird nicht von Anweisungen auf "0" zurückgesetzt, die sich nicht auf Zeilen beziehen, wie z.B. eine IF-Anweisung.</p>
@ @servername	Der Name des aktuellen Datenbankservers
@ @spid	Der Verbindungs-Handle der wartenden Verbindung. Dies ist derselbe Wert, der auch von der Prozedur "sa_conn_info" angezeigt wird.
@ @sqlstatus	<p>Enthält Statusinformationen aus der letzten Abruf-Anweisung. @ @sqlstatus kann die folgenden Werte enthalten:</p> <ul style="list-style-type: none"> • 0 Die Abruf-Anweisung wurde erfolgreich abgeschlossen. • 1 Die Abruf-Anweisung ergab einen Fehler. • 2 Es gibt keine weiteren Daten in der Ergebnismenge.

Variablenname	Bedeutung
@ @textsize	Der aktuelle Wert der SET TEXTSIZE-Option, der die maximale Länge von Text- oder Bilddaten in Byte angibt, die mit einer SELECT-Anweisung zurückgegeben werden. Die Standardeinstellung ist 32765, die längste Byte-Zeichenfolge, die mithilfe von READTEXT zurückgegeben werden kann. Der Wert kann mithilfe der SET-Anweisung gesetzt werden.
@ @thresh_hysteresis	0 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@ @timeticks	0 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@ @total_errors	0 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@ @total_read	0 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@ @total_write	0 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@ @tranchained	Aktueller Transaktionsmodus: 0 für unverkettet oder 1 für verkettet.
@ @trancount	Die Verschachtelungsebene der Transaktionen. Jedes BEGIN TRANSACTION in einem Batch zählt die Transaktionsanzahl hoch.
@ @transtate	-1 (aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt)
@ @version	Die Versionsnummer der aktuellen Version von Adaptive Server Anywhere

Siehe auch

- „Globale Variable @ @identity“ auf Seite 91

Globale Variable @ @identity

Die @ @identity-Variable enthält den aktuellsten Wert, der von der aktuellen Verbindung in eine IDENTITY-, DEFAULT AUTOINCREMENT- oder DEFAULT GLOBAL AUTOINCREMENT-Spalte eingefügt wurde, oder Null, wenn die letzte Einfügung keine solche Spalte hatte.

Der Wert von @ @identity ist abhängig von der Verbindung. Wenn eine Anweisung mehrere Zeilen einfügt, gibt @ @identity den IDENTITY-Wert für die zuletzt eingefügte Zeile wieder. Wenn die betroffene Tabelle keine IDENTITY-Spalte enthält, wird @ @identity auf Null gesetzt.

Der Wert von @ @identity wird vom Fehlschlag einer INSERT- oder SELECT INTO-Anweisung oder dem Zurücksetzen der Transaktion, die ihn enthalten hat, nicht betroffen. @ @identity behält den zuletzt in eine IDENTITY-Spalte eingefügten Wert, selbst wenn das Festschreiben der Anweisung, die ihn eingefügt hat, fehlschlägt.

@@identity und Trigger

Wenn ein Einfügen referenzielle Integritätsaktionen bewirkt oder einen Trigger auslöst, verhält sich @@identity wie ein Stack. Wenn zum Beispiel ein Einfügevorgang in eine Tabelle T1 (mit einer IDENTITY- oder AUTOINCREMENT-Spalte) einen Trigger auslöst, der eine Zeile in Tabelle T2 (ebenfalls mit einer IDENTITY- oder AUTOINCREMENT-Spalte) einfügt, dann ist der Wert, der an die Anwendung oder Prozedur zurückgegeben wird, derjenige, der in T1 eingefügt wurde. Im Trigger hat @@identity den T1-Wert vor dem Einfügen in T2, und danach den T2-Wert. Der Trigger kann die Werte in lokale Variablen kopieren, falls er auf beide zugreifen muss.

Standards und Kompatibilität

- **SQL/2008** Globale Variablen sind eine Erweiterung des Herstellers.

Kommentare

Mit Kommentaren wird erklärender Text zu SQL-Anweisungen oder Anweisungsblöcken hinzugefügt. Der Datenbankserver führt Kommentare nicht aus.

Die folgenden Kommentarindikatoren werden in SQL Anywhere unterstützt:

- **-- (Doppel-Bindestrich)** Der Datenbankserver ignoriert alle restlichen Zeichen in der Zeile. Dies ist der Kommentarindikator von SQL/2008.

Sie können diesen Kommentarindikator in Interactive SQL oder auf der Registerkarte **SQL** des Fensters **Prozeduren und Funktionen** von Sybase Central hinzufügen und entfernen, indem Sie Text auswählen und Strg+Minuszeichen drücken.

Der SQL-Kommentarindikator wird am Anfang jeder Zeile des ausgewählten Texts eingefügt. Wenn kein Text ausgewählt wurde, wird der Kommentarindikator am Anfang der aktuellen Zeile eingefügt.

- **// (Doppel-Schrägstrich)** Der Doppel-Schrägstrich hat dieselbe Bedeutung wie der Doppel-Bindestrich.

Sie können diesen Kommentarindikator in Interactive SQL oder auf der Registerkarte **SQL** des Fensters **Prozeduren und Funktionen** von Sybase Central hinzufügen und entfernen, indem Sie Text auswählen und Strg+Schrägstrich drücken.

Der SQL-Kommentarindikator wird am Anfang jeder Zeile des ausgewählten Texts eingefügt. Wenn kein Text ausgewählt wurde, wird der Kommentarindikator am Anfang der aktuellen Zeile eingefügt.

- **/* ... */ (Schrägstrich-Stern)** Alle Zeichen zwischen den beiden Kommentarmarkierungen werden ignoriert. Die beiden Kommentarmarkierungen können sich in derselben oder in verschiedenen Zeilen befinden. Kommentare mit Indikatoren in diesem Stil können verschachtelt sein, aber verschachtelte Kommentare müssen ausgeglichen sein. Kommentare im Kommentarblock dürfen nicht einzelne Instanzen der Kommentaranfangsmarkierung enthalten. Dieser Kommentarstil wird auch **C-Stil-Kommentar** genannt.

Beispiele

Das folgende Beispiel zeigt die Verwendung von Doppel-Bindestrich-Kommentaren:

```

CREATE FUNCTION fullname ( firstname CHAR(30),
                           lastname CHAR(30))
RETURNS CHAR(61)
-- fullname concatenates the firstname and lastname
-- arguments with a single space between.
BEGIN
    DECLARE name CHAR(61);
    SET name = firstname || ' ' || lastname;
    RETURN ( name );
END;

```

Das folgende Beispiel zeigt die Verwendung von Kommentaren im C-Stil:

```

/* Lists the names and employee IDs of employees
   who work in the sales department. */
CREATE VIEW SalesEmployees AS
    SELECT EmployeeID, Surname, GivenName
    FROM GROUPO.Employees
    WHERE DepartmentID = 200;

```

Standards und Kompatibilität

- **SQL/2008** Die Verwendung von doppelten Minuszeichen für einen Kommentar ist eine Kernfunktion des ANSI SQL/2008-Standards. Die Verwendung von C-artigen, in Klammern eingeschlossenen Kommentaren (*/* ... */*) ist SQL-Sprachenfunktion T351 des SQL/2008-Standards. Kommentare mit Doppel-Schrägstrich (*//*) werden als Erweiterung des Herstellers unterstützt.

Siehe auch

- „Tastenkürzel für Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]

Benannte Parameter

SQL Anywhere-Funktionen und -Prozeduren, die aus der CALL-Anweisung, der EXECUTE-Anweisung (Transact-SQL) oder der FROM-Klausel einer DML-Anweisung referenziert werden, unterstützen positionsbasierte Parameter und benannte Parameter. Mit benannten Parametern können Sie beliebige Teilmengen der verfügbaren Parameter in beliebiger Reihenfolge angeben. Sie können benannte Parameter mit systemdefinierten Prozeduren und Funktionen sowie mit benutzerdefinierten Prozeduren verwenden. Benannte Parameter können nicht mit benutzerdefinierten Funktionen verwendet werden.

Die folgenden Syntaxvarianten für benannte Parameter werden unterstützt:

- =
- =>

Beispiel

Im folgenden Beispiel wird = verwendet, um einen benannten Parameter anzugeben:

```
CALL sa_conn_properties( connidparm = 1 );
```

Im folgenden Beispiel werden beim Aufrufen der Systemprozedur benannte Parameter verwendet. Zurückgegeben werden Informationen zu den entfernten Tabellen mit Fremdschlüsseln in der SYSOBJECTS-Tabelle, die sich in der Produktionsdatenbank auf einem Datenbankserver namens satest befindet:

```
CALL sp_remote_exported_keys(  
    @server_name=>'satest',  
    @sp_name=>'sysobjects',  
    @sp_qualifier=>'production' );
```

Standards und Kompatibilität

- **SQL/2008** = und => sind Erweiterungen des Herstellers.

Siehe auch

- „CALL-Anweisung“ auf Seite 564
- „EXECUTE IMMEDIATE-Anweisung [SP]“ auf Seite 843
- „EXECUTE-Anweisung [ESQL]“ auf Seite 846
- „EXECUTE-Anweisung [T-SQL]“ auf Seite 848
- „FROM-Klausel“ auf Seite 863
- „Die externe CLR-Umgebung“ [*SQL Anywhere Server - Programmierung*]

SQL-Datentypen

Zeichendatentypen

Zeichendatentypen werden zum Speichern von Zeichenfolgen verwendet, die aus Buchstaben, Ziffern und anderen Symbolen bestehen.

SQL Anywhere stellt zwei Klassen von Zeichendatentypen sowie Domänen zur Verfügung, die auf Basis dieser Typen definiert wurden.

- **CHAR, VARCHAR, LONG VARCHAR** Zeichendaten, die in einem Einbyte- oder Mehrbyte-Zeichensatz gespeichert sind und aufgrund ihrer Entsprechung zur in der Datenbank gespeicherten Primärsprache bzw. Sprachen ausgewählt werden.
- **NCHAR, NVARCHAR, LONG NVARCHAR** Zeichendaten, die in der Unicode UTF-8-Kodierung gespeichert sind. Alle Unicode-Codepunkte können unter Verwendung dieser Typen gespeichert werden, unabhängig von der in der Datenbank gespeicherten Primärsprache bzw. Primärsprachen.
- **TEXT, UNIQUEIDENTIFIER, XML** Domänen, die auf anderen Zeichendatentypen basieren

Speicherung

Alle Zeichendatenwerte werden auf die gleiche Art gespeichert. Standardmäßig werden Werte bis zu 128 Byte in einem Stück gespeichert. Werte, die länger als 128 Byte sind, werden mit einem 4-Byte-Präfix gespeichert, das lokal auf der Datenbankseite aufbewahrt wird, während der vollständige Wert auf einer oder mehreren anderen Datenbankseiten gespeichert wird. Diese Standardgrößen werden von den **INLINE-** und **PREFIX-**Klauseln der **CREATE TABLE**-Anweisung gesteuert.

Siehe auch

- „[CREATE TABLE-Anweisung](#)“ auf Seite 737
- „[string_truncation-Option](#)“ [*SQL Anywhere Server - Datenbankadministration*]

CHAR-Datentyp

Der CHAR-Datentyp speichert Zeichen von bis zu 32.767 Byte.

Syntax

CHAR [(*max-length* [**CHAR** | **CHARACTER**])]

Parameter

- **max-length** Die Maximallänge der Zeichenfolge. Wenn Bytelänge-Semantik verwendet wird (CHAR oder CHARACTER wird nicht als Teil der Länge angegeben), ist die Länge in Bytes, und sie muss im Bereich von 1 bis 32.767 liegen. Wenn keine Länge angegeben ist, ist ihr Wert "1".

Wenn Zeichenlängensemantik verwendet wird (CHAR oder CHARACTER als Teil der Länge angegeben wird), ist die Länge in Zeichen und Sie müssen *max-length* angeben. *max-length* kann bis zu 32767 Zeichen betragen.

Bemerkungen

Mehrbyte-Zeichen können als CHAR gespeichert werden, aber die deklarierte Länge bezieht sich auf Byte und nicht auf Zeichen, es sei denn es wird Zeichenlängensemantik verwendet.

CHAR kann auch als CHARACTER angegeben werden. Unabhängig von der verwendeten Syntax wird der Datentyp als CHAR beschrieben.

CHAR entspricht semantisch VARCHAR, auch wenn es sich um unterschiedliche Typen handelt. In SQL Anywhere ist CHAR ein Typ mit variabler Länge. In anderen relationalen Datenbankmanagementsystemen ist CHAR ein Typ mit fester Länge und Daten werden mit Leerzeichen auf *max-length* in Byte aufgefüllt. SQL Anywhere füllt gespeicherte Zeichendaten nicht mit Leerzeichen auf.

Wie CHAR-Spalten beschrieben werden, hängt ab von der verwendeten Client-Schnittstelle, von den verwendeten Zeichensätzen und davon, ob Zeichenlängensemantik verwendet wird. Zum Beispiel entspricht in Embedded SQL die beschriebene Länge der maximalen Byte-Anzahl im Clientzeichensatz. Wenn die beschriebene Länge größer wäre als 32767 Byte, würde die Spalte als Typ DT_LONGVARCHAR beschrieben. Die folgende Tabelle enthält einige Beispiele für Embedded SQL und die Ergebnisse, die bei einer DESCRIBE-Anweisung zurückgegeben werden:

Beschriebener Typ	Zeichensatz der Datenbank	Clientzeichensatz	Ergebnis der DESCRIBE-Anweisung
CHAR(10)	Windows-1252	Windows-1252	DT_FIXCHAR der Länge 10
CHAR(10)	UTF-8	UTF-8	DT_FIXCHAR der Länge 10
CHAR(10)	Windows-1252	UTF-8	DT_FIXCHAR der Länge 30
CHAR(20000)	Windows-31J	UTF-8	DT_LONGVARCHAR
CHAR(10 CHAR)	Windows-1252	Windows-1252	DT_FIXCHAR der Länge 10
CHAR(10 CHAR)	UTF-8	UTF-8	DT_FIXCHAR der Länge 40

Bei ODBC wird CHAR entweder als SQL_CHAR oder als SQL_VARCHAR beschrieben, abhängig von der Einstellung der Option `odbc_distinguish_char_and_varchar`.

Siehe auch

- „`odbc_distinguish_char_and_varchar`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „VARCHAR-Datentyp“ auf Seite 101
- „LONG VARCHAR-Datentyp“ auf Seite 97
- „NCHAR-Datentyp“ auf Seite 98

Standards und Kompatibilität

- **SQL/2008** Kompatibel mit SQL/2008. Im Standard ist die Zeichenlängensemantik die Standardeinstellung, während in SQL Anywhere die Bytelänge-Semantik die Standardeinstellung ist. Es gibt geringfügige Inkonsistenzen mit dem SQL-Standard aufgrund der Kollationsunterstützung ohne Berücksichtigung von Groß- und Kleinschreibung und der Unterstützung von SQL Anywhere für das Auffüllen mit Leerzeichen.

Der SQL/2008-Standard unterstützt explizite Zeichenlänge- oder Bytelänge-Semantik als SQL-Sprachenfunktion T061.

LONG NVARCHAR-Datentyp

Der LONG NVARCHAR-Datentyp speichert Unicode-Zeichendaten von beliebiger Länge.

Syntax

LONG NVARCHAR

Bemerkungen

Die maximal zulässige Länge beträgt 2 GB minus 1 Byte ($2^{31} - 1$).

Zeichen werden in UTF-8 gespeichert. Jedes Zeichen benötigt ein bis vier Byte. Die maximale Anzahl von Zeichen, die als LONG NVARCHAR gespeichert werden können, beträgt über 500 Millionen bzw., abhängig von der Länge der gespeicherten Zeichen, über 2 Milliarden.

Wenn ein Embedded SQL-Client eine DESCRIBE-Anweisung für eine LONG NVARCHAR-Spalte ausführt, ist der zurückgegebene Datentyp entweder DT_LONGVARCHAR oder DT_LONGNVARCHAR, abhängig davon, ob die Funktion db_change_nchar_charset aufgerufen wurde.

Bei ODBC wird ein LONG NVARCHAR-Ausdruck als SQL_WLONGVARCHAR beschrieben.

Siehe auch

- „db_change_nchar_charset-Funktion“ [[SQL Anywhere Server - Programmierung](#)]
- „NCHAR-Datentyp“ auf Seite 98
- „NVARCHAR-Datentyp“ auf Seite 99
- „LONG VARCHAR-Datentyp“ auf Seite 97

Standards und Kompatibilität

- **SQL/2008** Die Verwendung von LONG NVARCHAR zum Deklarieren eines nationalen Zeichensatzes ist eine Erweiterung des Herstellers.

LONG VARCHAR-Datentyp

Der LONG VARCHAR-Datentyp speichert Zeichendaten von beliebiger Länge.

Syntax

LONG VARCHAR

Bemerkungen

Die maximale Größe beträgt 2 GB minus 1 Byte ($2^{31} - 1$).

Mehrbyte-Zeichensätze können als LONG VARCHAR gespeichert werden, aber die Länge ist in Byte, nicht in Zeichen.

Siehe auch

- „CHAR-Datentyp“ auf Seite 95
- „VARCHAR-Datentyp“ auf Seite 101
- „LONG NVARCHAR-Datentyp“ auf Seite 97

Standards und Kompatibilität

- **SQL/2008** Die Unterstützung von großen Objekten ist SQL-Sprachenfunktion T041 des SQL/2008-Standards.

NCHAR-Datentyp

Der NCHAR-Datentyp speichert Unicode-Zeichendaten, und zwar bis zu 32767 Zeichen.

Syntax

NCHAR [(*max-length*)]

Parameter

- **max-length** Die maximale Länge der Zeichenfolge in Zeichen. Die Länge muss im Bereich von 1 bis 32.767 liegen. Wenn keine Länge angegeben wird, ist ihr Wert "1".

Bemerkungen

Zeichen werden unter Verwendung der UTF-8-Kodierung gespeichert. Die maximale zum Speichern erforderliche Byte-Anzahl beträgt vier mal *max-length*. Die tatsächliche Anzahl der zum Speichern benötigten Byte ist jedoch normalerweise viel kleiner.

NCHAR kann auch als NATIONAL CHAR oder NATIONAL CHARACTER angegeben werden. Unabhängig von der verwendeten Syntax wird der Datentyp als NCHAR beschrieben.

Wenn ein Embedded SQL-Client eine DESCRIBE-Anweisung für eine NCHAR-Spalte ausführt, ist der zurückgegebene Datentyp entweder DT_FIXCHAR oder DT_NFIXCHAR, abhängig davon, ob die Funktion `db_change_nchar_charset` aufgerufen wurde.

Weiterhin gilt: Wenn ein Embedded SQL-Client eine DESCRIBE-Anweisung für eine NCHAR-Spalte ausführt, ist die zurückgegebene Länge die maximale Bytelänge im NCHAR-Zeichensatz des Clients. Beispiel: Bei einem Embedded SQL-Client, der den westeuropäischen Zeichensatz cp1252 als den NCHAR-Zeichensatz verwendet, wird eine NCHAR(10)-Spalte als Typ DT_NFIXCHAR mit der Länge 10 (10 Zeichen multipliziert mit einem Maximum von einem Byte pro Zeichen) beschrieben. Bei einem Embedded SQL-Client, der den japanischen Zeichensatz cp932 verwendet, wird dieselbe Spalte als Typ DT_NFIXCHAR der Länge 20 (10 Zeichen multipliziert mit einem Maximum von zwei Bytes pro Zeichen) beschrieben. Wenn die beschriebene Länge mehr als 32767 Byte zurückgeben würde, würde die Spalte als Typ DT_LONGNVARCHAR beschrieben.

NCHAR entspricht semantisch NVARCHAR, auch wenn es sich um unterschiedliche Typen handelt. In SQL Anywhere ist NCHAR ein Typ mit variabler Länge. In anderen relationalen Datenbankmanagementsystemen ist NCHAR ein Typ mit fester Länge und Daten werden mit Leerzeichen auf *max-length* in Zeichen aufgefüllt. SQL Anywhere füllt gespeicherte Zeichendaten nicht mit Leerzeichen auf.

Bei ODBC wird NCHAR als SQL_WCHAR beschrieben.

Siehe auch

- „db_change_nchar_charset-Funktion“ [*SQL Anywhere Server - Programmierung*]
- „CHAR-Datentyp“ auf Seite 95
- „NVARCHAR-Datentyp“ auf Seite 99
- „LONG NVARCHAR-Datentyp“ auf Seite 97

Standards und Kompatibilität

- **SQL/2008** Die Unterstützung von nationalen Zeichen ist Funktion F421 des SQL/2008-Standards.

NTEXT-Datentyp

Der NTEXT-Datentyp speichert Unicode-Zeichendaten von beliebiger Länge.

Syntax

NTEXT

Bemerkungen

NTEXT ist eine Domäne, die als LONG NVARCHAR implementiert ist.

Siehe auch

- „LONG NVARCHAR-Datentyp“ auf Seite 97
- „TEXT-Datentyp“ auf Seite 100

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

NVARCHAR-Datentyp

Der NVARCHAR-Datentyp speichert Unicode-Zeichendaten, und zwar bis zu 32767 Zeichen.

Syntax

NVARCHAR [(*max-length*)]

Parameter

- **max-length** Die maximale Länge der Zeichenfolge in Zeichen. Die Länge muss im Bereich von 1 bis 32.767 liegen. Wenn keine Länge angegeben wird, ist ihr Wert "1".

Bemerkungen

Zeichen werden in UTF-8-Kodierung gespeichert. Die maximale zum Speichern erforderliche Byte-Anzahl beträgt vier mal *max-length*, auch wenn der tatsächlich benötigte Speicherplatz normalerweise viel kleiner ist.

NVARCHAR kann auch als NCHAR VARYING, NATIONAL CHAR VARYING oder NATIONAL CHARACTER VARYING angegeben werden. Unabhängig von der verwendeten Syntax wird der Datentyp als NVARCHAR beschrieben.

Wenn ein Embedded SQL-Client eine DESCRIBE-Anweisung für eine NVARCHAR-Spalte ausführt, ist der zurückgegebene Datentyp entweder DT_VARCHAR oder DT_NVARCHAR, abhängig davon, ob die Funktion db_change_nchar_charset aufgerufen wurde.

Weiterhin gilt: Wenn ein Embedded SQL-Client eine DESCRIB-Anweisung für eine NVARCHAR-Spalte ausführt, ist die zurückgegebene Länge die maximale Bytelänge im NCHAR-Zeichensatz des Clients. Beispiel: Bei einem Embedded SQL-Client, der den westeuropäischen Zeichensatz cp1252 als den NCHAR-Zeichensatz verwendet, wird eine NVARCHAR(10)-Spalte als Typ DT_NVARCHAR mit der Länge 10 (10 Zeichen multipliziert mit einem Maximum von einem Byte pro Zeichen) beschrieben. Bei einem Embedded SQL-Client, der den japanischen Zeichensatz cp932 verwendet, wird dieselbe Spalte als Typ DT_NVARCHAR der Länge 20 (10 Zeichen multipliziert mit einem Maximum von zwei Bytes pro Zeichen) beschrieben. Wenn die beschriebene Länge mehr als 32767 Byte zurückgeben würde, würde die Spalte als Typ DT_LONGNVARCHAR beschrieben.

Bei ODBC wird NVARCHAR als SQL_WVARCHAR beschrieben.

Siehe auch

- „db_change_nchar_charset-Funktion“ [[SQL Anywhere Server - Programmierung](#)]
- „NCHAR-Datentyp“ auf Seite 98
- „LONG NVARCHAR-Datentyp“ auf Seite 97
- „VARCHAR-Datentyp“ auf Seite 101

Standards und Kompatibilität

- **SQL/2008** Die Unterstützung von nationalen Zeichen ist SQL-Sprachenfunktion F421 des SQL/2008-Standards.

TEXT-Datentyp

Der TEXT-Datentyp speichert Zeichendaten von beliebiger Länge.

Syntax

TEXT

Bemerkungen

TEXT ist eine Domäne, die als LONG VARCHAR implementiert ist.

Siehe auch

- „LONG VARCHAR-Datentyp“ auf Seite 97
- „NTEXT-Datentyp“ auf Seite 99

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

UNIQUEIDENTIFIERSTR-Datentyp

UNIQUEIDENTIFIERSTR ist eine als CHAR(36) implementierte Domäne.

Syntax

UNIQUEIDENTIFIERSTR

Bemerkungen

Wird beim entfernten Datenzugriff verwendet, wenn uniqueidentifier-Spalten von Microsoft SQL Server zugeordnet werden

Siehe auch

- Datentypkonvertierungen: Microsoft SQL Server [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „STRTOUUID-Funktion [Zeichenfolge]“ auf Seite 399

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

VARCHAR-Datentyp

Der VARCHAR-Datentyp speichert Zeichen von bis zu 32.767 Byte.

Syntax

VARCHAR [(*max-length* [CHAR | CHARACTER])]

Parameter

- **max-length** Die Maximallänge der Zeichenfolge. Dieser Standardwert ist 1.

Wenn Bytelänge-Semantik verwendet wird (CHAR oder CHARACTER *nicht* als Teil der Länge angegeben werden), wird die Länge in Byte angegeben und muss im Bereich von 1 bis 32767 liegen.

Wenn Zeichenlängensemantik verwendet wird (CHAR oder CHARACTER als Teil der Länge angegeben wird), ist die Länge in Zeichen und Sie müssen *max-length* angeben. *max-length* kann bis zu 32767 Zeichen betragen.

Bemerkungen

Mehrbyte-Zeichensätze können als VARCHAR gespeichert werden, aber die angegebene Länge bezieht sich auf Bytes, nicht auf Zeichen.

VARCHAR kann auch als CHAR VARYING oder CHARACTER VARYING angegeben werden. Unabhängig von der verwendeten Syntax wird der Datentyp als VARCHAR beschrieben.

VARCHAR entspricht semantisch NVARCHAR, auch wenn es sich um unterschiedliche Typen handelt. In SQL Anywhere ist VARCHAR ein Typ mit variabler Länge. In anderen relationalen Datenbankmanagementsystemen ist CHAR ein Typ mit fester Länge und Daten werden mit Leerzeichen auf *max-length* in Byte aufgefüllt. SQL Anywhere füllt gespeicherte Zeichendaten nicht mit Leerzeichen auf.

Wie VARCHAR-Spalten beschrieben werden, hängt ab von der verwendeten Client-Schnittstelle, von den verwendeten Zeichensätzen und davon, ob Zeichenlängensemantik verwendet wird. Zum Beispiel entspricht in Embedded SQL die beschriebene Länge der maximalen Byte-Anzahl im Clientzeichensatz. Wenn die beschriebene Länge größer wäre als 32767 Byte, würde die Spalte als Typ DT_LONGVARCHAR beschrieben. Die folgende Tabelle enthält einige Beispiele für Embedded SQL und die Ergebnisse, die bei einer DESCRIBE-Anweisung zurückgegeben werden:

Beschriebener Typ	Zeichensatz der Datenbank	Clientzeichensatz	Ergebnis der DESCRIBE-Anweisung
VARCHAR(10)	Windows-1252	Windows-1252	DT_VARCHAR der Länge 10
VARCHAR(10)	UTF-8	UTF-8	DT_VARCHAR der Länge 10
VARCHAR(10)	Windows-1252	UTF-8	DT_VARCHAR der Länge 30
VARCHAR(20000)	Windows-31J	UTF-8	DT_LONGVARCHAR
VARCHAR(10 CHAR)	Windows-1252	Windows-1252	DT_VARCHAR der Länge 10
VARCHAR(10 CHAR)	UTF-8	UTF-8	DT_VARCHAR der Länge 40

Bei ODBC wird VARCHAR als SQL_VARCHAR beschrieben.

Siehe auch

- [„CHAR-Datentyp“ auf Seite 95](#)
- [„LONG VARCHAR-Datentyp“ auf Seite 97](#)
- [„NVARCHAR-Datentyp“ auf Seite 99](#)

Standards und Kompatibilität

- **SQL/2008** Kompatibel mit SQL/2008. Im Standard ist die Zeichenlängensemantik die Standardeinstellung, während in SQL Anywhere die Bytelänge-Semantik die Standardeinstellung ist. Es gibt geringfügige Inkonsistenzen mit dem SQL-Standard aufgrund der Kollationsunterstützung ohne Berücksichtigung von Groß- und Kleinschreibung und der Unterstützung von SQL Anywhere für das Auffüllen mit Leerzeichen.

Der SQL/2008-Standard unterstützt explizite Zeichenlänge- oder Bytelänge-Semantik als SQL-Sprachenfunktion T061.

XML-Datentyp

Der XML-Datentyp speichert Zeichendaten von beliebiger Länge und wird zum Speichern von XML-Dokumenten verwendet.

Syntax

XML

Bemerkungen

Die maximal zulässige Länge beträgt 2 GB minus 1 Byte ($2^{31} - 1$).

Daten vom Typ XML werden nicht in Anführungszeichen gesetzt, wenn Elementinhalte aus relationalen Daten erzeugt werden.

Sie können zwischen dem XML-Datentyp und jedem anderen Datentyp, der in oder aus einer Zeichenfolge umgewandelt werden kann, eine Umwandlung durchführen. Beim Umwandeln in XML wird nicht überprüft, ob die Zeichenfolge wohlgeformt ist.

Wenn eine Embedded SQL-Clientanwendung eine DESCRIBE-Anweisung für eine XML-Spalte ausführt, wird sie als LONG VARCHAR beschrieben.

Siehe auch

- „XML in der Datenbank“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Speichern von XML-Dokumenten in relationalen Datenbanken“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Der XML-Datentyp ist SQL-Sprachenfunktion X010 des SQL/2008-Standards.

Numerische Datentypen

Numerische Datentypen speichern numerische Daten.

Die Datentypen NUMERIC und DECIMAL sowie die verschiedenen INTEGER-Datentypen werden manchmal als **exakte** numerische Datentypen bezeichnet, im Gegensatz zu den **angenäherten** numerischen Datentypen FLOAT, DOUBLE und REAL.

Die numerisch exakten Datentypen sind jene, für die Gesamtstellen- und Dezimalstellenwerte angegeben werden können, während angenäherte numerische Datentypen in einer vordefinierten Weise gespeichert werden. *Nur bei genauen numerischen Daten ist nach einem arithmetischen Vorgang Genauigkeit bis zur festgelegten niederstwertigen Stelle garantiert.*

Datentypängen und Gesamtstellen mit weniger als "Eins" sind nicht zulässig.

Kompatibilität

Nur der NUMERIC-Datentyp mit Dezimalstellenzahl = 0 kann für die Transact-SQL Identity-Spalte verwendet werden.

Sie sollten Standardeinstellungen für Gesamtstellenzahl und Dezimalstellen bei den Datentypen NUMERIC und DECIMAL nur mit Vorsicht verwenden, da diese Einstellungen in anderen Datenbanklösungen abweichen können. Die Standard-Gesamtstellenzahl beträgt 30 und die Standard-Dezimalstellenzahl 6.

Sie sollten Standard-Gesamtstellenzahl- und Dezimalstellenzahl-Einstellungen für die NUMERIC- und DECIMAL-Datentypen vermeiden, da diese in SQL Anywhere und in Adaptive Server Enterprise unterschiedlich sind. In SQL Anywhere ist die Standard-Gesamtstellenzahl 30 und die Standard-Dezimalstellenzahl 6. In Adaptive Server Enterprise ist die Standard-Gesamtstellenzahl 18 und die Standard-Dezimalstellenzahl 0.

Der FLOAT (*p*)-Datentyp ist ein Synonym für REAL oder DOUBLE, abhängig vom Wert *p*. Für SQL Anywhere ist der Kürzungspunkt plattformabhängig, aber der Kürzungswert ist auf allen Plattformen jedenfalls größer als 15.

Hinweise zum Ändern der Standardwerte durch das Einstellen von Datenbankoptionen finden Sie unter „precision-Option“ [[SQL Anywhere Server - Datenbankadministration](#)] und „scale-Option“ [[SQL Anywhere Server - Datenbankadministration](#)].

BIGINT-Datentyp

Der BIGINT-Datentyp speichert BIGINT-Werte, d.h. Ganzzahlen, die 8 Byte Speicherplatz erfordern.

Syntax

[UNSIGNED] BIGINT

Bemerkungen

Der BIGINT-Datentyp ist ein numerisch exakter Datentyp. Seine Genauigkeit wird durch arithmetische Vorgänge nicht berührt.

Ein BIGINT-Wert erfordert 8 Byte Speicherplatz.

Der Bereich für BIGINT-Werte ist -2^{63} bis $2^{63} - 1$ oder -9223372036854775808 bis 9223372036854775807.

Der Bereich für UNSIGNED BIGINT-Werte ist 0 bis $2^{64} - 1$ oder 0 bis 18446744073709551615.

Standardmäßig ist der Datentyp mit Vorzeichen (SIGNED).

Wenn Sie eine Zeichenfolge in BIGINT konvertieren, werden führende und nachgestellte Leerzeichen entfernt. Wenn das führende Zeichen "+" ist, wird es ignoriert. Wenn das führende Zeichen "-" ist, werden die restlichen Ziffern als negative Zahl interpretiert. Führende 0-Zeichen werden übersprungen und die restlichen Zeichen werden in einen Ganzzahlwert umgewandelt. Eine Fehlermeldung wird zurückgegeben, wenn der Wert außerhalb des gültigen Bereichs für den Zieldatentyp liegt, die Zeichenfolge unzulässige Zeichen enthält oder die Zeichenfolge nicht als Ganzzahlwert dekodiert werden kann.

Siehe auch

- „BIT-Datentyp“ auf Seite 105
- „INTEGER-Datentyp“ auf Seite 109
- „SMALLINT-Datentyp“ auf Seite 112
- „TINYINT-Datentyp“ auf Seite 112
- „Numerische Funktionen“ auf Seite 161
- „Aggregatfunktionen“ auf Seite 153

Standards und Kompatibilität

- **SQL/2008** Der BIGINT-Datentyp ist SQL-Sprachenfunktion T071 des SQL/2008-Standards.
- **MySQL** Das UNSIGNED-Schlüsselwort kann auf BIGINT folgen.

BIT-Datentyp

Der BIT-Datentyp speichert ein Bit (0 oder 1).

Syntax

BIT

Bemerkungen

BIT ist ein Ganzzahltyp, der die Werte "0" oder "1" speichern kann.

Standardmäßig ist beim BIT-Datentyp NULL nicht zulässig.

Der BIT-Wert erfordert 1 Byte Speicherplatz.

Wenn Sie eine Zeichenfolge in BIT konvertieren, werden führende und nachgestellte Leerzeichen entfernt. Wenn das führende Zeichen "+" ist, wird es ignoriert. Wenn das führende Zeichen "-" ist, werden die restlichen Ziffern als negative Zahl interpretiert. Führende 0-Zeichen werden übersprungen und die restlichen Zeichen werden in einen Ganzzahlwert umgewandelt. Ein Fehler wird zurückgegeben, wenn der Wert nicht 0 oder 1 ist.

Siehe auch

- „BIGINT-Datentyp“ auf Seite 104
- „INTEGER-Datentyp“ auf Seite 109
- „SMALLINT-Datentyp“ auf Seite 112
- „TINYINT-Datentyp“ auf Seite 112
- „Numerische Funktionen“ auf Seite 161
- „Aggregatfunktionen“ auf Seite 153

Standards und Kompatibilität

- **SQL/1999** Der BIT-Datentyp ist SQL-Sprachenfunktion F511 des SQL/1999-Standards.
- **SQL/2008** Die Datentypen BIT und BIT VARYING wurden aus dem SQL/2003-Standard ausgeschlossen. Daher ist in Bezug auf den SQL/2008-Standard der BIT-Datentyp eine Erweiterung des Herstellers.

DECIMAL-Datentyp

Der DECIMAL-Datentyp ist eine Dezimalzahl mit insgesamt *precision* Ziffern und mit *scale* Ziffern nach dem Dezimalzeichen.

Syntax

DECIMAL [(*precision* [, *scale*])]

Parameter

- **precision** Ein ganzzahliger Ausdruck zwischen 1 und 127, der die Anzahl der Ziffern im Ausdruck festlegt. Die Standardeinstellung ist 30.
- **scale** Ein ganzzahliger Ausdruck zwischen 0 und 127, der die Anzahl der Ziffern nach dem Dezimalzeichen festlegt. Die Dezimalstellenzahl sollte immer kleiner oder gleich der Gesamtstellenzahl sein. Die Standardeinstellung ist 6.

Sie können die Standardwerte ändern, indem Sie Datenbankoptionen festlegen.

Bemerkungen

Der DECIMAL-Datentyp ist ein numerisch exakter Datentyp. Seine Genauigkeit bleibt nach arithmetischen Vorgänge bis zur letzten niederwertigen Stelle erhalten.

Die Anzahl von Byte, die zum Speichern einer Dezimalzahl erforderlich sind, kann folgendermaßen geschätzt werden:

```
2 + INT(((precision - scale) + 1) / 2) + INT((scale + 1) / 2);
```

Die INT-Funktion übernimmt den Ganzzahlteil ihres Arguments. Die Speicherung basiert auf dem gespeicherten Wert und nicht auf der in der Spalte zulässigen maximalen Gesamt- und Dezimalstellenzahl.

DECIMAL kann auch als DEC angegeben werden. Unabhängig von der verwendeten Syntax wird der Datentyp als DECIMAL beschrieben.

Bei einer Gesamtstellenzahl von 20 oder weniger und 0 Dezimalstellen kann es sinnvoll sein, stattdessen einen der Ganzzahl-Datentypen (BIGINT, INTEGER, SMALLINT oder TINYINT) zu verwenden. Ganzzahlige Werte erfordern bei einer ähnlichen Anzahl von signifikanten Stellen weniger Speicherplatz als NUMERIC- und DECIMAL-Werte. Vorgänge mit ganzzahligen Werten, wie etwa das Abrufen oder Einfügen, und arithmetische Operatoren liefern in der Regel eine bessere Performance als Vorgänge mit NUMERIC- und DECIMAL-Werten.

DECIMAL entspricht semantisch NUMERIC.

Hinweis

Wenn Sie eine Spalte oder Variable eines DECIMAL-Datentyps erstellen, bei der die Genauigkeit oder die Anzahl der Dezimalstellen die entsprechenden Einstellungen für die Datenbank überschreitet, werden Werte entsprechend den Datenbankeinstellungen gekürzt. Wenn Sie also feststellen, dass Werte in einer als DECIMAL definierten Spalte oder Variablen gekürzt wurden, überprüfen Sie, ob die Gesamtstellenzahl und die Anzahl der Dezimalstellen die Einstellungen der Datenbankoptionen überschreiten.

Siehe auch

- „FLOAT-Datentyp“ auf Seite 108
- „REAL-Datentyp“ auf Seite 111
- „DOUBLE-Datentyp“ auf Seite 107
- „NUMERIC-Datentyp“ auf Seite 109
- „Numerische Funktionen“ auf Seite 161
- „Aggregatfunktionen“ auf Seite 153
- „precision-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „scale-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Die Datentypen DECIMAL und NUMERIC sind Kernfunktionen des SQL/2008-Standards.

DOUBLE-Datentyp

Der DOUBLE-Datentyp speichert Gleitkommazahlen mit doppelter Genauigkeit.

Syntax

DOUBLE

Bemerkungen

Der DOUBLE-Datentyp ist ein angenäherter numerischer Datentyp, sodass bei arithmetischen Operationen Rundungsfehler auftreten können. Da DOUBLE-Werte angenähert sind, sollten Abfragen, die Gleichheiten verwenden, beim Vergleich von DOUBLE-Werten im Allgemeinen vermieden werden.

DOUBLE-Werte erfordern 8 Byte Speicherplatz.

Der Wertebereich liegt zwischen -1.79769313486231e+308 und 1.79769313486231e+308, wobei Zahlen nahe Null so klein sind wie 2.22507385850721e-308. Als DOUBLE gespeicherte Werte sind bis zu 15 signifikanten Stellen genau, können aber nach der fünfzehnten Stelle Rundungsfehler aufweisen.

Siehe auch

- „FLOAT-Datentyp“ auf Seite 108
- „REAL-Datentyp“ auf Seite 111
- „DECIMAL-Datentyp“ auf Seite 106
- „NUMERIC-Datentyp“ auf Seite 109
- „Numerische Funktionen“ auf Seite 161
- „Konvertierungen von numerischen Datentypen“ auf Seite 149
- „Aggregatfunktionen“ auf Seite 153

Standards und Kompatibilität

- **SQL/2008** Der DOUBLE PRECISION-Typ ist eine Kernfunktion des SQL/2008-Standards.

FLOAT-Datentyp

Der FLOAT-Datentyp speichert eine Gleitkommazahl mit einfacher oder doppelter Genauigkeit.

Syntax

FLOAT [(*precision*)]

Parameter

- **precision** Ein Ganzzahlausdruck, der die Anzahl der Bits in der Mantisse angibt, d.h. im dezimalen Teil eines Logarithmus. Im Logarithmus 5,63428 ist die Mantisse beispielsweise 0,63428. Für die Gleitkomma-Gesamtstellenzahl gilt nach IEEE-Standard 754 Folgendes:

Verfügbare Präzisionswerte	Dezimale Gesamtstellenzahl	Äquivalenter SQL-Datentyp	Speichergröße
1-24	7 Dezimalstellen	REAL	4 Byte
25-53	15 Dezimalstellen	DOUBLE	8 Byte

Bemerkungen

Wenn eine Spalte mit dem FLOAT-Datentyp (*precision*) erstellt wird, ist gewährleistet, dass die Spalten auf allen Plattformen die Werte zumindest bis zur festgelegten Mindest-Gesamtstellenzahl speichern. REAL und DOUBLE gewährleisten keine plattformunabhängige Mindest-Gesamtstellenzahl.

Wenn *precision* nicht angegeben wird, ist der FLOAT-Datentyp eine Gleitkommazahl mit einfacher Genauigkeit, äquivalent mit dem Datentyp REAL, und erfordert 4 Byte Speicherplatz.

Wenn *precision* angegeben wird, hat der FLOAT-Datentyp entweder einfache oder doppelte Genauigkeit, abhängig vom Wert der Gesamtstellenzahl. Der Unterschied zwischen REAL und DOUBLE ist plattformabhängig. FLOAT-Werte mit einfacher Genauigkeit erfordern 4 Byte Speicherplatz und FLOAT-Werte mit doppelter Genauigkeit 8 Byte.

Der FLOAT-Datentyp ist ein angenäherter numerischer Datentyp. Nach arithmetischen Vorgängen können bei ihm Rundungsfehler auftreten. Da FLOAT-Werte angenähert sind, sollten beim Vergleichen von FLOAT-Werten Abfragen mit Gleichheiten vermieden werden.

Siehe auch

- [„DOUBLE-Datentyp“ auf Seite 107](#)
- [„REAL-Datentyp“ auf Seite 111](#)
- [„DECIMAL-Datentyp“ auf Seite 106](#)
- [„NUMERIC-Datentyp“ auf Seite 109](#)
- [„Numerische Funktionen“ auf Seite 161](#)
- [„Aggregatfunktionen“ auf Seite 153](#)

Standards und Kompatibilität

- **SQL/2008** Der FLOAT-Typ ist eine Kernfunktion des SQL/2008-Standards.

INTEGER-Datentyp

Der INTEGER-Datentyp speichert Ganzzahlen, die 4 Byte Speicherplatz erfordern.

Syntax

[**UNSIGNED**] **INTEGER**

Bemerkungen

Der INTEGER-Datentyp ist ein numerisch exakter Datentyp. Seine Genauigkeit wird durch arithmetische Vorgänge nicht berührt.

Wenn Sie UNSIGNED festlegen, kann der Ganzzahl keine negative Zahl zugewiesen werden. Standardmäßig ist der Datentyp mit Vorzeichen (SIGNED).

Der Bereich für INTEGER-Werte ist -2^{31} bis $2^{31} - 1$ oder -2147483648 bis 2147483647.

Der Bereich für UNSIGNED INTEGER-Werte ist 0 bis $2^{32} - 1$ oder 0 bis 4294967295.

Wenn Sie eine Zeichenfolge in INTEGER konvertieren, werden führende und nachgestellte Leerzeichen entfernt. Wenn das führende Zeichen "+" ist, wird es ignoriert. Wenn das führende Zeichen "-" ist, werden die restlichen Ziffern als negative Zahl interpretiert. Führende 0-Zeichen werden übersprungen und die restlichen Zeichen werden in einen Ganzzahlwert umgewandelt. Eine Fehlermeldung wird zurückgegeben, wenn der Wert außerhalb des gültigen Bereichs für den Zieldatentyp liegt, die Zeichenfolge unzulässige Zeichen enthält oder die Zeichenfolge nicht als Ganzzahlwert dekodiert werden kann.

Siehe auch

- „BIGINT-Datentyp“ auf Seite 104
- „BIT-Datentyp“ auf Seite 105
- „SMALLINT-Datentyp“ auf Seite 112
- „TINYINT-Datentyp“ auf Seite 112
- „Numerische Funktionen“ auf Seite 161
- „Aggregatfunktionen“ auf Seite 153

Standards und Kompatibilität

- **SQL/2008** Der INTEGER-Typ ist eine Kernfunktion des SQL/2008-Standards. Das UNSIGNED-Schlüsselwort ist eine Erweiterung des Herstellers.
- **MySQL** Das UNSIGNED-Schlüsselwort kann auf INTEGER folgen.

NUMERIC-Datentyp

Der NUMERIC-Datentyp speichert Dezimalzahlen mit insgesamt *precision* Ziffern und mit *scale* Ziffern nach dem Dezimalzeichen.

Syntax

NUMERIC [(*precision* [, *scale*])]

Parameter

- **precision** Ein ganzzahliger Ausdruck zwischen 1 und 127, der die Anzahl der Ziffern im Ausdruck festlegt. Die Standardeinstellung ist 30.
- **scale** Ein ganzzahliger Ausdruck zwischen 0 und 127, der die Anzahl der Ziffern nach dem Dezimalzeichen festlegt. Die Dezimalstellenzahl sollte immer kleiner oder gleich der Gesamtstellenzahl sein. Die Standardeinstellung ist 6.

Bemerkungen

Der NUMERIC-Datentyp ist ein numerisch exakter Datentyp. Seine Genauigkeit wird nach arithmetischen Vorgängen bis zur letzten signifikanten Stelle erhalten.

Die Anzahl von Byte, die für die Speicherung einer Dezimalzahl erforderlich sind, kann wie folgt errechnet werden:

```
2 + INT( (BEFORE+1)/2 ) + INT( (AFTER+1)/2 );
```

Die Funktion INT übernimmt den ganzzahligen Teil ihres Arguments, und die BEFORE- und AFTER-Werte stellen die Anzahl der signifikanten Ziffern vor und nach dem Dezimalzeichen dar. Die Speicherung basiert auf dem gespeicherten Wert und nicht auf der in der Spalte zulässigen maximalen Gesamt- und Dezimalstellenzahl.

Bei einer Gesamtstellenzahl von 20 oder weniger und 0 Dezimalstellen kann es sinnvoll sein, stattdessen einen der Ganzzahl-Datentypen (BIGINT, INTEGER, SMALLINT oder TINYINT) zu verwenden. Ganzzahlige Werte erfordern bei einer ähnlichen Anzahl von signifikanten Stellen weniger Speicherplatz als NUMERIC- und DECIMAL-Werte. Vorgänge mit ganzzahligen Werten, wie etwa das Abrufen oder Einfügen, und arithmetische Operatoren liefern in der Regel eine bessere Performance als Vorgänge mit NUMERIC- und DECIMAL-Werten.

NUMERIC entspricht semantisch DECIMAL.

Hinweis

Wenn Sie eine Spalte oder Variable eines NUMERIC-Datentyps erstellen, bei der die Genauigkeit oder die Anzahl der Dezimalstellen die entsprechenden Einstellungen für die Datenbank überschreitet, werden Werte entsprechend den Datenbankeinstellungen gekürzt. Wenn Sie also feststellen, dass Werte in einer als NUMERIC definierten Spalte oder Variablen gekürzt wurden, überprüfen Sie, ob die Gesamtstellenzahl und die Anzahl der Dezimalstellen die Einstellungen der Datenbankoptionen überschreiten.

Siehe auch

- „FLOAT-Datentyp“ auf Seite 108
- „REAL-Datentyp“ auf Seite 111
- „DOUBLE-Datentyp“ auf Seite 107
- „DECIMAL-Datentyp“ auf Seite 106
- „Numerische Funktionen“ auf Seite 161
- „Aggregatfunktionen“ auf Seite 153
- „Konvertierungen von numerischen Datentypen“ auf Seite 149
- „precision-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „scale-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Kompatibel mit SQL/2008., wenn die SCALE-Option auf 0 gesetzt ist.

REAL-Datentyp

Der REAL-Datentyp speichert Gleitkommazahlen mit einfacher Genauigkeit, die in 4 Byte gespeichert werden.

Syntax

REAL

Bemerkungen

Der REAL-Datentyp ist ein angenäherter numerischer Datentyp, sodass bei arithmetischen Operationen Rundungsfehler auftreten können. Da REAL-Werte angenähert sind, sollten Abfragen, die Gleichheiten verwenden, beim Vergleich von REAL-Werten im Allgemeinen vermieden werden.

REAL-Werte erfordern 4 Byte Speicherplatz.

Der Wertebereich liegt zwischen -3.402823e+38 und 3.402823e+38, wobei Zahlen nahe Null so klein sind wie 1.175494351e-38. Die als REAL-Datentyp gespeicherten Werte sind bis zu 7 signifikanten Stellen genau, können aber nach der sechsten Stelle Rundungsfehler aufweisen.

Siehe auch

- „DOUBLE-Datentyp“ auf Seite 107
- „FLOAT-Datentyp“ auf Seite 108
- „DECIMAL-Datentyp“ auf Seite 106
- „NUMERIC-Datentyp“ auf Seite 109
- „Numerische Funktionen“ auf Seite 161
- „Aggregatfunktionen“ auf Seite 153

Standards und Kompatibilität

- **SQL/2008** Der REAL-Datentyp ist eine Kernfunktion des SQL/2008-Standards.

SMALLINT-Datentyp

Der SMALLINT-Datentyp speichert Ganzzahlen, die 2 Byte Speicherplatz erfordern.

Syntax

[UNSIGNED] SMALLINT

Bemerkungen

Der SMALLINT-Datentyp ist ein numerisch exakter Datentyp. Seine Genauigkeit wird durch arithmetische Vorgänge nicht berührt. Er erfordert 2 Byte Speicherplatz.

Der Bereich für SMALLINT-Werte ist -2^{15} bis $2^{15} - 1$ oder -32768 bis 32767.

Der Bereich für UNSIGNED SMALLINT-Werte ist 0 bis $2^{16} - 1$ oder 0 bis 65535.

Wenn Sie eine Zeichenfolge in SMALLINT konvertieren, werden führende und nachgestellte Leerzeichen entfernt. Wenn das führende Zeichen "+" ist, wird es ignoriert. Wenn das führende Zeichen "-" ist, werden die restlichen Ziffern als negative Zahl interpretiert. Führende 0-Zeichen werden übersprungen und die restlichen Zeichen werden in einen Ganzzahlwert umgewandelt. Eine Fehlermeldung wird zurückgegeben, wenn der Wert außerhalb des gültigen Bereichs für den Zieldatentyp liegt, die Zeichenfolge unzulässige Zeichen enthält oder die Zeichenfolge nicht als Ganzzahlwert dekodiert werden kann.

Siehe auch

- „BIGINT-Datentyp“ auf Seite 104
- „BIT-Datentyp“ auf Seite 105
- „INTEGER-Datentyp“ auf Seite 109
- „TINYINT-Datentyp“ auf Seite 112
- „Numerische Funktionen“ auf Seite 161
- „Aggregatfunktionen“ auf Seite 153

Standards und Kompatibilität

- **SQL/2008** Kompatibel mit SQL/2008. Das UNSIGNED-Schlüsselwort ist eine Erweiterung des Herstellers.
- **MySQL** Das UNSIGNED-Schlüsselwort kann auf SMALLINT folgen.

TINYINT-Datentyp

Der TINYINT-Datentyp speichert Ganzzahlen ohne Vorzeichen, die 1 Byte Speicherplatz erfordern.

Syntax

TINYINT

Bemerkungen

Der TINYINT-Datentyp ist ein numerisch exakter Datentyp. Seine Genauigkeit wird durch arithmetische Vorgänge nicht berührt.

Der Bereich für TINYINT-Werte ist 0 bis $2^8 - 1$ oder 0 bis 255.

In Embedded SQL dürfen TINYINT-Spalten nicht in Variablen abgerufen werden, die als CHAR oder UNSIGNED CHAR definiert sind, da dann versucht wird, den Wert der Spalte in eine Zeichenfolge zu konvertieren und anschließend das erste Byte der Variablen im Programm zuzuweisen. Stattdessen sollten TINYINT-Spalten in 2-Byte- oder 4-Byte-Ganzzahlspalten abgerufen werden. Damit ein TINYINT-Wert aus einer in C geschriebenen Anwendung an eine Datenbank gesendet werden kann, muss die C-Variable vom Typ INTEGER sein.

Wenn Sie eine Zeichenfolge in TINYINT konvertieren, werden führende und nachgestellte Leerzeichen entfernt. Wenn das führende Zeichen "+" ist, wird es ignoriert. Wenn das führende Zeichen "-" ist, werden die restlichen Ziffern als negative Zahl interpretiert. Führende 0-Zeichen werden übersprungen und die restlichen Zeichen werden in einen Ganzzahlwert umgewandelt. Eine Fehlermeldung wird zurückgegeben, wenn der Wert außerhalb des gültigen Bereichs für den Zieldatentyp liegt, die Zeichenfolge unzulässige Zeichen enthält oder die Zeichenfolge nicht als Ganzzahlwert dekodiert werden kann.

Siehe auch

- „BIGINT-Datentyp“ auf Seite 104
- „BIT-Datentyp“ auf Seite 105
- „INTEGER-Datentyp“ auf Seite 109
- „SMALLINT-Datentyp“ auf Seite 112
- „Numerische Funktionen“ auf Seite 161
- „Aggregatfunktionen“ auf Seite 153

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **MySQL** Das UNSIGNED-Schlüsselwort kann vor oder nach TINYINT stehen, hat aber keine Wirkung, da der Typ immer ohne Vorzeichen ist.

Währungsdatentypen

Währungsdatentypen werden zum Speichern von Währungsdaten verwendet.

MONEY-Datentyp

Der MONEY-Datentyp speichert Währungsdaten.

Syntax

MONEY

Bemerkungen

MONEY ist eine als NUMERIC(19,4) implementierte Domäne.

Siehe auch

- [„SMALLMONEY-Datentyp“ auf Seite 114](#)
- [„Numerische Funktionen“ auf Seite 161](#)
- [„Aggregatfunktionen“ auf Seite 153](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

SMALLMONEY-Datentyp

Der SMALLMONEY-Datentyp speichert Währungsdaten, die kleiner sind als eine Million Währungseinheiten.

Syntax

SMALLMONEY

Bemerkungen

SMALLMONEY ist eine als NUMERIC(10,4) implementierte Domäne.

Siehe auch

- [„MONEY-Datentyp“ auf Seite 113](#)
- [„Numerische Funktionen“ auf Seite 161](#)
- [„Aggregatfunktionen“ auf Seite 153](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Bit-Array-Datentypen

Ein Bit-Array ist einer Zeichenfolge ähnlich, außer dass die Bestandteile Bitdaten, also 0 (Nullen) und 1 (Einsen), anstelle von Zeichen sind. Bit-Arrays werden in der Regel verwendet, um eine Folge von Booleschen Werten aufzunehmen.

Die von SQL Anywhere unterstützten Bit-Array-Datentypen sind VARBIT und LONG VARBIT.

LONG VARBIT-Datentyp

Der LONG VARBIT-Datentyp speichert Bit-Arrays von beliebiger Länge.

Syntax

LONG VARBIT

Bemerkungen

Wird zum Speichern von Bit-Arrays (1 und 0) von beliebiger Länge bzw. von Bit-Arrays verwendet, die länger als 32.767 Bit sind.

LONG VARBIT kann auch als LONG BIT VARYING angegeben werden. Unabhängig von der verwendeten Syntax wird der Datentyp als LONG VARBIT beschrieben.

Siehe auch

- „BIT-Datentyp“ auf Seite 105
- „VARBIT-Datentyp“ auf Seite 115
- „Konvertierungen von Bit-Arrays“ auf Seite 148
- „Bit-Array-Funktionen“ auf Seite 155
- „Aggregatfunktionen“ auf Seite 153

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

VARBIT-Datentyp

Der VARBIT-Datentyp wird zum Speichern von Bit-Arrays verwendet, die kürzer als 32.767 Bit sind.

Syntax

VARBIT [(*max-length*)]

Parameter

- **max-length** Die maximale Länge des Bit-Arrays in Bit. Die Länge muss im Bereich von 1 bis 32.767 liegen. Wenn keine Länge angegeben wird, ist ihr Wert "1".

Bemerkungen

VARBIT kann auch als BIT VARYING angegeben werden. Unabhängig von der verwendeten Syntax wird der Datentyp als VARBIT beschrieben.

Siehe auch

- „BIT-Datentyp“ auf Seite 105
- „LONG VARBIT-Datentyp“ auf Seite 114
- „Konvertierungen von Bit-Arrays“ auf Seite 148
- „Bit-Array-Funktionen“ auf Seite 155
- „Aggregatfunktionen“ auf Seite 153
- „Bit-Operatoren“ auf Seite 20

Standards und Kompatibilität

- **SQL/1999** Der Datentyp BIT VARYING ist SQL-Sprachenfunktion F511 des SQL/1999-Standards.

- **SQL/2008** Die Datentypen BIT und BIT VARYING wurden aus dem SQL/2003-Standard ausgeschlossen. Daher ist in Bezug auf den SQL/2008-Standard der Datentyp BIT VARYING eine Erweiterung des Herstellers.

Datentypen für Datum und Uhrzeit

Die folgende Liste gibt einen kurzen Überblick, wie Datumsangaben verarbeitet werden:

- Für alle zulässigen arithmetischen und logischen Operationen mit Datumsangaben werden immer richtige Werte zurückgegeben, unabhängig davon, ob die berechneten Werte über Jahrhundertgrenzen hinwegreichen.
- Die interne Speicherung von Datumsangaben bezieht immer explizit den Jahrhundertteil eines Jahreswerts ein.
- Datumswerte können immer im vollen Jahrhundertformat ausgegeben werden.

Datum und Uhrzeit speichern

Datum und Uhrzeit werden in SQL Anywhere-Datenbanken mit einer der folgenden Datentypen gespeichert:

Datentyp	Segmentbenutzung	Gespeichert in	Bereich der möglichen Werte
DATE	Kalenderdatum (Jahr, Monat, Tag)	4 Byte	Datumsangaben von 0001-01-01 bis 9999-12-31.
TIME	Uhrzeit (Stunde, Minute, Sekunde und Sekundenbruchteil bis 6 Dezimalstellen)	8 Byte	Zeitangaben von 00:00:00.000000 bis 24:00:00.000000.
TIME-STAMP	Kalenderdatum und Uhrzeitstempel (Jahr, Monat, Tag, Stunde, Minute, Sekunde und Sekundenbruchteil bis 6 Dezimalstellen)	8 Byte	Datumsangaben von 0001-01-01 bis 9999-12-31 (Gesamtstellen der Stunden- und Minutenangabe in TIMESTAMP werden vor 1600-02-28 23:59:59 und nach 7911-01-01 00:00:00 ausgeschlossen.)

Datentyp	Segmentbenutzung	Gespeichert in	Bereich der möglichen Werte
TIME-STAMP WITH TIME ZONE	Kalenderdatum, Uhrzeit und Zeitzone-Offset (Jahr, Monat, Tag, Stunde, Minute, Sekunde, Sekundenbruchteil bis 6 Dezimalstellen sowie Zeitzone-Offset in Stunden und Minuten)	10 Byte	Datumsangaben von 0001-01-01 bis 9999-12-31 (Gesamtstellen der Stunden- und Minutenangabe in TIME-STAMP WITH TIME ZONE werden vor 1600-02-28 23:59:59 und nach 7911-01-01 00:00:00 ausgeschlossen.) Zeitzone-Offset von -14:59 bis +14:59.

Möglichkeiten zum Senden von Datums- und Zeitangaben an die Datenbank

Datum und Uhrzeit können auf eine der folgenden Arten an eine Datenbank gesendet werden:

- Über jede Schnittstelle als Zeichenfolge
- Über ODBC oder OLE DB als Binärwert (z.B. mit einer ODBC-TIMESTAMP_STRUCT-Struktur)
- Über Embedded SQL als SQLDATETIME-Struktur

Datum und Uhrzeit mit Zeitzone-Offset können nur als Zeichenfolge an die Datenbank gesendet werden.

Datumsformate

Wenn ein Datum als Zeichenfolge (beim DATE-Datentyp) bzw. als Teil einer Zeichenfolge (bei den Datentypen TIMESTAMP und TIMESTAMP WITH TIME ZONE) an die Datenbank gesendet wird, kann die Zeichenfolge mit einer Reihe von verschiedenen Methoden angegeben werden, einschließlich derjenigen nach ISO 8601, einem internationalen Standard für die Darstellung von Datums- und Zeitangaben.

Ein Datum kann in einem der folgenden ISO 8601-Formate angegeben werden.

- **Kalenderdatum** Das Kalenderdatumsformat lautet JJJJ-MM-TT, wobei JJJJ das Jahr im Gregorianischen Kalender ist, MM der Monat im Jahr zwischen 01 (Januar) und 12 (Dezember) und TT der Tag des Monats zwischen 01 und 31. Beispiel: "2010-04-01" stellt den ersten Tag des Monats April im Jahr 2010 dar. ISO 8601 erfordert nicht die Eingabe des Trennzeichens. Daher sieht "20100401" ebenfalls für den ersten Tag des Monats April im Jahr 2010.

ISO-Kalenderdatum	Format	Beispiel
Grundlegend	JJJJMMTT	20100401
Erweitert	YYYY-MM-DD	2010-04-01

- Wochendatum** Ein weiteres ISO-Datumsformat ist das Wochendatum. Das Format lautet JJJJ-Www-T, wobei JJJJ das Jahr im Gregorianischen Kalender ist, W der Buchstabe **W**, ww ist die Woche des Jahres zwischen 01 (der ersten Woche) und 52 oder 53 (der letzten Woche) und T der Tag der Woche zwischen 1 (Montag) und 7 (Sonntag). Beispiel: "2010-W13-4" stellt den vierten Tag der dreizehnten Woche des Jahres 2010 dar (den 1. April 2010). ISO 8601 erfordert nicht die Eingabe des Trennzeichens. Daher stellt "2010W134" ebenfalls den vierten Tag der dreizehnten Woche des Jahres 2010 dar. Bei geringerer Genauigkeit kann eine Ziffer aus der Darstellung weggelassen werden. ("2010W13" stellt den 29. März 2010 dar.)

ISO-Wochendatum	Format	Beispiel
Grundlegend	JJJJWwwT	2010W134
Erweitert	JJJJ-Www-T	2010-W13-4

- Ordinaldatum** Das letzte ISO-Datumsformat ist das Ordinaldatum. Das Format ist JJJJ-TTT, wobei JJJJ das Jahr im Gregorianischen Kalender ist und TTT die Ordinalzahl des Kalendertags innerhalb des Kalenderjahres. Beispiel: "2010-091" stellt den ersten Tag des Monats April im Jahr 2010 dar. ISO 8601 erfordert nicht die Eingabe des Trennzeichens. Beispiel: "2010091" stellt ebenfalls den 1. April 2010 dar. Das maximale Ordinaldatum ist 366 für Schaltjahre. Beispiel: "2008366" stellt den letzten Tag des Jahres 2008 dar (den 31. Dezember 2008).

ISO-Ordinaldatum	Format	Beispiel
Grundlegend	JJJJTTT	2010091
Erweitert	JJJJ-TTT	2010-091

Andere Datumsformate werden unterstützt. SQL Anywhere ist sehr flexibel in der Interpretation von Zeichenfolgen, die Datumsangaben enthalten. Im Falle von Mehrdeutigkeiten wird die Interpretation des Datumswerts von den Einstellungen für die Datenbankoptionen `date_order` und `nearest_century` gesteuert. Beispiel: Abhängig von der `date_order`-Einstellung kann "02/05/2002" vom Datenbankserver als 2. Mai (DMY), als 5. Februar (MDY) oder als unzulässiger Wert (YMD) interpretiert werden.

Die `nearest_century`-Einstellung legt fest, ob ein zweistelliger Jahreswert als Jahr im zwanzigsten oder einundzwanzigsten Jahrhundert interpretiert wird. Beispiel: Im Falle der Zeichenfolge "02/05/10" würde die `date_order`-Einstellung bestimmen, ob "02" oder "10" als Jahr interpretiert wird, und die `nearest_century`-Einstellung würde bestimmen, ob "02" das Jahr 1902 oder 2002 darstellt bzw. ob "10" das Jahr 1910 oder 2010 darstellt. Der Wert der Option `nearest_century` beeinflusst die Interpretation der zweistelligen Jahreszahlen: 2000 wird bei Werten unter `nearest_century` verwendet, 1900 bei allen anderen Werten. Der Standardwert für diese Option ist 50. Daher wird standardmäßig das Jahr 50 als 1950 interpretiert und das Jahr 49 als 2049.

Die folgende Tabelle zeigt, wie sich der erste Tag im April des Jahres 2010 angeben werden kann, mit der jeweiligen Einstellung für `date_order` und der `nearest_century`-Einstellung 50.

date_order	Format	Beispiel
YMD	JJJJ/MM/TT	2010/04/01
YMD	JJ/MM/TT	10/04/01
MDY	MM/TT/JJJJ	04/01/2010
MDY	MM/TT/JJ	04/01/10
DMY	TT/MM/JJJJ	01/04/2010
DMY	TT/MM/JJ	01/04/10

Da ISO 8601-Formate nicht zweideutig sind und nicht durch die Einstellungen des Benutzers für date_order und nearest_century beeinflusst werden, ist ihre Verwendung empfehlenswert.

Daten können auch mit Monatsnamen angegeben werden. Beispiele dafür sind "2010 April 01", "April 1, 2010" und "1 April 2010". Wenn das Jahr zweideutig angegeben ist, wird die date_order-Option verwendet, um zu ermitteln, welcher Datumsteil das Jahr darstellt und welcher den Tag des Monats. Daher wird "01 April 10" als 10. April 2001 interpretiert, wenn die date_order-Option auf "YMD" gesetzt ist, oder als 1. April 2010, wenn die date_order-Option auf "DMY" gesetzt ist.

Das Jahr in einem Datum kann im Bereich von 0001 bis 9999 liegen. Das in SQL Anywhere minimal mögliche Datum ist 0001-01-01.

Wenn eine Zeichenfolge nur eine teilweise Datumsangabe enthält, werden Standardwerte benutzt, um den Datumswert aufzufüllen. Folgende Standardwerte werden benutzt:

- **year** Das aktuelle Jahr wird verwendet, wenn kein Jahr angegeben ist (z.B. "April 1").
- **month** Der aktuelle Monat wird verwendet, wenn weder Jahr noch Monat angegeben ist (z.B. "23:59:59"), oder 01, wenn ein Jahr angegeben ist (z.B. "2010").
- **day** Der aktuelle Tag wird verwendet, wenn weder Jahr noch Monat angegeben ist (z.B. "23:59:59"), oder 01, wenn ein Monat angegeben ist (z.B. "April").

Im folgenden Beispiel wird der Datumswert aus dem aktuellen Datum erstellt.

```
SELECT CAST('23:59:59' AS TIMESTAMP);
```

Zeitformate

Die Tageszeit kann im ISO 8601-Format mit dem 24-Stunden-System angegeben werden. Sie lautet "hh:mm:ss", wobei hh die Anzahl der vollen Stunden seit Mitternacht ist, mm die Anzahl der vollen Minuten seit Beginn der Stunde und ss die Anzahl der vollen Sekunden seit Beginn der Minute. Beispiel: "23:59:59" stellt die Zeit eine Sekunde vor Mitternacht dar.

Der ISO 8601-Standard gestattet die Weglassung von Sekunden und Minuten. Beispiel: "23:59" stellt die Zeit sechzig Sekunden vor Mitternacht dar.

Der ISO 8601-Standard gestattet es Ihnen außerdem, eine Dezimalzahl in die Sekundeneinheit einzubeziehen. Sekundenbruchteile werden mit einem Komma (,) oder einem Punkt (.) angegeben. Der Sekundenbruchteil wird mit maximal sechs Dezimalstellen gespeichert. Beispiel: "23:59:59,500000" und "23:59:59.500000" stellen beide die Zeit eine halbe Sekunde vor Mitternacht dar. SQL Anywhere bietet keine Unterstützung für Minuten- oder Stundenbruchteile.

ISO 8601 erfordert nicht den Doppelpunkt als Trennzeichen, wenn die Uhrzeit in einer Datumsangabe enthalten ist. Beispiel: "235959" stellt die Zeit eine Sekunde vor Mitternacht dar.

Die maximale Uhrzeit ist "24:00:00". Dies stellt Mitternacht dar. In Kombination mit einem Datum stellt es entweder Mitternacht oder 00:00:00 des nächsten Tags dar. Beispiel: "2010-04-01 24:00:00" ist gleichwertig mit "2010-04-02 00:00:00".

ISO-Zeit	Format	Beispiel
Grundlegend (mit Datum)	hhmmss.ssssss	20100401 235959.500000
Grundlegend (mit Datum)	hhmmss,ssssss	20100401 235959,500000
Erweitert	hh:mm:ss.ssssss	23:59:59.500000
Erweitert	hh:mm:ss,ssssss	23:59:59,500000

Die Nicht-ISO-Bezeichner "AM" und "PM" werden ebenfalls unterstützt. Beispiel: "11:59:59 PM" ist gleichwertig mit "23:59:59".

AM/PM	Format	Beispiel
AM	hh:mm:ss.ssssss AM	11:59:59.500000 AM
AM	hh:mm:ss,ssssss AM	11:59:59,500000 AM
PM	hh:mm:ss.ssssss PM	11:59:59.500000 PM
PM	hh:mm:ss,ssssss PM	11:59:59,500000 PM

Datumsformate mit Uhrzeit

ISO 8601 ermöglicht es, Datum und Uhrzeit mit einem Leerzeichen oder dem Buchstaben **T** zu kombinieren. Beispiel: "2010-04-01 23:59:59" und "2010-04-01T23:59:59" stellen beide die Zeit eine Sekunde vor Mitternacht am ersten Tag des Monats April 2010 dar. Bindestrich und Doppelpunkt als Trennzeichen können weggelassen werden. Beispiel: "20100401T235959" stellt dasselbe Datum und dieselbe Uhrzeit dar. Als Erweiterung zu diesem Format unterstützt SQL Anywhere auch die Weglassung des Trennzeichens zwischen Datum und Uhrzeit. Beispiel: "20100401235959" stellt dasselbe Datum und dieselbe Uhrzeit dar.

SQL Anywhere unterstützt das Mischen von grundlegenden und erweiterten Datums- und Zeitformaten. Beispiel: "20100401T23:59:59" kombiniert das grundlegende und das erweiterte Format.

Zeitzoneformate

ISO 8601 bietet außerdem die Möglichkeit, einen Zeitzone-Offset in eine Zeichenfolge mit Datum und Uhrzeit einzufügen. Das Format lautet:

- **Z** (Zulu) Datum und Uhrzeit werden in der Coordinated Universal Time (UTC) angegeben. Beispiel: "2010-04-01 23:00:00Z" stellt 23:00 Uhr Coordinated Universal Time am ersten Tag des Monats April 2010 dar.
- **+hh:mm** Datum und Uhrzeit liegen um die angegebene Anzahl von Stunden und Minuten vor der UTC. Beispiel: "2010-04-01 23:00:00+04:00" stellt 23:00 Uhr am ersten Tag des Monats April 2010 in einer Zeitzone 4 Stunden östlich der UTC dar.
- **-hh:mm** Datum und Uhrzeit liegen um die angegebene Anzahl von Stunden und Minuten hinter der UTC. Beispiel: "2010-04-01 23:00:00-05:00" stellt 23:00 Uhr am ersten Tag des Monats April 2010 in einer Zeitzone 5 Stunden westlich der UTC dar.

Wenn die Minuten gleich 0 sind, müssen sie nicht im Zeitzone-Offset angegeben werden. Außerdem kann dem Zeitzone-Offset ein Leerzeichen vorangestellt werden. Beispiel: "2010-04-01 23:00:00 -03:30" stellt 23:00 Uhr am ersten Tag des Monats April 2010 in einer Zeitzone dreieinhalb Stunden westlich der UTC dar.

ISO-Zeitzone	Format	Beispiel
Grundlegend	Z	20100401 235959Z
Grundlegend	+hhmm	20100401 235959+0400
Grundlegend	+hh	20100401 235959+04
Grundlegend	-hhmm	20100401 235959-0500
Grundlegend	-hh	20100401 235959-05
Grundlegend	mit T, Sekundenbruchteil	20100401T235959.50-0330
Erweitert	Z	2010-04-01 23:59:59Z
Erweitert	+hh:mm	2010-04-01 23:59:59+04:00
Erweitert	-hh:mm	2010-04-01 23:59:59-05:00
Erweitert	mit T, Sekundenbruchteil	2010-04-01T23:59:59.50-03:30

SQL Anywhere unterstützt das Mischen von grundlegenden und erweiterten Datums-, Zeit- und Zeitzoneformaten. Beispiel: "20100401T23:59:59-05" kombiniert das grundlegende und das erweiterte Format.

Siehe auch

- „Vergleiche von Datumsangaben und Uhrzeiten“ auf Seite 143
- „DATE-Datentyp“ auf Seite 123
- „date_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „nearest_century-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Datentyp“ auf Seite 129
- „TIMESTAMP WITH TIME ZONE-Datentyp“ auf Seite 131

Abrufen von Datums- und Zeitangabe aus der Datenbank

Datums- und Zeitangaben können auf eine der folgenden Arten aus einer Datenbank abgerufen werden:

- Über jede Schnittstelle als Zeichenfolge
- Über ODBC oder OLE DB als Binärwert (z.B. mit einer ODBC-TIMESTAMP_STRUCT-Struktur)
- Über Embedded SQL als SQLDATETIME-Struktur

Datums- und Zeitangaben mit Zeitzonen-Offsets können nur als Zeichenfolgen aus der Datenbank abgerufen werden.

Wenn eine Datums- oder Zeitangabe, mit oder ohne Zeitzonen-Offset, als Zeichenfolge abgerufen wird, geschieht dies in dem durch die Datenbankoptionen `date_format`, `time_format`, `timestamp_format` und `timestamp_with_time_zone_format` angegebenen Format.

Die folgenden arithmetischen Operatoren sind in Datumsangaben zulässig:

- **Zeitstempel + Ganzzahl** Die angegebene Anzahl von Tagen zu einer Datumsangabe oder einem Zeitstempel addieren
- **Zeitstempel - Ganzzahl** Die angegebene Anzahl von Tagen von der Datumsangabe oder dem Zeitstempel subtrahieren
- **Datum - Datum** Die Anzahl der Tage zwischen zwei Datumsangaben oder Zeitstempeln berechnen
- **Datum + Zeit** Einen Zeitstempel durch die Kombination der gegebenen Datums- und Uhrzeitangabe erstellen

Schaltjahre

SQL Anywhere benutzt einen weltweit akzeptierten Algorithmus zur Ermittlung der Schaltjahre. Mit diesem Algorithmus wird ein Jahr als Schaltjahr angesehen, wenn es durch vier dividiert werden kann, außer wenn das Jahr ein Jahrhundertjahr ist (z.B. das Jahr 1900), das nur dann ein Schaltjahr ist, wenn es durch 400 geteilt werden kann.

SQL Anywhere verarbeitet alle Schaltjahre richtig. Die folgende SQL-Anweisung gibt beispielsweise "Dienstag" zurück:

```
SELECT DAYNAME( '2000-02-29' );
```

SQL Anywhere akzeptiert den 29. Februar 2000 - ein Schaltjahr - als Datumsangabe und legt damit den Wochentag fest.

Die folgende Anweisung wird von SQL Anywhere hingegen zurückgewiesen:

Diese Anweisung führt zu einer Fehlermeldung (Konvertierung von '2001-02-29' auf 'timestamp' nicht möglich), da der 29. Februar 2001 nicht existiert.

Siehe auch

- [„SET OPTION-Anweisung“ auf Seite 1039](#)
- [„Datums- und Zeitfunktionen“ auf Seite 156](#)

DATE-Datentyp

Der DATE-Datentyp speichert Kalenderdatumsangaben, z.B. Jahr, Monat und Tag.

Syntax

DATE

Bemerkungen

Das Format, in dem DATE-Werte von Anwendungen als Zeichenfolgen abgerufen werden, wird durch die Einstellung der date_format-Option gesteuert. Ein DATE-Wert, der den 19. Juli 2010 darstellt, kann beispielsweise als "2010/07/19" oder als "Jul 19, 2010" an eine Anwendung zurückgegeben werden, je nach Einstellung der date_format-Option.

Ein DATE-Wert benötigt 4 Byte Speicherplatz.

Siehe auch

- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „Datums- und Zeitfunktionen“ auf Seite 156
- Datumsformate auf Seite 117
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „date_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „date_order-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „DATETIME-Datentyp“ auf Seite 124
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „nearest_century-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Datentyp“ auf Seite 129
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „TIMESTAMP WITH TIME ZONE-Datentyp“ auf Seite 131
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Kompatibel mit SQL/2008.
- **Transact-SQL** Wird von Adaptive Server Enterprise unterstützt.

DATETIME-Datentyp

DATETIME speichert Datums- und Uhrzeitangaben.

Syntax

DATETIME

Bemerkungen

DATETIME ist ein Transact-SQL-Typ.

Das Format, in dem DATETIME-Werte von Anwendungen als Zeichenfolge abgerufen werden, wird durch die Einstellung der timestamp_format-Option gesteuert. Der DATETIME-Wert 2010/04/01T23:59:59.999999 kann beispielsweise als "2010/04/01 23:59:59" oder als "April 1, 2010 23:59:59.999999" an eine Anwendung zurückgegeben werden, je nach Einstellung der timestamp_format-Option.

Ein DATETIME-Wert benötigt 8 Byte Speicherplatz.

Obwohl der Bereich der möglichen Datumsangaben beim DATETIME-Datentyp derselbe ist wie beim DATE-Typ (abgedeckte Jahre 0001 bis 9999), liegt der nützliche Bereich des DATETIME-Datumstyps

zwischen 1600-02-28 23:59:59 und 7911-01-01 00:00:00. Vor und nach diesem Bereich wird der Stunden- und Minutenteil des DATETIME-Werts nicht gespeichert.

Hinweis

Wenn die Anzahl der Dezimalstellen in DATETIME verringert wird, erzeugen integrierte Funktionen, die sich auf Minuten oder Sekunden beziehen, bedeutungslose Ergebnisse.

Beim Konvertieren eines DATETIME-Werts in DATETIMEOFFSET wird die `time_zone_adjustment`-Einstellung der Verbindung für den Zeitzone-Offset im Ergebnis herangezogen. Mit anderen Worten, der Wert gilt für die Verbindung als "lokal". Beim Konvertieren eines DATETIMEOFFSET-Werts in DATETIME wird der Offset verworfen.

Siehe auch

- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75
- „Datums- und Zeitfunktionen“ auf Seite 156
- „Möglichkeiten zum Senden von Datums- und Zeitangaben an die Datenbank“ auf Seite 117
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „date_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „date_order-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „DATE-Datentyp“ auf Seite 123
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Datentyp“ auf Seite 129
- „TIMESTAMP WITH TIME ZONE-Datentyp“ auf Seite 131
- „timestamp_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „nearest_century-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** DATETIME wird von Adaptive Server Enterprise anstelle von TIMESTAMP verwendet. Der Datentyp DATETIME unterstützt in Adaptive Server Enterprise Datumsangaben zwischen dem 1. Januar 1753 und dem 31. Dezember 9999 und unterstützt eine geringere Genauigkeit für den Zeitanteil des Wertes. In SQL Anywhere wird DATETIME als TIMESTAMP ohne diese Einschränkungen implementiert. Sie sollten diese Unterschiede bei der Datenmigration zwischen SQL Anywhere und Adaptive Server Enterprise berücksichtigen.

DATETIMEOFFSET-Datentyp

Der DATETIMEOFFSET-Datentyp ist ein Alias für TIMESTAMP WITH TIME ZONE und wird zum Speichern von Datums-, Uhrzeit- und Zeitzoneangaben verwendet.

Syntax

DATETIMEOFFSET

Bemerkungen

Der DATETIMEOFFSET-Wert enthält Jahr, Monat, Tag, Stunde, Minute, Sekunde, Sekundenbruchteil und die Anzahl der Minuten vor oder nach der Coordinated Universal Time (UTC). Der Sekundenbruchteil wird mit 6 Dezimalstellen gespeichert.

Das Format, in dem DATETIMEOFFSET-Werte von Anwendungen als Zeichenfolge abgerufen werden, wird von der Einstellung der `timestamp_with_time_zone_format`-Option gesteuert. Der DATETIMEOFFSET-Wert 2010/04/01T23:59:59.999999-6:00 kann beispielsweise als "2010/04/01 23:59:59 -06:00" oder als "April 1, 2010 23:59:59.999999 -06:00" an eine Anwendung zurückgegeben werden, je nach Einstellung der `timestamp_with_time_zone_format`-Option.

Ein DATETIMEOFFSET-Wert benötigt 10 Byte Speicherplatz.

Obwohl der Bereich der möglichen Datumsangaben beim DATETIMEOFFSET-Datentyp derselbe ist wie beim DATE-Typ (abgedeckte Jahre 0001 bis 9999), liegt der nützliche Bereich der DATETIMEOFFSET-Datumstypen zwischen 1600-02-28 23:59:59 und 7911-01-01 00:00:00. Vor und nach diesem Bereich wird der Stunden- und Minutenteil des DATETIMEOFFSET-Werts nicht gespeichert.

Verwenden Sie DATETIMEOFFSET nicht für berechnete Spalten oder in materialisierten Ansichten, weil der Wert der steuernden `time_zone_adjustment`-Option sich nach dem Standort und der Jahreszeit der jeweiligen Verbindung richtet.

Zwei DATETIMEOFFSET-Werte gelten als identisch, wenn sie in UTC denselben Zeitpunkt darstellen, und zwar unabhängig vom angewendeten Zeitzone-Offset. Zum Beispiel liefert die folgende Anweisung **Yes**, weil die Ergebnisse als identisch gelten:

```
IF CAST('2009-07-15 08:00:00 -08:00' AS DATETIMEOFFSET) =  
CAST('2009-07-15 11:00:00 -05:00' AS DATETIMEOFFSET) THEN  
SELECT 'Yes'  
ELSE  
SELECT 'No'  
END IF;
```

Wenn Sie den Zeitzone-Offset bei einem DATETIMEOFFSET-Wert weglassen, wird standardmäßig der aktuelle UTC-Offset für den Client verwendet, unabhängig davon, ob der Zeitstempel Datum und Uhrzeit in Standard- oder Sommerzeit darstellt. Beispiel: Wenn sich der Client in der Eastern Standard-Zeitzone befindet und die folgende Anweisung ausführt, während Sommerzeit aktiv ist, wird ein Zeitstempel für die Atlantic Standard-Zeitzone (UTC -4 Stunden) zurückgegeben.

```
SELECT CAST('2009/01/30 12:34:55' AS DATETIMEOFFSET);
```

Der Vergleich von DATETIMEOFFSET-Werten mit Zeitstempeln ohne Zeitzone wird nicht empfohlen, weil sich der standardmäßige Zeitzone-Offset des Clients nach dem geografischen Standort des Clients und nach der Jahreszeit richtet.

Führen Sie die folgende Anweisung aus, um den aktuellen Zeitzone-Offset in Minuten für einen Client zu ermitteln:

```
SELECT CONNECTION_PROPERTY( 'TimeZoneAdjustment' );
```

Hinweis

Die Verbindungseigenschaft TimeZoneAdjustment wird in UltraLite-Datenbanken nicht unterstützt.

Siehe auch

- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75
- „Datums- und Zeitfunktionen“ auf Seite 156
- „Möglichkeiten zum Senden von Datums- und Zeitangaben an die Datenbank“ auf Seite 117
- „DATE-Datentyp“ auf Seite 123
- „DATETIME-Datentyp“ auf Seite 124
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „Ausdrücke“ auf Seite 22
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „TIMESTAMP-Datentyp“ auf Seite 129
- „TIMESTAMP WITH TIME ZONE-Datentyp“ auf Seite 131
- „timestamp_with_time_zone_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „nearest_century-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „date_order-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Die spezifische Verwendung von DATETIMEOFFSET ist eine Erweiterung des Herstellers. Um die Kompatibilität mit SQL/2008 zu gewährleisten, verwenden Sie TIMESTAMP WITH TIME ZONE. Der Typ TIMESTAMP WITH TIME ZONE ist die optionale SQL-Sprachenfunktion F411 des SQL/2008-Standards.

SMALLDATETIME-Datentyp

SMALLDATETIME ist eine als TIMESTAMP implementierte Domäne zum Speichern von Datums- und Uhrzeitangaben. SMALLDATETIME ist ein Transact-SQL-Datentyp.

Syntax

SMALLDATETIME

Bemerkungen

Weitere Hinweise zur Angabe von Datum und Uhrzeit finden Sie unter „[Möglichkeiten zum Senden von Datums- und Zeitangaben an die Datenbank](#)“ auf Seite 117.

Siehe auch

- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75
- „Datums- und Zeitfunktionen“ auf Seite 156
- „DATE-Datentyp“ auf Seite 123
- „DATETIME-Datentyp“ auf Seite 124
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „Ausdrücke“ auf Seite 22
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „TIMESTAMP-Datentyp“ auf Seite 129
- „TIMESTAMP WITH TIME ZONE-Datentyp“ auf Seite 131
- „timestamp_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „timestamp_with_time_zone_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** SMALLDATETIME wird von Adaptive Server Enterprise unterstützt. In Adaptive Server Enterprise unterstützt der SMALLDATETIME-Typ Datumsangaben zwischen dem 1. Januar 1900 und dem 6. Juni 2079 und unterstützt eine geringere Genauigkeit für den Zeitanteil des Werts. In SQL Anywhere ist SMALLDATETIME als TIMESTAMP ohne diese Einschränkungen implementiert. Sie sollten diese Unterschiede bei der Datenmigration zwischen SQL Anywhere und Adaptive Server Enterprise berücksichtigen.

TIME-Datentyp

Der TIME-Datentyp speichert die Uhrzeit mit Stunden, Minuten, Sekunden und Sekundenbruchteil.

Syntax

TIME

Bemerkungen

Das Format, in dem TIME-Werte von Anwendungen als Zeichenfolgen abgerufen werden, wird durch die Einstellung der time_format-Option gesteuert. Der TIME-Wert 23:59:59.999999 kann beispielsweise als 23:59:59, 23:59:59.999 oder 23:59:59.999999 an eine Anwendung zurückgegeben werden, je nach Einstellung der time_format-Option.

Ein TIME-Wert benötigt 8 Byte Speicherplatz.

Wenn Sie ODBC verwenden, wird ein TIME-Wert, der als Binärwert (mit einer ODBC-TIME_STRUCT-Struktur) gesendet oder abgerufen wurde, auf die Angabe von Stunden, Minuten und Sekunden beschränkt. Sekundenbruchteile sind nicht Teil der Struktur. Aus diesem Grund sollten TIME-Werte als Zeichenfolgen gesendet oder abgefragt werden, wenn eine höhere Genauigkeit gewünscht wird.

Siehe auch

- [Zeitformate auf Seite 119](#)
- [„CURRENT TIME-Spezialwert“ auf Seite 72](#)
- [„CURRENT TIMESTAMP-Spezialwert“ auf Seite 73](#)
- [„CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75](#)
- [„Datums- und Zeitfunktionen“ auf Seite 156](#)
- [„DATE-Datentyp“ auf Seite 123](#)
- [„DATETIME-Datentyp“ auf Seite 124](#)
- [„DATE-Funktion \[Datum und Uhrzeit\]“ auf Seite 216](#)
- [„DATETIME-Funktion \[Datum und Uhrzeit\]“ auf Seite 222](#)
- [„Ausdrücke“ auf Seite 22](#)
- [„GETDATE-Funktion \[Datum und Uhrzeit\]“ auf Seite 268](#)
- [„ISDATE-Funktion \[Datentypkonvertierung\]“ auf Seite 293](#)
- [„NOW-Funktion \[Datum und Uhrzeit\]“ auf Seite 330](#)
- [„SMALLDATETIME-Datentyp“ auf Seite 127](#)
- [„TIMESTAMP-Spezialwert“ auf Seite 82](#)
- [„TIMESTAMP-Datentyp“ auf Seite 129](#)
- [„TIMESTAMP WITH TIME ZONE-Datentyp“ auf Seite 131](#)
- [„timestamp_format-Option“ \[*SQL Anywhere Server - Datenbankadministration*\]](#)
- [„timestamp_with_time_zone_format-Option“ \[*SQL Anywhere Server - Datenbankadministration*\]](#)
- [„UTC TIMESTAMP-Spezialwert“ auf Seite 84](#)

Standards und Kompatibilität

- **SQL/2008** Kompatibel mit SQL/2008.
- **Transact-SQL** Der TIME-Datentyp wird von Adaptive Server Enterprise unterstützt. Sybase Adaptive Server Enterprise unterstützt jedoch eine Millisekundenauflösung (drei Stellen) statt einer Mikrosekundenauflösung (sechs Stellen). Bei der Datenmigration zwischen SQL Anywhere und Adaptive Server Enterprise sollten Sie diese Unterschiede berücksichtigen. Verwenden Sie zum Migrieren von TIME-Werten den Adaptive Server Enterprise-Datentyp BIGTIME.

TIMESTAMP-Datentyp

Der TIMESTAMP-Datentyp speichert einen Zeitpunkt mit Jahr, Monat, Tag, Stunde, Minute, Sekunde und Sekundenbruchteil, auf sechs Dezimalstellen genau.

Syntax

TIMESTAMP

Bemerkungen

Das Format, in dem **TIMESTAMP**-Werte von Anwendungen als Zeichenfolgen abgerufen werden, wird durch die Einstellung der `timestamp_format`-Option gesteuert. Der **TIMESTAMP**-Wert 2010/04/01T23:59:59.999999 kann beispielsweise als "2010/04/01 23:59:59" oder als "April 1, 2010 23:59:59.999999" an eine Anwendung zurückgegeben werden, je nach Einstellung der `timestamp_format`-Option.

Der **TIMESTAMP**-Wert benötigt 8 Byte Speicherplatz.

Obwohl der Bereich der möglichen Datumsangaben beim **TIMESTAMP**-Datentyp derselbe ist wie beim **DATE**-Typ (abgedeckte Jahre 0001 bis 9999), liegt der nützliche Bereich der **TIMESTAMP**-Datumstypen zwischen 1600-02-28 23:59:59 und 7911-01-01 00:00:00. Vor und nach diesem Bereich wird der Stunden- und Minutenteil des **TIMESTAMP**-Werts nicht gespeichert.

Hinweis

Wenn die Anzahl der Dezimalstellen in **TIMESTAMP** verringert wird, erzeugen integrierte Funktionen, die sich auf Minuten oder Sekunden beziehen, bedeutungslose Ergebnisse.

Beim Konvertieren eines **TIMESTAMP**-Werts in **TIMESTAMP WITH TIME ZONE** wird die `time_zone_adjustment`-Einstellung der Verbindung für den Zeitzone-Offset im Ergebnis herangezogen. Mit anderen Worten, der Wert gilt für die Verbindung als "lokal". Beim Konvertieren eines **TIMESTAMP WITH TIME ZONE**-Werts in **TIMESTAMP** wird der Offset verworfen.

Siehe auch

- „Möglichkeiten zum Senden von Datums- und Zeitangaben an die Datenbank“ auf Seite 117
- „**CURRENT TIME**-Spezialwert“ auf Seite 72
- „**CURRENT TIMESTAMP**-Spezialwert“ auf Seite 73
- „**CURRENT UTC TIMESTAMP**-Spezialwert“ auf Seite 75
- „Datums- und Zeitfunktionen“ auf Seite 156
- „**DATE**-Datentyp“ auf Seite 123
- „**DATETIME**-Datentyp“ auf Seite 124
- „**DATE**-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „**DATETIME**-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „`date_order`-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Ausdrücke“ auf Seite 22
- „**GETDATE**-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „**ISDATE**-Funktion [Datentypkonvertierung]“ auf Seite 293
- „`nearest_century`-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „**NOW**-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „**SMALLDATETIME**-Datentyp“ auf Seite 127
- „**TIME**-Datentyp“ auf Seite 128
- „**TIMESTAMP**-Spezialwert“ auf Seite 82
- „**TIMESTAMP WITH TIME ZONE**-Datentyp“ auf Seite 131
- „`timestamp_format`-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „`timestamp_with_time_zone_format`-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „**UTC TIMESTAMP**-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Kompatibel mit SQL/2008.
- **Transact-SQL** Adaptive Server Enterprise benutzt den DATETIME-Typ für Zeitstempelwerte.

TIMESTAMP WITH TIME ZONE-Datentyp

Der TIMESTAMP WITH TIME ZONE-Datentyp speichert einen Zeitpunkt mit Zeitzonen-Offset.

Syntax**TIMESTAMP WITH TIME ZONE****Bemerkungen**

Der TIMESTAMP WITH TIME ZONE-Wert enthält Jahr, Monat, Tag, Stunde, Minute, Sekunde, Sekundenbruchteil und die Anzahl der Minuten vor oder nach der Coordinated Universal Time (UTC). Der Sekundenbruchteil wird mit sechs Dezimalstellen gespeichert.

Das Format, in dem TIMESTAMP WITH TIME ZONE-Werte von Anwendungen als Zeichenfolgen abgerufen werden, wird durch die Einstellung der `timestamp_with_time_zone_format`-Option gesteuert. Der TIMESTAMP WITH TIME ZONE-Wert 2010/04/01T23:59:59.999999-6:00 kann beispielsweise als "2010/04/01 23:59:59 -06:00" oder als "April 1, 2010 23:59:59.999999 -06:00" an eine Anwendung zurückgegeben werden, je nach Einstellung der `timestamp_with_time_zone_format`-Option.

Ein TIMESTAMP WITH TIME ZONE-Wert benötigt 10 Byte Speicherplatz.

Obwohl der Bereich der möglichen Datumsangaben beim TIMESTAMP WITH TIME ZONE-Datentyp derselbe ist wie beim DATE-Typ (abgedeckte Jahre 0001 bis 9999), liegt der nützliche Bereich der TIMESTAMP WITH TIME ZONE-Datumstypen zwischen 1600-02-28 23:59:59 und 7911-01-01 00:00:00. Vor und nach diesem Bereich wird der Stunden- und Minutenteil des TIMESTAMP WITH TIME ZONE-Werts nicht gespeichert.

Verwenden Sie TIMESTAMP WITH TIME ZONE nicht für berechnete Spalten oder in materialisierten Ansichten, weil der Wert der steuernden `time_zone_adjustment`-Option sich nach dem Standort und der Jahreszeit der jeweiligen Verbindung richtet.

Zwei TIMESTAMP WITH TIME ZONE-Werte gelten als identisch, wenn sie in UTC denselben Zeitpunkt darstellen, und zwar unabhängig vom angewendeten Zeitzonen-Offset. Zum Beispiel liefert die folgende Anweisung **Yes**, weil die Ergebnisse als identisch gelten:

```
IF CAST('2009-07-15 08:00:00 -08:00' AS TIMESTAMP WITH TIME ZONE) =
CAST('2009-07-15 11:00:00 -05:00' AS TIMESTAMP WITH TIME ZONE) THEN
SELECT 'Yes'
ELSE
SELECT 'No'
END IF;
```

Wenn Sie den Zeitzonen-Offset bei einem TIMESTAMP WITH TIME ZONE-Wert weglassen, wird standardmäßig der aktuelle UTC-Offset für den Client verwendet, unabhängig davon, ob der Zeitstempel Datum und Uhrzeit in Standard- oder Sommerzeit darstellt. Beispiel: Wenn sich der Client in der Eastern

Standard-Zeitzone befindet und die folgende Anweisung ausführt, während Sommerzeit aktiv ist, wird ein Zeitstempel für die Atlantic Standard-Zeitzone (UTC -4 Stunden) zurückgegeben.

```
SELECT CAST('2009/01/30 12:34:55' AS TIMESTAMP WITH TIME ZONE)
```

- **Vergleich von TIMESTAMP WITH TIME ZONE mit anderen Datentypen** Der Vergleich von TIMESTAMP WITH TIME ZONE-Werten mit Zeitstempeln ohne Zeitzonen wird nicht empfohlen, weil sich der standardmäßige Zeitzonen-Offset des Clients nach dem geografischen Standort des Clients und nach der Jahreszeit richtet.

Führen Sie die folgende Anweisung aus, um den aktuellen Zeitzonen-Offset in Minuten für einen Client zu ermitteln:

```
SELECT CONNECTION_PROPERTY( 'TimeZoneAdjustment' );
```

- **Konvertierung in oder aus TIMESTAMP WITH TIME ZONE** Beim Konvertieren eines TIMESTAMP-Werts in TIMESTAMP WITH TIME ZONE wird die time_zone_adjustment-Einstellung der Verbindung für den Zeitzonen-Offset im Ergebnis herangezogen. Mit anderen Worten, der Wert gilt für die Verbindung als "lokal". Beim Konvertieren eines TIMESTAMP WITH TIME ZONE-Werts in TIMESTAMP wird der Offset verworfen. Konvertierungen in oder aus Typen, bei denen es sich nicht um Zeichenfolgen, Datumsangaben oder Datums- und Uhrzeitangaben handelt, werden nicht unterstützt.

Siehe auch

- „Vergleiche von Datumsangaben und Uhrzeiten“ auf Seite 143
- „Möglichkeiten zum Senden von Datums- und Zeitangaben an die Datenbank“ auf Seite 117
- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75
- „Datums- und Zeitfunktionen“ auf Seite 156
- „DATE-Datentyp“ auf Seite 123
- „DATETIME-Datentyp“ auf Seite 124
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „date_order-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Ausdrücke“ auf Seite 22
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „nearest_century-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „TIMESTAMP-Datentyp“ auf Seite 129
- „timestamp_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „timestamp_with_time_zone_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Die Unterstützung für **TIMESTAMP WITH TIME ZONE** ist die optionale SQL-Sprachenfunktion F411 des SQL/2008-Standards.

Binärdatentypen

Binäre Datentypen speichern Binärdaten, darunter auch Bilder und andere Informationstypen, die von der Datenbank nicht interpretiert werden.

BINARY-Datentyp

Der BINARY-Datentyp speichert Binärdaten mit einer festgelegten Maximallänge (in Byte).

Syntax

BINARY [(*max-length*)]

Parameter

- **max-length** Die maximale Länge des Werts in Byte. Wenn die Länge nicht angegeben wird, ist sie gleich 1.

Der Wert muss im Bereich 1 bis 32767 liegen.

Bemerkungen

Bei Vergleichsvorgängen werden BINARY-Werte Byte für Byte miteinander verglichen. Dies ist ein Unterschied zum CHAR-Datentyp, bei dem Werte anhand der Kollationssequenz der Datenbank verglichen werden.

Wenn eine binäre Zeichenfolge ein Präfix der anderen ist, wird die kürzere Zeichenfolge als kleiner als die längere Zeichenfolge angesehen.

Im Gegensatz zu CHAR-Werten werden BINARY-Werte während einer Zeichensatzkonvertierung nicht umgewandelt.

BINARY entspricht semantisch VARBINARY. Es ist ein Typ mit variabler Länge. In anderen Datenbankverwaltungssystemen ist BINARY ein Typ mit fester Länge.

Siehe auch

- „[VARBINARY-Datentyp](#)“ auf Seite 135
- „[LONG BINARY-Datentyp](#)“ auf Seite 134
- „[Zeichenfolgenfunktionen](#)“ auf Seite 163
- „[Bit-Operatoren](#)“ auf Seite 20

Standards und Kompatibilität

- **SQL/2008** Der BINARY-Datentyp ist SQL-Sprachenfunktion T021 des SQL/2008-Standards.

IMAGE-Datentyp

Der IMAGE-Datentyp speichert Binärdaten von beliebiger Länge.

Syntax

IMAGE

Bemerkungen

IMAGE ist eine als LONG BINARY implementierte Domäne.

Siehe auch

- [„LONG BINARY-Datentyp“ auf Seite 134](#)
- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

LONG BINARY-Datentyp

Der LONG BINARY-Datentyp speichert Binärdaten von beliebiger Länge.

Syntax

LONG BINARY

Bemerkungen

Die maximal zulässige Länge beträgt 2 GB minus 1 Byte ($2^{31} - 1$).

Siehe auch

- [„BINARY-Datentyp“ auf Seite 133](#)
- [„VARBINARY-Datentyp“ auf Seite 135](#)

Standards und Kompatibilität

- **SQL/2008** Der LONG BINARY-Datentyp umfasst die SQL-Sprachenfunktionen T021 (die Datentypen BINARY und VARBINARY) und T041 (grundlegende Unterstützung des LOB-Datentyps) des SQL/2008-Standards.

UNIQUEIDENTIFIER-Datentyp

Der UNIQUEIDENTIFIER-Datentyp speichert UUID-Werte (auch als GUID-Werte bezeichnet).

Syntax

UNIQUEIDENTIFIER

Bemerkungen

Der UNIQUEIDENTIFIER-Datentyp wird üblicherweise für eine Primärschlüsselspalte oder andere eindeutige Spalten verwendet, um UUID-(Universally Unique Identifier-)Werte aufzunehmen, die Zeilen eindeutig identifizieren. Die NEWID-Funktion generiert UUID-Werte so, dass ein auf einem Computer erzeugter Wert nicht mit einer auf einem anderen Computer erzeugten UUID übereinstimmt. Mit NEWID generierte UNIQUEIDENTIFIER-Werte können daher als Schlüssel in einer Synchronisationsumgebung verwendet werden.

Beispiel:

```
CREATE TABLE T1 (
    pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    c1 INT );
```

UUID-Werte werden auch als GUID-Werte (Globally Unique Identifier) bezeichnet. UUID-Werte enthalten Bindestriche, um mit anderen RDBMS kompatibel zu sein. Sie können dies ändern, indem Sie die Option `uuid_has_hyphens` auf OFF einstellen.

UNIQUEIDENTIFIER-Werte werden bei Bedarf automatisch zwischen Zeichenfolgenwerten und binären Werten konvertiert.

UNIQUEIDENTIFIER-Werte werden als BINARY(16) gespeichert, aber für Clientanwendungen als BINARY(36) beschrieben. Das gewährleistet, dass der Client, wenn er einen Wert als Zeichenfolge abrufen, genügend Speicher für das Ergebnis zugeordnet hat. Bei ODBC-Clientanwendungen werden UNIQUEIDENTIFIER-Werte als ein SQL_GUID-Typ angezeigt.

Siehe auch

- „Der Standardwert NEWID“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „NEWID-Funktion [Verschiedene]“ auf Seite 321
- „UIDTOSTR-Funktion [Zeichenfolge]“ auf Seite 424
- „STRTOUUID-Funktion [Zeichenfolge]“ auf Seite 399
- „uuid_has_hyphens-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

VARBINARY-Datentyp

Der VARBINARY-Datentyp speichert Binärdaten mit einer festgelegten Maximallänge (in Byte).

Syntax

```
VARBINARY [ ( max-length ) ]
```

Parameter

- **max-length** Die maximale Länge des Werts in Byte. Wenn die Länge nicht angegeben wird, ist sie gleich 1.

Der Wert muss im Bereich 1 bis 32767 liegen.

Bemerkungen

Bei Vergleichsvorgängen werden VARBINARY-Werte Byte für Byte miteinander verglichen. Dies ist ein Unterschied zum CHAR-Datentyp, bei dem Werte anhand der Kollationssequenz der Datenbank verglichen werden.

Wenn eine binäre Zeichenfolge ein Präfix der anderen ist, wird die kürzere Zeichenfolge als kleiner als die längere Zeichenfolge angesehen.

VARBINARY-Werte werden während einer Zeichensatzkonvertierung nicht umgewandelt.

VARBINARY kann auch als BINARY VARYING angegeben werden. Unabhängig von der verwendeten Syntax wird der Datentyp als VARBINARY beschrieben.

Siehe auch

- „BINARY-Datentyp“ auf Seite 133
- „LONG BINARY-Datentyp“ auf Seite 134
- „Zeichenfolgenfunktionen“ auf Seite 163
- „Bit-Operatoren“ auf Seite 20

Standards und Kompatibilität

- **SQL/2008** Der VARBINARY-Datentyp umfasst die SQL-Sprachenfunktion T021 (die Datentypen BINARY und VARBINARY) des SQL/2008-Standards.

Zusammengesetzte Datentypen

Zusammengesetzte Datentypen sind Werte, die aus null oder mehr Elementen bestehen, wobei jedes Element den Wert eines bestimmten Datentyps aufweist.

SQL Anywhere unterstützt die zusammengesetzten Datentypen ROW und ARRAY. Diese Datentypen stellen eine effizientere Methode zum Speichern von Listen dar, weil sie die Möglichkeit bieten, Struktur und Datentyp ihrer Werte zu definieren. Außerdem erleichtern sie den Zugriff auf Listenelemente, entweder direkt mit doppelten eckigen Klammern oder als Ergebnismenge mit dem UNNEST-Operator. Ziehen Sie den ARRAY-Datentyp in Erwägung, wenn Sie Listen als begrenzte Zeichenfolgen in VARCHAR-Spalten speichern und bei der syntaktischen Analyse `sa_split_list` verwenden. ARRAY-Daten sind sehr hilfreich, wenn Sie verschiedene Objekte speichern möchten, die miteinander verknüpft sind. ROW-Daten sind hilfreich, wenn Sie mehrere Werte für ein Objekt speichern möchten.

ARRAY-Typen deklarieren

Ein ARRAY-Typ ist eine homogene, sortierte Sammlung, die ganz oder teilweise als Argument an gespeicherte SQL-Prozeduren oder an SQL-Funktionen übergeben werden kann.

Ein ARRAY-Typ kann aus bis zu 6,4 Millionen Elementen bestehen. Ein ARRAY-Wert wird als ARRAY-Wert des deklarierten Typs mit Null-Länge initialisiert, wobei jedes Element NULL ist.

Ein ARRAY-Konstruktor erstellt einen ARRAY-Wert so, dass das Array in einer Abfrage verarbeitet oder als Argument an eine gespeicherte SQL-Prozedur oder eine SQL-Funktion übergeben werden kann.

Der Datenbankserver unterstützt die folgenden Syntaxvarianten zum Deklarieren von Variablen vom Typ ARRAY:

1. **DECLARE** *variable-name* *element-type-name* **ARRAY** [(*maximum-size*)]
2. **DECLARE** *variable-name* **ARRAY** [(*maximum-size*)] **OF** *element-type-name*

Das Array kann bis zu 6,4 Millionen Elemente enthalten, wenn *maximum-size* weggelassen wird. Variablen vom Typ ARRAY werden im Gegensatz zu allen anderen nicht zusammengesetzten Variablen nicht auf NULL initialisiert. Stattdessen werden sie als Array mit Null-Länge initialisiert.

ARRAY-Typen können nicht als Spalten in einer Basistabelle oder einer temporären Tabelle gespeichert werden und werden in folgenden Kontexten nicht unterstützt:

- Äußerste SELECT-Liste einer Ansichtsdefinition
- SELECT-Block der obersten Ebene oder Abfrageausdruck, der an den Client zurückgegeben wird
- Basistabelle
- Temporäre Tabelle
- Embedded SQL-Anweisung FETCH

Das folgende Beispiel veranschaulicht, wie ein Array von 5 Ganzzahlen deklariert wird:

```
DECLARE NewArray INTEGER ARRAY(5);
```

Das folgende Beispiel veranschaulicht eine alternative Methode zum Deklarieren eines Arrays von 5 Ganzzahlen, die mit der Oracle-Syntax kompatibel ist:

```
DECLARE NewArray ARRAY(5) OF INTEGER;
```

Das folgende Beispiel veranschaulicht, wie ein zweidimensionales Array deklariert wird, wobei New2DArray 10 Elemente enthält, von denen jedes ein aus fünf Elementen bestehendes Array von Ganzzahlen ist:

```
DECLARE New2DArray INTEGER ARRAY(5) ARRAY(10);
```

Das folgende Beispiel veranschaulicht eine alternative Methode zum Deklarieren eines zweidimensionalen Arrays, die mit der Oracle-Syntax kompatibel ist, wobei New2DArray 10 Elemente enthält, von denen jedes ein aus fünf Elementen bestehendes Array von Ganzzahlen ist:

```
DECLARE New2DArray ARRAY(10) OF ARRAY(5) OF INTEGER;
```

ROW-Typen deklarieren

Ein ROW-Typ wird durch einen Zeilentyp-Deskriptor beschrieben, der aus den Felddeskriptoren für die einzelnen Felder in dem ROW-Typ besteht. ROW-Typen sind auf 45000 Felder beschränkt. Die gleiche Grenze gilt auch für die Anzahl von Spalten in einer Tabelle.

Variablen im ROW-Typ werden als ROW-Wert vom deklarierten Typ initialisiert, wobei jedes Feld auf NULL initialisiert wird.

ROW unterstützt die Konstruktion von strukturierten Typen, die aus einer Gruppe von Feldern von möglicherweise unterschiedlichen Typen bestehen. ROW-Typen können Teil von Zeilentypen höherer Ordnung sein, was komplexe Strukturen mit anderen Zeilentypen und Arrays ermöglicht.

Das folgende Beispiel veranschaulicht, wie eine Variable namens student deklariert wird, die als strukturierter Typ aus vier verschiedenen Feldern definiert ist:

```
DECLARE student ROW(studentID INTEGER,  
    student_first_name VARCHAR(40),  
    student_last_name VARCHAR(50),  
    student address LONG VARCHAR);
```

Das folgende Beispiel veranschaulicht, wie Sie einen ROW-Wert als vollständige Struktur zuordnen können:

```
DECLARE employee ROW(empID INTEGER,  
    address ROW(street_address LONG VARCHAR,  
        city VARCHAR(50),  
        province VARCHAR(30),  
        country VARCHAR(40)  
    )  
);  
DECLARE temp_address ROW(street_address LONG VARCHAR,  
    city VARCHAR(50),  
    province VARCHAR(30),  
    country VARCHAR(40)  
);  
SET temp_address = employee.address;
```

Siehe auch

- „ARRAY-Konstruktor [zusammengesetzt]“ auf Seite 168
- „ROW-Konstruktor [zusammengesetzt]“ auf Seite 371

Räumliche Datentypen

SQL Anywhere unterstützt viele räumliche Datentypen. Die Dokumentation für diese Datentypen befinden sich in der Dokumentation der entsprechenden SQL-API.

Siehe auch

- „Unterstützte räumliche Datentypen und ihre Hierarchie“ [*SQL Anywhere Server - Unterstützung für räumliche Daten*]

Domänen

Domänen sind Aliasnamen für integrierte Datentypen, gegebenenfalls auch für Gesamtstellen- und Dezimalstellenwerte, und können DEFAULT-Werte und CHECK-Bedingungen enthalten. Manche Domänen, wie die Währungsdatentypen, sind in SQL Anywhere im Voraus festgelegt, aber Sie können eigene hinzufügen.

Domänen, auch **benutzerdefinierte Datentypen** genannt, ermöglichen eine automatische Definition der Spalten in einer Datenbank für einen bestimmten Datentyp, wobei dieselben NULL- oder NOT NULL-

Bedingungen, dieselben DEFAULT-Einstellungen und dieselben CHECK-Bedingungen gelten. Domänen fördern die Konsistenz in der gesamten Datenbank und können manche Typen von Fehlern verhindern.

Einfache Domänen

Domänen werden mit der Anweisung CREATE DOMAIN erstellt.

Die folgende Anweisung erstellt einen Datentyp namens street_address, bei dem es sich um eine 35-stellige Zeichenfolge handelt.

```
CREATE DOMAIN street_address CHAR( 35 );
```

CREATE DATATYPE kann als Alternative zu CREATE DOMAIN verwendet werden, was aber nicht empfehlenswert ist.

Sie müssen das CREATE DATATYPE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben, um Domänen erstellen zu können. Nachdem ein Datentyp erstellt wurde, wird der Benutzer, der die CREATE DOMAIN-Anweisung ausgeführt hat, zum Eigentümer des Datentyps. Jeder Benutzer kann den Datentyp verwenden. Anders als bei anderen Datenbankobjekten wird der Name des Eigentümers nicht als Präfix für den Datentypnamen verwendet.

Der street_address-Datentyp kann auf exakt dieselbe Weise benutzt werden wie jeder andere Datentyp, wenn Spalten definiert werden. Beispiel: Die folgende Tabelle mit zwei Spalten hat die zweite Spalte als street_address:

```
CREATE TABLE twocol (
    ID INT,
    street street_address
);
```

Sie können auch eine Domäne löschen:

```
DROP DOMAIN street_address;
```

Diese Anweisung kann nur ausgeführt werden, wenn der Datentyp in keiner Tabelle der Datenbank benutzt wird. Wenn Sie versuchen, eine Domäne, die verwendet wird, zu löschen, wird eine Fehlermeldung angezeigt.

Siehe auch

- [„CREATE DOMAIN-Anweisung“ auf Seite 598](#)
- [„DROP DOMAIN-Anweisung“ auf Seite 807](#)

Integritätsregeln und Standardwerte bei Domänen

Viele mit Spalten verbundene Attribute, wie beispielsweise das Zulassen von NULL, die Einrichtung eines DEFAULT-Werts usw., können in eine Domäne integriert werden. Jede Spalte, die mit dem Datentyp definiert wird, übernimmt automatisch die NULL-Einstellung, die CHECK-Bedingung und die DEFAULT-Werte. Damit können in der ganzen Datenbank Spalten mit ähnlichen Bedeutungen einheitlich angelegt werden.

Beispiel: Viele Primärschlüsselspalten in der SQL Anywhere-Beispieldatenbank sind Ganzzahlspalten, die ID-Nummern enthalten. Die folgende Anweisung erstellt einen Datentyp, der für solche Spalten sinnvoll verwendet werden kann:

```
CREATE DOMAIN ID INT  
NOT NULL  
DEFAULT AUTOINCREMENT  
CHECK( @col > 0 );
```

Standardmäßig gilt: Eine Spalte, die mit dem id-Datentyp erstellt wurde, lässt keine NULL-Werte zu, übernimmt als Standard einen autoinkrementierten Wert und muss eine positive Zahl enthalten. Jeder beliebige Bezeichner kann an Stelle von *col* in der Variablen *@col* verwendet werden.

Die Attribute eines Datentyps können aufgehoben werden, indem man explizit Attribute für die Spalte liefert. Eine mit dem id-Datentyp erstellte Spalte mit explizit erlaubten NULL-Werten lässt NULL zu, unabhängig von der Einstellung des id-Datentyps.

Kompatibilität

- **Benannte Integritätsregeln und Standardwerte** In SQL Anywhere werden Domänen mit einem Basis-Datentyp und optional einer NULL- oder NOT-NULL-Bedingung, einem Standardwert und einer CHECK-Bedingung erstellt. Benannte Integritätsregeln und benannte Standardwerte werden nicht unterstützt.
- **Datentypen erstellen** In SQL Anywhere können Sie die Systemprozedur `sp_addtype` benutzen, um eine Domäne hinzuzufügen, oder aber die Anweisung `CREATE DOMAIN` verwenden.

Datentypvergleiche

Wenn ein Vergleich (z.B. =) zwischen Argumenten mit verschiedenen Datentypen durchgeführt wird, muss mindestens eines der Argumente konvertiert werden, damit der Vergleich an einem einheitlichen Datentyp vorgenommen werden kann.

Einige Regeln können zu Konvertierungen führen, die fehlschlagen, oder unerwartete Ergebnisse des Vergleichs liefern. Führen Sie in diesen Fällen eine explizite Konvertierung eines der Argumente mit `CAST` oder `CONVERT` aus.

Sie können diese Konvertierungsregeln überschreiben, indem Sie ein explizites Casting des Arguments in den jeweils anderen Datentyp durchführen. Wenn Sie beispielsweise einen DATE-Wert und einen CHAR-Wert als CHAR vergleichen möchten, müssen Sie den DATE-Wert explizit in einen CHAR-Wert umwandeln.

Siehe auch

- „[CAST-Funktion \[Datentypkonvertierung\]](#)“ auf Seite 186

Verlustreiche Konvertierung und Ersetzungszeichen

Wenn ein Zeichen nicht in dem Zeichensatz dargestellt werden kann, in den es konvertiert wird, wird stattdessen ein Ersetzungszeichen verwendet. Konvertierungen dieser Art werden als **verlustreich** betrachtet, da das ursprüngliche Zeichen verloren geht, wenn es im Zielzeichensatz nicht dargestellt werden kann.

Unterschiedliche Zeichensätze können nicht nur unterschiedliche Ersetzungszeichen enthalten. Das Ersetzungszeichen für einen Zeichensatz kann darüber hinaus ein Nicht-Ersetzungszeichen in einem anderen Zeichensatz sein. Dies ist wichtig zu wissen, wenn mehrere Konvertierungen eines Zeichens durchgeführt werden, da das letzte Zeichen möglicherweise nicht als das erwartete Ersetzungszeichen des Zielzeichensatzes angezeigt wird.

Beispiel: Der Client-Zeichensatz ist Windows-1252 und der Datenbank-Zeichensatz ist ISO_8859-1:1987 (der US-Standard für einige Unix-Versionen). Eine Nicht-Unicode-Clientanwendung (z.B. Embedded SQL) versucht, das Euro-Symbol in eine CHAR-, VARCHAR- oder LONG VARCHAR-Spalte einzufügen. Da das Zeichen nicht im CHAR-Zeichensatz enthalten ist, wird das Ersetzungszeichen für ISO_8859-1:1987, 0x1A, eingefügt.

Wenn dieses ISO_8859-1:1987-Ersetzungszeichen dann als Unicode (z.B. durch Ausführen des Befehls `SELECT * FROM t` in einer gebundenen SQL_C_WCHAR-Spalte in ODBC) abgerufen wird, wird es zum Unicode-Punktzeichen U+001A. (In Unicode ist der Code-Point U+001A das Steuerzeichen für die Trennung von Datensätzen.) Allerdings ist das Ersetzungszeichen für Unicode der Code Point U+FFFD. Dieses Beispiel zeigt: Auch wenn in Ihren Daten Ersetzungszeichen enthalten sind, können diese Zeichen aufgrund von mehreren Konvertierungen möglicherweise nicht in die Ersetzungszeichen des Zielzeichensatzes konvertiert werden.

Daher ist es wichtig, die Verwendung von Ersetzungszeichen zu verstehen und zu testen, wenn zwischen mehreren Zeichensätzen konvertiert wird.

Die `on_charset_conversion_failure`-Option kann das Verhalten während der Konvertierung ermitteln, wenn ein Zeichen nicht im Zielzeichensatz dargestellt werden kann.

Siehe auch

- „Datentypkonvertierungen“ auf Seite 146
- „Vergleiche zwischen CHAR und NCHAR“ auf Seite 141
- „on_charset_conversion_failure-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Vergleiche zwischen CHAR und NCHAR

Wenn ein Vergleich zwischen einem Wert des CHAR-Typs (CHAR, VARCHAR, LONG VARCHAR) und einem Wert des NCHAR-Typs (NCHAR, NVARCHAR, LONG NVARCHAR) vorgenommen wird, benutzt SQL Anywhere Inferenzregeln, um den Typ zu ermitteln, in dem der Vergleich erfolgen soll. Im Allgemeinen gilt: Wenn ein Wert auf einer Spaltenreferenz basiert und der andere nicht, wird der Vergleich in dem Typ des Werts vorgenommen, der die Spaltenreferenz enthält.

Die Inferenzregeln beschäftigen sich damit, ob ein Wert auf einer Spaltenreferenz basiert. Wenn ein Wert eine Variable, eine Hostvariable, eine Literalkonstante oder ein komplexer Ausdruck ist, der nicht auf einer Spaltenreferenz basiert, und der andere Wert auf einer Spaltenreferenz basiert, wird der konstantenbasierte Wert implizit auf den Typ des spaltenbasierten Werts angepasst.

Nachfolgend sind alle Inferenzregeln in der Reihenfolge aufgeführt, in der sie angewendet werden:

- Wenn der NCHAR-Wert auf einer Spaltenreferenz basiert, wird der CHAR-Wert implizit auf NCHAR angepasst und der Vergleich erfolgt als NCHAR. Dies umfasst auch den Fall, wenn sowohl der NCHAR- als auch der CHAR-Wert auf Spaltenreferenzen basieren.

- Sonst gilt: Wenn der NCHAR-Wert nicht auf einer Spaltenreferenz basiert und der CHAR-Wert auf einer Spaltenreferenz basiert, wird der NCHAR-Wert implizit auf CHAR angepasst und der Vergleich erfolgt als CHAR.

Es ist wichtig, die `on_charset_conversion_failure`-Option zu berücksichtigen, wenn NCHAR-Werte in CHAR konvertiert werden, da diese Option das Verhalten steuert, wenn ein NCHAR-Zeichen nicht im CHAR-Zeichensatz dargestellt werden kann.

- Wenn keiner der beiden Werte auf einer Spaltenreferenz basiert, wird der CHAR-Wert implizit auf NCHAR angepasst und der Vergleich erfolgt als NCHAR.

Beispiele

Die Bedingung `Employees.GivenName = N'Susan'` vergleicht eine CHAR-Spalte (`Employees.GivenName`) mit dem Literal `N'Susan'`. Der Wert `N'Susan'` wird an CHAR angepasst und der Vergleich wird so durchgeführt, als sei er folgendermaßen geschrieben:

```
Employees.GivenName = CAST( N'Susan' AS CHAR );
```

Alternativ ergibt die Bedingung `Employees.GivenName = T.nchar_column`, dass der `T.nchar_column`-Wert nicht in CHAR umgewandelt werden kann. In diesem Fall wird der Vergleich so ausgeführt, als sei er folgendermaßen geschrieben, wobei ein Index für `Employees.GivenName` nicht verwendet werden kann:

```
CAST( Employees.GivenName AS NCHAR ) = T.nchar_column;
```

Siehe auch

- „Konvertierungen von NCHAR in CHAR“ auf Seite 147
- „Konvertierungen von NCHAR in CHAR“ auf Seite 147
- „Verlustreiche Konvertierung und Ersetzungszeichen“ auf Seite 140
- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186
- „CONVERT-Funktion [Datentypkonvertierung]“ auf Seite 200
- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186
- „on_charset_conversion_failure-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Vergleiche zwischen numerischen Datentypen

SQL Anywhere führt Vergleiche zwischen numerischen Datentypen anhand der nachfolgenden Regeln durch. Die Regeln werden in der aufgeführten Reihenfolge durchsucht. Die erste zutreffende Regel wird angewendet:

1. Wenn ein Argument TINYINT und das andere INTEGER ist, werden beide in INTEGER konvertiert und verglichen.
2. Wenn ein Argument TINYINT und das andere SMALLINT ist, werden beide in SMALLINT konvertiert und verglichen.
3. Wenn ein Argument UNSIGNED SMALLINT und das andere INTEGER ist, werden beide in INTEGER konvertiert und verglichen.

4. Wenn die Datentypen der Argumente einen gemeinsamen übergeordneten Datentyp haben, wird in den übergeordneten Datentyp konvertiert und dann verglichen. Die übergeordneten Datentypen sind die jeweils zuletzt aufgeführten Datentypen der nachstehend genannten Datentypreihen:

- BIT » TINYINT » UNSIGNED SMALLINT » UNSIGNED INTEGER » UNSIGNED BIGINT » NUMERIC
- SMALLINT » INTEGER » BIGINT » NUMERIC
- REAL » DOUBLE
- CHAR » LONG VARCHAR
- BINARY » LONG BINARY

Wenn beispielsweise die beiden Argumente die Datentypen BIT und TINYINT haben, werden sie in NUMERIC konvertiert.

Vergleiche von Datumsangaben und Uhrzeiten

Die folgende Tabelle enthält eine Übersicht über die impliziten Konvertierungen beim Vergleich von bestimmten Datentypen mit Datums-, Uhrzeit- oder Datum-Uhrzeit-Datentypen.

Datentyp	Datentyp	Konvertierung
CHAR	DATE	CHAR in TIMESTAMP, DATE in TIMESTAMP
CHAR	TIME	CHAR in TIME
CHAR	TIMESTAMP	CHAR in TIMESTAMP
CHAR	TIMESTAMP WITH TIME ZONE	CHAR in TIMESTAMP WITH TIME ZONE
DATE	TIME	Unzulässig
DATE	TIMESTAMP	DATE in TIMESTAMP
DATE	TIMESTAMP WITH TIME ZONE	DATE in TIMESTAMP WITH TIME ZONE
DATE	SMALLINT, INTEGER, BIGINT und NUMERIC	SMALLINT-, INTEGER-, BIGINT- und NUMERIC-Werte werden als Datumszeichenfolge behandelt und in TIMESTAMP konvertiert, DATE in TIMESTAMP
DATE	REAL, FLOAT und DOUBLE	REAL-, FLOAT- und DOUBLE-Werte werden als Anzahl von Tagen seit 0000-02-29 behandelt und in TIMESTAMP konvertiert, DATE in TIMESTAMP

Datentyp	Datentyp	Konvertierung
TIME	TIMESTAMP	TIMESTAMP in TIME
TIME	TIMESTAMP WITH TIME ZONE	Unzulässig
TIME-STAMP	TIMESTAMP WITH TIME ZONE	TIMESTAMP in TIMESTAMP WITH TIME ZONE
TIME-STAMP	SMALLINT, INTEGER, BIGINT und NUMERIC	SMALLINT-, INTEGER-, BIGINT- und NUMERIC-Werte werden als Datumszeichenfolge behandelt und in TIMESTAMP konvertiert
TIME-STAMP	REAL, FLOAT und DOUBLE	REAL-, FLOAT- und DOUBLE-Werte werden als Anzahl von Tagen seit 0000-02-29 behandelt und in TIME-STAMP konvertiert

Die folgenden Punkte bauen auf den in der obigen Tabelle vorgestellten Informationen auf.

1. Nur Werte vom Typ TIME, TIMESTAMP und CHAR können mit einem Wert vom Typ TIME verglichen werden. Ein Vergleich mit Werten anderer Datentypen führt zu einem Konvertierungsfehler. Beim Vergleichen eines Uhrzeitwerts mit einem Wert eines anderen Datentyps ist der Vergleichsdatentyp TIME.
2. Beim Vergleichen eines Werts vom Typ TIMESTAMP, SMALLINT, INTEGER, BIGINT, NUMERIC, REAL, FLOAT oder DOUBLE mit einem DATE-Wert ist der Vergleichsdatentyp immer TIMESTAMP.
3. Beim Vergleichen eines TIMESTAMP WITH TIME ZONE-Werts mit einem DATE-Wert ist der Vergleichsdatentyp TIMESTAMP WITH TIME ZONE.
4. Wenn ein Zeitwert in einen TIMESTAMP-Wert konvertiert wird, ist das Ergebnis die Kombination aus dem aktuellen Datum und dem Zeitwert.
5. Exakte numerische Werte vom Typ SMALLINT, INTEGER, BIGINT oder NUMERIC können in Datumswerte konvertiert werden. Die Konvertierung erfolgt durch Behandlung der Zahl als Zeichenfolge. Beispiel: Der Ganzzahlwert "20100401" stellt den ersten Tag des Monats April im Jahr dar.
6. Die exakten numerischen Datentypen ohne Vorzeichen (BIT, TINYINT, UNSIGNED SMALLINT, UNSIGNED INTEGER und UNSIGNED BIGINT) können nicht in Datumswerte konvertiert werden.
7. Angenäherte numerische Werte vom Datentyp REAL, FLOAT oder DOUBLE können in Datumswert konvertiert werden kann, indem die Zahl als die Anzahl der Tage seit dem fiktiven Datum 0000-02-29 behandelt wird. Beispiel: 307 stellt den Datumswert "0001-01-01" dar und 734169 den Datumswert "2010-04-01".

Siehe auch

- „Datentypen für Datum und Uhrzeit“ auf Seite 116

Vergleiche von zusammengesetzten Typen

Array-Elemente werden verglichen, beginnend mit dem ersten Element. Wenn ein Unterschied gefunden wurde, wird der Vergleich gestoppt und das Ergebnis des Vergleichs zwischen den zuletzt verglichenen Elementen zurückgegeben. Wenn der Vergleich für alle Elemente Gleichheit ergibt, sind die Arrays gleich. Die ausgeführten Vergleiche entsprechen denjenigen für Ausdrücke, die nicht in Arrays gespeichert sind. Wenn ein Array kürzer ist als ein anderes und alle Elemente des kürzeren Arrays gleich den entsprechenden Elementen des längeren Arrays sind, lautet das Ergebnis, dass das kürzere Array kleiner ist als das längere Array.

Beim Vergleichen von Arrays müssen die Arrays Werte mit vereinigungskompatiblen Datentypen enthalten. Duplikateliminierung und GROUP BY werden für Array-Ausdrücke ebenfalls unterstützt. Beim folgenden Array-Vergleich gibt die Abfrage beispielsweise den Wert 1 zurück:

```
SELECT IF ARRAY(3,4,5) > ARRAY(2,3,4) THEN 1 ELSE 0 ENDIF;
```

Zeilentypen können verglichen sowie in Joins, zur Duplikateliminierung und zum Gruppieren verwendet werden. Die beiden folgenden Zeilentypen ähneln dem oben genannten Beispiel eines Zeilenausdrucks:

```
DECLARE test1 ROW(x INT, w ROW(y INT, z INT));
DECLARE test2 ROW(a INT, b ROW(c INT, d CHAR(3)));
SET test1 = ROW(3, ROW(6,7));
SET test2 = ROW(3, ROW(8,'7'));
SELECT (IF (test1 > test2) THEN 1 ELSE 0 ENDIF) AS RESULT FROM dummy;
```

Die beiden Zeilenausdrücke können nur verglichen werden, wenn ihre Strukturen übereinstimmen. Obwohl jedoch die Zeilenausdrücke dieselbe Struktur aufweisen müssen, brauchen die Namen der Attribute für einen Zeilentyp nicht identisch zu sein und auch die Datentypen der einzelnen Blattwerte müssen nicht identisch, sondern nur vereinigungskompatibel sein.

Alle ROW-Vergleiche mit Ausnahme von Gleichheits- und Ungleichheitsvorgängen liefern das Ergebnis UNKNOWN.

Transact-SQL - Konvertierung von Zeichenfolgen in Datums-/Zeitdatentypen

Wenn eine Zeichenfolge, die nur einen Zeitwert (keine Datumsangabe) enthält, in einen Datums-/Zeitdatentyp konvertiert wird, verwendet SQL Anywhere das aktuelle Datum.

Wenn der Sekundenbruchteilabschnitt einer Zeitangabe weniger als 3 Stellen hat, interpretiert SQL Anywhere den Wert auf die gleiche Art, unabhängig davon, ob vorher ein Punkt oder ein Doppelpunkt steht: Ein einstelliger Wert stellt eine Zehntelsekunde, ein zweistelliger Wert eine Hunderstelsekunde und ein dreistelliger Wert eine Tausendstelsekunde dar.

Beispiele

SQL Anywhere konvertiert den Millisekundenwert unabhängig vom Trennzeichen auf dieselbe Art und Weise.

```
12:34:56.7 bis 12:34:56.700
12:34:56:7 bis 12:34:56.700
12.34.56.78 bis 12:34:56.780
12.34.56:78 bis 12:34:56.780
12:34:56.789 bis 12:34:56.789
12:34:56:789 bis 12:34:56.789
```

Sonstige Vergleiche

1. Wenn die Datentypen verschiedene CHAR-Werte (z.B. CHAR, VARCHAR, LONG VARCHAR usw., jedoch keine NCHAR-Typen) umfassen, wird es in LONG VARCHAR konvertiert und verglichen.
2. Wenn der Datentyp eines Arguments UNIQUEIDENTIFIER ist, wird es in UNIQUEIDENTIFIER konvertiert und verglichen.
3. Wenn der Datentyp eines Arguments ein Bit-Array ist (VARBIT oder LONG VARBIT), wird es in LONG VARBIT konvertiert und dann verglichen.
4. Wenn ein Argument den CHARACTER-Datentyp und das andere den BINARY-Datentyp hat, werden beide in BINARY konvertiert und verglichen.
5. Wenn ein Argument ein CHAR-Typ und das andere ein NCHAR-Typ ist, werden vordefinierte Inferenzregeln verwendet.
6. Wenn keine Regel vorhanden ist, wird in NUMERIC konvertiert und verglichen.

Wenn beispielsweise die beiden Argumente vom Datentyp REAL und CHAR sind, werden sie beide in NUMERIC konvertiert.

Siehe auch

- [„Vergleiche zwischen CHAR und NCHAR“ auf Seite 141](#)

Datentypkonvertierungen

Konvertierungen von Datentypen können automatisch erfolgen oder mit der CAST- oder CONVERT-Funktion explizit erzwungen werden. Auch die nachstehenden Funktionen können verwendet werden, um die Konvertierung von Datentypen zu erzwingen:

- **DATE-Funktion** Konvertiert den Ausdruck in einen DATE-Wert und entfernt Stunden, Minuten oder Sekunden. Konvertierungsfehler können gemeldet werden.
- **DATETIME-Funktion** Konvertiert den Ausdruck in einen TIMESTAMP-Wert und entfernt die Zeitzone. Konvertierungsfehler können gemeldet werden.

- **STRING-Funktion** Diese Funktion ist äquivalent mit CAST(Wert AS LONG VARCHAR).
- **VALUE+0.0** Gleichwertig zu CAST(Wert AS DECIMAL)

Die folgende Liste ist eine allgemeine Übersicht zu automatischen Datentypkonvertierungen:

- Wenn eine Zeichenfolge in einem numerischen Ausdruck als Argument einer Funktion verwendet wird, die ein numerisches Argument erwartet, wird die Zeichenfolge in eine Zahl umgewandelt.
- Wenn eine Zahl in einem Zeichenfolgenausdruck oder als Zeichenfolgenargument verwendet wird, erfolgt vorher die Konvertierung in eine Zeichenfolge.
- Alle Datumskonstanten werden als Zeichenfolge angegeben. Die Zeichenfolge wird vor der Verwendung automatisch in ein Datum umgewandelt.

Es gibt Fälle, in denen automatische Datentypkonvertierungen nicht sinnvoll sind. Im nachfolgenden Beispiel schlägt die automatische Datentypkonvertierung fehl.

```
'12/31/90' + 5  
'a' > 0
```

Siehe auch

- „Funktionen zur Datentypkonvertierung“ auf Seite 156
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „STRING-Funktion [Zeichenfolge]“ auf Seite 398
- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186

Konvertierungen von NCHAR in CHAR

Konvertierungen von NCHAR in CHAR können als Teil eines Vergleichs zwischen CHAR- und NCHAR-Daten oder auf Anforderung erfolgen. Dieser Konvertierungstyp ist verlustreich, weil je nach dem CHAR-Zeichensatz möglicherweise einige NCHAR-Zeichen nicht im CHAR-Datentyp dargestellt werden können. Wenn ein NCHAR-Zeichen nicht in CHAR umgewandelt werden kann, wird anstelle dessen ein Ersetzungszeichen aus dem CHAR-Zeichensatz verwendet. Bei Einbyte-Zeichensätzen ist das üblicherweise 'hex 1A'.

Je nach Einstellung der `on_charset_conversion_failure`-Option tritt eine der folgenden Situationen ein, wenn ein Zeichen nicht konvertiert werden kann:

- Ein Ersetzungszeichen wird verwendet, ohne dass eine Warnung ausgegeben wird
- Ein Ersetzungszeichen wird verwendet und eine Warnung wird ausgegeben
- Ein Fehler wird ausgegeben

Aus diesem Grund empfiehlt es sich, diese Option zu berücksichtigen, wenn Sie NCHAR in CHAR konvertieren.

Siehe auch

- „Vergleiche zwischen CHAR und NCHAR“ auf Seite 141
- „on_charset_conversion_failure-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Konvertierung von NULL-Konstanten in NUMERIC- oder Zeichenfolgentypen

Beim Konvertieren einer NULL-Konstanten in einen NUMERIC- bzw. einen Zeichenfolgentyp wie CHAR, VARCHAR, LONG VARCHAR, BINARY, VARBINARY und LONG BINARY stellen Sie die Größe auf '0' ein. Zum Beispiel:

```
SELECT CAST( NULL AS CHAR ) gibt CHAR(0) zurück
```

```
SELECT CAST( NULL AS NUMERIC ) gibt NUMERIC(1,0) zurück
```

Konvertierungen von Bit-Arrays

Ganzzahlen in Bit-Arrays konvertieren

Wenn eine Ganzzahl in ein Bit-Array konvertiert wird, ist die Länge des Bit-Arrays die Anzahl von Bits im Ganzzahltyp, und der Wert des Bit-Arrays ist die binäre Darstellung. Das signifikanteste Bit der Ganzzahl wird zum ersten Bit im Array.

Beispiele

```
SELECT CAST( CAST( 1 AS BIT ) AS VARBIT ) gibt ein VARBIT(1) zurück, das '1' enthält.
```

```
SELECT CAST( CAST( 8 AS TINYINT ) AS VARBIT ) gibt ein VARBIT(8) zurück, das '00001000' enthält.
```

```
SELECT CAST( CAST( 194 AS INTEGER ) AS VARBIT ) gibt ein VARBIT(32) zurück, das '0000000000000000000000000000000011000010' enthält.
```

Binärwerte in Bit-Arrays konvertieren

Wenn ein Binärtyp der Länge n in ein Bit-Array konvertiert wird, beträgt die Länge des Arrays $n * 8$ Bit. Die ersten 8 Bits des Bit-Arrays werden zum ersten Byte des Binärwerts. Das signifikanteste Bit des Binärwerts wird zum ersten Bit im Array. Die nächsten 8 Bits des Bit-Arrays werden zum zweiten Byte des Binärwerts, usw.

Beispiele

```
SELECT CAST( 0x8181 AS VARBIT ) gibt ein VARBIT(16) zurück, das '1000000110000001' enthält.
```

Zeichen in Bit-Arrays konvertieren

Wenn ein Zeichendatentyp der Länge n in ein Bit-Array konvertiert wird, beträgt die Länge des Arrays n Bit. Jedes Zeichen muss '0' oder '1' sein, und dem entsprechenden Bit des Arrays wird der Wert "0" oder "1" zugeordnet.

Beispiel

SELECT CAST('001100' AS VARBIT) gibt ein VARBIT(6) zurück, das '001100' enthält.

Bit-Arrays in Ganzzahlen konvertieren

Wenn ein Bit-Array in einen Ganzzahl-Datentyp konvertiert wird, wird der Binärwert des Bit-Arrays anhand des Speicherformats des Ganzzahltyps interpretiert, wobei das signifikanteste Bit zuerst verwendet wird.

Beispiel

SELECT CAST(CAST('11000010' AS VARBIT) AS INTEGER) gibt 194 zurück
(11000010₂ = 0xC2 = 194).

Bit-Arrays in Binärwerte konvertieren

Wenn ein Bit-Array in einen Binärwert konvertiert wird, werden die ersten 8 Bits des Arrays zum ersten Byte des Binärwerts. Das erste Bit des Arrays wird zum signifikantesten Bit des Binärwerts. Die nächsten 8 Bits werden als zweites Byte verwendet, usw. Wenn die Länge des Bit-Arrays kein Vielfaches von 8 ist, werden zusätzliche Nullen verwendet, um die am wenigsten signifikante Bits im letzten Bytes des Binärwerts aufzufüllen.

Beispiele

SELECT CAST(CAST('1111' AS VARBIT) AS BINARY) gibt 0xF0 zurück (1111₂ wird 11110000₂ = 0xF0).

SELECT CAST(CAST('0011000000110001' AS VARBIT) AS BINARY) gibt 0x3031 zurück (0011000000110001₂ = 0x3031).

Bit-Arrays in Zeichen konvertieren

Wenn ein Bit-Array der Länge n in einen Zeichendatentyp konvertiert wird, beträgt die Länge des Ergebnisses n Zeichen. Jedes Zeichen im Ergebnis ist '0' oder '1', abhängig vom Bit im Array.

Beispiel

SELECT CAST(CAST('01110' AS VARBIT) AS VARCHAR) gibt die Zeichenfolge '01110' zurück.

Konvertierungen von numerischen Datentypen

Wenn ein DOUBLE-Typ in einen NUMERIC-Typ konvertiert wird, wird die Gesamtstellenzahl bei den ersten 15 signifikanten Stellen beibehalten.

Siehe auch

- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186
- „CONVERT-Funktion [Datentypkonvertierung]“ auf Seite 200

Konvertierung von Java- und SQL-Datentypen

Eine Datentypkonvertierung zwischen Java-Typen und SQL-Typen ist sowohl für gespeicherte Java-Prozeduren als auch für JDBC-Anwendungen erforderlich. Konvertierungen von Datentypen zwischen Java und SQL werden nach dem JDBC-Standard durchgeführt. Die Konvertierungen werden in den folgenden Tabellen beschrieben.

Konvertierung von Java- in SQL-Datentypen

Java-Datentyp	SQL-Datentyp
String	CHAR
String	VARCHAR
String	TEXT
java.math.BigDecimal	NUMERIC
java.math.BigDecimal	MONEY
java.math.BigDecimal	SMALLMONEY
boolean	BIT
byte	TINYINT
short	SMALLINT
int	INTEGER
long	BIGINT
float	REAL
double	DOUBLE
byte[]	VARBINARY
byte[]	IMAGE
java.sql.Date	DATE
java.sql.Time	TIME
java.sql.Timestamp	TIMESTAMP
java.lang.Double	DOUBLE

Java-Datentyp	SQL-Datentyp
java.lang.Float	REAL
java.lang.Integer	INTEGER
java.lang.Long	BIGINT

Konvertierung von SQL- in Java-Datentypen

SQL-Datentyp	Java-Datentyp
CHAR	String
VARCHAR	String
TEXT	String
NUMERIC	java.math.BigDecimal
DECIMAL	java.math.BigDecimal
MONEY	java.math.BigDecimal
SMALLMONEY	java.math.BigDecimal
UNSIGNED BIGINT	java.math.BigDecimal (precision=20, scale=0)
BIT	boolean
TINYINT	byte
SMALLINT	short
UNSIGNED SMALLINT	int
INTEGER	int
UNSIGNED INTEGER	long
BIGINT	long
REAL	float
FLOAT	double
DOUBLE	double

SQL-Datentyp	Java-Datentyp
BINARY	byte[]
VARBINARY	byte[]
LONG BINARY	byte[]
IMAGE	byte[]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp

SQL-Funktionen

Funktionen werden zur Rückgabe von Informationen aus der Datenbank verwendet. Sie können überall dort aufgerufen werden, wo ein Ausdruck erlaubt ist.

Wenn in der Dokumentation nichts anderes angegeben ist, wird NULL für eine Funktion zurückgegeben, wenn eines der Argumente NULL ist.

Wenn ein Argument optional ist, kann DEFAULT als Argument angegeben werden.

Funktionen verwenden dieselben Syntaxkonventionen wie SQL-Anweisungen.

Siehe auch

- [„Syntaxkonventionen“ auf Seite 446](#)

Funktionstypen

In diesem Abschnitt werden die verfügbaren Funktionen nach ihrem Typ gruppiert.

Siehe auch

- [„ST_GEOMETRY-Datentyp“ \[*UltraLite - Datenbankverwaltung*\]](#)

Aggregatfunktionen

Aggregatfunktionen fassen Daten über eine Gruppe von Zeilen in der Datenbank zusammen. Die Gruppen werden durch die GROUP BY-Klausel der SELECT-Anweisung gebildet. Aggregatfunktionen sind nur in der Auswahlliste und in den Klauseln HAVING und ORDER BY einer SELECT-Anweisung erlaubt.

Liste der Funktionen

Folgende Aggregatfunktionen stehen zur Verfügung:

- „ARRAY_AGG-Funktion [Aggregat]“ auf Seite 170
- „AVG-Funktion [Aggregat]“
- „BIT_AND-Funktion [Aggregat]“
- „BIT_OR-Funktion [Aggregat]“
- „BIT_XOR-Funktion [Aggregat]“
- „COVAR_POP-Funktion [Aggregat]“
- „COVAR_SAMP-Funktion [Aggregat]“
- „COUNT-Funktion [Aggregat]“
- „COUNT_BIG-Funktion [Aggregat]“
- „CORR-Funktion [Aggregat]“
- „FIRST_VALUE-Funktion [Aggregat]“
- „GROUPING-Funktion [Aggregat]“
- „LAST_VALUE-Funktion [Aggregat]“
- „LIST-Funktion [Aggregat]“
- „MAX-Funktion [Aggregat]“
- „MEDIAN-Funktion [Aggregat]“
- „MIN-Funktion [Aggregat]“
- „REGR_AVGX-Funktion [Aggregat]“
- „REGR_AVGY-Funktion [Aggregat]“
- „REGR_COUNT-Funktion [Aggregat]“
- „REGR_INTERCEPT-Funktion [Aggregat]“
- „REGR_R2-Funktion [Aggregat]“
- „REGR_SLOPE-Funktion [Aggregat]“
- „REGR_SXX-Funktion [Aggregat]“
- „REGR_SXY-Funktion [Aggregat]“
- „REGR_SYY-Funktion [Aggregat]“
- „SET_BITS-Funktion [Aggregat]“
- „STDDEV-Funktion [Aggregat]“
- „STDDEV_POP-Funktion [Aggregat]“
- „STDDEV_SAMP-Funktion [Aggregat]“
- „SUM-Funktion [Aggregat]“
- „VAR_POP-Funktion [Aggregat]“
- „VAR_SAMP-Funktion [Aggregat]“
- „VARIANCE-Funktion [Aggregat]“
- „XMLAGG-Funktion [Aggregat]“

Zusammengesetzte Funktionen

Mit zusammengesetzten Funktionen können Sie Aufgaben für Arrays ausführen.

Liste der Funktionen

Die folgenden zusammengesetzten Funktionen sind verfügbar:

- „ARRAY-Konstruktor [zusammengesetzt]“ auf Seite 168
- „ROW-Konstruktor [zusammengesetzt]“ auf Seite 371
- „ARRAY_MAX_CARDINALITY-Funktion [zusammengesetzt]“ auf Seite 171
- „CARDINALITY-Funktion [zusammengesetzt]“ auf Seite 185
- „TRIM_ARRAY-Funktion [zusammengesetzt]“ auf Seite 416

Siehe auch

- „ARRAY_AGG-Funktion [Aggregat]“ auf Seite 170
- „Array-Operator UNNEST“

Bit-Array-Funktionen

Mit Bit-Array-Funktionen können Sie Aufgaben auf Bit-Arrays durchführen.

Liste der Funktionen

Folgende Bit-Array-Funktionen sind verfügbar:

- „BIT_AND-Funktion [Aggregat]“
- „BIT_OR-Funktion [Aggregat]“
- „BIT_XOR-Funktion [Aggregat]“
- „BIT_LENGTH-Funktion [Bit-Array]“
- „BIT_SUBSTR-Funktion [Bit-Array]“
- „COUNT_SET_BITS-Funktion [Bit-Array]“
- „GET_BIT-Funktion [Bit-Array]“
- „SET_BIT-Funktion [Bit-Array]“
- „SET_BITS-Funktion [Aggregat]“

Siehe auch

- „Bit-Operatoren“ auf Seite 20
- „sa_get_bits-Systemprozedur“ auf Seite 1221

Rangfolgefunktionen

Mit Rangfolgefunktionen können Sie einen Rangwert für jede Zeile in einer Ergebnismenge basierend auf einer in der Abfrage festgelegten Rangordnung berechnen.

Liste der Funktionen

Die folgenden Rangfunktionen sind verfügbar:

- „CUME_DIST-Funktion [Rangfolge]“
- „DENSE_RANK-Funktion [Rangfolge]“
- „PERCENT_RANK-Funktion [Rangfolge]“
- „RANK-Funktion [Rangfolge]“

Funktionen zur Datentypkonvertierung

Datentypkonvertierungsfunktionen werden benutzt, um Argumente von einem Datentyp in einen anderen zu konvertieren oder um zu testen, ob eine Konvertierung möglich ist.

Liste der Funktionen

Die folgenden Datentypkonvertierungsfunktionen sind verfügbar:

- „BINTOHEX-Funktion [Datentypkonvertierung]“ auf Seite 178
- „CAST-Funktion [Datentypkonvertierung]“
- „CONVERT-Funktion [Datentypkonvertierung]“
- „HEXTOBIN-Funktion [Datentypkonvertierung]“ auf Seite 274
- „HEXTOINT-Funktion [Datentypkonvertierung]“
- „INTTOHEX-Funktion [Datentypkonvertierung]“
- „ISDATE-Funktion [Datentypkonvertierung]“
- „ISNUMERIC-Funktion [Verschiedene]“
- „TREAT-Funktion [Datentypkonvertierung]“

Datums- und Zeitfunktionen

Datums- und Uhrzeitfunktionen führen Vorgänge mit den Datentypen DATE, TIME, TIMESTAMP, und TIMESTAMP WITH TIME ZONE aus.

SQL Anywhere umfasst Kompatibilitätsunterstützung für Datums und Zeittypen von Transact-SQL, einschließlich DATETIME und SMALLDATETIME. Diese Transact-SQL-Datentypen werden als Domänen über den nativen SQL Anywhere-Datentyp TIMESTAMP implementiert.

Angeben von Datumsteilen

Viele Datumsfunktionen benutzen Datumsangaben, die aus **Datumsteilen** zusammengesetzt sind. Die folgende Tabelle zeigt zulässige Werte der Datumsteile.

Bei Datums- und Zeitfunktionen können Sie ein Minuszeichen angeben, um von einem Datum oder einer Uhrzeit zu subtrahieren Sie können z.B. Folgendes ausführen, um einen Zeitstempel von vor 31 Tagen zu erzeugen:

```
SELECT DATEADD(day, -31, NOW());
```

Datumsteil	Abkürzung	Werte
Year	YY	1-9999
Quarter	QQ	1-4
Month	MM	1-12
Week	WK	1-54. Wochen beginnen am Sonntag. Ein Jahr mit 54 Wochen tritt in Schaltjahren auf, die an einem Samstag beginnen.
Day	DD	1-31
Dayofyear	DY	1-366
Weekday	DW	1-7 (Sonntag = 1, ..., Samstag = 7)
Hour	HH	0-23
Minute	MI	0-59
Second	SS	0-59
Millisecond	MS	0-999
Microsecond	MCS oder US	0-999999
Calyearofweek	CYR	1-9999. Das Jahr, in dem die Woche beginnt. Die Woche mit den ersten Tagen des neuen Jahres beginnt möglicherweise schon im Vorjahr, je nachdem, an welchem Wochentag das Jahr beginnt. Jahre, die von Montag bis Donnerstag beginnen, haben keine Tage, die Teil des vorhergehenden Jahres sind, aber Jahre, die von Freitag bis Sonntag beginnen, beginnen ihre erste Woche am ersten Montag des Jahres.
Calweekofyear	CWK	1-53. Die Nummer der Kalenderwoche im Jahr, die das angegebene Datum enthält. Weitere Hinweise zum ISO-Wochensystem und zum Datums- und Zeitstandard ISO 8601 finden Sie unter http://en.wikipedia.org/wiki/ISO_week_date .
Caldayofweek	CDW	1-7. (Montag = 1, ..., Sonntag = 7)
TZOffset	TZ	-840 bis 840

Liste der Funktionen

Die folgenden Datums- und Zeitfunktionen stehen zur Verfügung:

- „DATE-Funktion [Datum und Uhrzeit]“
- „DATEADD-Funktion [Datum und Uhrzeit]“
- „DATEDIFF-Funktion [Datum und Uhrzeit]“
- „DATEFORMAT-Funktion [Datum und Uhrzeit]“
- „DATENAME-Funktion [Datum und Uhrzeit]“
- „DATEPART-Funktion [Datum und Uhrzeit]“
- „DATETIME-Funktion [Datum und Uhrzeit]“
- „DAY-Funktion [Datum und Uhrzeit]“
- „DAYNAME-Funktion [Datum und Uhrzeit]“
- „DAYS-Funktion [Datum und Uhrzeit]“
- „DOW-Funktion [Datum und Uhrzeit]“
- „GETDATE-Funktion [Datum und Uhrzeit]“
- „HOUR-Funktion [Datum und Uhrzeit]“
- „HOURS-Funktion [Datum und Uhrzeit]“
- „MINUTE-Funktion [Datum und Uhrzeit]“
- „MINUTES-Funktion [Datum und Uhrzeit]“
- „MONTH-Funktion [Datum und Uhrzeit]“
- „MONTHNAME-Funktion [Datum und Uhrzeit]“
- „MONTHS-Funktion [Datum und Uhrzeit]“
- „NOW-Funktion [Datum und Uhrzeit]“
- „QUARTER-Funktion [Datum und Uhrzeit]“
- „SECOND-Funktion [Datum und Uhrzeit]“
- „SECONDS-Funktion [Datum und Uhrzeit]“
- „SWITCHOFFSET-Funktion [Datum und Zeit]“
- „SYSDATETIMEOFFSET-Funktion [Datum und Zeit]“
- „TODAY-Funktion [Datum und Uhrzeit]“
- „TODATETIMEOFFSET-Funktion [Datum und Zeit]“
- „WEEKS-Funktion [Datum und Uhrzeit]“
- „YEAR-Funktion [Datum und Uhrzeit]“
- „YEARS-Funktion [Datum und Uhrzeit]“
- „YMD-Funktion [Datum und Uhrzeit]“

Siehe auch

- „Datentypen für Datum und Uhrzeit“ auf Seite 116
- „UltraLite, SQLDatentypen“ [*UltraLite - Datenbankverwaltung*]

Benutzerdefinierte Funktionen

Eine benutzerdefinierte Funktion (auch UDF für User Defined Function) wird von einem Benutzer eines Programms oder einer Umgebung erstellt. Benutzerdefinierte Funktionen unterscheiden sich von Funktionen, die in das Programm oder die Umgebung integriert sind.

Es gibt zwei Mechanismen zum Erstellen von benutzerdefinierten Funktionen in SQL Anywhere. Sie können SQL oder eine beliebige CLR-Sprache verwenden, um die Funktion zu schreiben.

Benutzerdefinierte Funktionen in SQL

Mithilfe der CREATE FUNCTION-Anweisung können Sie Ihre eigenen Funktionen in SQL implementieren.

Die RETURN-Anweisung innerhalb der CREATE FUNCTION-Anweisung legt den Datentyp der Funktion fest.

Sobald eine benutzerdefinierte SQL-Funktion erstellt ist, kann sie überall dort verwendet werden, wo eine integrierte Funktion desselben Datentyps verwendet werden kann.

Benutzerdefinierte Funktionen in Java und CLR

Java-Klassen bieten eine wirksamere und flexible Möglichkeit, benutzerdefinierte Funktionen zu implementieren, mit dem zusätzlichen Vorteil, dass sie ggf. vom Datenbankserver zu einer Clientanwendung verschoben werden können. Jede Klassen-Methode einer installierten Java-Klasse kann überall dort als benutzerdefinierte Funktion verwendet werden, wo eine integrierte Funktion desselben Datentyps verwendet werden kann. Instanzmethoden sind an bestimmte Instanzen einer Klasse gebunden und haben deshalb ein anderes Verhalten als standardmäßige benutzerdefinierte Funktionen.

SQL Anywhere stellt Unterstützung für gespeicherte CLR-Prozeduren und -Funktionen bereit. Eine gespeicherte CLR-Prozedur oder -Funktion verhält sich wie eine gespeicherte SQL-Prozedur oder -Funktion, abgesehen davon, dass der Code für die Prozedur oder Funktion in einer .NET-Sprache wie C# oder Visual Basic geschrieben ist und die Ausführung der Prozedur oder Funktion außerhalb des Datenbankservers stattfindet (d.h. innerhalb eines separaten .NET-Programms). Es wird nur .NET Version 2.0 unterstützt.

Entscheidung, ob eine benutzerdefinierte Funktion oder Prozedur erstellt wird

Funktionen sind ähnlich Prozeduren. Eine Entscheidung darüber, ob eine Funktion oder eine Prozedur erstellt wird, hängt davon ab, welche Rückgabe Sie erwarten und welches Objekt abgerufen wird. Wenn Sie darüber entscheiden, ob eine UDF oder eine Prozedur erstellt wird, beachten Sie die nachstehenden Hinweise zu ihren Besonderheiten.

Funktionen:

- Können einen einzelnen Wert eines beliebigen Typs zurückgeben und ermöglichen es Ihnen, den zurückgegebenen Typ mit der RETURNS-Klausel zu deklarieren.
- Können an den meisten Stellen verwendet werden, an denen auch Ausdrücke zulässig sind.
- Ermöglichen nur die Definition von IN-Parametern.

Procedures:

- Können mehrfache Werte mit INOUT- oder OUT-Parametern zurückgeben.
- Können Ergebnismengen zurückgeben.
- Können in der FROM-Klausel einer Abfrage oder mit einer CALL-Anweisung bzw. einer Transact-SQL EXECUTE-Anweisung referenziert werden.

- Können mit benannten Parametern aufgerufen werden.

Siehe auch

- „CREATE FUNCTION-Anweisung“ auf Seite 633
- „Erstellung von Klassendateien“ [*SQL Anywhere Server - Programmierung*]
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Benannte Parameter“ auf Seite 93
- „Die externe CLR-Umgebung“ [*SQL Anywhere Server - Programmierung*]

Verschiedene Funktionen

Verschiedene Funktionen führen Vorgänge auf arithmetischen, Zeichenfolge- oder Datum/Zeit-Ausdrücken, einschließlich der Rückgabewerte anderer Funktionen, aus.

Liste der Funktionen

Folgende verschiedene Funktionen stehen zur Verfügung:

- „ARGN-Funktion [Verschiedene]“
- „COALESCE-Funktion [Verschiedene]“
- „CONFLICT-Funktion [Verschiedene]“
- „ERRORMSG-Funktion [Verschiedene]“
- „ESTIMATE-Funktion [Verschiedene]“
- „ESTIMATE_SOURCE-Funktion [Verschiedene]“
- „EXPERIENCE_ESTIMATE-Funktion [Verschiedene]“
- „EXPLANATION-Funktion [Verschiedene]“
- „EXPRTYPE-Funktion [Verschiedene]“
- „GET_IDENTITY-Funktion [Verschiedene]“
- „GRAPHICAL_PLAN-Funktion [Verschiedene]“
- „GREATER-Funktion [Verschiedene]“
- „IDENTITY-Funktion [Verschiedene]“
- „IFNULL-Funktion [Verschiedene]“
- „INDEX_ESTIMATE-Funktion [Verschiedene]“
- „ISNULL-Funktion [Verschiedene]“
- „LESSER-Funktion [Verschiedene]“
- „NEWID-Funktion [Verschiedene]“
- „NULLIF-Funktion [Verschiedene]“
- „NUMBER-Funktion [Verschiedene]“
- „PLAN-Funktion [Verschiedene]“
- „REWRITE-Funktion [Verschiedene]“
- „ROW_NUMBER-Funktion [Verschiedene]“
- „SQLDIALECT-Funktion [Verschiedene]“
- „SQLFLAGGER-Funktion [Verschiedene]“
- „ERROR_LINE-Funktion [Verschiedene]“ auf Seite 244
- „TRACEBACK-Funktion [Verschiedene]“
- „TRANSACTSQL-Funktion [Verschiedene]“
- „VAREXISTS-Funktion [Verschiedene]“
- „WATCOMSQL-Funktion [Verschiedene]“

Nummerische Funktionen

Nummerische Funktionen führen mathematische Vorgänge auf numerischen Datentypen aus oder geben numerische Informationen zurück.

Liste der Funktionen

Die folgenden numerischen Funktionen stehen zur Verfügung:

- „ABS-Funktion [Numerisch]“
- „ACOS-Funktion [Numerisch]“
- „ASIN-Funktion [Numerisch]“
- „ATAN-Funktion [Numerisch]“
- „ATAN2-Funktion [Numerisch]“
- „CEILING-Funktion [Numerisch]“
- „COS-Funktion [Numerisch]“
- „COT-Funktion [Numerisch]“
- „DEGREES-Funktion [Numerisch]“
- „EXP-Funktion [Numerisch]“
- „FLOOR-Funktion [Numerisch]“
- „LOG-Funktion [Numerisch]“
- „LOG10-Funktion [Numerisch]“
- „MOD-Funktion [Numerisch]“
- „PI-Funktion [Numerisch]“
- „POWER-Funktion [Numerisch]“
- „RADIANS-Funktion [Numerisch]“
- „RAND-Funktion [Numerisch]“
- „REMAINDER-Funktion [Numerisch]“
- „ROUND-Funktion [Numerisch]“
- „SIGN-Funktion [Numerisch]“
- „SIN-Funktion [Numerisch]“
- „SQRT-Funktion [Numerisch]“
- „TAN-Funktion [Numerisch]“
- „TRUNCNUM-Funktion [Numerisch]“

Webdienstfunktionen

HTTP-Funktionen sind beim Umgang mit HTTP-Anforderungen innerhalb von Webdiensten hilfreich. Auf ähnliche Weise sind SOAP-Funktionen beim Umgang mit SOAP-Anforderungen innerhalb von Webdiensten hilfreich.

Folgende Funktionen sind verfügbar:

- „HTML_DECODE-Funktion [Verschiedene]“
- „HTML_ENCODE-Funktion [Verschiedene]“
- „HTTP_BODY-Funktion [Webdienst]“
- „HTTP_DECODE-Funktion [Webdienst]“
- „HTTP_ENCODE-Funktion [Webdienst]“
- „HTTP_HEADER-Funktion [Webdienst]“
- „HTTP_RESPONSE_HEADER-Funktion [Webdienst]“
- „HTTP_VARIABLE-Funktion [Webdienst]“
- „NEXT_HTTP_HEADER-Funktion [Webdienst]“
- „NEXT_HTTP_RESPONSE_HEADER-Funktion [Webdienst]“
- „NEXT_HTTP_VARIABLE-Funktion [Webdienst]“
- „NEXT_SOAP_HEADER-Funktion [SOAP]“
- „SOAP_HEADER-Funktion [SOAP]“

Es stehen auch Systemprozeduren für Webdienste zur Verfügung.

Siehe auch

- „SQL Anywhere als HTTP-Webserver“ [*SQL Anywhere Server - Programmierung*]
- „Datenbankserveroption -xs“ [*SQL Anywhere Server - Datenbankadministration*]
- „Webdienst-Systemprozeduren“ auf Seite 1162

Zeichenfolgenfunktionen

Zeichenfolgenfunktionen führen Konvertierungs-, Extraktions- oder Manipulationsvorgänge auf Zeichenfolgen aus oder geben Informationen über Zeichenfolgen zurück.

Prüfen Sie beim Arbeiten in einem Mehrzeichen-Zeichensatz sorgfältig, ob die verwendete Funktion Informationen über Zeichen oder Byte zurückgibt.

Liste der Funktionen

Folgende Zeichenfolgenfunktionen stehen zur Verfügung: Systemfunktionen

- „ASCII-Funktion [Zeichenfolge]“
- „BASE64_DECODE-Funktion [Zeichenfolge]“
- „BASE64_ENCODE-Funktion [Zeichenfolge]“
- „BYTE_LENGTH-Funktion [Zeichenfolge]“
- „BYTE_SUBSTR-Funktion [Zeichenfolge]“
- „CHAR-Funktion [Zeichenfolge]“
- „CHARINDEX-Funktion [Zeichenfolge]“
- „CHAR_LENGTH-Funktion [Zeichenfolge]“
- „COMPARE-Funktion [Zeichenfolge]“
- „COMPRESS-Funktion [Zeichenfolge]“
- „CS_CONVERT-Funktion [Zeichenfolge]“
- „DECOMPRESS-Funktion [Zeichenfolge]“
- „DECRYPT-Funktion [Zeichenfolge]“
- „DIFFERENCE-Funktion [Zeichenfolge]“
- „ENCRYPT-Funktion [Zeichenfolge]“
- „HASH-Funktion [Zeichenfolge]“
- „INSERTSTR-Funktion [Zeichenfolge]“
- „LCASE-Funktion [Zeichenfolge]“
- „LEFT-Funktion [Zeichenfolge]“
- „LENGTH-Funktion [Zeichenfolge]“
- „LOCATE-Funktion [Zeichenfolge]“
- „LOWER-Funktion [Zeichenfolge]“
- „LTRIM-Funktion [Zeichenfolge]“
- „NCHAR-Funktion [Zeichenfolge]“
- „PATINDEX-Funktion [Zeichenfolge]“
- „READ_CLIENT_FILE-Funktion“
- „REGEXP_SUBSTR-Funktion [Zeichenfolge]“
- „REPEAT-Funktion [Zeichenfolge]“
- „REPLACE-Funktion [Zeichenfolge]“
- „REPLICATE-Funktion [Zeichenfolge]“
- „REVERSE-Funktion [Zeichenfolge]“
- „RIGHT-Funktion [Zeichenfolge]“
- „RTRIM-Funktion [Zeichenfolge]“
- „SIMILAR-Funktion [Zeichenfolge]“
- „SORTKEY-Funktion [Zeichenfolge]“
- „SOUNDEX-Funktion [Zeichenfolge]“
- „SPACE-Funktion [Zeichenfolge]“
- „STR-Funktion [Zeichenfolge]“
- „STRING-Funktion [Zeichenfolge]“
- „STRTOUUID-Funktion [Zeichenfolge]“
- „STUFF-Funktion [Zeichenfolge]“
- „SUBSTRING-Funktion [Zeichenfolge]“
- „TO_CHAR-Funktion [Zeichenfolge]“
- „TO_NCHAR-Funktion [Zeichenfolge]“

- „TRIM-Funktion [Zeichenfolge]“
- „UCASE-Funktion [Zeichenfolge]“
- „UNICODE-Funktion [Zeichenfolge]“
- „UNISTR-Funktion [Zeichenfolge]“
- „UPPER-Funktion [Zeichenfolge]“
- „UIDTOSTR-Funktion [Zeichenfolge]“
- „XMLCONCAT-Funktion [Zeichenfolge]“
- „XMLELEMENT-Funktion [Zeichenfolge]“
- „XMLFOREST-Funktion [Zeichenfolge]“
- „XMLGEN-Funktion [Zeichenfolge]“

Systemfunktionen

Systemfunktionen geben Systeminformationen zurück.

Liste der Funktionen

Die folgenden Systemfunktionen stehen zur Verfügung:

- „CONNECTION_EXTENDED_PROPERTY-Funktion [Zeichenfolge]“
- „CONNECTION_PROPERTY-Funktion [System]“
- „DATALENGTH-Funktion [System]“
- „DB_ID-Funktion [System]“
- „DB_NAME-Funktion [System]“
- „DB_EXTENDED_PROPERTY-Funktion [System]“
- „DB_PROPERTY-Funktion [System]“
- „EVENT_CONDITION-Funktion [System]“
- „EVENT_CONDITION_NAME-Funktion [System]“
- „EVENT_PARAMETER-Funktion [System]“
- „NEXT_CONNECTION-Funktion [System]“
- „NEXT_DATABASE-Funktion [System]“
- „PROPERTY-Funktion [System]“
- „PROPERTY_DESCRIPTION-Funktion [System]“
- „PROPERTY_NAME-Funktion [System]“
- „PROPERTY_NUMBER-Funktion [System]“
- „USER_ID-Funktion [System]“
- „USER_NAME-Funktion [System]“
- „TSEQUAL-Funktion [System] (nicht mehr empfohlen)“
- „USER_ID-Funktion [System]“
- „USER_NAME-Funktion [System]“

Hinweise

- Die Funktionen db_id, db_name und datalength sind als integrierte Funktionen implementiert.
- Einige Systemfunktionen sind in SQL Anywhere als gespeicherte Prozeduren implementiert.

Systemfunktionen, die nicht an anderer Stelle beschrieben werden, sind in der folgenden Tabelle aufgeführt.

Systemfunktion	Beschreibung
COL_LENGTH (@object_name, @column_name)	Gibt die als INTEGER definierte Länge der angegebenen Spalte zurück
COL_NAME (@object_id, @column_id [, @database_id])	Gibt den CHAR(128)-Spaltennamen zurück
INDEX_COL (@table_name, @index_id, @key_# [, @user_id])	Gibt den CHAR(128)-Namen der indizierten Spalte zurück
OBJECT_ID (@object_name)	Gibt die INTEGER-Objekt-ID zurück
OBJECT_NAME (@object_id [, @database_id])	Gibt den CHAR(128)-Objektnamen zurück

Text- und Bilddatenfunktionen

Text- und Bilddatenfunktionen bearbeiten TEXT- und IMAGE-Datentypen. SQL Anywhere unterstützt nur die Text- und Bilddatenfunktion textptr.

Liste der Funktionen

Die folgende Text- und Bilddatenfunktion ist verfügbar:

- [„TEXTPTR-Funktion \[Text und Bild\]“](#)

Funktionen

Jede Funktion ist mit ihrem Funktionstyp (numerisch, Zeichen, etc.) aufgelistet.

Verknüpfungen zu Funktionen eines gegebenen Typs finden Sie unter [„Funktionstypen“](#) auf Seite 153.

ABS-Funktion [Numerisch]

Gibt den absoluten Wert eines numerischen Ausdrucks zurück.

Syntax

ABS(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Die Zahl, deren absoluter Wert zurückgegeben werden soll

Rückgabe

Ein absoluter Wert des numerischen Ausdrucks.

Datentyp numerischer Ausdruck	Rückgabe
INT	INT
FLOAT	FLOAT
DOUBLE	DOUBLE
NUMERIC	NUMERIC

Standards und Kompatibilität

- **SQL/2008** Die ABS-Funktion ist Teil der optionalen SQL/2008-Sprachenfunktion T441.

Beispiel

Die folgende Anweisung gibt den Wert 66 zurück:

```
SELECT ABS( -66 );
```

ACOS-Funktion [Nummerisch]

Gibt den Arkuskosinus eines numerischen Ausdrucks im Bogenmaß zurück.

Syntax

ACOS(*numeric-expression*)

Parameter

- **Nummerischer_Ausdruck** Der Kosinus des Winkels.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch.

Siehe auch

- „ASIN-Funktion [Nummerisch]“ auf Seite 173
- „ATAN-Funktion [Nummerisch]“ auf Seite 173
- „ATAN2-Funktion [Nummerisch]“ auf Seite 174
- „COS-Funktion [Nummerisch]“ auf Seite 204

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Arkuskosinus-Wert für 0,52 zurück:

```
SELECT ACOS( 0.52 );
```

ARGN-Funktion [Verschiedene]

Gibt ein ausgewähltes Argument aus einer Argumentliste zurück.

Syntax

ARGN(*integer-expression*, *expression* [, ...])

Parameter

- **Ganzzahlausdruck** Die Position eines Arguments in der Liste der Ausdrücke.
- **expression** Ein Ausdruck eines beliebigen Datentyps, der an die Funktion übergeben wird. Alle übergebenen Ausdrücke müssen denselben Datentyp haben.

Rückgabe

Wenn der Wert von *Ganzzahlausdruck* n ist, wird das n-te Argument (beginnend bei 1) aus der verbleibenden Argumentliste zurückgegeben.

Bemerkungen

Für die Ausdrücke gibt es zwar keine Datentypbeschränkung, allerdings müssen alle den gleichen Datentyp haben. Der Ganzzahlausdruck muss zwischen eins und der Elementanzahl der Liste der Ausdrücke sein, weil sonst NULL zurückgegeben wird. Mehrfache Ausdrücke werden durch ein Komma getrennt.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 6 zurück:

```
SELECT ARGN( 6, 1,2,3,4,5,6 );
```

ARRAY-Konstruktor [zusammengesetzt]

Gibt Elemente eines bestimmten Datentyps zurück.

Syntax 1

ARRAY(*expression* [, *expression* ...])

Syntax 2

ARRAY(*single-column-query-expression*)

Parameter

- **Ausdruck** Ein Elementausdruck im ROW-Typ.
- **Einspaltiger_Abfrageausdruck** Eine Abfrage, die eine einzige Spalte zurückgibt.

Rückgabe

Array-Wert

Bemerkungen

Alle Ausdrücke müssen vereinigungskompatibel sein.

Alle Elemente werden auf NULL initialisiert und bleiben NULL, bis ein Wert implizit oder explizit innerhalb eines bestimmten Array-Elements platziert wird.

Ein ARRAY-Typ kann andere ARRAY- oder ROW-Werte enthalten oder Teil eines ROW-Typs sein.

Die FETCH-Anweisung unterstützt die Übertragung von Werten in ein Array. Sie können für einzelne Ausdrücke, für ganze Arrays oder für Teile von Arrays Werte in ein Array abrufen.

Bestimmte Werte oder Vektoren von Werten können mithilfe von doppelten eckigen Klammern entreferenziert werden.

Siehe auch

- „Zusammengesetzte Datentypen“ auf Seite 136
- „Zusammengesetzte Funktionen“ auf Seite 154
- „Vergleiche von zusammengesetzten Typen“ auf Seite 145
- „FETCH-Anweisung [ESQL] [SP]“ auf Seite 853

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Oracle** VARRAY kann als Synonym für ARRAY verwendet werden.

Beispiel

Das folgende Beispiel veranschaulicht, wie ein Array konstruiert wird:

```
SELECT FIRST f[[2]] FROM ( SELECT ARRAY( ID,Quantity ) FROM GROUPO.Products )
      AS dt( f ) ORDER BY f[[1]] ASC;
```

In diesem Beispiel erstellt der ARRAY-Konstruktor für jede Zeile der Tabelle "Products" einen ARRAY-Typ mit zwei Elementen, der Spalten-ID und dem Wert der Quantity-Spalte. Beide Elemente sind Ganzzahlen. Das Ergebnis wird nach dem ersten Element des Arrays in jeder Zeile sortiert und das zurückgegebene Ergebnis ist das zweite Element aus dem Array, dessen erstes Element am kleinsten ist (Produkt-ID 300). Sie können ein Array auch direkt aus einem einspaltigen Abfrageausdruck konstruieren.

Das folgende Beispiel veranschaulicht eine alternative Methode zum Konstruieren eines Arrays:

```
SELECT * FROM GROUPO.SalesOrders S WHERE ARRAY( SELECT P.ID FROM
GROUPO.Products P JOIN GROUPO.SalesOrderItems SI ON( P.ID =
```

```
SI.ProductID )AND SI.ID = S.ID
ORDER BY P.ID )< ARRAY ( SELECT ID FROM GROUPO.Products ORDER BY ID );
```

Im folgenden Beispiel werden in der SELECT-Liste der Abfrage drei Arrays verwendet: Eine erzeugt einen GROUP BY-Ausdruck und die MAX-Funktion verwendet die anderen. Jeder ARRAY-Typ wird für ein bestimmtes Element entreferenziert, bevor das Ergebnis an den Client zurückgegeben wird:

```
SELECT FIRST ARRAY( Quantity )[[1]], MAX( ARRAY( ID,Quantity ) )[[1]],
       MAX( ARRAY( name,name ) )[[2]] FROM Products GROUP BY ARRAY( Quantity )
ORDER BY 1;
```

Das folgende Beispiel veranschaulicht, wie die FETCH-Anweisung verwendet werden kann, um Werte in ein Array zu übertragen:

```
BEGIN
DECLARE product_orders ARRAY(10) OF ARRAY OF INTEGER;
DECLARE products ARRAY(10) OF INTEGER;
DECLARE greatest_orders INTEGER = 0;
DECLARE i INTEGER = 1;
DECLARE curs CURSOR FOR
    SELECT ProductID,
    ARRAY_AGG( Quantity ) AS Quantities
    FROM GROUPO.SalesOrderItems
    GROUP BY ProductID
    ORDER BY ProductID;

OPEN curs;
lp: LOOP
    FETCH NEXT curs INTO products[[i]], product_orders[[i]];
    IF SQLCODE <> 0 THEN LEAVE lp; END IF;
    IF i = 1 THEN
        SET greatest_orders = 1;
    ELSE
        IF CARDINALITY( product_orders[[greatest_orders]] )
        < CARDINALITY( product_orders[[i]] ) THEN
            SET greatest_orders = i;
        END IF;
    END IF;
    SET i = i + 1;
END LOOP;

IF greatest_orders >= 1 THEN
    SELECT * FROM GROUPO.Products WHERE ID = products[[greatest_orders]];
END IF;
END;
```

ARRAY_AGG-Funktion [Aggregat]

Erstellt ein unbeschränktes eindimensionales Array aus dem angegebenen Ausdruck für jede Gruppe, in der der Array-Elementtyp mit dem angegebenen Ausdruck identisch ist.

Syntax

```
ARRAY_AGG( expression
[ ORDER BY order-by-expression [ ASC | DESC ], ... ] )
```

Parameter

- **Ausdruck** Der Ausdruck, auf dem das Array basieren soll.

Im erstellten Array erhält das erste Element den Wert der ersten Gruppe aus *Ausdruck*, das zweite Element den Wert der zweiten Gruppe usw.

- **s** Eine Gruppe von Attributen, die angeben, wie die vom *Ausdruck* zurückgegebenen Zeilen sortiert werden sollen.

Rückgabe

ARRAY

Bemerkungen

Array-Elemente werden aus der Eingabe gefüllt, beginnend mit dem ersten Element.

ARRAY_AGG ignoriert keine NULL-Werte in der Eingabe. NULL-Werte werden im Array als separate Elemente gespeichert, wie jeder andere Wert. Wenn die Gruppe leer ist, enthält das Ergebnis der ARRAY_AGG-Funktion ein NULL-Element für diese Gruppe.

ARRAY_AGG kann zwar nicht als Fensterfunktion verwendet werden, aber als Eingabe für eine Fensterfunktion.

Der UNNEST-Array-Operator kann verwendet werden, um eine Reihe von Zeilen aus einem Array zu erstellen, damit die einzelnen Array-Elemente mit anderen relationalen Ausdrücken verarbeitet werden können.

Siehe auch

- [„Array-Operator UNNEST“ auf Seite 19](#)
- [„LIST-Funktion \[Aggregat\]“ auf Seite 302](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

ARRAY_MAX_CARDINALITY-Funktion [zusammengesetzt]

Gibt die maximale Anzahl von Elementen im Array zurück.

Syntax

ARRAY_MAX_CARDINALITY(*array-expression*)

Parameter

- **Array-Ausdruck** Der auszuwertende Array-Ausdruck.

Wenn *Array-Ausdruck* NULL ist, gibt ARRAY_MAX_CARDINALITY den Wert NULL zurück.

Rückgabe

INTEGER

Bemerkungen

Die Kardinalität der Sammlung ist die Anzahl der in der Sammlung enthaltenen Elemente.

Bei einem unbeschränkten Array gibt `ARRAY_MAX_CARDINALITY` die maximale Größe für ein von SQL Anywhere unterstütztes Array zurück. Bei einem begrenzten Array oder einem über einen Konstruktor erstellten Array gibt `ARRAY_MAX_CARDINALITY` die maximale explizit oder implizit deklarierte Größe des Arrays zurück.

Siehe auch

- [„ARRAY_AGG-Funktion \[Aggregat\]“ auf Seite 170](#)
- [„CARDINALITY-Funktion \[zusammengesetzt\]“ auf Seite 185](#)
- [„TRIM_ARRAY-Funktion \[zusammengesetzt\]“ auf Seite 416](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

ASCII-Funktion [Zeichenfolge]

Gibt den Ganzzahl-ASCII-Wert des ersten Bytes in einem Zeichenfolgenausdruck zurück.

Syntax

ASCII(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge.

Rückgabe

SMALLINT

Bemerkungen

Wenn die Zeichenfolge leer ist, gibt ASCII 0 zurück. Literal-Zeichenfolgen müssen von Anführungszeichen umschlossen sein. Wenn der Zeichensatz der Datenbank ein Mehrbyte-Zeichensatz ist und das erste Zeichen der Parameterzeichenfolge aus mehr als einem Byte besteht, ist das Ergebnis NULL.

Siehe auch

- [„CHAR-Funktion \[Zeichenfolge\]“ auf Seite 188](#)
- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 90 zurück:

```
SELECT ASCII( 'Z' );
```

ASIN-Funktion [Nummerisch]

Gibt den Arkussinus einer Zahl im Bogenmaß zurück.

Syntax

ASIN(*numeric-expression*)

Parameter

- **Nummerischer_Ausdruck** Der Sinus des Winkels.

Rückgabe

DOUBLE

Bemerkungen

Die SIN- und ASIN-Funktionen sind inverse Vorgänge.

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch.

Siehe auch

- „ACOS-Funktion [Nummerisch]“ auf Seite 167
- „ATAN-Funktion [Nummerisch]“ auf Seite 173
- „ATAN2-Funktion [Nummerisch]“ auf Seite 174
- „SIN-Funktion [Nummerisch]“ auf Seite 383

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Arkussinus-Wert für 0,52 zurück:

```
SELECT ASIN( 0.52 );
```

ATAN-Funktion [Nummerisch]

Gibt den Arkustangens einer Zahl im Bogenmaß zurück.

Syntax

ATAN(*numeric-expression*)

Parameter

- **Nummerischer_Ausdruck** Der Tangens des Winkels.

Bemerkungen

Die ATAN- und TAN-Funktionen sind inverse Vorgänge.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch.

Siehe auch

- „ACOS-Funktion [Numerisch]“ auf Seite 167
- „ASIN-Funktion [Numerisch]“ auf Seite 173
- „ATAN2-Funktion [Numerisch]“ auf Seite 174
- „TAN-Funktion [Numerisch]“ auf Seite 407

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Arkustangens-Wert für 0,52 zurück:

```
SELECT ATAN( 0.52 );
```

ATAN2-Funktion [Numerisch]

Gibt den Arkustangens des Bruchs aus zwei Zahlen im Bogenmaß zurück.

Syntax

{ **ATN2** | **ATAN2** }(*numeric-expression-1*, *numeric-expression-2*)

Parameter

- **Numerischer_Ausdruck_1** Der Zähler des Bruchs, dessen Arkustangens berechnet wird.
- **Numerischer_Ausdruck_2** Der Nenner des Bruchs, dessen Arkustangens berechnet wird.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch.

Siehe auch

- „ACOS-Funktion [Numerisch]“ auf Seite 167
- „ASIN-Funktion [Numerisch]“ auf Seite 173
- „ATAN-Funktion [Numerisch]“ auf Seite 173
- „TAN-Funktion [Numerisch]“ auf Seite 407

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Arkustangens-Wert für den Bruch 0,52 durch 0,60 zurück:

```
SELECT ATAN2( 0.52, 0.60 );
```

AVG-Funktion [Aggregat]

Berechnet bei einer Zeilenmenge den Durchschnitt eines numerischen Ausdrucks oder einer Menge eindeutiger Werte.

Syntax 1

AVG([**ALL** | **DISTINCT**] *numeric-expression*)

Syntax 2

AVG([**ALL**] *numeric-expression*) **OVER** (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- [**ALL**] **Numerischer_Ausdruck** Der Ausdruck, dessen Durchschnitt über die Zeilen in jeder Gruppe berechnet wird.
- **DISTINCT-Klausel** Berechnet den Durchschnitt der eindeutigen numerischen Werte in jeder Gruppe.

Rückgabe

Gibt NULL für eine Gruppe zurück, die keine Zeilen enthält.

Gibt DOUBLE zurück, wenn das Argument DOUBLE ist, sonst NUMERIC.

Bemerkungen

Dieser Durchschnitt schließt keine Zeilen mit ein, in denen *Numerischer_Ausdruck* NULL ist.

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie in der *Fensterspezifikation*-Definition unter „**WINDOW-Klausel**“ auf Seite 1124.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „Fensterfunktionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Diese Funktion kann einen Überlauflfehler erzeugen, was dazu führt, dass ein Fehler zurückgegeben wird. Sie können die CAST-Funktion auf *Numerischer_Ausdruck* anwenden, um den Überlauflfehler zu vermeiden.

Siehe auch

- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186
- „SUM-Funktion [Aggregat]“ auf Seite 403
- „COUNT-Funktion [Aggregat]“ auf Seite 205

Standards und Kompatibilität

- **SQL/2008** AVG ist eine Kernfunktion des SQL/2008-Standards.
- **SQL/2008** Syntax 1 ist eine Kernfunktion des SQL/2008-Standards, während Syntax 2 einen Teil der optionalen SQL/2008-Sprachenfunktion T611, "Grundlegende OLAP-Vorgänge", umfasst. Die Möglichkeit, DISTINCT über einen Ausdruck anzugeben, der keine Spaltenreferenz ist, umfasst einen Teil der optionalen SQL-Sprachenfunktion F561, "Full value expressions". SQL Anywhere unterstützt auch die SQL/2008-Sprachenfunktion F441, "Extended set function support", die zulässt, dass die Operanden von Aggregatfunktionen beliebige Ausdrücke sein können, möglicherweise einschließlich äußerer Referenzen zu Ausdrücken in anderen Abfrageblöcken, die keine Spaltenreferenzen sind.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der AVG-Funktion als auch eine äußere Referenz enthalten.

Siehe auch

- „Fehlerbehandlung: Aggregatfunktionen und äußere Referenzen“ [[SQL Anywhere 16 - Änderungen und Upgrades](#)]

Beispiel

Die folgende Anweisung gibt den Wert 49988.623200 zurück, wenn eine Verbindung mit SQL Anywhere 16 Demo besteht:

```
SELECT AVG( Salary ) FROM Employees;
```

Die folgende Anweisung gibt den durchschnittlichen Produktpreis aus der Tabelle "Products" zurück, wenn eine Verbindung mit SQL Anywhere 16 Demo besteht:

```
SELECT AVG( DISTINCT UnitPrice ) FROM Products;
```

Die folgende Anweisung gibt einen Fehler mit dem SQLSTATE-Wert 42W68 zurück, da die Argumente von AVG sowohl einen quantifizierten Ausdruck aus der Unterabfrage als auch eine äußere Referenz (p.Quantity) aus dem äußeren SELECT-Block enthalten:

```
SELECT * from GROUPO.Products as p
WHERE p.Quantity > ( SELECT AVG( 0.5 * p.Quantity + 0.5 * s.Quantity )
                    from GROUPO.SalesOrderItems as s
                    WHERE s.ProductID = p.ProductID )
```

BASE64_DECODE-Funktion [Zeichenfolge]

Dekodiert Daten unter Verwendung des MIME base64-Formats und gibt die Zeichenfolge als LONG VARCHAR zurück.

Syntax

BASE64_DECODE(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die zu dekodierende Zeichenfolge. Die Zeichenfolge muss Base64-kodiert sein.

Rückgabe

LONG VARCHAR

Siehe auch

- „BASE64_ENCODE-Funktion [Zeichenfolge]“ auf Seite 177
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel fügt ein Bild von einem Embedded SQL-Programm in eine Bilddatentabelle ein. Die Eingabedaten (Hostvariable) müssen im base64-Format kodiert sein:

```
EXEC SQL INSERT INTO images ( image_data ) VALUES ( BASE64_DECODE ( :img ) );
```

BASE64_ENCODE-Funktion [Zeichenfolge]

Kodiert Daten unter Verwendung des MIME base64-Formats und gibt eine 7-Bit ASCII-Zeichenfolge zurück.

Syntax

BASE64_ENCODE(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die zu kodierende Zeichenfolge.

Rückgabe

LONG VARCHAR

Siehe auch

- „[BASE64_DECODE-Funktion \[Zeichenfolge\]](#)“ auf Seite 177
- „[Zeichenfolgenfunktionen](#)“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung ruft Daten aus einer fiktiven Tabelle ab, die Bilder enthält, und gibt sie im ASCII-Format zurück. Die resultierende Zeichenfolge kann in eine E-Mail-Nachricht eingebettet werden, die der Empfänger dekodiert, um das ursprüngliche Bild zu erhalten.

```
SELECT BASE64_ENCODE( image_data ) FROM IMAGES;
```

BINTOHEX-Funktion [Datentypkonvertierung]

Gibt das hexadezimale Äquivalent einer binären Zeichenfolge zurück.

Syntax

BINTOHEX(*binary-expression*)

Parameter

- **Binärausdruck** Die binäre Zeichenfolge, die in eine hexadezimale Zeichenfolge konvertiert werden soll.

Rückgabe

Die BINTOHEX Funktion gibt eine LONG VARCHAR-Zeichenfolge zurück. Die Länge des Ergebnisses ist die zweifache Länge der Eingabezeichenfolge.

Bemerkungen

Die Funktionen CAST, CONVERT, BINTOHEX, HEXTOBIN, HEXTOINT und INTTOHEX können für Konvertierungen in hexadezimale Werte bzw. aus hexadezimalen Werten verwendet werden. Weitere Hinweise zur Verwendung dieser Funktionen finden Sie unter [Konvertierung in und aus hexadezimalen Werten auf Seite 7](#).

Siehe auch

- „[HEXTOBIN-Funktion \[Datentypkonvertierung\]](#)“ auf Seite 274

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt eine Zeichenfolge zurück, die 313233 enthält:

```
SELECT BINTOHEX(0x313233);
```

BIT_AND-Funktion [Aggregat]

Gibt das bitweise AND des angegebenen Ausdrucks für jede Zeilengruppe zurück.

Syntax

BIT_AND(*bit-expression*)

Parameter

- **Bit-Ausdruck** Das Objekt, dessen Aggregat gebildet wird. Der Ausdruck kann ein VARBIT-Array, ein BINARY-Wert oder eine Ganzzahl (einschließlich aller Ganzzahl-Varianten wie etwa BIT und TINYINT) sein.

Rückgabe

Derselbe Datentyp wie das Argument. Für jede verglichene Bitposition gilt: Wenn jede Zeile eine 1 in der Bitposition enthält, wird 1 zurückgegeben, ansonsten wird 0 zurückgegeben.

Siehe auch

- „[BIT_OR-Funktion \[Aggregat\]](#)“ auf Seite 180
- „[BIT_XOR-Funktion \[Aggregat\]](#)“ auf Seite 182
- „[Bit-Operatoren](#)“ auf Seite 20

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel werden vier Zeilen generiert, die eine CHAR-Spalte enthalten, und anschließend werden die Werte in VARBIT konvertiert:

```
SELECT BIT_AND( CAST(row_value AS VARBIT) )
FROM dbo.sa_split_list('0001,0111,0100,0011');
```

Das Ergebnis (0000) wird folgendermaßen ermittelt:

1. Ein bitweises AND wird zwischen Zeile 1 (0001) und Zeile 2 (0111) durchgeführt und ergibt 0001 (beide Werte hatten ein 1 als viertes Bit).
2. Ein bitweises AND wird zwischen dem Ergebnis des vorherigen Vergleichs (0001) und Zeile 3 (0100) durchgeführt und ergibt 0000 (kein Wert hatte 1 in demselben Bit).

- Ein bitweises AND wird zwischen dem Ergebnis des vorherigen Vergleichs (0000) und Zeile 4 (0011) durchgeführt und ergibt 0000 (kein Wert hatte 1 in demselben Bit).

BIT_LENGTH-Funktion [Bit-Array]

Gibt die Anzahl der im Array gespeicherten Bits zurück.

Syntax

BIT_LENGTH(*bit-expression*)

Parameter

- **Bit-Ausdruck** Der Bitausdruck, für den die Länge bestimmt werden soll.

Rückgabe

INT

Siehe auch

- [„CHAR_LENGTH-Funktion \[Zeichenfolge\]“ auf Seite 189](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **SQL/1999** Die BIT_LENGTH Funktion war eine Kernfunktion des SQL/1999-Standards. Der BIT VARYING-Datentyp war die optionale SQL-Sprachenfunktion F511 des SQL/1999-Standards. Die Unterstützung für BIT_LENGTH und den Datentyp BIT VARYING wurde im SQL/2003-Standard entfernt.

Beispiel

Die folgende Anweisung gibt den Wert 8 zurück:

```
SELECT BIT_LENGTH( '01101011' );
```

BIT_OR-Funktion [Aggregat]

Gibt das bitweise OR des angegebenen Ausdrucks für jede Zeilengruppe zurück.

Syntax

BIT_OR(*bit-expression*)

Parameter

- **Bit-Ausdruck** Das Objekt, dessen Aggregat gebildet wird. Der Ausdruck kann ein VARBIT-Array, ein BINARY-Wert oder eine Ganzzahl (einschließlich aller Ganzzahl-Varianten wie etwa BIT und TINYINT) sein.

Rückgabe

Derselbe Datentyp wie das Argument. Für jede verglichene Bitposition gilt: Wenn irgendeine Zeile eine 1 in der Bitposition enthält, wird 1 zurückgegeben, ansonsten wird 0 zurückgegeben.

Siehe auch

- [„BIT_AND-Funktion \[Aggregat\]“ auf Seite 179](#)
- [„BIT_XOR-Funktion \[Aggregat\]“ auf Seite 182](#)
- [„Bit-Operatoren“ auf Seite 20](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel werden vier Zeilen generiert, die eine CHAR-Spalte enthalten, und anschließend werden die Werte in VARBIT konvertiert:

```
SELECT BIT_OR( CAST(row_value AS VARBIT) )
FROM dbo.sa_split_list('0001,0111,0100,0011');
```

Das Ergebnis (0111) wird folgendermaßen ermittelt:

1. Ein bitweises OR wird zwischen Zeile 1 (0001) und Zeile 2 (0111) durchgeführt und ergibt 0111.
2. Ein bitweises OR wird zwischen dem Ergebnis aus dem vorherigen Vergleich (0111) und Zeile 3 (0100) durchgeführt und ergibt 0111.
3. Ein bitweises OR wird zwischen dem Ergebnis aus dem vorherigen Vergleich (0111) und Zeile 4 (0011) durchgeführt und ergibt 0111.

BIT_SUBSTR-Funktion [Bit-Array]

Gibt ein Sub-Array eines Bit-Arrays zurück.

Syntax

BIT_SUBSTR(*bit-expression* [, *start* [, *length*]])

Parameter

- **Bit-Ausdruck** Das Bit-Array, aus dem das Sub-Array extrahiert werden soll.
- **start** Die Startposition des zurückzugebenden Sub-Arrays. Eine negative Startposition legt eine Anzahl von Bits vom Ende anstatt vom Anfang des Arrays aus fest. Das erste Bit im Array hat die Position 1.
- **length** Die Länge des zurückzugebenden Sub-Arrays. Eine positive Länge legt fest, dass das Sub-Array in *length* Bits rechts von der Startposition endet, während eine negative Länge links von der Startposition maximal *length* Bits bis zur und einschließlich der Startposition zurückgibt.

Rückgabe

LONG VARBIT

Bemerkungen

Sowohl *start* als auch *length* können entweder positiv oder negativ sein. Mithilfe von entsprechenden Kombinationen von negativen und positiven Zahlen können Sie ein Sub-Array entweder vom Anfang oder vom Ende der Zeichenfolge bekommen. Die Verwendung einer negativen Zahl für *length* wirkt sich nicht auf die Reihenfolge der zurückgegebenen Bits im Sub-Array aus.

Wenn die *length* angegeben ist, ist das Sub-Array auf diese Länge begrenzt. Wenn *start* Null und die Länge nicht-negativ ist, wird ein start-Wert von "1" verwendet. Wenn *start* Null und *length* negativ ist, wird ein Start-Wert von "-1" verwendet.

Wenn keine *length* angegeben ist, wird die Auswahl bis zum Ende des Arrays fortgesetzt.

Die BIT_SUBSTR-Funktion ist äquivalent zu, aber schneller als Folgendes:

```
CAST( SUBSTR( CAST( bit-expression AS VARCHAR ),  
start [, length ] )  
AS VARBIT );
```

Siehe auch

- [„SUBSTRING-Funktion \[Zeichenfolge\]“ auf Seite 401](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt 1101 zurück:

```
SELECT BIT_SUBSTR( '001101', 3 );
```

Die folgende Anweisung gibt 10110 zurück:

```
SELECT BIT_SUBSTR( '01011011101111011111', 2, 5 );
```

Die folgende Anweisung gibt 11111 zurück:

```
SELECT BIT_SUBSTR( '01011011101111011111', -5, 5 );
```

BIT_XOR-Funktion [Aggregat]

Gibt das bitweise XOR des angegebenen Ausdrucks für jede Zeilengruppe zurück.

Syntax

BIT_XOR(*bit-expression*)

Parameter

- **Bit-Ausdruck** Das Objekt, dessen Aggregat gebildet wird. Der Ausdruck kann ein VARBIT-Array, ein BINARY-Wert oder eine Ganzzahl (einschließlich aller Ganzzahl-Varianten wie etwa BIT und TINYINT) sein.

Rückgabe

Derselbe Datentyp wie das Argument. Für jede verglichene Bitposition gilt: Wenn ein ungerade Anzahl von Zeilen eine 1 in der Bitposition enthält, wird 1 zurückgegeben, ansonsten wird 0 zurückgegeben.

Siehe auch

- „BIT_AND-Funktion [Aggregat]“ auf Seite 179
- „BIT_OR-Funktion [Aggregat]“ auf Seite 180
- „Bit-Operatoren“ auf Seite 20

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel werden vier Zeilen generiert, die eine CHAR-Spalte enthalten, und anschließend werden die Werte in VARBIT konvertiert:

```
SELECT BIT_XOR( CAST(row_value AS VARBIT) )
FROM dbo.sa_split_list('0001,0111,0100,0011');
```

Das Ergebnis (0001) wird folgendermaßen ermittelt:

1. Ein bitweises exklusives OR (XOR) wird zwischen Zeile 1 (0001) und Zeile 2 (0111) durchgeführt und ergibt 0110.
2. Ein bitweises XOR wird zwischen dem Ergebnis aus dem vorherigen Vergleich (0110) und Zeile 3 (0100) durchgeführt und ergibt 0010.
3. Ein bitweises XOR wird zwischen dem Ergebnis aus dem vorherigen Vergleich (0010) und Zeile 4 (0011) durchgeführt und ergibt 0001.

BYTE_LENGTH-Funktion [Zeichenfolge]

Gibt die Anzahl der Byte in einer Zeichenfolge zurück.

Syntax

BYTE_LENGTH(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, deren Länge berechnet werden soll.

Rückgabe

INT

Bemerkungen

Nachgestellte Leerzeichen in *Zeichenfolgenausdruck* sind in der zurückgegebenen Länge enthalten.

Der Rückgabewert einer NULL-Zeichenfolge ist NULL.

Wenn die Zeichenfolge zu einem Mehrbyte-Zeichensatz gehört, kann sich der BYTE_LENGTH-Wert von der Anzahl der Zeichen, die CHAR_LENGTH zurückgibt, unterscheiden.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- „CHAR_LENGTH-Funktion [Zeichenfolge]“ auf Seite 189
- „DATALENGTH-Funktion [System]“ auf Seite 214
- „LENGTH-Funktion [Zeichenfolge]“ auf Seite 301
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Die entsprechende Funktion im SQL/2008-Standard ist die OCTET_LENGTH-Funktion.

Beispiel

Die folgende Anweisung gibt den Wert 12 zurück:

```
SELECT BYTE_LENGTH( 'Test Message' );
```

BYTE_SUBSTR-Funktion [Zeichenfolge]

Gibt eine Teilzeichenfolge einer Zeichenfolge zurück. Die Teilzeichenfolge wird nach Bytes berechnet, nicht nach Zeichen.

Syntax

BYTE_SUBSTR(*string-expression*, *start* [, *length*])

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, der die Teilzeichenfolge entnommen wird.
- **start** Ein Ganzzahl-Ausdruck, der den Start der Teilkette angibt. Eine positive Ganzzahl startet am Anfang der Zeichenfolge, mit dem ersten Zeichen in der Position 1. Eine negative Ganzzahl gibt eine Teilzeichenfolge an, die am Ende der Zeichenfolge beginnt, wobei sich das letzte Zeichen in der Position -1 befindet.
- **length** Ein Ganzzahl-Ausdruck, der die Länge der Teilzeichenfolge angibt. Eine positive *length* gibt die Anzahl von Byte an, die genommen werden sollen, wobei mit der Startposition *begonnen* wird. Eine negative *length* gibt maximal *length* Bytes links von der Startposition bis zur und einschließlich der Startposition zurück.

Rückgabe

BINARY, VARCHAR oder NVARCHAR. Der zurückgegebene Wert hängt vom Typ von *Zeichenfolgenausdruck* ab. Außerdem bestimmen die von Ihnen angegebenen Argumente, ob der zurückgegebene Wert LONG ist. Beispiel: LONG wird nicht zurückgegeben, wenn Sie eine Konstante < 32 kB für die Länge angeben.

Bemerkungen

Wenn *length* angegeben ist, ist die Teilzeichenfolge auf diese Anzahl von Byte begrenzt. Sowohl *start* als auch *length* können entweder positiv oder negativ sein. Mithilfe von entsprechenden Kombinationen von negativen und positiven Zahlen können Sie eine Teilzeichenfolge entweder vom Anfang oder vom Ende der Zeichenfolge bekommen.

Wenn *start* Null und die Länge nicht-negativ ist, wird ein *start*-Wert von "1" verwendet. Wenn *start* Null und *length* negativ ist, wird ein Start-Wert von "-1" verwendet.

Siehe auch

- „SUBSTRING-Funktion [Zeichenfolge]“ auf Seite 401
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert Test zurück:

```
SELECT BYTE_SUBSTR( 'Test Message', 1, 4 );
```

CARDINALITY-Funktion [zusammengesetzt]

Gibt die höchste Nummer eines Array-Elements zurück, dem ein Wert zugeordnet wurde, einschließlich NULL.

Syntax

CARDINALITY(*array-expression*)

Parameter

- **Array-Ausdruck** Der Array-Ausdruck, für den die Kardinalität berechnet wird.

Wenn *Array-Ausdruck* NULL ist, gibt CARDINALITY den Wert NULL zurück.

Rückgabe

INTEGER

Bemerkungen

Die Kardinalität der Sammlung ist die Anzahl der in der Sammlung enthaltenen Elemente.

Das Ergebnis ist eine Ganzzahl zwischen 0 und der maximalen Größe des Arrays.

Siehe auch

- [„ARRAY_AGG-Funktion \[Aggregat\]“ auf Seite 170](#)
- [„ARRAY_MAX_CARDINALITY-Funktion \[zusammengesetzt\]“ auf Seite 171](#)
- [„TRIM_ARRAY-Funktion \[zusammengesetzt\]“ auf Seite 416](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird der Wert 4 zurückgegeben:

```
SELECT CARDINALITY( ARRAY( 3,4 ) || ARRAY( 5,6 ) );
```

CAST-Funktion [Datentypkonvertierung]

Gibt den in einen angegebenen Datentyp konvertierten Wert eines Ausdrucks zurück.

Syntax

CAST(*expression* **AS** *datatype*)

Parameter

- ***expression*** Der zu konvertierende Ausdruck.
- ***datatype*** Der Zieldatentyp.

Rückgabe

Abhängig vom angeforderten Datentyp.

Bemerkungen

Wenn Sie bei Zeichenfolgentypen keine Länge angeben, wird eine geeignete Länge gewählt. Wenn bei einer DECIMAL-Konvertierung weder Gesamtstellenzahl noch Dezimalstellen angegeben wurden, wählt der Datenbankserver geeignete Werte aus.

Wenn Sie die CAST-Funktion zum Kürzen von Zeichenfolgen verwenden, muss die Datenbankoption `string_rtruncation` auf OFF gesetzt sein, weil sonst ein Fehler auftritt. Es wird empfohlen, zum Kürzen von Zeichenfolgen die Funktion LEFT zu verwenden.

Die Funktionen HEXTOINT und INTTOHEX können zum Konvertieren in und aus hexadezimalen Werten verwendet werden. Weitere Hinweise zur Verwendung dieser Funktionen finden Sie unter [Konvertierung in und aus hexadezimalen Werten auf Seite 7](#).

Siehe auch

- „Datentypkonvertierungen“ auf Seite 146
- „CONVERT-Funktion [Datentypkonvertierung]“ auf Seite 200
- „LEFT-Funktion [Zeichenfolge]“ auf Seite 300

Standards und Kompatibilität

- **SQL/2008** Die CAST-Funktion ist eine Kernfunktion des SQL/2008-Standards. In SQL Anywhere unterstützt CAST jedoch eine Reihe von Datentypkonvertierungen, die im SQL-Standard nicht zulässig sind. Zum Beispiel können Sie in SQL Anywhere mit CAST einen Ganzzahlwert in einen DATE-Typ konvertieren, während diese Typkonvertierung im SQL-Standard nicht zulässig ist. Weitere Hinweise finden Sie unter „Datentypkonvertierungen“ auf Seite 146.

Beispiel

Die folgende Funktion stellt sicher, dass eine Zeichenfolge als Datum verwendet wird:

```
SELECT CAST( '2000-10-31' AS DATE );
```

Der Wert des Ausdrucks `1 + 2` wird berechnet und das Ergebnis als Zeichenfolge, bestehend aus einem Zeichen, ausgegeben.

```
SELECT CAST( 1 + 2 AS CHAR );
```

Das Casting zwischen VARCHAR und ST_GEOMETRY erfolgt normalerweise implizit. Die folgende Anweisung fügt beispielsweise Werte zu ST_GEOMETRY-Spalten hinzu und verwendet dafür die ST_POINT-Funktion und einen VARCHAR-Parameter. Jeder Wert wird implizit in einen mit den Spalten der Tabelle konsistenten ST_GEOMETRY-Datentyp umgewandelt, aber Ergebnisse werden weiterhin als VARCHAR angezeigt.

```
INSERT INTO T1 VALUES (2, ST_POINT(1,2,0), 'SRID=2163;Point(1 2)');
```

CEILING-Funktion [Numerisch]

Gibt die erste Ganzzahl zurück, die größer oder gleich einem gegebenen Wert ist. Bei positiven Zahlen wird dies "Aufrunden" genannt.

Syntax

```
{ CEILING | CEIL } ( numeric-expression )
```

Parameter

- **Numerischer_Ausdruck** Die Zahl, deren Obergrenze berechnet werden soll.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch.

Siehe auch

- [„FLOOR-Funktion \[Numerisch\]“](#) auf Seite 265

Standards und Kompatibilität

- **SQL/2008** Die CEILING Funktion umfasst Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Die folgende Anweisung gibt den Wert 60 zurück:

```
SELECT CEILING( 59.84567 );
```

CHAR-Funktion [Zeichenfolge]

Gibt das zu einem ASCII-Code gehörige Zeichen zurück.

Syntax

CHAR(*integer-expression*)

Parameter

- **Ganzzahlausdruck** Die Zahl, die in ein ASCII-Zeichen konvertiert werden soll. Die Zahl muss im Bereich 0 bis einschließlich 255 liegen.

Rückgabe

VARCHAR

Bemerkungen

Das zurückgegebene Zeichen entspricht dem angegebenen numerischen Ausdruck im aktuellen Zeichensatz nach binärer Sortierreihenfolge.

CHAR gibt NULL für Ganzzahl-Ausdrücke mit Werten größer als 255 und kleiner als Null zurück.

Siehe auch

- [„Zeichenfolgenfunktionen“](#) auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert Y zurück:

```
SELECT CHAR( 89 );
```

CHAR_LENGTH-Funktion [Zeichenfolge]

Gibt die Anzahl der Zeichen in einer Zeichenfolge zurück.

Syntax

CHAR_LENGTH (*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, deren Länge berechnet werden soll.

Rückgabe

INT

Bemerkungen

Nachgestellte Leerzeichen sind in der zurückgegebenen Länge enthalten.

Der Rückgabewert einer NULL-Zeichenfolge ist NULL.

Wenn die Zeichenfolge ein Mehrbyte-Zeichensatz ist, kann sich der von der CHAR_LENGTH-Funktion zurückgegebene Wert von der Anzahl von Bytes, die die BYTE_LENGTH-Funktion zurückgibt, unterscheiden.

Hinweis

Sie können die CHAR_LENGTH-Funktion und die LENGTH-Funktion gleichermaßen für CHAR, VARCHAR-, LONG VARCHAR- und NCHAR-Datentypen verwenden. Sie müssen allerdings die LENGTH-Funktion bei BINARY- und Bit-Array-Datentypen verwenden.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- „[BYTE_LENGTH-Funktion \[Zeichenfolge\]](#)“ auf Seite 183
- „[Zeichenfolgenfunktionen](#)“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** CHAR_LENGTH ist eine Kernfunktion des SQL/2008-Standards. Die Verwendung von CHAR_LENGTH über einen Ausdruck vom Datentyp NCHAR umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion F421.

Beispiel

Die folgende Anweisung gibt den Wert 8 zurück:

```
SELECT CHAR_LENGTH( 'Chemical' );
```

CHARINDEX-Funktion [Zeichenfolge]

Gibt die Position einer Zeichenfolge in einer anderen zurück.

Syntax

CHARINDEX(*string-expression-1*, *string-expression-2*)

Parameter

- **Zeichenfolgenausdruck_1** Die Zeichenfolge, nach der Sie suchen.
- **Zeichenfolgenausdruck_2** Die zu durchsuchende Zeichenfolge.

Rückgabe

INT

Bemerkungen

Das erste Zeichen von *Zeichenfolgenausdruck_1* ist als 1 definiert. Wenn die durchsuchte Zeichenfolge mehr als eine Instanz der anderen Zeichenfolge enthält, gibt die CHARINDEX-Funktion die Position der ersten Instanz zurück.

Wenn die durchsuchte Zeichenfolge die andere Zeichenfolge nicht enthält, gibt die CHARINDEX-Funktion "0" zurück.

Falls eines der Argumente NULL ist, ist auch das Ergebnis NULL.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- „SUBSTRING-Funktion [Zeichenfolge]“ auf Seite 401
- „REPLACE-Funktion [Zeichenfolge]“ auf Seite 365
- „LOCATE-Funktion [Zeichenfolge]“ auf Seite 305
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die Nach- und Vornamen aus den Spalten Surname und GivenName der Tabelle "Employees" zurück, aber nur, wenn der Nachname mit dem Buchstaben K anfängt:

```
SELECT Surname, GivenName
FROM GROUPO.Employees
WHERE CHARINDEX( 'K', Surname ) = 1;
```

Zurückgegebene Ergebnisse:

Surname	GivenName
Klobucher	James
Kuo	Felicia

Surname	GivenName
Kelly	Moir

COALESCE-Funktion [Verschiedene]

Gibt den ersten Nicht-NULL-Ausdruck in einer Liste zurück. Diese Funktion ist mit der ISNULL-Funktion identisch.

Syntax

COALESCE(*expression*, *expression* [, ...])

Parameter

- **expression** Ein beliebiger Ausdruck.

Mindestens zwei Ausdrücke müssen an die Funktion übergeben werden und alle Ausdrücke müssen vergleichbar sein.

Rückgabe

Der Rückgabetyt dieser Funktion hängt von den angegebenen Ausdrücken ab. Das bedeutet, dass der Datenbankserver beim Auswerten der Funktion zuerst nach einem Datentyp sucht, mit dem alle Ausdrücke verglichen werden können. Wenn er gefunden wird, vergleicht der Datenbankserver die Ausdrücke und gibt das Ergebnis in dem für den Vergleich verwendeten Datentyp zurück. Wenn der Datenbankserver keinen gemeinsamen Vergleichsdattentyp findet, wird ein Fehler zurückgegeben.

Bemerkungen

Das Ergebnis ist nur NULL, wenn alle Argumente NULL sind.

Der Parameter kann ein beliebiger Skalartyp sein, aber nicht notwendigerweise derselbe Typ.

Eine detaillierte Beschreibung, wie der Datenbankserver diese Funktion verarbeitet, finden Sie unter „ISNULL-Funktion [Verschiedene]“ auf Seite 295.

Siehe auch

- „ISNULL-Funktion [Verschiedene]“ auf Seite 295

Standards und Kompatibilität

- **SQL/2008** Kernfunktion.

Beispiel

Die folgende Anweisung gibt den Wert 34 zurück:

```
SELECT COALESCE( NULL, 34, 13, 0 );
```

COMPARE-Funktion [Zeichenfolge]

Damit können Sie zwei Zeichenfolgen basierend auf unterschiedlichen Kollationsregeln vergleichen.

Syntax

```
COMPARE(  
  string-expression-1,  
  string-expression-2  
  [, { collation-id  
      | collation-name[(collation-tailoring-string) ] } ]  
)
```

Parameter

- **Zeichenfolgenausdruck_1** Der erste Zeichenfolgenausdruck.
- **Zeichenfolgenausdruck_2** Der zweite Zeichenfolgenausdruck.

Der Zeichenfolgenausdruck kann nur Zeichen enthalten, die im Zeichensatz der Datenbank kodiert sind.

- **Kollations-ID** Eine Variable oder Ganzzahlkonstante, die die zu verwendende Sortierreihenfolge angibt. Sie können eine *Kollations-ID* nur für integrierte Kollationen verwenden. Siehe „[SORTKEY-Funktion \[Zeichenfolge\]](#)“ auf Seite 385.

Wenn Sie keinen Kollationsnamen bzw. keine Kollations-ID eingeben, wird als Standard "Default Unicode multilingual" verwendet.

- **Kollationsname** Eine Zeichenfolge oder Zeichenvariable, die den Namen der zu verwendenden Kollation angibt. Sie können auch *char_collation* oder *db_collation* angeben (z.B. `COMPARE('abc', 'ABC', 'char_collation');`), um die CHAR-Kollation der Datenbank zu verwenden. Ebenso können Sie *nchar_collation* angeben, um die NCHAR-Kollation der Datenbank zu verwenden. Eine Liste der gültigen Kollationen finden Sie unter „[SORTKEY-Funktion \[Zeichenfolge\]](#)“ auf Seite 385.
- **Zeichenfolge_Kollationsanpassung** Optional können Sie Optionen der Kollationsanpassung (*Zeichenfolge_Kollationsanpassung*) als zusätzliche Steuerung bei Zeichenvergleichen angeben. Diese Optionen werden in der Form von Schlüsselwort=Wert-Paaren, die in Klammern gesetzt werden, hinter dem Kollationsnamen angegeben. Zum Beispiel:
`'UCA(locale=es;case=LowerFirst;accent=respect)'`. Die Syntax zur Angabe dieser Optionen ist identisch mit der für die COLLATION-Klausel der CREATE DATABASE-Anweisung definierten Syntax.

Hinweis

Alle Optionen der Kollationsanpassung werden unterstützt, wenn die UCA-Kollation angegeben ist. Bei allen anderen Kollationen wird nur die Kollationsanpassungsoption für Groß-/Kleinschreibung unterstützt.

Siehe auch

- „Optionen der Kollationsanpassung“ [[SQL Anywhere Server - Datenbankadministration](#)]

Rückgabe

Ein INTEGER-Wert, basierend auf den von Ihnen gewählten Kollationsregeln:

Wert	Bedeutung
1	<i>Zeichenfolgenausdruck_1</i> ist größer als <i>Zeichenfolgenausdruck_2</i>
0	<i>Zeichenfolgenausdruck_1</i> ist gleich <i>Zeichenfolgenausdruck_2</i>
-1	<i>Zeichenfolgenausdruck_1</i> ist kleiner als <i>Zeichenfolgenausdruck_2</i>

Bemerkungen

Die COMPARE-Funktion vergleicht keine leeren Zeichenfolgen und Zeichenfolgen mit Leerzeichen, auch wenn bei der Datenbank das Auffüllen mit Leerzeichen aktiviert ist. Die COMPARE-Funktion verwendet die SORTKEY-Funktion, um Kollationsschlüssel für den Vergleich zu generieren. Daher erzielt der Vergleich einer leeren Zeichenfolge, einer Zeichenfolge mit einem Leerzeichen und einer Zeichenfolge mit zwei Leerzeichen nicht dasselbe Ergebnis.

Wenn *Zeichenfolgenausdruck_1* oder *Zeichenfolgenausdruck_2* NULL ist, ist das Ergebnis NULL.

Siehe auch

- „SORTKEY-Funktion [Zeichenfolge]“ auf Seite 385
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel führt drei Vergleiche mithilfe der COMPARE-Funktion durch:

```
SELECT COMPARE( 'abc', 'ABC', 'UCA(case=LowerFirst)' ),  
       COMPARE( 'abc', 'ABC', 'UCA(case=Ignore)' ),  
       COMPARE( 'abc', 'ABC', 'UCA(case=UpperFirst)' );
```

Die zurückgegebenen Werte sind -1, 0, 1, was das Ergebnis der einzelnen Vergleiche anzeigt. Der erste Vergleich ergibt -1, weil *Zeichenfolgenausdruck_2* ("ABC") kleiner ist als *Zeichenfolgenausdruck_1* ("abc"). Dies liegt daran, dass die Berücksichtigung der Groß-/Kleinschreibung in der ersten COMPARE-Anweisung auf LowerFirst gesetzt ist.

COMPRESS-Funktion [Zeichenfolge]

Komprimiert die Zeichenfolge und gibt einen Wert vom Typ LONG BINARY zurück

Syntax

COMPRESS(*string-expression* [, 'compression-algorithm-alias'])

Parameter

- **Zeichenfolgenausdruck** Die zu komprimierende Zeichenfolge. Binäre Werte können an diese Funktion übergeben werden. Dieser Parameter beachtet die Groß- und Kleinschreibung, sogar in Datenbanken, die das nicht tun.
- **compression-algorithm-alias** Alias für den bei der Komprimierung zu verwendenden Algorithmus. Die unterstützten Werte sind "zip" und "gzip". (Beide basieren auf demselben Algorithmus, verwenden aber unterschiedliche Header und Trailer.)

Zip ist ein allgemein unterstützter Komprimierungsalgorithmus. Gzip ist mit dem gzip-Dienstprogramm unter Unix kompatibel, der zip-Algorithmus hingegen nicht.

Die Dekomprimierung muss mit demselben Algorithmus durchgeführt werden.

Rückgabe

LONG BINARY

Bemerkungen

Der von COMPRESS zurückgegebene Wert ist nicht in lesbarer Form. Wenn der zurückgegebene Wert länger als die ursprüngliche Zeichenfolge ist, wird die maximale Größe (ursprüngliche Zeichenfolge + 12 Byte) nicht mehr als eine Zunahme von 0,1% aufweisen. Einen komprimierten *Zeichenfolgenausdruck* können Sie mit der DECOMPRESS-Funktion dekomprimieren.

Wenn Sie komprimierte Werte in einer Tabelle speichern, sollte die Spalte BINARY oder LONG BINARY sein, damit keine Zeichensatzkonvertierung an den Daten durchgeführt wird.

Siehe auch

- [„DECOMPRESS-Funktion \[Zeichenfolge\]“ auf Seite 233](#)
- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel gibt die Länge der binären Zeichenfolge zurück, die erstellt wird, wenn die Zeichenfolge "Hello World" mit dem gzip-Algorithmus komprimiert wird. Dieses Beispiel ist nützlich, wenn Sie ermitteln wollen, ob ein Wert eine geringere Länge hat, wenn er komprimiert wird.

```
SELECT LENGTH( COMPRESS( 'Hello world', 'gzip' ) );
```

CONFLICT-Funktion [Verschiedene]

Zeigt an, ob eine Spalte die Ursache eines Konflikts in einem UPDATE ist, das gegenüber einer konsolidierten Datenbank in einer SQL Remote-Umgebung durchgeführt wird.

Syntax

CONFLICT(*column-name*)

Parameter

- **Spaltenname** Der Name der Spalte, die auf Konflikte untersucht wird.

Rückgabe

Gibt TRUE zurück, wenn die Spalte in der VERIFY-Liste einer UPDATE-Anweisung angezeigt wird, die vom SQL Remote-Nachrichtenagenten ausgeführt wird, und wenn der in der VALUES-Liste gelieferte Wert dieser Anweisung nicht mit dem ursprünglichen Wert der Spalte in der Zeile übereinstimmt, die aktualisiert wird. Ansonsten wird FALSE zurückgegeben.

Siehe auch

- „CREATE TRIGGER-Anweisung“ auf Seite 762
- „Standardlösung für Aktualisierungskonflikte“ [[SQL Remote](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die CONFLICT-Funktion ist für die Verwendung in SQL Remote RESOLVE UPDATE-Triggern gedacht, um Fehlermeldungen zu vermeiden. Um die Verwendung der CONFLICT-Funktion zu illustrieren, nehmen wir die folgende Tabelle:

```
CREATE TABLE Admin (
    PKey bigint NOT NULL DEFAULT GLOBAL AUTOINCREMENT,
    TextCol CHAR(20) NULL, PRIMARY KEY ( PKey ) );
```

Nehmen wir an, dass sowohl konsolidierte als auch entfernte Datenbanken die folgende Zeile in der Admin-Tabelle haben:

```
1, 'Initial'
```

Jetzt aktualisieren Sie in der konsolidierten Datenbank die Zeile folgendermaßen:

```
UPDATE Admin SET TextCol = 'Consolidated Update' WHERE PKey = 1;
```

In der entfernten Datenbank aktualisieren Sie die Zeile zu einem anderen Wert, und zwar folgendermaßen:

```
UPDATE Admin SET TextCol = 'Remote Update' WHERE PKey = 1;
```

Als nächstes führen Sie dbremote in der entfernten Datenbank aus. Es wird eine Meldungsdatei generiert, die die folgenden Anweisungen enthält, die in der konsolidierten Datenbank ausgeführt werden sollen:

```
UPDATE Admin SET TextCol='Remote Update'
VERIFY ( TextCol )
VALUES ( 'Initial' )
WHERE PKey=1;
```

Wenn der SQL Remote-Nachrichtenagent auf der konsolidierten Datenbank läuft und diese UPDATE-Anweisung anwendet, verwendet SQL Anywhere die VERIFY- und VALUES-Klauseln, um zu ermitteln,

ob ein RESOLVE UPDATE-Trigger ausgelöst wird. Ein RESOLVE UPDATE-Trigger wird nur ausgelöst, wenn das Update vom SQL Remote-Nachrichtenagenten gegenüber einer konsolidierten Datenbank ausgeführt wird. Hier ist ein RESOLVE UPDATE-Trigger:

```
CREATE TRIGGER ResolveUpdateAdmin
RESOLVE UPDATE ON Admin
REFERENCING OLD AS OldConsolidated
          NEW AS NewRemote
          REMOTE as OldRemote
FOR EACH ROW BEGIN
  MESSAGE 'OLD';
  MESSAGE OldConsolidated.PKey || ',' || OldConsolidated.TextCol;
  MESSAGE 'NEW';
  MESSAGE NewRemote.PKey || ',' || NewRemote.TextCol;
  MESSAGE 'REMOTE';
  MESSAGE OldRemote.PKey || ',' || OldRemote.TextCol;
END;
```

Der RESOLVE UPDATE-Trigger wird ausgelöst, weil der aktuelle Wert der TextCol-Spalte in der konsolidierten Datenbank ('Consolidated Update') nicht mit dem Wert in der VALUES-Klausel für die zugeordnete Spalte ('Initial') übereinstimmt.

Der Trigger führt zu einem Fehlschlag, weil die PKey-Spalte in der UPDATE-Anweisung, die auf der entfernten Datenbank ausgeführt wurde, nicht geändert wurde. Dadurch gibt es keinen OldRemote.PKey-Wert, auf den dieser Trigger zugreifen könnte.

Die CONFLICT-Funktion hilft, diesen Fehler zu vermeiden, indem die folgenden Werte zurückgegeben werden:

- Wenn es keinen OldRemote.PKey-Wert gibt, wird FALSE zurückgegeben.
- Wenn es einen OldRemote.PKey-Wert gibt, der aber mit OldConsolidated.PKey übereinstimmt, wird FALSE zurückgegeben.
- Wenn es einen OldRemote.PKey-Wert gibt, der von OldConsolidated.PKey verschieden ist, wird TRUE zurückgegeben.

Sie können die CONFLICT-Funktion verwenden, um den Trigger neu zu schreiben und den Fehler zu vermeiden:

```
CREATE TRIGGER ResolveUpdateAdmin
RESOLVE UPDATE ON Admin
REFERENCING OLD AS OldConsolidated
          NEW AS NewRemote
          REMOTE as OldRemote
FOR EACH ROW BEGIN
  message 'OLD';
  message OldConsolidated.PKey || ',' || OldConsolidated.TextCol;
  message 'NEW';
  message NewRemote.PKey || ',' || NewRemote.TextCol;
  message 'REMOTE';
  if CONFLICT( PKey ) then
    message OldRemote.PKey;
  end if;
  if CONFLICT( TextCol ) then
    message OldRemote.TextCol;
  end if;
END;
```

CONNECTION_EXTENDED_PROPERTY-Funktion [Zeichenfolge]

Gibt den Wert der angegebenen Eigenschaft zurück. Es kann ein optionaler eigenschaftsspezifischer Zeichenfolgenparameter angegeben werden.

Syntax

```
CONNECTION_EXTENDED_PROPERTY(  
  { property-id | property-name }  
  [, property-specific-argument [, connection-id ] ]  
)
```

Parameter

- **Eigenschafts-ID** Die ID der Verbindungseigenschaft.
- **Eigenschaftsname** Der Name der Verbindungseigenschaft. Die unterstützten Eigenschaftsnamen lauten:
 - **CharSet** Gibt das CHAR-Zeichensatzlabel für die Verbindung zurück, wie es vom angegebenen Standard erkannt wird. Mögliche Werte sind: ASE, IANA, MIME, JAVA, WINDOWS, UTR22, IBM und ICU. Der Standard ist IANA, außer die Datenbankverbindung wurde mittels TDS hergestellt. In diesem Fall ist ASE der Standard.
 - **NcharCharSet** Gibt das NCHAR-Zeichensatzlabel für die Verbindung zurück, wie es vom angegebenen Standard erkannt wird. Die möglichen Werte sind dieselben wie oben für "CharSet" aufgelistet.
 - **Progress** Gibt Informationen darüber zurück, wie lange eine Anweisung gelaufen ist. Geben Sie einen Wert für *Eigenschaftsspezifisches_Argument* an, gefolgt von *Verbindungs-ID*, um spezifische Informationen zum Verarbeitungsfortschritt der Anweisung zurückzugeben.

Die folgenden Anweisungen und Prozeduren unterstützen die Progress-Eigenschaft:

- BACKUP DATABASE (sowohl Bild als auch Archiv)
- LOAD TABLE (nur USING FILE und USING CLIENT FILE)
- MESSAGE
- REORGANIZE TABLE
- RESTORE DATABASE
- UNLOAD (alle Typen)
- sa_table_page_usage-Systemprozedur
- **Eigenschaftsspezifisches_Argument** Optionaler eigenschaftsspezifischer Zeichenfolgenparameter, der folgender Verbindungseigenschaft zugeordnet ist:
 - **Progress**
 - **PercentComplete** Geben Sie "PercentComplete" an, um den Prozentsatz der verarbeiteten Anweisung zu erhalten.

- **Abgeschlossen** Geben Sie "Completed" an, um die abgeschlossene Anzahl der Einheiten zu erhalten.
- **Summe** Geben Sie "Total" an, um die Gesamtzahl der noch zu verarbeitenden Einheiten zu erhalten.
- **Einheit** Geben Sie "Units" an, um den Typ der noch zu verarbeitenden Einheiten (Seiten, Zeilen oder Byte) zu erhalten.
- **Elapsed** Geben Sie "Elapsed" an, um die aktuell verstrichene Zeit in Millisekunden zu erhalten.
- **Remaining** Geben Sie "Remaining" an, um die geschätzte verbleibende Zeit in Millisekunden zu erhalten.
- **RAW** Geben Sie "Raw" an, um eine Zeichenfolge zu erhalten, die alle oben genannten Werte in der angegebenen Reihenfolge, getrennt durch Semikola, umfasst. z. B.
`43;9728;22230;pages;5025;6138.`
- **Formatted** Geben Sie "Formatted" an, um das visuell lesbare Format zu erhalten. Beispiel:

```
43% ( 9728 of 22230 pages ) complete after 00:00:05; estimated  
00:00:06 remaining
```

Der Remaining-Wert kann leer sein, wenn die verbleibende Zeit noch nicht geschätzt wurde oder wenn die Anzahl der abgeschlossenen Einheiten größer ist als die ursprüngliche Schätzung.

Für alle eigenschaftsspezifischen Argumente außer "Formatted" werden große Byte-Werte nie in KB oder MB konvertiert.

- **Verbindungs-ID** Die Verbindungs-ID-Nummer einer Datenbankverbindung. Die ID-Nummer der aktuellen Verbindung wird benutzt, wenn kein Wert angegeben ist.

Rückgabe

Gibt erweiterte Verbindungseigenschaften zurück. Der zurückgegebene Wert ist VARCHAR.

Bemerkungen

Entweder die Eigenschafts-ID oder der Eigenschaftsname muss angegeben werden.

Die CONNECTION_EXTENDED_PROPERTY-Funktion ähnelt der CONNECTION_PROPERTY-Funktion, nur dass ein optionaler eigenschaftsspezifischer Zeichenfolgenparameter angegeben werden kann. Die Interpretation des eigenschaftsspezifischen Arguments hängt von der Eigenschafts-ID bzw. dem im ersten Argument angegebenen Namen ab.

Sie können die CONNECTION_EXTENDED_PROPERTY-Funktion verwenden, um den Wert von jeder Verbindungseigenschaft zurückzugeben. Erweiterte Informationen sind allerdings nur für die erweiterten Eigenschaften verfügbar.

🔥 **Cloud-Hinweis:** Aufgrund von Isolationsregeln für Tenant-Datenbanken gibt diese Funktion bei einem Aufruf in der Cloud nur Informationen über die aktuelle Tenant-Datenbank zurück.

Privilegien

Keine Privilegien sind erforderlich, um diese Funktion für die aktuelle Verbindungs-ID auszuführen. Um diese Funktion für andere Verbindungen ausführen zu können, benötigen Sie das SERVER OPERATOR-Systemprivileg, das MONITOR-Systemprivileg oder das DROP CONNECTION-Systemprivileg.

NULL wird zurückgegeben, wenn Sie einen ungültigen Parameterwert angeben oder eines der erforderlichen Systemprivilegien nicht haben.

Siehe auch

- „Liste der Verbindungseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „progress_messages-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „CONNECTION_PROPERTY-Funktion [System]“ auf Seite 199
- „DB_EXTENDED_PROPERTY-Funktion [System]“ auf Seite 226
- „DB_PROPERTY-Funktion [System]“ auf Seite 232

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel gibt den CHAR-Zeichensatz der aktuellen Verbindung zurück, wie er vom Java-Standard erkannt wird:

```
SELECT CONNECTION_EXTENDED_PROPERTY( 'charset', 'Java' );
```

CONNECTION_PROPERTY-Funktion [System]

Gibt den Wert einer angegebenen Verbindungseigenschaft als Zeichenfolge zurück.

Syntax

```
CONNECTION_PROPERTY(
{ property-id | property-name }
[ , connection-id ] )
```

Parameter

- **Eigenschafts-ID** Die ID der Verbindungseigenschaft.
- **Eigenschaftsname** Der Name der Verbindungseigenschaft.
- **Verbindungs-ID** Die Verbindungs-ID-Nummer einer Datenbankverbindung. Die ID-Nummer der aktuellen Verbindung wird benutzt, wenn kein Wert angegeben ist.

Rückgabe

VARCHAR, LONG VARCHAR

Bemerkungen

Entweder die Eigenschafts-ID oder der Eigenschaftsname muss angegeben werden.

🔥 **Cloud-Hinweis:** Aufgrund von Isolationsregeln für Tenant-Datenbanken gibt diese Funktion bei einem Aufruf in der Cloud nur Informationen über die aktuelle Tenant-Datenbank zurück.

Privilegien

Keine Privilegien sind erforderlich, um diese Funktion für die aktuelle Verbindungs-ID auszuführen. Um diese Funktion für andere Verbindungen ausführen zu können, benötigen Sie das SERVER OPERATOR-Systemprivileg, das MONITOR-Systemprivileg oder das DROP CONNECTION-Systemprivileg.

NULL wird zurückgegeben, wenn Sie einen ungültigen Parameterwert angeben oder eines der erforderlichen Systemprivilegien nicht haben.

Siehe auch

- „Liste der Verbindungseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Zugreifen auf Werte von Verbindungseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „PROPERTY_NUMBER-Funktion [System]“ auf Seite 342

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die Anzahl der vorbereiteten Anweisungen zurück, die verwaltet werden:

```
SELECT CONNECTION_PROPERTY( 'PrepStmt' );
```

CONVERT-Funktion [Datentypkonvertierung]

Gibt einen in einen angegebenen Datentyp konvertierten Ausdruck zurück.

Syntax

CONVERT(*datatype*, *expression* [, *format-style*])

Parameter

- **datatype** Der Datentyp, in den der Ausdruck konvertiert wird.
- **expression** Der zu konvertierende Ausdruck.
- **format-style** Der Stil-Code, der für den ausgegebenen Wert gilt. Verwenden Sie diesen Parameter, wenn Sie Zeichenfolgen in Datumsangaben oder Zeitdatumsangaben bzw. umgekehrt konvertieren. Die untenstehende Tabelle enthält die unterstützten Stil-Codes, gefolgt von einer Darstellung der Ausgabeformate, die vom jeweiligen Stil-Code erzeugt werden. Die Stil-Codes sind auf zwei Spalten aufgeteilt, abhängig davon, ob das Jahrhundert im Ausgabeformat enthalten ist (z.B. 06 bzw. 2006).

Stilcode 0 wird verwendet, wenn kein Argument angegeben wurde.

Stil-Codes ohne Jahrhundert (jj)	Stil-Codes mit Jahrhundert (jjjj)	Ausgabeformat
-	0 oder 100	Mmm tt jjjj hh:nnAA
1	101	mm/tt/jj[jj]
2	102	[jj]jj.mm.tt
3	103	tt/mm/jj[jj]
4	104	tt.mm.jj[jj]
5	105	tt-mm-jj[jj]
6	106	tt Mmm jj[jj]
7	107	Mmm tt, jj[jj]
8	108	hh:nn:ss
-	9 oder 109	Mmm tt jjjj hh:nn:ss:sssAA
10	110	mm-tt-jj[jj]
11	111	[jj]jj/mm/tt
12	112	[jj]jjmmtt
-	13 oder 113	tt Mmm jjjj hh:nn:ss:sss (24-Stundenuhr, Europa-Standard + Millisekunden, vierstelliges Jahr)
-	14 oder 114	hh:nn:ss:sss (24-Stundenuhr)
-	20 oder 120	jjjj-mm-tt hh:nn:ss (24-Stundenuhr, ODBC entsprechend, vierstelliges Jahr)
-	21 oder 121	jjjj-mm-tt hh:nn:ss:sss (24-Stundenuhr, ODBC entsprechend mit Millisekunden, vierstelliges Jahr)

Rückgabe

Abhängig vom angegebenen Datentyp.

Bemerkungen

Die CONVERT-Funktion kann verwendet werden, um eine Zeichenfolge in einen DATE-, TIME- oder TIMESTAMP-Datentyp zu konvertieren, wenn es keine Mehrdeutigkeit bei der syntaktischen Analyse der Zeichenfolge gibt. Wenn *format-style* angegeben ist, verwendet der Datenbankserver diesen

möglicherweise als Hinweis zur syntaktischen Analyse der Zeichenfolge. Der Datenbankserver gibt einen Fehler zurück, wenn er die Zeichenfolge syntaktisch nicht eindeutig analysieren kann.

Informationen über die Stile, die von den Ausgabesymbolen erzeugt werden (z.B. Mmm), finden Sie unter „date_format-Option“ [[SQL Anywhere Server - Datenbankadministration](#)].

Siehe auch

- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186
- „CSCONVERT-Funktion [Zeichenfolge]“ auf Seite 211

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Die CONVERT-Funktion ist im SQL/2008-Standard definiert. Im SQL-Standard besteht der Zweck von CONVERT jedoch darin, die Transkodierung des Eingabezeichenfolge-Ausdrucks in einen anderen Zeichensatz durchzuführen, was in SQL Anywhere als CSCONVERT-Funktion implementiert ist.

Beispiel

Die folgenden Anweisungen veranschaulichen die Verwendung des Formatstils:

```
SELECT CONVERT( CHAR( 20 ), OrderDate, 104 ) FROM GROUP0.SalesOrders;
```

OrderDate
16.03.2000
20.03.2000
23.03.2000
25.03.2000
...

```
SELECT CONVERT( CHAR( 20 ), OrderDate, 7 ) FROM GROUP0.SalesOrders;
```

OrderDate
Mar 16, 00
Mar 20, 00
Mar 23, 00
Mar 25, 00
...

Die folgende Anweisung veranschaulicht die Konvertierung in eine Ganzzahl und gibt den Wert 5 zurück:

```
SELECT CONVERT( integer, 5.2 );
```

CORR-Funktion [Aggregat]

Gibt den Korrelationskoeffizienten einer Menge von Zahlenpaaren zurück.

Syntax

CORR(*dependent-expression, independent-expression*)

Parameter

- **Abhängiger_Ausdruck** Die Variable, auf die sich der unabhängige Ausdruck auswirkt.
- **Unabhängiger_Ausdruck** Die Variable, die das Ergebnis beeinflusst

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch. Wenn die Funktion bei einer leeren Menge angewendet wird, gibt sie NULL zurück.

Abhängiger_Ausdruck und *Unabhängiger_Ausdruck* sind beide numerisch. Die Funktion wird auf die Menge von (*Abhängiger_Ausdruck*, *Unabhängiger_Ausdruck*)-Paaren angewendet, nachdem die Paare eliminiert wurden, bei denen *Abhängiger_Ausdruck* oder *Unabhängiger_Ausdruck* NULL ist. Die folgende Berechnung wird durchgeführt:

COVAR_POP (*y*, *x*) / **STDDEV_POP** (*y*) * **STDDEV_POP** (*x*)

Hier gilt: *y* stellt *abhängiger_Ausdruck* und *x* stellt *unabhängiger_Ausdruck* dar.

Siehe auch

- „Aggregatfunktionen“ auf Seite 153
- „COVAR_POP-Funktion [Aggregat]“ auf Seite 209
- „STDDEV_POP-Funktion [Aggregat]“ auf Seite 393

Standards und Kompatibilität

- **SQL/2008** Die CORR-Funktion umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Im folgenden Beispiel wird eine Korrelation ausgeführt, um herauszufinden, ob das Alter mit der Einkommenshöhe zusammenhängt. Die Anweisung gibt den Wert 0,44022675645996 zurück:

```
SELECT CORR( Salary, ( YEAR( NOW( ) ) - YEAR( BirthDate ) ) ) FROM
GROUPO.Employees;
```

COS-Funktion [Numerisch]

Gibt den Kosinus des Winkels im Bogenmaß zurück, der im Argument angegeben ist.

Syntax

COS(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Der Winkel im Bogenmaß.

Rückgabe

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltpgenauem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück. Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Siehe auch

- „ACOS-Funktion [Numerisch]“ auf Seite 167
- „COT-Funktion [Numerisch]“ auf Seite 204
- „SIN-Funktion [Numerisch]“ auf Seite 383
- „TAN-Funktion [Numerisch]“ auf Seite 407

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Kosinuswert eines Winkels von 0,52 Bogenmaß zurück:

```
SELECT COS( 0.52 );
```

COT-Funktion [Numerisch]

Gibt den Kotangens des Winkels im Bogenmaß zurück, der im Argument angegeben ist.

Syntax

COT(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Der Winkel im Bogenmaß.

Rückgabe

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltpgenauem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück. Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Siehe auch

- „COS-Funktion [Nummerisch]“ auf Seite 204
- „SIN-Funktion [Nummerisch]“ auf Seite 383
- „TAN-Funktion [Nummerisch]“ auf Seite 407

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Kotangenswert für 0,52 zurück:

```
SELECT COT( 0.52 );
```

COUNT-Funktion [Aggregat]

Zählt die Anzahl der Zeilen in einer Gruppe, abhängig von den angegebenen Parametern.

Syntax 1

```
COUNT( [ * | [ ALL | DISTINCT ] expression ] )
```

Syntax 2

```
COUNT( [ * | [ ALL ] expression ] ) OVER ( window-spec )
```

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- ***** Gibt die Anzahl der Zeilen in jeder Gruppe zurück. COUNT(*) und COUNT() sind semantisch gleichwertig.
- **[ALL] expression** Gibt die Anzahl der Zeilen in jeder Gruppe zurück, wobei *Ausdruck* nicht den Wert NULL hat.
- **DISTINCT expression** Gibt die Anzahl der unterschiedlichen Werte von *expression* für alle Zeilen in jeder Gruppe zurück, in denen *expression* nicht den Wert NULL hat.

Rückgabe

Die COUNT Funktion gibt einen Wert vom Datentyp INT zurück.

COUNT liefert nie den Wert NULL. Wenn eine Gruppe keine Zeilen oder keine Nicht-NULL-Werte von *expression* enthält, gibt COUNT den Wert 0 zurück.

Bemerkungen

Die COUNT Funktion gibt einen Maximalwert von 2147483647 zurück. Verwenden Sie die COUNT_BIG Funktion beim Zählen großen Ergebnismengen bzw. wenn es möglich ist, dass das Ergebnis mehrere Zeilen hat oder ein Überlauf auftritt.

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie in der *Fensterspezifikation*-Definition unter „[WINDOW-Klausel](#)“ auf Seite 1124.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „[COUNT\(* \)](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „[AVG-Funktion \[Aggregat\]](#)“ auf Seite 175
- „[SUM-Funktion \[Aggregat\]](#)“ auf Seite 403
- „[COUNT_BIG-Funktion \[Aggregat\]](#)“ auf Seite 206
- „[Fehlerbehandlung: Aggregatfunktionen und äußere Referenzen](#)“ [*SQL Anywhere 16 - Änderungen und Upgrades*]

Standards und Kompatibilität

- **SQL/2008** Kernfunktion. Wenn COUNT als Fensterfunktion verwendet wird (Syntax 2), umfasst sie einen Teil der optionalen SQL/2008-Sprachenfunktion T611, "Grundlegende OLAP-Vorgänge".

Die Möglichkeit, DISTINCT über einen Ausdruck anzugeben, der keine Spaltenreferenz ist, umfasst einen Teil der optionalen SQL-Sprachenfunktion F561, "Full value expressions". SQL Anywhere unterstützt auch die SQL/2008-Sprachenfunktion F441, "Extended set function support", die zulässt, dass die Operanden von Aggregatfunktionen beliebige Ausdrücke sein können, möglicherweise einschließlich äußerer Referenzen zu Ausdrücken in anderen Abfrageblöcken, die keine Spaltenreferenzen sind.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der COUNT-Funktion als auch eine äußere Referenz enthalten.

Ein Beispiel finden Sie unter „[AVG-Funktion \[Aggregat\]](#)“ auf Seite 175.

Beispiel

Die folgende Anweisung gibt jede eindeutige Stadt und die Anzahl der Mitarbeiter in der jeweiligen Stadt zurück:

```
SELECT City, COUNT( * ) FROM GROUPO.Employees GROUP BY City;
```

COUNT_BIG-Funktion [Aggregat]

Zählt die Anzahl der Zeilen in einer Gruppe, abhängig von den angegebenen Parametern.

Syntax 1

COUNT_BIG([*] [**ALL** | **DISTINCT**] *expression*)

Syntax 2

COUNT_BIG([*] [**ALL**] *expression*) **OVER** (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- ***** Gibt die Anzahl der Zeilen in jeder Gruppe zurück. COUNT_BIG(*) und COUNT_BIG() sind semantisch gleichwertig.
- **[ALL] expression** Gibt die Anzahl der Zeilen in jeder Gruppe zurück, wobei *Ausdruck* nicht den Wert NULL hat.
- **DISTINCT expression** Gibt die Anzahl der unterschiedlichen Werte von *expression* für alle Zeilen in jeder Gruppe zurück, in denen *expression* nicht den Wert NULL hat.

Rückgabe

COUNT_BIG gibt einen Wert vom Datentyp BIGINT zurück.

COUNT_BIG liefert nie den Wert NULL. Wenn eine Gruppe keine Zeilen oder keine Nicht-NULL-Werte von *Ausdruck* enthält, gibt COUNT_BIG den Wert 0 zurück.

Bemerkungen

Es wird empfohlen, die COUNT_BIG-Funktion beim Zählen von großen Ergebnismengen zu verwenden bzw. wenn es möglich ist, dass das Ergebnis mehrere Zeilen hat oder ein Überlauf auftritt. Andernfalls können Sie die COUNT-Funktion verwenden, deren Maximalwert bei 2147483647 liegt.

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie in der *Fensterspezifikation*-Definition unter „[WINDOW-Klausel](#)“ auf Seite 1124.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Siehe auch

- „AVG-Funktion [Aggregat]“ auf Seite 175
- „SUM-Funktion [Aggregat]“ auf Seite 403
- „COUNT-Funktion [Aggregat]“ auf Seite 205

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der COUNT_BIG-Funktion als auch eine äußere Referenz enthalten.

Ein Beispiel finden Sie unter „AVG-Funktion [Aggregat]“ [[UltraLite - Datenbankverwaltung](#)].

Siehe auch

- „Fehlerbehandlung: Aggregatfunktionen und äußere Referenzen“ [[SQL Anywhere 16 - Änderungen und Upgrades](#)]

Beispiel

Die folgende Anweisung gibt jede eindeutige Stadt und die Anzahl der Mitarbeiter in der jeweiligen Stadt zurück.

```
SELECT City, COUNT_BIG( * ) FROM GROUPO.Employees GROUP BY City;
```

COUNT_SET_BITS-Funktion [Bit-Array]

Gibt die Summe der Anzahl von Bits zurück, die im Array auf "1" (TRUE) gesetzt sind.

Syntax

COUNT_SET_BITS(*bit-expression*)

Parameter

- **Bit-Ausdruck** Das Bit-Array, bei dem die gesetzten Bits ermittelt werden sollen.

Rückgabe

UNSIGNED INT

Bemerkungen

Gibt NULL zurück, wenn *Bit-Ausdruck* NULL ist.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung liefert den Wert 4:

```
SELECT COUNT_SET_BITS( '00110011' );
```

Die folgende Anweisung gibt den Wert "12" zurück:

```
SELECT COUNT_SET_BITS( '0011001111111111' );
```

COVAR_POP-Funktion [Aggregat]

Gibt die Populationskovarianz einer Menge von Zahlenpaaren zurück.

Syntax 1

COVAR_POP(*dependent-expression*, *independent-expression*)

Syntax 2

COVAR_POP(*dependent-expression*, *independent-expression*)
OVER (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Abhängiger_Ausdruck** Die Variable, auf die sich der unabhängige Ausdruck auswirkt.
- **Unabhängiger_Ausdruck** Die Variable, die das Ergebnis beeinflusst

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelte genaue Gleitkommazahl durch. Wenn die Funktion bei einer leeren Menge angewendet wird, gibt sie NULL zurück.

Abhängiger_Ausdruck und *Unabhängiger_Ausdruck* sind beide numerisch. Die Funktion wird auf die Menge von (*Abhängiger_Ausdruck*, *Unabhängiger_Ausdruck*)-Paaren angewendet, nachdem alle Paare eliminiert wurden, bei denen *Abhängiger_Ausdruck* oder *Unabhängiger_Ausdruck* NULL ist. Dann wird die folgende Berechnung durchgeführt:

$$(\text{SUM}(y * x) - \text{SUM}(x) * \text{SUM}(y) / n) / n$$

Hier gilt: *y* stellt *abhängiger_Ausdruck* und *x* stellt *unabhängiger_Ausdruck* dar.

Weitere Hinweise zur durchgeführten statistischen Berechnung finden Sie unter „[Mathematische Formeln für die Aggregatfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie in der *Fensterspezifikation*-Definition unter „[WINDOW-Klausel](#)“ auf Seite 1124.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „COVAR_SAMP-Funktion [Aggregat]“ auf Seite 210
- „SUM-Funktion [Aggregat]“ auf Seite 403

Standards und Kompatibilität

- **SQL/2008** Die COVAR_POP-Funktion umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Das folgende Beispiel berechnet die Größe des Zusammenhangs zwischen dem Alter und dem Gehalt von Mitarbeitern. Die Funktion gibt den Wert "73.785,84005866687" zurück.

```
SELECT COVAR_POP( Salary, ( YEAR( NOW( ) ) - YEAR( BirthDate ) ) )  
FROM GROUPO.Employees;
```

COVAR_SAMP-Funktion [Aggregat]

Gibt die Stichproben-Kovarianz einer Menge von Zahlenpaaren zurück.

Syntax 1

COVAR_SAMP(*dependent-expression*, *independent-expression*)

Syntax 2

COVAR_SAMP(*dependent-expression*, *independent-expression*)
OVER (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Abhängiger_Ausdruck** Die Variable, auf die sich der unabhängige Ausdruck auswirkt.
- **Unabhängiger_Ausdruck** Die Variable, die das Ergebnis beeinflusst

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelte genaue Gleitkommazahl durch. Wenn die Funktion bei einer leeren Menge angewendet wird, gibt sie NULL zurück.

Abhängiger_Ausdruck und *Unabhängiger_Ausdruck* sind beide numerisch. Die Funktion wird auf die Menge von (*Abhängiger_Ausdruck*, *Unabhängiger_Ausdruck*)-Paaren angewendet, nachdem alle Paare eliminiert wurden, bei denen *Abhängiger_Ausdruck* oder *Unabhängiger_Ausdruck* NULL ist.

Weitere Hinweise zur durchgeführten statistischen Berechnung finden Sie unter „[Mathematische Formeln für die Aggregatfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie in der *Fensterspezifikation*-Definition unter „[WINDOW-Klausel](#)“ auf Seite 1124.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „[COVAR_POP-Funktion \[Aggregat\]](#)“ auf Seite 209
- „[SUM-Funktion \[Aggregat\]](#)“ auf Seite 403

Standards und Kompatibilität

- **SQL/2008** Die COVAR_SAMP-Funktion umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Das folgende Beispiel gibt den Wert "74782,9460054052" zurück.

```
SELECT COVAR_SAMP( Salary, ( 2008 - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

CSCONVERT-Funktion [Zeichenfolge]

Konvertiert Zeichenfolgen zwischen Zeichensätzen.

Syntax

```
CSCONVERT(
  string-expression,
  target-charset-string [, source-charset-string [, options ] ] )
```

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, die konvertiert werden soll.
- **target-charset-string** Der Zielzeichensatz. *target-charset-string* kann ein beliebiges von SQL Anywhere unterstütztes Zeichensatzlabel sein. Er kann auch Folgendes sein:
 - **os_charset** Geben Sie diese Klausel an, um den Zeichensatz zu verwenden, der von dem Betriebssystem verwendet wird, das den Datenbankserver hostet.
 - **char_charset** Geben Sie diese Klausel an, um den von der Datenbank verwendeten CHAR-Zeichensatz zu verwenden.
 - **nchar_charset** Geben Sie diese Klausel an, um den von der Datenbank verwendeten NCHAR-Zeichensatz zu verwenden.

- **Optionen** Sie können eine der folgenden Optionen festlegen:
 - **Bytereihenfolge-Markierung (BOM) lesen oder schreiben** Geben Sie **read_bom=on** oder **read_bom=off** an, um Lese-Bytereihenfolge-Markierungen zu aktivieren oder zu deaktivieren. Geben Sie **write_bom=on** oder **write_bom=off** an, um Schreib-Bytereihenfolge-Markierungen zu aktivieren oder zu deaktivieren. Standardmäßig ist das Verhalten **read_bom=on** und **write_bom=off**.
 - **source-charset-string** Der für *Zeichenfolgenausdruck* verwendete Zeichensatz. Der Standardwert ist "db_charset" (der Zeichensatz der Datenbank). *source-charset-string* kann ein beliebiges von SQL Anywhere unterstütztes Zeichensatzlabel sein. Er kann auch Folgendes sein:
 - **os_charset** Geben Sie diese Klausel an, um den Zeichensatz zu verwenden, der von dem Betriebssystem verwendet wird, das den Datenbankserver hostet.
 - **char_charset** Geben Sie diese Klausel an, um den von der Datenbank verwendeten CHAR-Zeichensatz zu verwenden.
 - **nchar_charset** Geben Sie diese Klausel an, um den von der Datenbank verwendeten NCHAR-Zeichensatz zu verwenden.

Rückgabe

LONG BINARY

Bemerkungen

Sie können eine Liste der Zeichensätze aufrufen, die von SQL Anywhere unterstützt werden, indem Sie den folgenden Befehl ausführen:

```
dbinit -le
```

Weitere Hinweise zu Zeichensatzlabels, die Sie mit dieser Funktion verwenden können, finden Sie unter „Unterstützte Zeichensätze“ [[SQL Anywhere Server - Datenbankadministration](#)].

Siehe auch

- „[Zeichenfolgenfunktionen](#)“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Im SQL/2008-Standard erfolgt die Konvertierung von Zeichenfolgendaten aus einem Zeichensatz in einen anderen mithilfe der CONVERT-Funktion (nicht zu verwechseln mit der SQL Anywhere-Funktion CONVERT), die andere Argumente besitzt als CSCONVERT.

Beispiele

Dieses Fragment konvertiert die Spalte mytext vom Zeichensatz für Traditionelles Chinesisch in den Zeichensatz für Vereinfachtes Chinesisch:

```
SELECT CSCONVERT( mytext, 'cp936', 'cp950' )  
FROM mytable;
```

Dieses Fragment konvertiert die Spalte `mytext` vom Zeichensatz der Datenbank in den Zeichensatz für Vereinfachtes Chinesisch:

```
SELECT CSCONVERT( mytext, 'cp936' )
FROM mytable;
```

Wird ein Dateiname in der Datenbank gespeichert, erfolgt die Speicherung im Zeichensatz der Datenbank. Wenn der Server aus einer Datei lesen oder in eine Datei schreiben soll, deren Name in einer Datenbank gespeichert ist (beispielsweise in einer externen gespeicherten Prozedur), muss der Dateiname explizit in den Zeichensatz des Betriebssystems konvertiert werden, bevor ein Zugriff erfolgen kann. Bei in der Datenbank gespeicherten Dateinamen, die von einem Client abgerufen werden, erfolgt eine automatische Umwandlung in den Zeichensatz des Clients, sodass eine explizite Konvertierung hier nicht notwendig ist.

Dieses Fragment konvertiert den Wert in der Spalte "filename" vom Datenbank-Zeichensatz in den Zeichensatz des Betriebssystems:

```
SELECT CSCONVERT( filename, 'os_charset' )
FROM mytable;
```

Eine Tabelle enthält eine Liste von Dateinamen. Eine externe gespeicherte Prozedur nimmt einen Dateinamen aus dieser Tabelle als Parameter und liest direkt von dieser Datei Informationen aus. Wird eine Konvertierung des Zeichensatzes nicht benötigt, funktioniert die folgende Anweisung:

```
SELECT MYFUNC( filename )
FROM mytable;
```

Die Klausel `mytable` gibt eine Tabelle mit einer Dateinamensspalte an. Wenn Sie jedoch den Dateinamen in den Zeichensatz des Betriebssystems konvertieren müssen, sollten Sie die folgende Anweisung verwenden:

```
SELECT MYFUNC( cscconvert( filename, 'os_charset' ) )
FROM mytable;
```

CUME_DIST-Funktion [Rangfolge]

Berechnet die relative Position eines Werts innerhalb einer Gruppe von Zeilen.

Syntax

CUME_DIST() OVER (*window-spec*)

Fensterspezifikation: Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Rückgabe

DOUBLE-Wert zwischen 0 und 1

Bemerkungen

Zusammengesetzte Sortierschlüssel sind derzeit in der CUME_DIST-Funktion nicht zulässig. Sie können allerdings zusammengesetzte Sortierschlüssel mit jeder anderen Rangfunktion verwenden.

Elemente von *Fensterspezifikation* können entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Bei Verwendung als Fensterfunktion

müssen Sie eine ORDER BY-Klausel angeben und dürfen eine PARTITION BY-Klausel angeben, können jedoch keine ROWS- oder RANGE-Klausel angeben. Weitere Hinweise finden Sie in der *Fensterspezifikation*-Definition unter „[WINDOW-Klausel](#)“ auf Seite 1124.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „[DENSE_RANK-Funktion \[Rangfolge\]](#)“ auf Seite 237
- „[PERCENT_RANK-Funktion \[Rangfolge\]](#)“ auf Seite 335
- „[RANK-Funktion \[Rangfolge\]](#)“ auf Seite 346

Standards und Kompatibilität

- **SQL/2008** Die CUME_DIST-Funktion umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T612, "Erweiterte OLAP-Vorgänge".

Beispiel

Das folgende Beispiel gibt eine Ergebnismenge zurück, die eine kumulative Verteilung der Gehälter von Mitarbeitern liefert, die in Kalifornien leben:

```
SELECT DepartmentID, Surname, Salary,
CUME_DIST() OVER (PARTITION BY DepartmentID
ORDER BY Salary DESC) "Rank"
FROM GROUP0.Employees
WHERE State IN ('CA');
```

Die Ergebnismenge sieht folgendermaßen aus:

DepartmentID	Surname	Salary	Rank
200	Savarino	72300.000	0.3333333333333333
200	Clark	45000.000	0.6666666666666667
200	Overbey	39300.000	1

DATALENGTH-Funktion [System]

Gibt in Byte die Länge der Basisspeicherung für das Ergebnis eines Ausdrucks zurück.

Syntax

DATALENGTH(*expression*)

Parameter

- **expression** In der Regel ein Spaltenname. Wenn *expression* eine Zeichenfolgenkonstante ist, muss er in Anführungszeichen eingeschlossen werden.

Rückgabe

UNSIGNED INT

Bemerkungen

Die Rückgabewerte der DATALENGTH-Funktion sind Folgende:

Datentyp	DATALENGTH
BIT	1
TINYINT	1
SMALLINT	2
INTEGER	4
BIGINT	8
REAL	4
DOUBLE	8
TIME	8
DATE	4
TIMESTAMP	8
DATETIME	8
TIMESTAMP WITH TIME ZONE	29
UNIQUEIDENTIFIER	16
CHAR	Länge der Daten
VARCHAR	Länge der Daten
BINARY	Länge der Daten
VARBINARY	Länge der Daten
NCHAR	Länge der Daten
NVARCHAR	Länge der Daten

Datentyp	DATALENGTH
TEXT	Länge der Daten
NTEXT	Länge der Daten
IMAGE	Länge der Daten
XML	Länge der Daten

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Siehe auch

- [„SQL-Datentypen“ auf Seite 95](#)

Beispiel

Die folgende Anweisung gibt die Länge der längsten Zeichenfolge in der CompanyName-Spalte zurück:

```
SELECT MAX( DATALENGTH( CompanyName ) )  
FROM GROUPO.Customers;
```

Die folgende Anweisung gibt die Länge der Zeichenfolge "8sdofinsv8s7a7s7gehe4h" zurück:

```
SELECT DATALENGTH( '8sdofinsv8s7a7s7gehe4h' );
```

DATE-Funktion [Datum und Uhrzeit]

Konvertiert den Ausdruck in ein Datum und entfernt Stunden, Minuten oder Sekunden.

Hinweise zur Steuerung der Interpretation von Datumsformaten finden Sie unter „[date_order-Option](#)“ [[SQL Anywhere Server - Datenbankadministration](#)] und „[nearest_century-Option](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Syntax

DATE(*expression*)

Rückgabe

DATE

Parameter

- **expression** Der Wert, der ins Datumsformat konvertiert werden soll. Dies ist üblicherweise eine Zeichenfolge.

Siehe auch

- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75
- „DATE-Datentyp“ auf Seite 123
- „DATETIME-Datentyp“ auf Seite 124
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „Ausdrücke“ auf Seite 22
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „TIMESTAMP-Datentyp“ auf Seite 129
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 1999-01-02 als Datum zurück:

```
SELECT DATE( '1999-01-02 21:20:53' );
```

Die folgende Anweisung gibt das Erstellungsdatum aller Objekte zurück, die in der SYSOBJECT-Systemansicht aufgelistet sind:

```
SELECT DATE( creation_time ) FROM SYSOBJECT;
```

DATEADD-Funktion [Datum und Uhrzeit]

Gibt einen TIMESTAMP- oder TIMESTAMP WITH TIME ZONE-Wert zurück, der durch Addieren eines Datumsteils zu ihrem Argument erzeugt wird.

Syntax

```
DATEADD( date-part, integer-expression, timestamp-expression )
```

date-part :

```
year
quarter
month
week
day
dayofyear
hour
minute
second
```

| millisecond
| microsecond

Parameter

- **Datumsteil** Der Datumsteil, den der *Zeichenfolgenausdruck* darstellt.

Eine komplette Enumeration zulässiger Datumsteile finden Sie unter [Angaben von Datumsteilen auf Seite 156](#).
- **Ganzzahlausdruck** Die Anzahl der *Datumsteil*-Werte, die zu *Zeitstempelausdruck* addiert werden sollen. *Zeichenfolgenausdruck* kann jeder numerische Typ sein, aber der Wert wird auf INTEGER gekürzt.
- **Zeitstempelausdruck** Der TIMESTAMP- oder TIMESTAMP WITH TIME ZONE-Wert, der geändert werden muss.

Rückgabe

TIMESTAMP WITH TIME ZONE, wenn *Zeitstempelausdruck* vom Typ TIMESTAMP WITH TIME ZONE ist, andernfalls TIMESTAMP.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den TIMESTAMP-Wert 1995-11-02 00:00:00.000 zurück:

```
SELECT DATEADD( month, 102, '1987/05/02' );
```

Die folgende Anweisung gibt den TIMESTAMP-Wert 1987-05-02 04:00:00.000 zurück:

```
SELECT DATEADD( hour, 4, '1987/05/02' );
```

Die folgende Anweisung gibt den TIMESTAMP WITH TIME ZONE-Wert 1987-05-06 11:33:00.000+04:00 zurück:

```
SELECT DATEADD( day, 4, CAST( '1987/05/02 11:33:00.000000+04:00' as  
TIMESTAMP WITH TIME ZONE ));
```

DATEDIFF-Funktion [Datum und Uhrzeit]

Gibt das Intervall zwischen zwei Datumsangaben zurück.

Syntax

DATEDIFF(*date-part*, *date-expression-1*, *date-expression-2*)

date-part :
year
| quarter
| month
| week

day
dayofyear
hour
minute
second
millisecond
microsecond

Parameter

- **Datumsteil** Bestimmt den Datumsteil, in dem das Intervall gemessen werden soll.

Wählen Sie eines der oben aufgelisteten Datumsobjekte. Eine komplette Liste der Datumsteile finden Sie unter [Angaben von Datumsteilen auf Seite 156](#).
- **Datumsausdruck_1** Das Startdatum für das Intervall. Dieser Wert wird von *Datumsausdruck_2* abgezogen, um die Anzahl von *Datumsteilen* zwischen den beiden Argumenten zu erhalten.
- **Datumsausdruck_2** Das Enddatum für das Intervall. *Datumsausdruck_1* wird von diesem Wert abgezogen, um die Anzahl von *Datumsteilen* zwischen den beiden Argumenten zu erhalten.

Rückgabe

INT mit Jahr, Quartal, Monat, Woche, Tag und Jahrestag. BIGINT mit Stunde, Minute, Sekunde, Millisekunde und Mikrosekunde.

Bemerkungen

Diese Funktion berechnet die Anzahl von Datumsteilen zwischen zwei angegebenen Datumsangaben. Das Ergebnis ist ein Ganzzahlwert mit Vorzeichen gleich (*Datumsausdruck_2* - *Datumsausdruck_1*), in Datumsteilen.

Ergebnisse der DATEDIFF-Funktion werden gekürzt und nicht gerundet, wenn das Ergebnis nicht ein gerades Mehrfaches des Datumsteils ist.

Wenn Sie **day** als den Datumsteil verwenden, gibt die DATEDIFF-Funktion die Anzahl der Mitternächte zwischen den angegebenen Zeitpunkten zurück, einschließlich des zweiten Datums, aber nicht des ersten.

Wenn Sie **month** als den Datumsteil angeben, gibt die DATEDIFF-Funktion die Anzahl der Monatsersten zwischen den Datumsangaben zurück, einschließlich des zweiten Datums, aber nicht des ersten.

Wenn Sie **week** als den Datumsteil verwenden, gibt die DATEDIFF-Funktion die Anzahl der Sonntage zwischen den Datumsangaben zurück, einschließlich des zweiten Datums, aber nicht des ersten.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 1 zurück:

```
SELECT DATEDIFF( hour, '4:00AM', '5:50AM' );
```

Die folgende Anweisung gibt den Wert 102 zurück:

```
SELECT DATEDIFF( month, '1987/05/02', '1995/11/15' );
```

Die folgende Anweisung gibt den Wert 0 zurück:

```
SELECT DATEDIFF( day, '00:00', '23:59' );
```

Die folgende Anweisung gibt den Wert 4 zurück:

```
SELECT DATEDIFF( day,  
    '1999/07/19 00:00',  
    '1999/07/23 23:59' );
```

Die folgende Anweisung gibt den Wert 0 zurück:

```
SELECT DATEDIFF( month, '1999/07/19', '1999/07/23' );
```

Die folgende Anweisung gibt den Wert 1 zurück:

```
SELECT DATEDIFF( month, '1999/07/19', '1999/08/23' );
```

DATEFORMAT-Funktion [Datum und Uhrzeit]

Gibt eine Zeichenfolge zurück, die einen Datumsausdruck im angegebenen Format darstellt.

Syntax

DATEFORMAT(*datetime-expression*, *string-expression*)

Parameter

- **DATETIME-Ausdruck** Die zu konvertierende Zeitangabe
- **Zeichenfolgenausdruck** Das Format des konvertierten Datums

Hinweise zu den Datumsformatbeschreibungen finden Sie unter „[timestamp_format-Option](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Rückgabe

VARCHAR

Bemerkungen

Jedes erlaubte Datumsformat kann für den Zeichenfolgenausdruck verwendet werden.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert "Jan 01, 1989" zurück:

```
SELECT DATEFORMAT( '1989-01-01', 'Mmm dd, yyyy' );
```

DATENAME-Funktion [Datum und Uhrzeit]

Gibt den Namen des angegebenen Teils eines TIMESTAMP- oder TIMESTAMP WITH TIME ZONE-Wertes als Zeichenfolge zurück (z.B. Monat Juni).

Syntax

DATENAME(*date-part*, *timestamp-expression*)

Parameter

- **Datumsteil** Der Datumsteil, der benannt werden soll.

Eine komplette Enumeration zulässiger Datumsteile finden Sie unter [Angaben von Datumsteilen auf Seite 156](#).

- **Zeitstempelausdruck** Der TIMESTAMP- oder TIMESTAMP WITH TIME ZONE-Wert, für den der Datumsteil zurückgegeben werden soll. Damit die Funktion sinnvolle Ergebnisse liefert, sollte *Zeitstempelausdruck* den angeforderten *Datumsteil* enthalten.

Rückgabe

VARCHAR

Bemerkungen

Die DATENAME-Funktion gibt eine Zeichenfolge zurück, sogar wenn das Ergebnis numerisch ist, wie "23" für den Tag. Wenn der Datumsteil TZOffset angegeben ist, gibt DATENAME den Offset als Zeichenfolge in der folgenden Form zurück: { + | - }hh:nn.

Siehe auch

- [„DATEPART-Funktion \[Datum und Uhrzeit\]“ auf Seite 221](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert "Mai" zurück:

```
SELECT DATENAME( month, '1987/05/02' );
```

DATEPART-Funktion [Datum und Uhrzeit]

Gibt einen Teil eines TIMESTAMP- oder TIMESTAMP WITH TIME ZONE-Werts zurück.

Syntax

DATEPART(*date-part*, *timestamp-expression*)

Parameter

- **Datumsteil** Der Datumsteil, der zurückgegeben werden soll.

Eine komplette Enumeration zulässiger Datumsteile finden Sie unter [Angeben von Datumsteilen auf Seite 156](#).

- **Zeitstempelausdruck** Der TIMESTAMP- oder TIMESTAMP WITH TIME ZONE-Wert, für den der Teil zurückgegeben werden soll.

Rückgabe

INT

Bemerkungen

Damit die Funktion sinnvolle Ergebnisse liefert, sollte *Zeitstempelausdruck* den angeforderten *Datumsteil*-Anteil enthalten.

Die den Wochentagen entsprechenden Zahlen hängen von der Einstellung der `first_day_of_week`-Option ab. Standardmäßig gilt: Sonntag=7.

Siehe auch

- „`first_day_of_week`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „SET-Anweisung [T-SQL]“ auf Seite 1056

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 5 zurück:

```
SELECT DATEPART( month , '1987/05/02' );
```

Im folgenden Beispiel wird eine Tabelle namens `TableStatistics` erstellt und in diese wird die Gesamtanzahl der Aufträge pro Jahr eingefügt, wie sie in der Tabelle "SalesOrders" gespeichert sind:

```
CREATE TABLE TableStatistics (
    ID INTEGER NOT NULL DEFAULT AUTOINCREMENT,
    Year INT,
    NumberOrders INT );
INSERT INTO TableStatistics ( Year, NumberOrders )
SELECT DATEPART( Year, OrderDate ), COUNT(*)
FROM GROUPO.SalesOrders
GROUP BY DATEPART( Year, OrderDate );
```

DATETIME-Funktion [Datum und Uhrzeit]

Konvertiert einen Ausdruck in einen TIMESTAMP-Wert.

Syntax

DATETIME(*expression*)

Parameter

- **expression** Der zu konvertierende Ausdruck. Er ist üblicherweise eine Zeichenfolge.

Rückgabe

TIMESTAMP

Bemerkungen

Der Versuch, numerische Werte zu konvertieren, erzeugt einen Fehler.

Siehe auch

- „Ausdrücke“ auf Seite 22
- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186
- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75
- „DATE-Datentyp“ auf Seite 123
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Datentyp“ auf Seite 124
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „TIMESTAMP-Datentyp“ auf Seite 129
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt einen Zeitstempel mit dem Wert "1998-09-09 12:12:12.000" zurück:

```
SELECT DATETIME( '1998-09-09 12:12:12.000' );
```

DAY-Funktion [Datum und Uhrzeit]

Gibt den Tag des Monats für das Argument als Ganzzahl zwischen 1 und 31 zurück.

Syntax

DAY(*date-expression*)

Parameter

- **Datumsausdruck** Das Datum als DATE-Datentyp.

Rückgabe

SMALLINT

Bemerkungen

Die DAY-Funktion gibt eine Ganzzahl zwischen 1 und 31 zurück, entsprechend dem Tag des Monats im Argument.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 12 zurück:

```
SELECT DAY( '2001-09-12' );
```

DAYNAME-Funktion [Datum und Uhrzeit]

Gibt den Namen des Wochentags von einem Datum zurück.

Syntax

DAYNAME(*date-expression*)

Parameter

- **Datumsausdruck** Das Datum.

Rückgabe

VARCHAR

Bemerkungen

Die Tagesnamen im Deutschen werden folgendermaßen zurückgegeben: Sonntag, Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert "Samstag" zurück:

```
SELECT DAYNAME ( '1987/05/02' );
```

DAYS-Funktion [Datum und Uhrzeit]

Bearbeitet einen TIMESTAMP-Wert oder gibt die Anzahl der Tage zwischen zwei TIMESTAMP-Werten zurück. Spezifische Details finden Sie im folgenden Abschnitt "Bemerkungen".

Syntax 1

DAYS(*timestamp-expression*)

Syntax 2

DAYS(*timestamp-expression*, *timestamp-expression*)

Syntax 3

DAYS(*timestamp-expression*, *integer-expression*)

Parameter

- **Zeitstempelausdruck** Ein TIMESTAMP-Wert.
- **Ganzzahlausdruck** Die Anzahl der Tage, die zu *Zeitstempelausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl an Tagen von *Zeitstempelausdruck* abgezogen. Wenn Sie einen Ganzzahlausdruck angeben, muss *Zeitstempelausdruck* explizit als TIME-, DATE- oder TIMESTAMP-Wert festgelegt sein. Wenn *Zeitstempelausdruck* ein TIME-Wert ist, wird das aktuelle Datum angenommen.

Hinweise zum Casting von Datentypen finden Sie unter „[CAST-Funktion \[Datentypkonvertierung\]](#)“ auf Seite 186.

Rückgabe

INTEGER bei Syntax 1 und Syntax 2.

TIMESTAMP bei Syntax 3.

Bemerkungen

Das Ergebnis der DAYS-Funktion hängt von deren Argumenten ab. Die DAYS-Funktion ignoriert Stunden, Minuten, und Sekunden in ihren Argumenten.

- **Syntax 1** Wenn Sie einen einzelnen *Zeitstempelausdruck* an die DAYS-Funktion übergeben, wird die Anzahl der Tage zwischen 0000-02-29 und *Zeitstempelausdruck* als INTEGER-Wert zurückgegeben.

Hinweis

"0000-02-29" gibt kein tatsächliches Datum wieder. Es ist das von der DAYS-Funktion verwendete Standarddatum.

- **Syntax 2** Wenn Sie zwei TIMESTAMP-Werte an die DAYS-Funktion übergeben, liefert die Funktion die als Ganzzahl ausgedrückte Anzahl der Tage zwischen den beiden.
- **Syntax 3** Wenn Sie einen TIMESTAMP-Wert und eine Ganzzahl an die DAYS-Funktion übergeben, liefert die Funktion das TIMESTAMP-Ergebnis nach dem Addieren der durch die Ganzzahl ausgedrückten Anzahl an Tagen zum *Zeitstempelausdruck*-Argument.

Verwenden Sie statt Syntax 2 die DATEDIFF-Funktion. Verwenden Sie statt Syntax 3 die DATEADD-Funktion.

Siehe auch

- „DATEDIFF-Funktion [Datum und Uhrzeit]“ auf Seite 218
- „DATEADD-Funktion [Datum und Uhrzeit]“ auf Seite 217

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die Ganzzahl 729889 zurück:

```
SELECT DAYS( '1998-07-13 06:07:12' );
```

Die folgenden Anweisungen geben den Ganzzahlwert -366 zurück, womit angezeigt wird, dass der zweite DATE-Wert 366 Tage vor dem ersten liegt. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEDIFF):

```
SELECT DAYS( '1998-07-13 06:07:12',  
            '1997-07-12 10:07:12' );
```

```
SELECT DATEDIFF( day,  
                '1998-07-13 06:07:12',  
                '1997-07-12 10:07:12' );
```

Die folgenden Anweisungen geben den TIMESTAMP-Wert 1999-07-14 00:00:00.000 zurück. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEADD):

```
SELECT DAYS( CAST('1998-07-13' AS DATE ), 366 );  
  
SELECT DATEADD( day, 366, '1998-07-13' );
```

DB_EXTENDED_PROPERTY-Funktion [System]

Gibt den Wert der gegebenen Eigenschaft zurück. Es kann ein optionaler eigenschaftsspezifischer Zeichenfolgenparameter angegeben werden.

Syntax

```
DB_EXTENDED_PROPERTY(  
  { property-id | property-name }  
  [, property-specific-argument  
  [, database-id | database-name ] ]  
)
```

Parameter

- ***property-id*** Die abzufragende Kennung der Datenbankeigenschaft.
- ***Eigenschaftsname*** Der abzufragende Name der Datenbankeigenschaft

Eine vollständige Liste der Datenbankeigenschaften finden Sie unter „Liste der Datenbankeigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)].

- **Eigenschaftsspezifisches_Argument** Mit den folgenden Datenbankeigenschaften können Sie zusätzliche, im Folgenden beschriebene Argumente angeben, um spezifische Informationen zur Eigenschaft zurückzugeben.
 - **CharSet-Eigenschaft** Gibt den Namen eines Standardwerts an, um das Standard-CHAR-Zeichensatzlabel für den Standardwert zu erhalten. Mögliche Werte sind: ASE, IANA, MIME, JAVA, WINDOWS, UTR22, IBM und ICU. Wenn kein Standard angegeben ist, wird standardmäßig IANA verwendet, außer die Datenbankverbindung wurde durch TDS hergestellt. In diesem Fall ist ASE der Standardwert.
 - **Eigenschaften CatalogCollation, Collation und NcharCollation** Bei der Abfrage dieser Eigenschaften können Sie die folgenden Werte als *Eigenschaftsspezifisches_Argument* angeben, um spezifische Informationen für die Kollation zurückzugeben:
 - **AccentSensitive** Geben Sie AccentSensitive an, um die Einstellung der Akzentberücksichtigung für die Kollation abzurufen. Beispiel: Die folgende Anweisung gibt die Akzentberücksichtigung bei der NCHAR-Kollation zurück:

```
SELECT DB_EXTENDED_PROPERTY( 'NcharCollation', 'AccentSensitive');
```

Mögliche Rückgabewerte sind: Ignore, Respect und French. Weitere Hinweise finden Sie unter „Optionen der Kollationsanpassung“ [[SQL Anywhere Server - Datenbankadministration](#)].

- **CaseSensitivity** Geben Sie CaseSensitivity an, um die Berücksichtigung der Groß-/Kleinschreibung für die Kollation abzurufen. Mögliche Rückgabewerte sind: Ignore, Respect, UpperFirst und LowerFirst. Weitere Hinweise finden Sie unter „Optionen der Kollationsanpassung“ [[SQL Anywhere Server - Datenbankadministration](#)].
- **PunctuationSensitivity** Geben Sie PunctuationSensitivity an, um die Berücksichtigung der Satzzeichen für die Kollation abzurufen. Mögliche Rückgabewerte sind: Ignore, Primary und Quaternary. Weitere Hinweise finden Sie unter „Optionen der Kollationsanpassung“ [[SQL Anywhere Server - Datenbankadministration](#)].
- **Eigenschaften** Geben Sie Properties an, um eine Zeichenfolge zu erhalten, die alle für die Kollation angegebenen Optionen der Kollationsanpassung enthält. Eine Beschreibung der Schlüsselwörter und der Werte in der zurückgegebenen Zeichenfolge finden Sie unter „Optionen der Kollationsanpassung“ [[SQL Anywhere Server - Datenbankadministration](#)].
- **Spezifikation** Geben Sie Specification an, um eine Zeichenfolge mit der vollständigen Spezifikation zu erhalten, die für die Kollation verwendet wird. Eine Beschreibung der Schlüsselwörter und der Werte in der zurückgegebenen Zeichenfolge finden Sie unter „Optionen der Kollationsanpassung“ [[SQL Anywhere Server - Datenbankadministration](#)].
- **DriveType-Eigenschaft** Geben Sie den Namen eines DBSpaces oder die Datei-ID für den DBSpace an, um seinen Speichermediumtyp zu erhalten. Der zurückgegebene Wert ist einer der folgenden: CD, FIXED, RAMDISK, REMOTE, REMOVABLE oder UNKNOWN. Wenn keine Angabe gemacht wird, wird der Typ des Speichermediums des System-DBSpaces zurückgegeben. Wenn der angegebene DBSpace nicht existiert, gibt die Eigenschaftsfunktion NULL zurück. Wenn der Name des DBSpaces angegeben ist und die ID einer Datenbank, die nicht die

Datenbank der aktuellen Verbindung ist, auch angegeben ist, gibt die Funktion ebenfalls NULL zurück.

- **File-Eigenschaft** Geben Sie einen DBSpace-Namen an, um den Dateinamen der Datenbank-Stammdatei einschließlich des Pfads zu erhalten. Wenn keine Angabe gemacht wird, werden Informationen für den System-DBSpace zurückgegeben. Wenn die angegebene Datei nicht existiert, gibt die Funktion NULL zurück.
- **FileSize-Eigenschaft** Geben Sie den Namen eines DBSpaces oder die Datei-ID für den DBSpace an, um die Größe der angegebenen Datei in Seiten zu erhalten. Sie können auch "temporary" angeben, um die Größe des temporären DBSpace zurückzugeben, oder "translog", um die Größe der Logdatei zurückzugeben. Wenn keine Angabe gemacht wird, wird die Größe des System-DBSpaces zurückgegeben. Wenn die angegebene Datei nicht existiert, gibt die Funktion NULL zurück.
- **FreePages-Eigenschaft** Geben Sie den Namen eines DBSpaces oder die Datei-ID für den DBSpace an, um die Anzahl der freien Seiten zu erhalten. Sie können auch "temporary" angeben, um die Anzahl der freien Seiten im temporären DBSpace zurückzugeben, oder "translog", um die Anzahl der freien Seiten in der Logdatei zurückzugeben. Wenn keine Angabe gemacht wird, wird die Anzahl der freien Seiten im System-DBSpace zurückgegeben. Wenn die angegebene Datei nicht existiert, gibt die Funktion NULL zurück.
- **IOParallelism-Eigenschaft** Geben Sie den Namen eines DBSpaces an, um die geschätzte Anzahl der simultanen vom DBSpace unterstützten I/O-Vorgänge zu erhalten. Wenn kein DBSpace angegeben ist, wird der aktuelle System-DBSpace verwendet.
- **MirrorServerState-Eigenschaft** Geben Sie einen Servernamen an, um den Verbindungsstatus des Spiegelservers zu ermitteln. Gibt "connected", "disconnected", "incoming only", "outgoing only" oder NULL zurück.
- **MirrorState-Eigenschaft** Geben Sie einen Servernamen an, um den Synchronisationsstatus des Spiegelservers zu ermitteln. Gibt "synchronizing", "synchronized" oder NULL zurück.
- **NextScheduleTime-Eigenschaft** Geben Sie einen Ereignisnamen an, um seine nächste geplante Ausführungszeit zu erhalten.
- **Datenbank-ID** Die Datenbank-ID-Nummer, wie von der DB_ID-Funktion zurückgegeben. Üblicherweise wird der Datenbankname verwendet.
- **Datenbankname** Der Name der Datenbank, wie von der DB_NAME-Funktion zurückgegeben

Rückgabe

VARCHAR

Bemerkungen

Die DB_EXTENDED_PROPERTY-Funktion ist ähnlich der DB_PROPERTY-Funktion, mit dem Unterschied, dass hier ein optionaler Zeichenfolgenparameter *Eigenschaftsspezifisches_Argument* angegeben werden kann. Die Interpretation von *Eigenschaftsspezifisches_Argument* hängt von der Eigenschafts-ID bzw. dem im ersten Argument angegebenen Namen ab.

Wenn das dritte Argument weggelassen wurde, wird die aktuelle Datenbank verwendet.

Beim Vergleichen von Katalogzeichenfolgen, wie z.B. Tabellen- und Prozedurnamen, verwendet der Datenbankserver die CHAR-Kollation. Bei UCA-Kollationen ist die Katalogkollation zwar mit der CHAR-Kollation identisch, aber die Anpassung ist so geändert, dass die Groß- und Kleinschreibung sowie Akzente nicht berücksichtigt und Satzzeichen auf der ersten Ebene sortiert werden. Bei Sortierkollationen ist die Katalogkollation zwar mit der CHAR-Kollation identisch, aber die Anpassung ist so geändert, dass die Groß- und Kleinschreibung nicht berücksichtigt wird. Sie können die für Katalogkollationen verwendeten Anpassung zwar nicht explizit festlegen, haben aber die Möglichkeit, die Specification-Eigenschaft abzufragen, um die vollständige Kollationsspezifikation zu erhalten, die vom Datenbankserver für den Vergleich von Katalogzeichenfolgen verwendet wird. Das Abfragen der Specification-Eigenschaft kann nützlich sein, wenn Sie die Differenz zwischen der CHAR- und der Katalogkollation nutzen müssen. Beispiel: Sie haben eine CHAR-Kollation, bei der keine Satzzeichen berücksichtigt werden, und möchten ein Upgrade-Skript ausführen, das eine Prozedur mit dem Namen `my_procedure` definiert und versucht, eine alte Version mit dem Namen `myprocedure` zu löschen. Die folgenden Anweisungen können die gewünschten Ergebnisse nicht erzielen, da `my_procedure` bei Verwendung der CHAR-Kollation mit `myprocedure` gleichwertig ist:

```
CREATE PROCEDURE my_procedure( ) ...;
IF EXISTS ( SELECT * FROM SYS.SYSPROCEDURE WHERE proc_name = 'myprocedure' )
THEN DROP PROCEDURE myprocedure
END IF;
```

Stattdessen können Sie folgende Anweisungen ausführen, um die gewünschten Ergebnisse zu erzielen:

```
CREATE PROCEDURE my_procedure( ) ...;
IF EXISTS ( SELECT * FROM SYS.SYSPROCEDURE
  WHERE COMPARE( proc_name, 'myprocedure',
    DB_EXTENDED_PROPERTY( 'CatalogCollation', 'Specification' ) ) = 0 )
THEN DROP PROCEDURE myprocedure
END IF;
```

☛ **Cloud-Hinweis:** Aufgrund von Isolationsregeln für Tenant-Datenbanken gibt diese Funktion bei einem Aufruf in der Cloud nur Informationen über die aktuelle Tenant-Datenbank zurück.

Privilegien

Keine Privilegien sind erforderlich, um diese Funktion für die aktuelle Datenbank auszuführen. Um diese Funktion für andere Datenbanken ausführen zu können, benötigen Sie das `SERVER OPERATOR`-Systemprivileg oder das `MONITOR`-Systemprivileg.

NULL wird zurückgegeben, wenn Sie einen ungültigen Parameterwert angeben oder eines der erforderlichen Systemprivilegien nicht haben.

Siehe auch

- „[DB_ID-Funktion \[System\]](#)“ auf Seite 230
- „[DB_NAME-Funktion \[System\]](#)“ auf Seite 231
- „[Liste der Datenbankeigenschaften](#)“ [*SQL Anywhere Server - Datenbankadministration*]
- „[CONNECTION_PROPERTY-Funktion \[System\]](#)“ auf Seite 199
- „[CONNECTION_EXTENDED_PROPERTY-Funktion \[Zeichenfolge\]](#)“ auf Seite 197

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Speicherort der aktuellen Datenbank zurück:

```
SELECT DB_EXTENDED_PROPERTY( 'File' );
```

Die folgende Anweisung gibt die Dateigröße des System-DBSpace in Seiten zurück:

```
SELECT DB_EXTENDED_PROPERTY( 'FileSize' );
```

Die folgende Anweisung gibt die Dateigröße des Transaktionslogs in Seiten zurück:

```
SELECT DB_EXTENDED_PROPERTY( 'FileSize', 'translog' );
```

Die folgende Anweisung gibt die Einstellung in Bezug auf die Berücksichtigung der Groß-/Kleinschreibung bei der NCHAR-Kollation zurück:

```
SELECT DB_EXTENDED_PROPERTY( 'NcharCollation', 'CaseSensitivity' );
```

Die folgende Anweisung gibt die Optionen zurück, die für die CHAR-Kollation der Datenbank festgelegt wurden:

```
SELECT DB_EXTENDED_PROPERTY ( 'Collation', 'Properties' );
```

Die folgende Anweisung gibt die vollständige Kollationsspezifikation zurück, die für die NCHAR-Kollation der Datenbank festgelegt wurde:

```
SELECT DB_EXTENDED_PROPERTY( 'NcharCollation', 'Specification' );
```

Die folgende Anweisung liefert den Verbindungsstatus des Spiegelservers "Test":

```
SELECT DB_EXTENDED_PROPERTY( 'MirrorServerState', 'Test' );
```

Die folgende Anweisung liefert den Synchronisationsstatus der Spiegelservers "Test":

```
SELECT DB_EXTENDED_PROPERTY( 'MirrorState', 'Test' );
```

DB_ID-Funktion [System]

Gibt die Datenbank-ID-Nummer zurück.

Syntax

```
DB_ID( [ database-name ] )
```

Parameter

- **Datenbankname** Eine Zeichenfolge, die den Datenbanknamen enthält. Wenn kein *Datenbankname* angegeben ist, wird die ID-Nummer der aktuellen Datenbank zurückgegeben.

Rückgabe

INT

Bemerkungen

🔥 **Cloud-Hinweis:** Aufgrund von Isolationsregeln für Tenant-Datenbanken gibt diese Funktion bei einem Aufruf in der Cloud nur Informationen über die aktuelle Tenant-Datenbank zurück.

Privilegien

Keine

Siehe auch

- „global_database_id-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 0 zurück, wenn sie in der SQL Anywhere-Beispieldatenbank als einziger Datenbank auf dem Server ausgeführt wird:

```
SELECT DB_ID( 'demo' );
```

Die folgende Anweisung gibt den Wert 0 zurück, wenn sie in der einzigen laufenden Datenbank ausgeführt wird:

```
SELECT DB_ID( );
```

DB_NAME-Funktion [System]

Gibt den Namen einer Datenbank mit einer angegebenen ID-Nummer zurück.

Syntax

```
DB_NAME( [ database-id ] )
```

Parameter

- **Datenbank-ID** Die Kennung der Datenbank. Die *database-id* muss ein numerischer Ausdruck sein.

Rückgabe

VARCHAR

Bemerkungen

Wenn keine Datenbank-ID angegeben wird, wird der Name der aktuellen Datenbank zurückgegeben.

🔥 **Cloud-Hinweis:** Aufgrund von Isolationsregeln für Tenant-Datenbanken gibt diese Funktion bei einem Aufruf in der Cloud nur Informationen über die aktuelle Tenant-Datenbank zurück.

Privilegien

Keine Privilegien sind erforderlich, um diese Funktion für die aktuelle Datenbank auszuführen. Um diese Funktion für andere Datenbanken ausführen zu können, benötigen Sie das SERVER OPERATOR-Systemprivileg oder das MONITOR-Systemprivileg.

Siehe auch

- [„sa_db_list-Systemprozedur“ auf Seite 1197](#)
- [„NEXT_DATABASE-Funktion \[System\]“ auf Seite 324](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Datenbanknamen "demo" zurück, wenn sie in der SQL Anywhere-Beispieldatenbank als einziger Datenbank auf dem Server ausgeführt wird:

```
SELECT DB_NAME( 0 );
```

DB_PROPERTY-Funktion [System]

Gibt den Wert der gegebenen Eigenschaft zurück.

Syntax

```
DB_PROPERTY(  
  { property-id | property-name }  
  [, database-id | database-name ]  
)
```

Parameter

- ***property-id*** Die ID der Datenbankeigenschaft.
- ***Eigenschaftsname*** Der Name der Datenbankeigenschaft.
- ***Datenbank-ID*** Die Datenbank-ID-Nummer, wie von der DB_ID-Funktion zurückgegeben. Üblicherweise wird der Datenbankname verwendet.
- ***Datenbankname*** Der Name der Datenbank, wie von der DB_NAME-Funktion zurückgegeben

Rückgabe

VARCHAR, LONG VARCHAR

Bemerkungen

Gibt eine Zeichenfolge zurück. Wenn das zweite Argument weggelassen wurde, wird die aktuelle Datenbank verwendet.

🔥 **Cloud-Hinweis:** Aufgrund von Isolationsregeln für Tenant-Datenbanken gibt diese Funktion bei einem Aufruf in der Cloud nur Informationen über die aktuelle Tenant-Datenbank zurück.

Privilegien

Keine Privilegien sind erforderlich, um diese Funktion für die aktuelle Datenbank auszuführen. Um diese Funktion für andere Datenbanken ausführen zu können, benötigen Sie das SERVER OPERATOR-Systemprivileg oder das MONITOR-Systemprivileg.

NULL wird zurückgegeben, wenn Sie einen ungültigen Parameterwert angeben oder eines der erforderlichen Systemprivilegien nicht haben.

Siehe auch

- „DB_ID-Funktion [System]“ auf Seite 230
- „DB_NAME-Funktion [System]“ auf Seite 231
- „Liste der Datenbankeigenschaften“ [*SQL Anywhere Server - Datenbankadministration*]
- „PROPERTY-Funktion [System]“ auf Seite 339

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die Seitengröße der aktuellen Datenbank in Byte zurück:

```
SELECT DB_PROPERTY( 'PageSize' );
```

DECOMPRESS-Funktion [Zeichenfolge]

Dekomprimiert die Zeichenfolge und gibt einen LONG BINARY-Wert zurück.

Syntax

DECOMPRESS(*string-expression* [, *compression-algorithm-alias*])

Parameter

- **Zeichenfolgenausdruck** Die zu dekomprimierende Zeichenfolge. Auch binäre Werte können an diese Funktion übergeben werden. Dieser Parameter beachtet die Groß- und Kleinschreibung, sogar in Datenbanken, die das nicht tun.
- **compression-algorithm-alias** Alias (Zeichenfolge) für den bei der Dekomprimierung zu verwendenden Algorithmus. Die unterstützten Werte sind "zip" und "gzip". (Beide basieren auf demselben Algorithmus, verwenden aber unterschiedliche Header und Trailer.)

Zip ist ein allgemein unterstützter Komprimierungsalgorithmus. Gzip ist mit dem gzip-Dienstprogramm unter Unix kompatibel, der zip-Algorithmus hingegen nicht.

Wenn kein Algorithmus angegeben ist, versucht die Funktion herauszufinden, welcher Algorithmus zum Komprimieren der Zeichenfolge verwendet wurde. Wenn ein falscher Algorithmus angegeben wird oder wenn der korrekte Algorithmus nicht ermittelt werden kann, wird die Zeichenfolge nicht dekomprimiert.

Weitere Hinweise zur Komprimierung finden Sie unter „[COMPRESS-Funktion \[Zeichenfolge\]](#)“ auf Seite 193.

Rückgabe

LONG BINARY

Bemerkungen

Diese Funktion kann zur Dekomprimierung eines Werts verwendet werden, der mithilfe der COMPRESS-Funktion komprimiert wurde.

Sie sollten die DECOMPRESS-Funktion nicht bei Werten verwenden, die in einer komprimierten Spalte gespeichert werden. Das Komprimieren und Dekomprimieren von Werten in einer komprimierten Spalte wird vom Datenbankserver automatisch durchgeführt.

Siehe auch

- „Hinweise zur Spaltenkomprimierung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „COMPRESS-Funktion [Zeichenfolge]“ auf Seite 193
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel verwendet die DECOMPRESS-Funktion, um Werte in der Spalte "Attachment" einer fiktiven Tabelle namens "TableA" zu dekomprimieren:

```
SELECT DECOMPRESS ( Attachment, 'gzip' )  
FROM TableA;
```

Da DECOMPRESS binäre Werte zurückgibt, wenn die ursprünglichen Werte von einem Zeichentyp wie z.B. LONG VARCHAR waren, kann CAST verwendet werden, um lesbare Werte zu erhalten:

```
SELECT CAST ( DECOMPRESS ( Attachment, 'gzip' )  
AS LONG VARCHAR ) FROM TableA;
```

DECRYPT-Funktion [Zeichenfolge]

Dechiffriert die Zeichenfolge unter Verwendung des gelieferten Schlüssels und gibt einen LONG BINARY-Wert zurück.

Syntax

```
DECRYPT( string-expression, key  
[, algorithm ]  
)
```

algorithm :

```
'AES'  
| 'AES256'  
| 'AES_FIPS'  
| 'AES256_FIPS'  
[ format ]
```

format:
 (**FORMAT=RAW** [:padding]) [initialization-vector])

padding:
PADDING=PKCS5
 | **ZEROES**
 | **NONE**]

Parameter

- **Zeichenfolgenausdruck** Die zu dechiffrierende Zeichenfolge. Auch binäre Werte können an diese Funktion übergeben werden. Dieser Parameter beachtet die Groß- und Kleinschreibung, sogar in Datenbanken, die das nicht tun.
- **key** Der zum Entschlüsseln von *Zeichenfolgenausdruck* erforderliche Chiffrierschlüssel (Zeichenfolge). Dieser Wert muss derselbe Chiffrierschlüssel sein, der zum Verschlüsseln von *Zeichenfolgenausdruck* verwendet wurde, um den ursprünglichen Wert zu erhalten. Dieser Parameter beachtet die Groß- und Kleinschreibung, sogar in Datenbanken, die das nicht tun.

Vorsicht

Achten Sie bei stark verschlüsselten Datenbanken darauf, eine Kopie des Schlüssels an einem sicheren Ort zu verwahren. Wenn Sie den Chiffrierschlüssel verlieren, gibt es keine Möglichkeit, auf die Daten zuzugreifen, auch nicht mit Unterstützung durch den technischen Support. Die Datenbank muss verworfen und eine neue Datenbank muss erstellt werden.

- **algorithm** Dieser optionale Parameter legt den Algorithmus fest, der ursprünglich zum Verschlüsseln von *Zeichenfolgenausdruck* verwendet wird.

FORMAT=RAW Dieser optionale Parameter gibt an, dass die zu entschlüsselnden Daten im RAW-Format vorliegen. Der *Initialisierungsvektor*-Parameter ist erforderlich.

Auffüllen_mit_Zeichen Geben Sie den Typ des Auffüllens mit Zeichen an, der zum Verschlüsseln der Daten verwendet wurde. Wenn *Auffüllen_mit_Zeichen* nicht angegeben ist, wird standardmäßig PKCS5 verwendet.

Die folgenden Formate zum Auffüllen mit Zeichen werden unterstützt:

PKCS5 Die Daten werden gemäß dem PKCS#5-Algorithmus mit Zeichen aufgefüllt. Die entschlüsselten Daten enthalten Auffüllzeichen.

ZEROES Die Daten werden mit Nullen (0) aufgefüllt. Die entschlüsselten Daten sind mit Nullen aufgefüllt.

NONE Die Daten werden nicht mit Zeichen aufgefüllt. Die entschlüsselten Daten enthalten keine Auffüllzeichen.

Initialisierungsvektor Geben Sie den Initialisierungsvektor an, der zum Verschlüsseln der Daten verwendet wurde. Dieser Parameter ist erforderlich.

Rückgabe

LONG BINARY

Bemerkungen

Sie können die DECRYPT-Funktion verwenden, um einen *Zeichenfolgenausdruck* zu entschlüsseln, der mit der ENCRYPT-Funktion verschlüsselt wurde. Diese Funktion gibt einen LONG BINARY-Wert mit derselben Anzahl von Bytes wie die Eingabezeichenfolge zurück, es sei denn, die Daten liegen im RAW-Format vor. Wenn FORMAT=RAW angegeben ist, hängt die Länge des zurückgegebenen Werts vom Format zum Auffüllen mit Zeichen ab.

Um einen *Zeichenfolgenausdruck* erfolgreich zu entschlüsseln, müssen Sie denselben Chiffrierschlüssel verwenden wie zum Verschlüsseln der Daten. Wenn FORMAT=RAW angegeben ist, müssen Sie außerdem denselben Initialisierungsvektor und dasselbe Format zum Auffüllen mit Zeichen verwenden wie zum Verschlüsseln der Daten. Daten im RAW-Format können außerhalb des Datenbankservers entschlüsselt werden.

Wenn Sie einen falschen Chiffrierschlüssel angeben, wird ein Fehler generiert, es sei denn, FORMAT=RAW ist angegeben. Wenn Sie FORMAT=RAW angeben und der angegebene Chiffrierschlüssel oder Initialisierungsvektor falsch ist, schlägt die Entschlüsselung ohne Meldung fehl.

Vorsicht

Achten Sie bei stark verschlüsselten Daten darauf, eine Kopie des Schlüssels an einem sicheren Ort zu verwahren. Wenn Sie den Chiffrierschlüssel verlieren, gibt es keine Möglichkeit, auf die Daten zuzugreifen, auch nicht mit Unterstützung durch den technischen Support.

Hinweis

Nicht alle Plattformen unterstützen FIPS-zertifizierte Verschlüsselung. Eine Liste der unterstützten Plattformen finden Sie unter <http://www.sybase.com/detail?id=1061806>.

Siehe auch

- „RAW-Verschlüsselung“ [*SQL Anywhere Server - Datenbankadministration*]
- „ENCRYPT-Funktion [Zeichenfolge]“ auf Seite 241
- „ISENCRYPTED Funktion [System]“ auf Seite 294
- „Spalten- und Tabellenverschlüsselung“ [*SQL Anywhere Server - Datenbankadministration*]
- „Zeichenfolgenfunktionen“ auf Seite 163
- „Datenbankserveroption -fips“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel dechiffriert das Kennwort eines Benutzers aus der Tabelle "user_info". Die CAST-Funktion wird verwendet, um das Kennwort zurück in einen CHAR-Datentyp zu konvertieren, weil die DECRYPT-Funktion Werte in den LONG BINARY-Datentyp konvertiert, der nicht darstellbar ist.

```
SELECT CAST( DECRYPT( user_pwd, '8U3dkA' ) AS CHAR(100) ) FROM user_info;
```

Im folgenden Beispiel werden Daten entschlüsselt, die im RAW-Format verschlüsselt wurden. Die Daten wurden mit dem Chiffrierschlüssel TheEncryptionKey und dem Initialisierungsvektor ThisIsTheIV verschlüsselt.

```
SELECT DECRYPT( binary_data, 'TheEncryptionKey',
'AES(format=raw;padding=zeros)', 'ThisIsTheIV'),
LENGTH(binary_data) FROM SensitiveData;
```

DEGREES-Funktion [Numerisch]

Konvertiert eine Zahl von Bogenmaß in Grad.

Syntax

DEGREES(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Ein Winkel im Bogenmaß

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihr Argument in DOUBLE, führt die Berechnung als doppelgenaue Gleitkommazahl durch und gibt die Gradzahl des durch *Numerischer_Ausdruck* angegebenen Winkels zurück. Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 29,79380534680281 zurück:

```
SELECT DEGREES( 0.52 );
```

DENSE_RANK-Funktion [Rangfolge]

Berechnet den Rang eines Werts in einer Partition. Bei gebundenen Werten verhindert die DENSE_RANK-Funktion Lücken in der Rangfolge-Sequenz.

Syntax

DENSE_RANK() **OVER** (*window-spec*)

Fensterspezifikation: Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Rückgabe

INTEGER

Bemerkungen

Elemente von *Fensterspezifikation* können entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Bei Verwendung als Fensterfunktion

müssen Sie eine ORDER BY-Klausel angeben und dürfen eine PARTITION BY-Klausel angeben, können jedoch keine ROWS- oder RANGE-Klausel angeben. Weitere Hinweise finden Sie in der *Fensterspezifikation*-Definition der WINDOW-Klausel.

Siehe auch

- „WINDOW-Klausel“ auf Seite 1124
- „CUME_DIST-Funktion [Rangfolge]“ auf Seite 213
- „PERCENT_RANK-Funktion [Rangfolge]“ auf Seite 335
- „RANK-Funktion [Rangfolge]“ auf Seite 346
- „Fensterfunktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Die DENSE_RANK-Funktion umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T612, "Erweiterte OLAP-Vorgänge".

SQL Anywhere unterstützt die SQL/2008 -Sprachenfunktion F441, "Extended set function support", die es zulässt, dass Operanden von Fensterfunktionen beliebige Ausdrücke sind, die keine Spaltenreferenzen darstellen.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der DENSE_RANK-Funktion als auch eine äußere Referenz enthalten. Ein Beispiel finden Sie unter „AVG-Funktion [Aggregat]“ [*UltraLite - Datenbankverwaltung*].

Beispiel

Das folgende Beispiel gibt eine Ergebnismenge zurück, die die Rangfolge der Gehälter von Mitarbeitern in Utah und New York liefert. Auch wenn 19 Datensätze in der Ergebnismenge zurückgegeben werden, sind nur 18 Ränge angeführt, weil der 7. Rang vom 7. und 8. Mitarbeiter, die beide das gleiche Gehalt erhalten, geteilt wird. Der 9. Mitarbeiter wird nicht an 9., sondern an 8. Stelle aufgereiht, weil die DENSE_RANK-Funktion Lücken in der Rangfolge verhindert.

```
SELECT DepartmentID, Surname, Salary, State,  
       DENSE_RANK() OVER (ORDER BY Salary DESC) AS SalaryRank  
FROM GROUPO.Employees  
WHERE State IN ('NY','UT');
```

Die Ergebnismenge sieht folgendermaßen aus:

DepartmentID	Surname	Salary	State	SalaryRank
100	Shishov	72995.000	UT	1
100	Wang	68400.000	UT	2
100	Cobb	62000.000	UT	3

DepartmentID	Surname	Salary	State	SalaryRank
400	Morris	61300.000	UT	4
300	Davidson	57090.000	NY	5
200	Martel	55700.000	NY	6
400	Blaikie	54900.000	NY	7
100	Diaz	54900.000	UT	7
100	Driscoll	48023.000	UT	8
400	Hildebrand	45829.000	UT	9
100	Whitney	45700.000	NY	10
100	Guevara	42998.000	NY	11
100	Soo	39075.000	NY	12
200	Goggin	37900.000	UT	13
400	Wetherby	35745.000	NY	14
400	Ahmed	34992.000	NY	15
500	Rebeiro	34576.000	UT	16
300	Bigelow	31200.000	UT	17
500	Lynch	24903.000	UT	18

DIFFERENCE-Funktion [Zeichenfolge]

Gibt die Differenz der SOUNDEX-Werte zwischen den beiden Zeichenfolgenausdrücken zurück.

Syntax

DIFFERENCE (*string-expression-1*, *string-expression-2*)

Parameter

- **Zeichenfolgenausdruck_1** Das erste SOUNDEX-Argument.
- **Zeichenfolgenausdruck_2** Das zweite SOUNDEX-Argument.

Rückgabe

SMALLINT

Bemerkungen

Die DIFFERENCE-Funktion vergleicht die SOUNDEX-Werte von zwei Zeichenfolgen und berechnet die Ähnlichkeit zwischen ihnen, wobei ein Wert von "0" bis "4" zurückgegeben wird ("4" entspricht der höchsten Übereinstimmung).

Diese Funktion gibt immer einen Wert zurück. Das Ergebnis ist nur NULL, wenn eines der Argumente NULL ist.

Siehe auch

- [„SOUNDEX-Funktion \[Zeichenfolge\]“ auf Seite 388](#)
- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung liefert die Ähnlichkeit zwischen den Wörtern "test" und "chest":

```
SELECT DIFFERENCE( 'test', 'chest' );
```

DOW-Funktion [Datum und Uhrzeit]

Gibt eine Zahl von 1 bis 7 zurück, die den Wochentag eines angegebenen Datums repräsentiert, wobei Sonntag=1, Montag=2 usw. ist.

Syntax

DOW(*date-expression*)

Parameter

- **Datumsausdruck** Der Wert (vom Typ DATE), der ausgewertet werden soll.

Rückgabe

SMALLINT

Bemerkungen

Die DOW-Funktion wird durch den in der Datenbankoption `first_day_of_week` angegebenen Wert nicht beeinflusst. Beispiel: Selbst wenn `first_day_of_week` auf Montag gesetzt ist, gibt die DOW-Funktion eine 2 für Montag zurück.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 5 zurück:

```
SELECT DOW( '1998-07-09' );
```

Die folgende Anweisung gibt den Wert 1 zurück:

```
SELECT DOW( CAST( '2010/05/30 11:33:00.000000+04:00' as TIMESTAMP WITH TIME
ZONE ) );
```

Die folgende Anweisung fragt die Tabelle "Employees" ab und gibt den StartDate-Wert des Mitarbeiters zurück, ausgedrückt als die Nummer des Wochentags:

```
SELECT DOW( StartDate ) FROM GROUPO.Employees;
```

ENCRYPT-Funktion [Zeichenfolge]

Verschlüsselt die angegebenen Werte unter Verwendung des gelieferten Chiffrierschlüssels und gibt einen LONG BINARY-Wert zurück.

Syntax

```
ENCRYPT( string-expression, key
[, algorithm [ format ] ]
)
```

algorithm :

```
'AES'
| 'AES256'
| 'AES_FIPS'
| 'AES256_FIPS'
[ format ]
```

format:

```
( FORMAT=RAW [ ;padding ] ) [ initialization-vector ] )
```

padding:

```
PADDING=PKCS5
| ZEROES
| NONE ]
```

Parameter

Zeichenfolgenausdruck Die zu verschlüsselnden Daten. Auch binäre Werte können an diese Funktion übergeben werden. Dieser Parameter beachtet die Groß- und Kleinschreibung, sogar in Datenbanken, die das nicht tun.

key Der zum Verschlüsseln von *Zeichenfolgenausdruck* verwendete Chiffrierschlüssel. Dieser Schlüssel muss auch zum Dechiffrieren des Werts verwendet werden, um den ursprünglichen Wert zu erhalten. Dieser Parameter beachtet die Groß- und Kleinschreibung, sogar in Datenbanken, die das nicht tun.

Wie bei den meisten Kennwörtern sollte ein Schlüssel gewählt werden, der nur schwer erraten werden kann. Es wird empfohlen, dass Sie einen Wert für Ihren Schlüssel wählen, der mindestens 16 Zeichen umfasst und eine Mischung aus Groß- und Kleinbuchstaben, Zahlen und Sonderzeichen enthält. Diesen Schlüssel benötigen Sie jedes Mal, wenn Sie die Daten entschlüsseln möchten.

Vorsicht

Achten Sie bei stark verschlüsselten Spalten darauf, eine Kopie des Schlüssels an einem sicheren Ort zu verwahren. Wenn Sie den Chiffrierschlüssel verlieren, gibt es keine Möglichkeit, auf die Daten zuzugreifen, auch nicht mit Unterstützung durch den technischen Support. Sie müssen die Spalte verwerfen und eine neue Spalte erstellen.

algorithm Dieser optionale Parameter gibt den Algorithmus an, der beim Verschlüsseln von *Zeichenfolgenausdruck* verwendet werden soll. Der zur Implementierung der starken Verschlüsselung verwendete Algorithmus ist Rijndael: ein Block-Verschlüsselungsalgorithmus, der als Advanced Encryption Standard (AES) für Block-Chiffren vom amerikanischen National Institute of Standards and Technology (NIST) ausgewählt wurde).

Sie können einen der FIPS-zertifizierten Algorithmen als *Algorithmus* für jede Plattform festlegen, die FIPS-zertifizierte Verschlüsselung unterstützt.

Hinweis

RSA- und FIPS-zertifizierte Verschlüsselung ist nicht auf allen Plattformen verfügbar. Weitere Hinweise dazu, welche Plattformen welche Verschlüsselungsmethode unterstützen, finden Sie unter <http://www.sybase.com/detail?id=1061806>.

Wenn kein *algorithm* angegeben ist, wird standardmäßig AES verwendet. Wenn der Datenbankserver mit der Serveroption "fips" gestartet wurde, wird statt dessen AES_FIPS verwendet.

FORMAT=RAW Dieser optionale Parameter verschlüsselt die Daten im RAW-Format. Der Initialisierungsvektor-Parameter ist erforderlich.

Auffüllen_mit_Zeichen Dieser optionale Parameter gibt den zu verwendenden Typ des Auffüllens mit Zeichen an. Wenn *Auffüllen_mit_Zeichen* nicht angegeben ist, wird standardmäßig PKCS5 verwendet.

PKCS5 Die Daten werden gemäß dem PKCS#5-Algorithmus mit Zeichen aufgefüllt. Die ausgegebenen (verschlüsselten) Daten sind zwischen 1 und 16 Byte länger als die eingegebenen Daten.

ZEROES Die Daten werden mit Nullen (0) aufgefüllt. Die ausgegebenen (verschlüsselten) Daten sind bis zu 15 Byte länger als die eingegebenen Daten. Diese Ausgabe wird beim Entschlüsseln auch mit Nullen aufgefüllt.

NONE Die Daten werden nicht mit Zeichen aufgefüllt. Die Länge der eingegebenen Daten muss ein Vielfaches der Chiffrierblock-Länge (16 Byte) sein.

Initialisierungsvektor Dieser Initialisierungsvektor-Parameter ist erforderlich, wenn FORMAT=RAW angegeben ist. Die Zeichenfolge kann nicht länger sein als 16 Byte. Jeder Wert unter 16 Byte wird mit 0-Bytes aufgefüllt. Diese Zeichenfolge kann nicht auf NULL gesetzt werden.

Rückgabe

LONG BINARY

Bemerkungen

Der von dieser Funktion zurückgegebene LONG BINARY-Wert ist maximal 31 Byte länger als der eingegebene *Zeichenfolgendruck*. Der von dieser Funktion zurückgegebene Wert ist nicht in lesbarer Form. Sie können die DECRYPT-Funktion verwenden, um einen *Zeichenfolgendruck* zu entschlüsseln, der mit der ENCRYPT-Funktion verschlüsselt wurde. Um einen *Zeichenfolgendruck* erfolgreich zu entschlüsseln, müssen Sie denselben Chiffrierschlüssel und Algorithmus verwenden wie zum Verschlüsseln der Daten. Wenn Sie einen ungültigen Chiffrierschlüssel angeben, wird ein Fehler generiert. Der Verlust eines Schlüssels führt dazu, dass die betreffenden Daten nicht mehr verfügbar sind und sich nicht wiederherstellen lassen.

Wenn Sie verschlüsselte Werte in einer Tabelle speichern, sollte die Spalte BINARY oder LONG BINARY sein, damit keine Zeichensatzkonvertierung an den Daten durchgeführt wird.

Wenn FORMAT=RAW angegeben ist, werden die Daten mit RAW-Verschlüsselung verschlüsselt. Sie müssen den Chiffrierschlüssel angeben, den Initialisierungsvektor und optional ein Format zum Auffüllen mit Zeichen. Dieselben Werte müssen beim Entschlüsseln der Daten angegeben werden. Die Entschlüsselung kann außerhalb des Datenbankservers ausgeführt werden. Sie können allerdings auch die DECRYPT-Funktion verwenden.

RAW-Verschlüsselung ist nicht empfehlenswert, wenn die Daten nur innerhalb des Datenbankservers verschlüsselt und entschlüsselt werden sollen, weil Sie den Initialisierungsvektor und das Auffüllen mit Zeichen angeben müssen und der Chiffrierschlüssel nicht während der Entschlüsselung überprüft werden kann.

Siehe auch

- „DECRYPT-Funktion [Zeichenfolge]“ auf Seite 234
- „ISENCRYPTED Funktion [System]“ auf Seite 294
- „Spalten- und Tabellenverschlüsselung“ [SQL Anywhere Server - Datenbankadministration]
- „Datenbankserveroption -fips“ [SQL Anywhere Server - Datenbankadministration]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Der folgende Trigger verschlüsselt die user_pwd-Spalte der user_info-Tabelle. Diese Spalte enthält Benutzerkennwörter und der Trigger wird ausgelöst, sobald ein Kennwortwert geändert wird.

```
CREATE TRIGGER encrypt_updated_pwd
BEFORE UPDATE OF user_pwd
ON user_info
REFERENCING NEW AS new_pwd
FOR EACH ROW
BEGIN
    SET new_pwd.user_pwd=ENCRYPT( new_pwd.user_pwd, '8U3dkA' );
END;
```

Die folgende SELECT-Anweisung verwendet RAW-Verschlüsselung zum Verschlüsseln der binary_data-Spalte der Tabelle "SensitiveData". Die verschlüsselten Daten werden mit Nullen aufgefüllt:

```
SELECT ENCRYPT( binary_data, 'TheEncryptionKey',
'AES(format=raw;padding=zeros)', 'ThisIsTheIV'),
```

```
LENGTH(binary_data)
FROM SensitiveData;
```

ERROR_LINE-Funktion [Verschiedene]

Gibt die Zeilennummer in der Prozedur oder dem Batch zurück, bei der der Fehler aufgetreten ist, durch den der CATCH-Block einer TRY...CATCH-Anweisung aufgerufen wurde.

Syntax

ERROR_LINE()

Rückgabe

Ein UNSIGNED INTEGER-Wert, der die Zeilennummer in der Prozedur oder dem Batch darstellt, bei der ein Fehler aufgetreten ist.

Bemerkungen

Rufen Sie diese Funktion an einer beliebigen Stelle innerhalb eines CATCH-Blocks auf. Diese Funktion meldet Informationen zum aktuellen Fehler, wenn sie innerhalb einer Fehlerbehandlungsroutine, einer verschachtelten zusammengesetzten Anweisung, einer Funktion oder einer Prozedur aufgerufen wird.

Siehe auch

- „TRY-Anweisung“ auf Seite 1093
- „BEGIN-Anweisung“ auf Seite 557
- „ERROR_MESSAGE-Funktion [Verschiedene]“ auf Seite 245
- „ERROR_PROCEDURE-Funktion [Funktionstyp]“ auf Seite 246
- „ERROR_SQLCODE-Funktion [Verschiedene]“ auf Seite 247
- „ERROR_SQLSTATE-Funktion [Verschiedene]“ auf Seite 248
- „ERROR_STACK_TRACE-Funktion [Verschiedene]“ auf Seite 249
- „STACK_TRACE-Funktion [Verschiedene]“ auf Seite 392
- „sa_error_stack_trace-Systemprozedur“ auf Seite 1216
- „sa_stack_trace-Systemprozedur“ auf Seite 1335
- „Verschachtelte zusammengesetzte Anweisungen und Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Beispiel: Fehlerprotokollierungsprozedur erstellen, sodass sie von einer Ausnahmeroutine aufgerufen werden kann“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Bei Ausführung in einem Handler, der durch einen Fehler wegen Division durch Null in Zeile 15 der Prozedur u1.proc1 aufgerufen wurde, gibt die Anweisung `SELECT ERROR_LINE(), ERROR_MESSAGE(), ERROR_PROCEDURE()` ein Ergebnis wie das folgende zurück:

```
15, 'Division by zero', '"u1"."proc1"'
```

ERROR_MESSAGE-Funktion [Verschiedene]

Gibt den Meldungstext für den Fehler zurück, durch den der CATCH-Block einer TRY...CATCH-Anweisung aufgerufen wurde.

Syntax

ERROR_MESSAGE()

Rückgabe

VARCHAR-Wert, der die Fehlermeldung des Fehlers enthält, durch den der CATCH-Block aufgerufen wurde.

Bemerkungen

Rufen Sie diese Funktion an einer beliebigen Stelle innerhalb eines CATCH-Blocks auf. Diese Funktion gibt die aktive Fehlermeldung an einer beliebigen Stelle in der Fehlerbehandlungsroutine zurück, während die ERRORMSG-Funktion bei Aufruf ohne Parameter nur die Fehlermeldung zurückgibt, wenn sie in der ersten Anweisung der Fehlerbehandlungsroutine aufgerufen wird.

Die Parameter der Fehlermeldung werden durch tatsächlichen Werte ersetzt.

Siehe auch

- „TRY-Anweisung“ auf Seite 1093
- „BEGIN-Anweisung“ auf Seite 557
- „ERROR_LINE-Funktion [Verschiedene]“ auf Seite 244
- „ERROR_PROCEDURE-Funktion [Funktionstyp]“ auf Seite 246
- „ERROR_SQLCODE-Funktion [Verschiedene]“ auf Seite 247
- „ERROR_SQLSTATE-Funktion [Verschiedene]“ auf Seite 248
- „ERROR_STACK_TRACE-Funktion [Verschiedene]“ auf Seite 249
- „STACK_TRACE-Funktion [Verschiedene]“ auf Seite 392
- „sa_error_stack_trace-Systemprozedur“ auf Seite 1216
- „sa_stack_trace-Systemprozedur“ auf Seite 1335
- „Verschachtelte zusammengesetzte Anweisungen und Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Beispiel: Fehlerprotokollierungsprozedur erstellen, sodass sie von einer Ausnahmeroutine aufgerufen werden kann“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Bei Ausführung in einem Handler, der durch einen Fehler wegen Division durch Null in Zeile 15 der Prozedur u1.proc1 aufgerufen wurde, gibt die Anweisung `SELECT ERROR_LINE() , ERROR_MESSAGE() , ERROR_PROCEDURE()` das folgende Ergebnis zurück:

```
15, 'Division by zero', ' "u1"."proc1" '
```

ERROR_PROCEDURE-Funktion [Funktionstyp]

Gibt den Namen der Prozedur zurück, in der der Fehler aufgetreten ist, aufgrund dessen die Ausnahmeroutine ausgeführt wurde.

Syntax

ERROR_PROCEDURE()

Rückgabe

VARCHAR-Wert, der den qualifizierten Namen der Prozedur enthält, in der der Fehler aufgetreten ist, bei Aufruf innerhalb einer TRY...CATCH-Anweisung. Wenn die zusammengesetzte Anweisung nicht Teil einer Prozedur, einer Funktion, eines Triggers oder eines Ereignisses ist, wird statt des Prozedurnamens der Batch-Typ (watcom_batch oder tsql_batch) zurückgegeben.

Bemerkungen

ERROR_PROCEDURE kann an einer beliebigen Stelle innerhalb eines TRY...CATCH-Blocks aufgerufen werden.

Siehe auch

- „TRY-Anweisung“ auf Seite 1093
- „BEGIN-Anweisung“ auf Seite 557
- „ERROR_LINE-Funktion [Verschiedene]“ auf Seite 244
- „ERROR_MESSAGE-Funktion [Verschiedene]“ auf Seite 245
- „ERROR_SQLCODE-Funktion [Verschiedene]“ auf Seite 247
- „ERROR_SQLSTATE-Funktion [Verschiedene]“ auf Seite 248
- „ERROR_STACK_TRACE-Funktion [Verschiedene]“ auf Seite 249
- „STACK_TRACE-Funktion [Verschiedene]“ auf Seite 392
- „sa_error_stack_trace-Systemprozedur“ auf Seite 1216
- „Verschachtelte zusammengesetzte Anweisungen und Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Beispiel: Fehlerprotokollierungsprozedur erstellen, sodass sie von einer Ausnahmeroutine aufgerufen werden kann“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Sehen Sie sich die folgende Anweisung an:

```
SELECT ERROR_LINE(), ERROR_MESSAGE(), ERROR_PROCEDURE();
```

Wenn diese Anweisung in einem Handler ausgeführt wird, der durch einen Fehler wegen Division durch Null aufgerufen wurde, gibt sie das folgende Ergebnis zurück:

```
15, 'Division by zero', '"u1"."proc1"'
```

Bei Ausführung in einem Handler, der durch einen Fehler wegen Division durch Null in der Prozedur u1.proc1 aufgerufen wurde, gibt die Anweisung `SELECT ERROR_PROCEDURE()` ; das folgende Ergebnis zurück:

```
'u1"."proc1'
```

ERROR_SQLCODE-Funktion [Verschiedene]

Gibt den SQLCODE-Wert für den Fehler zurück, durch den die Fehlerbehandlungsroutine aufgerufen wurde.

Syntax

```
ERROR_SQLCODE( )
```

Rückgabe

SIGNED INTEGER-Wert mit dem SQLCODE-Wert für den Fehler, durch den die Fehlerbehandlungsroutine aufgerufen wurde.

Bemerkungen

Diese Funktion kann an einer beliebigen Stelle innerhalb einer TRY-Anweisung aufgerufen werden und gibt den SQLCODE-Wert an einer beliebigen Stelle in der Fehlerbehandlungsroutine zurück.

Siehe auch

- „TRY-Anweisung“ auf Seite 1093
- „BEGIN-Anweisung“ auf Seite 557
- „ERROR_LINE-Funktion [Verschiedene]“ auf Seite 244
- „ERROR_MESSAGE-Funktion [Verschiedene]“ auf Seite 245
- „ERROR_PROCEDURE-Funktion [Funktionstyp]“ auf Seite 246
- „ERROR_SQLSTATE-Funktion [Verschiedene]“ auf Seite 248
- „ERROR_STACK_TRACE-Funktion [Verschiedene]“ auf Seite 249
- „STACK_TRACE-Funktion [Verschiedene]“ auf Seite 392
- „sa_error_stack_trace-Systemprozedur“ auf Seite 1216
- „sa_stack_trace-Systemprozedur“ auf Seite 1335
- „Verschachtelte zusammengesetzte Anweisungen und Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Beispiel: Fehlerprotokollierungsprozedur erstellen, sodass sie von einer Ausnahmeroutine aufgerufen werden kann“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Wenn die Anweisung `SELECT ERROR_SQLSTATE(), ERROR_SQLCODE()` ; in einem Handler ausgeführt wird, der durch einen Fehler wegen Division durch Null aufgerufen wurde, gibt sie das folgende Ergebnis zurück:

'22012', -628

ERROR_SQLSTATE-Funktion [Verschiedene]

Gibt den SQLSTATE-Wert für den Fehler zurück, durch den die Fehlerbehandlungsroutine aufgerufen wurde.

Syntax

ERROR_SQLSTATE()

Rückgabe

CHAR(5)-Wert, der den SQLSTATE-Wert für den Fehler enthält, durch den die Fehlerbehandlungsroutine aufgerufen wurde.

Bemerkungen

Diese Funktion kann an einer beliebigen Stelle innerhalb einer TRY-Anweisung aufgerufen werden und gibt den SQLSTATE-Wert an einer beliebigen Stelle in der Fehlerbehandlungsroutine zurück.

Siehe auch

- „TRY-Anweisung“ auf Seite 1093
- „BEGIN-Anweisung“ auf Seite 557
- „ERROR_LINE-Funktion [Verschiedene]“ auf Seite 244
- „ERROR_MESSAGE-Funktion [Verschiedene]“ auf Seite 245
- „ERROR_PROCEDURE-Funktion [Funktionstyp]“ auf Seite 246
- „ERROR_SQLCODE-Funktion [Verschiedene]“ auf Seite 247
- „ERROR_STACK_TRACE-Funktion [Verschiedene]“ auf Seite 249
- „STACK_TRACE-Funktion [Verschiedene]“ auf Seite 392
- „sa_error_stack_trace-Systemprozedur“ auf Seite 1216
- „sa_stack_trace-Systemprozedur“ auf Seite 1335
- „Verschachtelte zusammengesetzte Anweisungen und Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Beispiel: Fehlerprotokollierungsprozedur erstellen, sodass sie von einer Ausnahmeroutine aufgerufen werden kann“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Wenn die Anweisung `SELECT ERROR_SQLSTATE(), ERROR_SQLCODE();` in einem Handler ausgeführt wird, der durch einen Fehler wegen Division durch Null aufgerufen wurde, gibt sie das folgende Ergebnis zurück:

'22012', -628

ERROR_STACK_TRACE-Funktion [Verschiedene]

Gibt Informationen zum Stack-Trace für den Fehler zurück, durch den die Fehlerbehandlungsroutine aufgerufen wurde.

Syntax

ERROR_STACK_TRACE()

Rückgabe

LONG VARCHAR-Wert, der das Stack-Trace für den Fehler darstellt, der durch den die Fehlerbehandlungsroutine aufgerufen wurde. Wenn die zusammengesetzte Anweisung nicht Teil einer Prozedur, einer Funktion, eines Triggers oder eines Ereignisses ist, wird statt des Prozedurnamens der Batch-Typ (watcom_batch oder tsql_batch) zurückgegeben.

Bemerkungen

Jede Zeile des zurückgegebenen Werts enthält den qualifizierten Prozedurnamen oder Batch-Typ (falls vorhanden) für die Anweisung auf dem Stack, gefolgt von der Zeilennummer der Anweisung. Die letzte Zeile des zurückgegebenen Werts endet nicht mit einer Zeilenendmarke. Wenn eine Prozedur ausgeblendet ist, gibt der Datenbankserver nicht den Namen der Prozedur aus, durch die der Fehler aufgerufen wurde.

Diese Funktion gibt dieselben Informationen zurück wie die sa_error_stack_trace-Systemprozedur.

Siehe auch

- „TRY-Anweisung“ auf Seite 1093
- „BEGIN-Anweisung“ auf Seite 557
- „ERROR_LINE-Funktion [Verschiedene]“ auf Seite 244
- „ERROR_MESSAGE-Funktion [Verschiedene]“ auf Seite 245
- „ERROR_PROCEDURE-Funktion [Funktionstyp]“ auf Seite 246
- „ERROR_SQLCODE-Funktion [Verschiedene]“ auf Seite 247
- „ERROR_SQLSTATE-Funktion [Verschiedene]“ auf Seite 248
- „STACK_TRACE-Funktion [Verschiedene]“ auf Seite 392
- „sa_error_stack_trace-Systemprozedur“ auf Seite 1216
- „sa_stack_trace-Systemprozedur“ auf Seite 1335
- „Verschachtelte zusammengesetzte Anweisungen und Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Beispiel: Fehlerprotokollierungsprozedur erstellen, sodass sie von einer Ausnahmeroutine aufgerufen werden kann“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Gruppe von Prozeduren (mit hinzugefügten Zeilennummern) kann verwendet werden, um den Fehler-Stack abzurufen:

```
1 CREATE PROCEDURE proc1()
2 BEGIN TRY
```

```
3      CALL proc2();
4 END TRY
5 BEGIN CATCH
6     SELECT ERROR_STACK_TRACE();
7 END CATCH;

1CREATE PROCEDURE proc2()
2 BEGIN
3     CALL proc3();
4 END;

1CREATE PROCEDURE proc3()
2 BEGIN
3     DECLARE v INTEGER = 0;
4     SET v = 1 / v;
5 END;

CALL proc1();
```

Dieser Abruf gibt die folgende Ergebniszeichenfolge zurück:

```
' "DBA"."proc1" : 3
"DBA"."proc2" : 3
"DBA"."proc3" : 4
```

Wenn in der Fehlerbehandlungsroutine **RESIGNAL** verwendet wird und der erneut signalisierte Fehler behandelt wird, enthält der im zweiten Handler gemeldete Fehler-Stack das Stack-Trace des ursprünglichen Fehlers, den Datensatz für das **RESIGNAL** und den Stack der erneut signalisierten Ausnahmebedingung. Beispiel:

```
CREATE PROCEDURE proc1()
BEGIN TRY
    BEGIN TRY
        DECLARE v INTEGER = 0;
        SET v = 1 / v;
    END TRY
    BEGIN CATCH
        CALL proc2();
    END CATCH
END TRY
BEGIN CATCH
    SELECT ERROR_STACK_TRACE();
END CATCH;

CREATE PROCEDURE proc2()
BEGIN
    CALL proc3();
END;

CREATE PROCEDURE proc3()
BEGIN
    RESIGNAL;
END;

CALL proc1();
```

Dieser Abruf gibt die folgende Ergebniszeichenfolge zurück:

```
' "DBA"."proc1" : 8
"DBA"."proc2" : 3
```

```
RESIGNAL: "DBA"."proc3" : 3
"DBA"."proc1" : 5
```

ERRORMSG-Funktion [Verschiedene]

Liefert die Fehlermeldung für den aktuellen Fehler oder für einen angegebenen SQLSTATE- oder SQLCODE-Wert.

Syntax

ERRORMSG([*sqlstate* | *sqlcode*])

sqlstate: string

sqlcode: integer

Parameter

- **sqlstate** Der SQLSTATE-Wert, für den die Fehlermeldung zurückgegeben werden soll.
- **sqlcode** Der SQLCODE-Wert, für den die Fehlermeldung zurückgegeben werden soll.

Rückgabe

VARCHAR die die Fehlermeldung enthält.

Bemerkungen

Wenn kein Argument eingegeben wurde, wird die Fehlermeldung für den aktuellen Status ("state") ausgegeben. Für Platzhalter werden die jeweiligen Echtbezeichnungen (z.B. Tabellennamen und Spaltennamen) ausgegeben.

Wenn ein Argument übergeben wurde, wird die Fehlermeldung für den jeweiligen SQLSTATE- oder SQLCODE-Wert ohne Ersetzung der Platzhalter durch Echtnamen zurückgegeben. Tabellennamen und Spaltennamen werden als Platzhalter (%1) geliefert.

Siehe auch

- „SQL Anywhere-Fehlermeldungen - sortiert nach SQLSTATE“ [[Fehlermeldungen](#)]
- „SQL Anywhere-Fehlermeldungen - sortiert nach SQLCODE“ [[Fehlermeldungen](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die Fehlermeldung für SQLCODE -813 zurück:

```
SELECT ERRORMSG( -813 );
```

ESTIMATE-Funktion [Verschiedene]

Gibt Selektivitätsschätzungen als Prozentsatz zurück, der durch den Abfrageoptimierer basierend auf angegebenen Parametern berechnet wurde.

Syntax

```
ESTIMATE( column-name [, value [, relation-string ] ] )
```

Parameter

- **Spaltenname** Die Spalte, die in der Schätzung verwendet wird
- **value** Der Wert, mit dem die Spalte verglichen wird. Der Standardwert ist NULL.
- **relation-string** Der Vergleichsoperator, der für den Vergleich verwendet wird, in Apostrophe gesetzt. Mögliche Werte für diesen Parameter sind: '=', '>', '<', '>=', '<=', '<>', '!=', '!'<', und '!'>'. Der Standardwert ist '='.

Rückgabe

REAL

Bemerkungen

Diese Funktion gibt Selektivitätsschätzungen für das Prädikat `column-name relation-string value` zurück. Wenn *Wert* NULL ist und die Beziehungszeichenfolge '=', ist die Selektivität für das Prädikat `column-name IS NULL`. Wenn *value* NULL ist und die Relationszeichenfolge '!=' oder '<>', ist die Selektivität für das Prädikat `column-name IS NOT NULL`.

Siehe auch

- „Quellen der Selektivitätsschätzung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Selektivitätsinformationen im grafischen Plan“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „INDEX_ESTIMATE-Funktion [Verschiedene]“ auf Seite 290
- „ESTIMATE_SOURCE-Funktion [Verschiedene]“ auf Seite 253
- „EXPERIENCE_ESTIMATE-Funktion [Verschiedene]“ auf Seite 259

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Prozentsatz der EmployeeID-Werte zurück, die größer als 200 geschätzt werden. Der genaue Wert hängt von den Aktionen ab, die Sie auf der Datenbank durchgeführt haben.

```
SELECT FIRST ESTIMATE( EmployeeID, 200, '>' )
FROM GROUPO.Employees
ORDER BY 1;
```

ESTIMATE_SOURCE-Funktion [Verschiedene]

Liefert die Quelle für Selektivitätsschätzungen, die vom Abfrageoptimierer verwendet werden

Syntax

```
ESTIMATE_SOURCE(
  column-name
  [, value
  [, relation-string ] ]
)
```

Parameter

- **Spaltenname** Der Name der Spalte, die untersucht wird.
- **value** Der Wert, mit dem die Spalte verglichen wird. Der Standardwert ist NULL.
- **relation-string** Der Vergleichsoperator, der für den Vergleich verwendet wird, in Apostrophe gesetzt. Mögliche Werte für diesen Parameter sind: '=', '>', '<', '>=', '<=', '<>', '!=', '!'<', und '!'>'. Der Standardwert ist '='.

Rückgabe

Die folgende Liste enthält die Selektivitätsschätzungsquellen, die ESTIMATE_SOURCE zurückgibt.

Weitere Hinweise zu den Quellen finden Sie unter „Quellen der Selektivitätsschätzung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Wert	Selektivitätsschätzungsquelle
Statistiken (Statistics)	Gespeicherte Spaltenstatistiken
Spalte	Mittelwert aller in den Spaltenstatistiken gespeicherten Werte
Index	Prüfungen des Indexes
Annahme	Integrierte Annahmen für jeden Typ von Prädikat. Dies wird nur dann zurückgegeben, wenn kein relevanter Index benutzt werden kann, keine Statistiken für die referenzierten Spalten gesammelt wurden oder das Prädikat ein komplexes Prädikat ist.
Berechnet	Andere Quellen als die oben beschriebenen
Immer	Wird zurückgegeben, wenn das angegebene Prädikat immer TRUE ist
Kombiniert	Eine oder mehrere der oben genannten Quellen
Beschränkt	Wird zurückgegeben, wenn für die Selektivitätsschätzung eine obere und/oder untere Grenzen festgelegt ist

Bemerkungen

Diese Funktion gibt die Quelle der Selektivitätsschätzung für das Prädikat `column-name relation-string value` zurück. Wenn *Wert* NULL ist und die Beziehung Zeichenfolge '=', ist die Selektivitätsquelle für das Prädikat `column-name IS NULL`. Wenn *value* NULL ist und die Relationszeichenfolge '!=' oder '<>', ist die Selektivitätsquelle für das Prädikat `column-name IS NOT NULL`.

Siehe auch

- „Quellen der Selektivitätsschätzung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „ESTIMATE-Funktion [Verschiedene]“ auf Seite 252
- „INDEX_ESTIMATE-Funktion [Verschiedene]“ auf Seite 290

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Selektivitätsquellenindex zurück, mit dem ermittelt wird, ob der erste Wert in der Spalte `EmployeeID` größer ist als 200. Die Rückgabe eines Index bedeutet, dass der Abfrageoptimierer zur Schätzung der Selektivität einen Index verwendet hat.

```
SELECT FIRST ESTIMATE_SOURCE( EmployeeID, 200, '>' )  
FROM GROUPO.Employees  
ORDER BY 1;
```

EVENT_CONDITION-Funktion [System]

Legt fest, wann ein Event-Handler ausgelöst wird.

Syntax

EVENT_CONDITION(*condition-name*)

Parameter

- **Bedingungsname** Die Bedingung, die das Ereignis auslöst. Die möglichen Werte werden in der Datenbank im Voraus eingestellt und beachten die Groß- und Kleinschreibung nicht. Jede Bedingung ist nur für bestimmte Ereignistypen gültig. Die Bedingungen und die Ereignisse, für die sie gelten, sind die folgenden:

Bedingungsname	Einheit	Gilt für...	Kommentare
DBFreePercent	Nicht zutreffend	DBDiskSpace	
DBFreeSpace	MB	DBDiskSpace	
DBSize	MB	GrowDB	
ErrorNumber	Nicht zutreffend	RAISERROR	

Bedingungsname	Einheit	Gilt für...	Kommentare
IdleTime	Sekunden	ServerIdle	
Interval	Sekunden	All	Zeit, seitdem Verarbeitungsroutine zuletzt ausgeführt worden ist
LogFreePercent	Nicht zutreffend	LogDiskSpace	
LogFreeSpace	MB	LogDiskSpace	
LogSize	MB	GrowLog	
RemainingValues	integer	GlobalAutoincrement	Die Anzahl der verbleibenden Werte
TempFreePercent	Nicht zutreffend	TempDiskSpace	
TempFreeSpace	MB	TempDiskSpace	
TempSize	MB	GrowTemp	

Rückgabe

INT

Bemerkungen

Die EVENT_CONDITION-Funktion gibt NULL zurück, wenn sie nicht von einem Ereignis aufgerufen wird.

Siehe auch

- [„CREATE EVENT-Anweisung“ auf Seite 606](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Ereignisdefinition verwendet die EVENT_CONDITION-Funktion:

```
CREATE EVENT LogNotifier
TYPE LogDiskSpace
WHERE event_condition( 'LogFreePercent' ) < 50
HANDLER
BEGIN
    MESSAGE 'LogNotifier message'
END;
```

EVENT_CONDITION_NAME-Funktion [System]

Listet die möglichen Parameter für EVENT_CONDITION auf.

Syntax

EVENT_CONDITION_NAME(*integer*)

Parameter

- **integer** Muss größer oder gleich Null sein

Rückgabe

VARCHAR

Bemerkungen

Sie können mit der EVENT_CONDITION_NAME-Funktion eine Liste aller Argumente für die EVENT_CONDITION-Funktion erhalten, wobei ein Ganzzahlen-Schleifendurchlauf solange durchgeführt wird, bis die Funktion NULL zurückgibt.

Die EVENT_CONDITION_NAME-Funktion gibt NULL zurück, wenn sie nicht von einem Ereignis aufgerufen wird.

Siehe auch

- [„CREATE EVENT-Anweisung“ auf Seite 606](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

EVENT_PARAMETER-Funktion [System]

Liefert Kontextinformationen bei Event-Handlern.

Syntax

EVENT_PARAMETER(*context-name*)

context-name :

ApplInfo

ConnectionID
DisconnectReason
EventName
Executions
MirrorServerName
NumActive
ScheduleName
SQLCODE
TableName
User

condition-name

Parameter

- **Kontextname** Eine der voreingestellten Zeichenfolgen. Die Zeichenfolgen müssen in Anführungszeichen gesetzt werden, beachten die Groß- und Kleinschreibung nicht und enthalten die folgenden Informationen:
- **AppInfo** Der Wert der AppInfo-Verbindungseigenschaft für die Verbindung, die das Auslösen des Ereignisses bewirkt hat. Verwenden Sie die folgende Anweisung, um den Wert der Eigenschaft außerhalb des Kontexts des Ereignisses anzuzeigen:

```
SELECT CONNECTION_PROPERTY( 'AppInfo' );
```

Die AppInfo-Zeichenfolge enthält den Computernamen und den Anwendungsnamen der Clientverbindung für Embedded SQL-, ODBC-, OLE DB-, ADO.NET- und iAnywhere JDBC-Treiberverbindungen.

- **ConnectionID** Die Verbindungs-ID der Verbindung, die das Auslösen des Ereignisses bewirkt hat
- **DisconnectReason** Eine Zeichenfolge, die darauf hinweist, warum die Verbindung beendet wurde. Dieser Parameter gilt nur für Disconnect-Ereignisse. Zu den möglichen Ergebnissen gehören:
 - **abnormal** Eine Verbindung wurde getrennt, weil die Clientanwendung vor der Trennung von der Datenbank abnormal beendet wurde oder weil ein Kommunikationsfehler zwischen Client- und dem Servercomputer aufgetreten ist.
 - **connect failed** Ein Verbindungsversuch ist fehlgeschlagen.
 - **drop connection** Eine Anweisung DROP CONNECTION ist ausgeführt worden.
 - **from client** Die Clientanwendung hat die Verbindung getrennt.
 - **inactive** Für die mit der Option -ti angegebene Serveroption wurden für die angegebene Periode keine Anforderungen empfangen.
 - **liveness** Für die mit der Option -tl angegebene Serveroption wurden keine Verfügbarkeitspakete empfangen.
- **EventName** Der Name des Ereignisses, das ausgelöst wurde
- **Executions** Gibt an, wie oft der Event-Handler ausgeführt wurde.
- **MirrorServerName** Der Name des Spiegel- oder Arbiterservers, der in einem Datenbankspiegelungssystem seine Verbindung zum Primärserver verloren hat
- **NumActive** Die Anzahl der aktiven Instanzen eines Event-Handlers. Dies ist nützlich, wenn Sie einen Event-Handler auf jeweils eine Instanz begrenzen möchten.
- **ScheduleName** Der Name des Plans, der zur Auslösung des Ereignisses geführt hat. Wenn das Ereignis manuell mit TRIGGER EVENT oder als Systemereignis ausgelöst wird, gibt das System

eine leere Zeichenfolge zurück. Wenn dem Plan beim Erstellen nicht ausdrücklich ein Name zugeordnet wurde, übernimmt er den Namen des Ereignisses.

- **SQLCODE** Der SQLCODE für den Fehler, der während einer fehlgeschlagenen Verbindung aufgetreten ist. Dieser Parameter gilt nur für ConnectFailed-Ereignisse.
- **TableName** Der Name der Tabelle für RemainingValues-Werte
- **Benutzer** Die Benutzer-ID des Benutzers, der das ausgelöst hat.

Außerdem können Sie über die EVENT_PARAMETER-Funktion auf alle gültigen *Bedingungsname*-Argumente der EVENT_CONDITION-Funktion zugreifen.

Aus der folgenden Tabelle ist ersichtlich, welche Kontextnamen-Werte für welche Systemereignistypen gelten.

Systemereignistypen	Kontextnamenwert
BackupEnd	AppInfo, ConnectionID, EventName, Executions, NumActive, User
Verbinden	AppInfo, ConnectionID, EventName, Executions, NumActive, User
ConnectFailed	AppInfo, EventName, Executions, NumActive, SQLCODE, User
"Disconnect"	AppInfo, ConnectionID, EventName, Executions, NumActive, User
GlobalAutoincrement	ConnectionID, EventName, Executions, NumActive, TableName, User
"RAISERROR"	AppInfo, ConnectionID, EventName, Executions, NumActive, User
Benutzerereignisse	AppInfo, ConnectionID, EventName, Executions, NumActive, User

Rückgabe

VARCHAR

Bemerkungen

Die maximale Größe für Werte, die an ein Ereignis übergeben werden, ist mit der maximalen Seitengröße des Servers begrenzt (-gp Serveroption). Längere Werte werden gekürzt, damit sie unter der maximalen Seitengröße liegen.

Siehe auch

- [„EVENT_CONDITION-Funktion \[System\]“ auf Seite 254](#)
- [„CREATE EVENT-Anweisung“ auf Seite 606](#)
- [„TRIGGER EVENT-Anweisung“ auf Seite 1088](#)
- [„Datenbankserveroption -gp“ \[SQL Anywhere Server - Datenbankadministration\]](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel zeigt, wie ein Zeichenfolgenparameter an ein Ereignis übergeben wird. Das Ereignis zeigt den Zeitpunkt, an dem es ausgelöst wird, im Meldungsfenster des Datenbankservers.

```
CREATE EVENT ev_PassedParameter
HANDLER
BEGIN
    MESSAGE 'ev_PassedParameter - was triggered at ' ||
event_parameter( 'time' );
END;
TRIGGER EVENT ev_PassedParameter( "Time"=string(current timestamp ) );
```

EXP-Funktion [Numerisch]

Gibt das Ergebnis der Berechnung der Basis des natürlichen Logarithmus (e) zurück, potenziert mit dem angegebenen Argument.

Syntax

EXP(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Der Exponent.

Rückgabe

DOUBLE

Bemerkungen

Das Ergebnis der EXP-Funktion ist die Basis des natürlichen Logarithmus (e), potenziert mit dem durch *Numerischer_Ausdruck* angegebenen Wert.

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltgenauem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück. Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Standards und Kompatibilität

- **SQL/2008** Die EXP Funktion umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Die Anweisung gibt den Wert 3269017,3724721107 zurück:

```
SELECT EXP( 15 );
```

EXPERIENCE_ESTIMATE-Funktion [Verschiedene]

Gibt Selektivitätsschätzungen als Prozentsatz zurück, der durch den Abfrageoptimierer basierend auf angegebenen Parametern berechnet wurde.

Syntax

```
EXPERIENCE_ESTIMATE(  
  column-name  
  [, value  
  [, relation-string ] ]  
)
```

Parameter

- **Spaltenname** Der Name der Spalte, die untersucht wird.
- **value** Der Wert, mit dem die Spalte verglichen wird.
- **relation-string** Der für den Vergleich verwendete Vergleichsoperator. Mögliche Werte für diesen Parameter sind: '=', '>', '<', '>=', '<=', '<>', '!=', '!<', und '!>'. Der Standardwert ist '='.

Rückgabe

REAL

Bemerkungen

Wenn *value* NULL ist, werden die Relationszeichenfolgen "=" und "!=" als die Bedingung IS NULL bzw. IS NOT NULL interpretiert.

Siehe auch

- [„ESTIMATE-Funktion \[Verschiedene\]“ auf Seite 252](#)
- [„INDEX_ESTIMATE-Funktion \[Verschiedene\]“ auf Seite 290](#)
- [„ESTIMATE_SOURCE-Funktion \[Verschiedene\]“ auf Seite 253](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 90,3262405396 zurück:

```
SELECT DISTINCT EXPERIENCE_ESTIMATE( EmployeeID, 200, '>' )  
FROM GROUPO.Employees;
```

EXPLANATION-Funktion [Verschiedene]

Gibt die Optimierungsstrategie einer SQL-Anweisung als normale Textzeichenfolge zurück.

Syntax

```
EXPLANATION(  
  string-expression  
  [ , cursor-type ]  
  [, update-status ]  
)
```

Parameter

- **Zeichenfolgenausdruck** Die SQL-Anweisung, die gewöhnlich eine SELECT-Anweisung ist, aber auch eine UPDATE-, MERGE- oder DELETE-Anweisung sein kann.
- **Cursortyp** Ein als Zeichenfolge ausgedrückter Cursortyp. Mögliche Werte sind asensitiv, insensitiv, sensitiv oder keyset-driven. Wenn *Cursortyp* nicht angegeben ist, wird standardmäßig "asensitive" verwendet.
- **update-status** Ein Zeichenfolgenparameter, der einen der folgenden Werte akzeptiert, die angeben, wie der Optimierer die existierenden Cursor behandeln soll:

Wert	Beschreibung
READ-ONLY	Der Cursor ist schreibgeschützt.
READ-WRITE (Standardwert)	Der Cursor kann gelesen oder beschrieben werden.
FOR UPDATE	Der Cursor kann gelesen oder beschrieben werden. Dieser Wert ist derselbe wie READ-WRITE.

Rückgabe

LONG VARCHAR

Bemerkungen

Der Zugriffsplan der Anweisung wird als Zeichenfolge zurückgegeben. Hinweise zum Interpretieren des Ergebnisses finden Sie unter „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Die GRAPHICAL_PLAN-Funktion bietet deutlich mehr Informationen über Zugriffspläne, einschließlich der Systemeigenschaften die möglicherweise beeinflusst haben, wie die Anweisung optimiert wurde.

Anhand dieser Informationen können Sie entscheiden, ob Sie Indizes hinzufügen oder wie Sie Ihre Datenbank zur Steigerung der Performance strukturieren sollen.

Siehe auch

- „Ausführungspläne in UltraLite“ [[UltraLite - Datenbankverwaltung](#)]
- „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „PLAN-Funktion [Verschiedene]“ auf Seite 337
- „GRAPHICAL_PLAN-Funktion [Verschiedene]“ auf Seite 269

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung übergibt eine SELECT-Anweisung als Zeichenfolgenparameter und gibt den Ausführungsplan für die Abfrage zurück:

```
SELECT EXPLANATION( 'SELECT * FROM Departments WHERE DepartmentID > 100' );
```

Die folgende Anweisung gibt eine Zeichenfolge zurück, die die Kurzform des Textplans für einen INSENSITIVE-Cursor über die Abfrage "select * from Department where" enthält.:

```
SELECT EXPLANATION( 'SELECT * FROM GROUPO.Departments WHERE DepartmentID > 100',  
    'insensitive', 'read-only' );
```

EXPRTYPE-Funktion [Verschiedene]

Gibt eine Zeichenfolge zurück, die den Datentyp eines Ausdrucks angibt.

Syntax

EXPRTYPE(*string-expression*, *integer-expression*)

Parameter

- **Zeichenfolgenausdruck** Eine SELECT-Anweisung. Der Ausdruck, dessen Datentyp abgerufen werden soll, muss in der Select-Liste erscheinen. Sollte die Zeichenfolge keine gültige SELECT-Anweisung sein, wird NULL zurückgegeben.
- **Ganzzahlausdruck** Die Position in der SELECT-Liste des gewünschten Ausdrucks. Das erste Element in der SELECT-Liste erhält die Nummer 1. Wenn der Wert des Ganzzahlausdrucks mit keinem der SELECT-Listen-Elemente übereinstimmt, wird NULL zurückgegeben.

Rückgabe

LONG VARCHAR

Bemerkungen

Für benutzerdefinierte Domänen gibt EXPRTYPE die Beschreibung des zugrunde liegenden Datentyps und nicht des Domänennamens an. Beispiel: Sie erstellen eine Domäne mydomain und definieren eine Tabellenspalte mit mydomain wie folgt:

```
CREATE DOMAIN mydomain CHAR(20);  
CREATE TABLE mytable( colA mydomain, colB DATETIME );
```

Wenn Sie `SELECT EXPRTYPE('SELECT * FROM mytable', 1)` ausführen, ist der zurückgegebene Datentyp `char(20)`, nicht `mydomain`.

Siehe auch

- „SQL-Datentypen“ auf Seite 95
- „sa_describe_query-Systemprozedur“ auf Seite 1206

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt `smallint` zurück, wenn sie in der SQL Anywhere-Beispieldatenbank ausgeführt wird:

```
SELECT EXPRTYPE( 'SELECT LineID FROM SalesOrderItems', 1 );
```

FIRST_VALUE-Funktion [Aggregat]

Liefert Werte aus der ersten Zeile eines Fensters.

Syntax

```
FIRST_VALUE( [ ALL ] expression [ { RESPECT | IGNORE } NULLS ] )  
OVER ( window-spec )
```

Fensterspezifikation: Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Parameter

- ***expression*** Der auszuwertende Ausdruck. Z.B. ein Spaltenname.

Rückgabe

Datentyp der Werte aus der ersten Zeile eines Fensters.

Bemerkungen

Mit der FIRST_VALUE-Funktion können Sie den ersten Wert (entsprechend der Sortierung) in einer Tabelle auswählen, ohne einen Selbst-Join verwenden zu müssen. Dies ist nützlich, wenn Sie den ersten Wert als Basiswert in Berechnungen verwenden wollen.

Die FIRST_VALUE-Funktion nimmt den ersten Datensatz im Fenster. Anschließend wird der *expression* anhand des ersten Datensatzes berechnet und Ergebnisse werden zurückgegeben.

Wenn IGNORE NULLS angegeben ist, wird der erste Nicht-NULL-Wert von *expression* zurückgegeben. Wenn RESPECT NULLS angegeben ist (Standardwert), wird der erste Wert zurückgegeben, unabhängig davon, ob er NULL ist.

Die FIRST_VALUE-Funktion unterscheidet sich von den meisten anderen Aggregatfunktionen, weil sie nur mit einer Fensterspezifikation verwendet werden kann.

Elemente von *Fensterspezifikation* können entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie in der *Fensterspezifikation*-Definition der WINDOW-Klausel. Siehe „[WINDOW-Klausel](#)“ auf Seite 1124.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Siehe auch

- „Fenster-Aggregatfunktionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „LAST_VALUE-Funktion [Aggregat]“ auf Seite 297

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

SQL Anywhere unterstützt die SQL/2008 -Sprachenfunktion F441, "Extended set function support", die es zulässt, dass Operanden von Fensterfunktionen beliebige Ausdrücke sind, die keine Spaltenreferenzen darstellen.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der FIRST_VALUE-Funktion als auch eine äußere Referenz enthalten. Ein Beispiel finden Sie unter „AVG-Funktion [Aggregat]“ [[UltraLite - Datenbankverwaltung](#)].

Beispiel

Das folgende Beispiel gibt die Beziehung (als Prozentsatz) zwischen dem Gehalt aller Mitarbeiter und dem Gehalt des zuletzt eingestellten Mitarbeiters in der Abteilung zurück:

```
SELECT DepartmentID, EmployeeID,
       100 * Salary / ( FIRST_VALUE( Salary ) OVER (
                           PARTITION BY DepartmentID ORDER BY StartDate
                           DESC ) )
       AS percentage
FROM GROUPO.Employees;
```

DepartmentID	EmployeeID	percentage
500	1658	100
500	1615	110.4284624
500	1570	138.8427097
500	1013	109.5851905
500	921	167.4497049
500	868	113.2393688
500	750	137.7344095
500	703	222.8679276

DepartmentID	EmployeeID	percentage
500	191	119.6642975
400	1751	100
400	1740	99.705647
400	1684	130.969936
400	1643	83.9734797
400	1607	175.1828989
400	1576	197.0164609
...

Mitarbeiter 1658 ist die erste Zeile für Department 500, womit angezeigt wird, dass es sich dabei um den jüngsten Neuzugang in der Abteilung handelt. Der Prozentsatz ist 100%. Die Prozentsätze für die restlichen Department-500-Mitarbeiter werden relativ zu Mitarbeiter 1658 berechnet. Mitarbeiter 1570 verdient beispielsweise 139% des Gehalts von 1658.

Wenn andere Mitarbeiter in derselben Abteilung gleich viel verdienen wie der zuletzt eingestellte Mitarbeiter, haben sie ebenfalls den Prozentsatz 100%.

FLOOR-Funktion [Nummerisch]

Gibt die größte Ganzzahl zurück, die nicht größer ist als die angegebene Zahl.

Syntax

FLOOR(*numeric-expression*)

Parameter

- **Nummerischer Ausdruck** Der Wert, der gekürzt werden soll, in der Regel eine fester numerischer Typ, bei dem die Anzahl der Dezimalstellen nicht Null ist, oder ein angenäherter numerischer Datentyp (DOUBLE, REAL oder FLOAT).

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch.

Siehe auch

- [„CEILING-Funktion \[Numerisch\]“ auf Seite 187](#)

Standards und Kompatibilität

- **SQL/2008** Die FLOOR-Funktion umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Die folgende Anweisung gibt einen FLOOR-Wert von "123" zurück:

```
SELECT FLOOR (123);
```

Die folgende Anweisung gibt einen FLOOR-Wert von 123 zurück:

```
SELECT FLOOR (123.45);
```

Die folgende Anweisung gibt einen FLOOR-Wert von 124 zurück:

```
SELECT FLOOR (-123.45);
```

GET_BIT-Funktion [Bit-Array]

Gibt den Wert (1 oder 0) eines angegebenen Bits in einem Bit-Array zurück.

Syntax

```
GET_BIT( bit-expression, position )
```

Parameter

- **Bit-Ausdruck** Das Bit-Array, das das Bit enthält.
- **position** Die Position des Bits, dessen Status zurückgegeben werden soll

Rückgabe

BIT

Bemerkungen

Die Positionen im Array werden von links mit 1 beginnend abgezählt.

Wenn *Position* die Länge des Arrays überschreitet, wird FALSE (0) zurückgegeben.

Siehe auch

- [„Bit-Operatoren“ auf Seite 20](#)
- [„SET_BIT-Funktion \[Bit-Array\]“ auf Seite 379](#)
- [„SET_BITS-Funktion \[Aggregat\]“ auf Seite 381](#)
- [„sa_get_bits-Systemprozedur“ auf Seite 1221](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung liefert den Wert 1:

```
SELECT GET_BIT( '00110011' , 4 );
```

Die folgende Anweisung liefert den Wert 0:

```
SELECT GET_BIT( '00110011' , 5 );
```

GET_IDENTITY-Funktion [Verschiedene]

Weist einer AUTOINCREMENT-Spalte Werte zu. Diese Funktion kann als Alternative zur AUTOINCREMENT-Funktion zum Generieren von Zahlen verwendet werden.

Syntax

```
GET_IDENTITY( table_name [, number_to_allocate ] )
```

Parameter

- ***table_name*** Eine Zeichenfolge, die den Namen der Tabelle und optional auch den Eigentümernamen enthält.
- ***number_to_allocate*** Die Anzahl der zu reservierenden Werte. Standardwert ist "1".

Rückgabe

UNSIGNED BIGINT

Bemerkungen

Zum Generieren von IDs ist die Verwendung von AUTOINCREMENT oder GLOBAL AUTOINCREMENT nach wie vor die effizienteste Methode. Diese Funktion kann jedoch als Alternative verwendet werden. Die Funktion geht davon aus, dass für die Tabelle eine AUTOINCREMENT-Spalte definiert ist. Sie gibt den nächsten verfügbaren Wert zurück, der für die AUTOINCREMENT-Spalte der Tabelle generiert würde, und reserviert diesen Wert, damit ihn nicht eine andere Verbindung standardmäßig verwendet.

Die Funktion gibt eine Fehlermeldung zurück, sollte die Tabelle nicht gefunden werden, und NULL, wenn die Tabelle keine AUTOINCREMENT-Spalte enthält. Wenn mehrere AUTOINCREMENT-Spalten vorhanden sind, verwendet die Funktion die erste gefundene Spalte.

number_to_allocate ist die Anzahl der zu reservierenden Werte. Wenn der Wert *number_to_allocate* größer als 1 ist, reserviert die Funktion auch die restlichen Werte. Die nächste Zuweisung verwendet die aktuelle Zahl und addiert den Wert von *number_to_allocate*. Dies führt dazu, dass die Anwendung *get_identity* weniger häufig ausführen muss. Wenn *number_to_allocate* gleich 0 ist, wird der nächste verfügbare Wert zurückgegeben, ohne Werte zu reservieren.

Nach dem Ausführen der GET_IDENTITY-Funktion ist kein COMMIT mehr notwendig, und daher kann dieselbe Verbindung zum Aufruf der Funktion und zum Einfügen von Zeilen verwendet werden. Wenn ID-Werte für mehrere Tabellen benötigt werden, können diese über einmaliges Aufrufen der SELECT-Anweisung, die mehrere Aufrufe der GET_IDENTITY-Funktion beinhaltet, bezogen werden. Siehe hierzu das Beispiel.

Die GET_IDENTITY-Funktion ist eine nicht-deterministische Funktion. Aufeinanderfolgende Aufrufe können unterschiedliche Werte zurückgeben. Der Optimierer behält die Ergebnisse der Funktion GET_IDENTITY nicht im Cache.

Weitere Hinweise zu nicht-deterministischen Funktionen finden Sie unter [Caching von Funktionen \[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#).

Siehe auch

- „CREATE TABLE-Anweisung“ auf Seite 737
- „ALTER TABLE-Anweisung“ auf Seite 516
- „NUMBER-Funktion [Verschiedene]“ auf Seite 331

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den nächsten verfügbaren Wert für die AUTOINCREMENT-Spalte (ID) der Tabelle Customers zurück. Die zurückgegebene Nummer und die folgenden neun Werte sind reserviert:

```
SELECT GET_IDENTITY( 'GROUPO.Customers', 10 );
```

GETDATE-Funktion [Datum und Uhrzeit]

Gibt die aktuellen Werte für Jahr, Monat, Tag, Stunde, Minute, Sekunde und Sekundenbruchteil zurück.

Syntax

GETDATE()

Rückgabe

TIMESTAMP

Bemerkungen

Die Genauigkeit ist durch die Genauigkeit der Systemuhr begrenzt.

Die von der GETDATE-Funktion zurückgegebenen Werte ähneln den von der NOW-Funktion zurückgegebenen Angaben und dem Spezialwert von CURRENT_TIMESTAMP.

Siehe auch

- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75
- „DATE-Datentyp“ auf Seite 123
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Datentyp“ auf Seite 124
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „Ausdrücke“ auf Seite 22
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „TIMESTAMP-Datentyp“ auf Seite 129
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt das Systemdatum und die Systemzeit zurück:

```
SELECT GETDATE( );
```

GRAPHICAL_PLAN-Funktion [Verschiedene]

Gibt die Optimierungsstrategie einer SQL-Anweisung des Zeichenfolgenausdrucks im XML-Format als Zeichenfolge zurück.

Syntax

```
GRAPHICAL_PLAN(  
  string-expression  
  [, statistics-level  
  [, cursor-type  
  [, update-status ] ] ] )
```

Parameter

- **Zeichenfolgenausdruck** Die SQL-Anweisung, die gewöhnlich eine SELECT-Anweisung ist, aber auch eine UPDATE- oder DELETE-Anweisung sein kann.
- **statistics-level** Eine Ganzzahl. *statistics-level* kann einer der folgenden Werte sein:

Wert	Beschreibung
0	Nur Schätzungen des Optimierers (Standard)
2	Detaillierte Statistiken, einschließlich Knotenstatistiken
3	Detaillierte Statistiken

- **Cursortyp** Ein als Zeichenfolge ausgedrückter Cursortyp. Mögliche Werte sind: asensitiv, insensitiv, sensitiv oder keyset-driven. Wenn *Cursortyp* nicht angegeben ist, wird standardmäßig "asensitive" verwendet.
- **update-status** Ein Zeichenfolgenparameter, der einen der folgenden Werte akzeptiert, die angeben, wie der Optimierer die existierenden Cursor behandeln soll:

Wert	Beschreibung
READ-ONLY	Der Cursor ist schreibgeschützt.
READ-WRITE (Standardwert)	Der Cursor kann gelesen oder beschrieben werden.
FOR UPDATE	Der Cursor kann gelesen oder beschrieben werden. Dieser Wert ist exakt derselbe wie READ-WRITE.

Rückgabe

LONG VARCHAR

Siehe auch

- „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „PLAN-Funktion [Verschiedene]“ auf Seite 337
- „EXPLANATION-Funktion [Verschiedene]“ auf Seite 260

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Die folgende Interactive SQL-Beispielanweisung übergibt eine SELECT-Anweisung als Zeichenfolgenparameter und gibt den Ausführungsplan einer Abfrage zurück. Sie speichert den Plan in der Datei *plan.saplan*, die mit Interactive SQL geöffnet und gelesen werden kann.

```
SELECT GRAPHICAL_PLAN( 'SELECT * FROM GROUPO.Departments WHERE DepartmentID
> 100' );
OUTPUT TO 'plan.saplan' FORMAT TEXT QUOTE '' HEXADECIMAL ASIS;
```

Die folgende Anweisung gibt eine Zeichenfolge zurück, die einen grafischen Plan für einen Keyset-gesteuerten aktualisierbaren Cursor auf die Abfrage `SELECT * FROM Departments WHERE`

GROUPO.DepartmentID > 100 enthält. Sie lässt des weiteren den Server die Abfrage ausführen, um den wiedergegebenen Plan mit den tatsächlichen Ausführungs-Statistiken zu versehen, zusätzlich zu den vom Optimierer verwendeten geschätzten Statistiken.

```
SELECT GRAPHICAL_PLAN(
  'SELECT * FROM GROUPO.Departments WHERE DepartmentID > 100',
  2,
  'keyset-driven', 'for update' );
```

GREATER-Funktion [Verschiedene]

Gibt den größeren von zwei Parameterwerten zurück.

Syntax

GREATER(*expression-1*, *expression-2*)

Parameter

- ***expression-1*** Der erste Parameterwert, der verglichen werden soll.
- ***expression-2*** Der zweite Parameterwert, der verglichen werden soll.

Rückgabe

Der Rückgabotyp dieser Funktion hängt von den angegebenen Ausdrücken ab. Das bedeutet, dass der Datenbankserver beim Auswerten der Funktion zuerst nach einem Datentyp sucht, mit dem alle Ausdrücke verglichen werden können. Wenn er gefunden wird, vergleicht der Datenbankserver die Ausdrücke und gibt das Ergebnis in dem für den Vergleich verwendeten Datentyp zurück. Wenn der Datenbankserver keinen gemeinsamen Vergleichsdattentyp findet, wird ein Fehler zurückgegeben.

Bemerkungen

Wenn die Parameter gleich sind, wird der erste zurückgegeben.

Siehe auch

- „[LESSER-Funktion \[Verschiedene\]](#)“ auf Seite 302

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 10 zurück:

```
SELECT GREATER( 10, 5 ) FROM dummy;
```

GROUPING-Funktion [Aggregat]

Gibt an, ob eine Spalte in einer GROUP BY-Vorgangs-Ergebnismenge NULL ist, weil sie Teil einer Zwischensummenzeile ist oder aufgrund der zugrunde liegenden Daten.

Syntax

GROUPING(*group-by-expression*)

Parameter

- **GROUP_BY-Ausdruck** Ein Ausdruck, der als Grouping-Spalte in einer Ergebnismenge einer Abfrage erscheint, die eine GROUP BY-Klausel verwendet. Die Funktion kann auch verwendet werden, um Zwischensummenzeilen zu kennzeichnen, die von einem ROLLUP- oder CUBE-Vorgang hinzugefügt wurden.

Rückgabe

- **1** Gibt an, dass *GROUP_BY-Ausdruck* NULL ist, weil es Teil einer Zwischensummenzeile ist. Die Spalte ist keine Prefix-Spalte für diese Zeile.
- **0** Gibt an, dass *GROUP_BY-Ausdruck* eine Prefix-Spalte einer Zwischensummenzeile ist.

Siehe auch

- „ROLLUP-Klausel“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „CUBE-Klausel“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „GROUP BY GROUPING SETS“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „SELECT-Anweisung“ auf Seite 1020
- „Erkennen von NULL-Platzhaltern mit der GROUPING-Funktion“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Die GROUPING-Funktion ist Teil der optionalen SQL/2008-Sprachenfunktion T431, "Extended grouping capabilities".

Beispiel

Beispiele für diese Funktion finden Sie unter „Erkennen von NULL-Platzhaltern mit der GROUPING-Funktion“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

HASH-Funktion [Zeichenfolge]

Gibt den angegebenen Wert in einem Hash-Format zurück.

Syntax

HASH(*string-expression*[, *algorithm*])

Parameter

- **Zeichenfolgenausdruck** Der Wert, der im Hash-Format zurückgegeben werden soll. Dieser Parameter beachtet die Groß- und Kleinschreibung, sogar in Datenbanken, die das nicht tun.
- **algorithm** Der für den Hash-Vorgang zu verwendende Algorithmus. Mögliche Werte sind CRC32, MD5, SHA1, SHA1_FIPS, SHA256 und SHA256_FIPS. Standardmäßig wird der MD5-Algorithmus verwendet. FIPS-zertifizierte Hash-Algorithmen erfordern eine getrennte Lizenz.

Rückgabe

Es folgen die Rückgabetypen, abhängig vom verwendeten Algorithmus:

- CRC32 gibt eine hexadezimale Zeichenfolge zurück. Verwenden Sie die HEXTOINT-Funktion, um eine hexadezimale Zeichenfolge in eine 32-Bit-Ganzzahl zu konvertieren.
- MD5 gibt VARCHAR(32) zurück.
- SHA1 gibt VARCHAR(40) zurück.
- SHA1_FIPS gibt VARCHAR(40) zurück.
- SHA256 gibt VARCHAR(64) zurück.
- SHA256_FIPS gibt VARCHAR(64) zurück.

Bemerkungen

Wenn die Hash-Funktion verwendet wird, wird der Wert in eine Bytesequenz konvertiert, die für jeden Wert, der an die Funktion übergeben wird, eindeutig ist.

Wenn der Datenbankserver mit der Option `-fips` gestartet wurde, kann sich der verwendete Algorithmus oder das Verhalten ändern, und zwar folgendermaßen:

- SHA1_FIPS wird verwendet, wenn SHA1 angegeben ist.
- SHA256_FIPS wird verwendet, wenn SHA256 angegeben ist.
- Ein Fehler wird zurückgegeben, wenn MD5 angegeben ist.
- Der CRC32-Algorithmus, der nicht FIPS-zertifiziert ist, ist im FIPS-Modus zulässig, weil er nicht als Verschlüsselungsalgorithmus betrachtet wird.

Vorsicht

Jeder Algorithmus ist ein Einweg-Hash. Es ist nicht möglich, den ursprünglichen Wert anhand des Hash-Werts wieder herzustellen.

Siehe auch

- „Zeichenfolgenfunktionen“ auf Seite 163
- „Datenbankserveroption `-fips`“ [[SQL Anywhere Server - Datenbankadministration](#)]
- [SQL Anywhere-Sicherheitsoption](#) [[SQL Anywhere 16 - Einführung](#)]
- „HEXTOINT-Funktion [Datentypkonvertierung]“ auf Seite 275

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel erstellt eine Tabelle namens `user_info`, um Daten über die Benutzer einer Anwendung, wie Benutzer-ID und Kennwort, zu speichern. In die Tabelle wird auch eine Zeile eingefügt.

Das Kennwort wird mit der HASH-Funktion und dem SHA256-Algorithmus verarbeitet. Das Speichern von Hash-Kennwörtern auf diese Weise kann nützlich sein, wenn Sie Kennwörter nicht in lesbarer Form speichern wollen, aber eine externe Anwendung haben, die Kennwörter vergleicht.

```
CREATE TABLE user_info (  
    employee_id    INTEGER NOT NULL PRIMARY KEY,  
    user_name CHAR(80),  
    user_pwd CHAR(80) );  
INSERT INTO user_info  
VALUES ( '1', 's_phillips', HASH( 'mypass', 'SHA256' ) );
```

HEXTOBIN-Funktion [Datentypkonvertierung]

Gibt das LONG BINARY-Äquivalent einer hexadezimalen Zeichenfolge zurück

Syntax

HEXTOBIN(*hexadecimal-string*)

Parameter

- **hexadecimal-string** Die Zeichenfolge, die in eine binäre Zeichenfolge konvertiert werden soll.

Rückgabe

Die HEXTOBIN-Funktion gibt eine LONG BINARY-Zeichenfolge zurück. Wenn die Anzahl der Zeichen in der Eingabe ungerade ist, wird die Eingabe von links mit einer Null aufgefüllt. Die Länge des Ergebnisses ist die Länge der Eingabezeichenfolge, geteilt durch 2. Wenn die Eingabezeichenfolge nicht hexadezimale Zeichen enthält, wird ein Fehler zurückgegeben.

Bemerkungen

Die HEXTOINT-Funktion akzeptiert Zeichenfolgenlitterale oder Variable, die nur aus Ziffern und den groß- oder kleingeschriebenen Buchstaben A-F bestehen.

Die Funktionen BINTOHEX, CAST, CONVERT, HEXTOBIN, HEXTOINT und INTTOHEX können für Konvertierungen in hexadezimale Werte bzw. aus hexadezimalen Werten verwendet werden. Weitere Hinweise zur Verwendung dieser Funktionen finden Sie unter [Konvertierung in und aus hexadezimalen Werten auf Seite 7](#).

Siehe auch

- [„BINTOHEX-Funktion \[Datentypkonvertierung\]“ auf Seite 178](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt eine binäre Zeichenfolge zurück, die 0x313233 enthält:

```
SELECT HEXTOBIN( '313233' );
```

HEXTOINT-Funktion [Datentypkonvertierung]

Gibt das dezimale Ganzzahl-Äquivalent einer hexadezimalen Zeichenfolge zurück

Die Funktionen CAST, CONVERT, HEXTOINT und INTTOHEX können benutzt werden, um in und aus hexadezimalen Werten zu konvertieren. Weitere Hinweise zur Verwendung dieser Funktionen finden Sie unter [Konvertierung in und aus hexadezimalen Werten auf Seite 7](#).

Syntax

HEXTOINT(*hexadecimal-string*)

Parameter

- **hexadecimal-string** Die Zeichenfolge, die in eine Ganzzahl konvertiert werden soll.

Rückgabe

Die HEXTOINT-Funktion gibt das plattformunabhängige SQL INTEGER-Äquivalent der hexadezimalen Zeichenfolge als INT-Wert zurück. Der hexadezimale Wert stellt eine negative Ganzzahl dar, wenn die 8. Ziffer von rechts aus den Ziffern 8-9 besteht und wenn die Buchstaben A-F in Groß- oder Kleinschreibung und die vorhergehenden führenden Ziffern alle aus dem Buchstaben F in Groß- oder Kleinschreibung bestehen. Das Folgende ist keine zulässige Verwendung von HEXTOINT, weil das Argument einen positiven Ganzzahlwert darstellt, der nicht als 32-Bit-Ganzzahl mit Vorzeichen dargestellt werden kann:

```
SELECT HEXTOINT( '0x0080000001' );
```

Bemerkungen

Die HEXTOINT-Funktion akzeptiert Zeichenfolgenlitterale oder Variable, die ausschließlich aus Ziffern und den groß- oder kleingeschriebenen Buchstaben A-F mit oder ohne 0x-Präfix bestehen. Die folgenden Verwendungen von HEXTOINT sind zulässig:

```
SELECT HEXTOINT( '0xFFFFFFFF' );
SELECT HEXTOINT( '0x00000100' );
SELECT HEXTOINT( '100' );
SELECT HEXTOINT( '0xffffffff80000001' );
```

Die HEXTOINT-Funktion entfernt ggf. das 0x-Präfix. Wenn die Daten 8 Stellen überschreiten, müssen sie einen Wert darstellen, der als 32-Bit-Ganzzahlwert mit Vorzeichen dargestellt werden kann.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- [„INTTOHEX-Funktion \[Datentypkonvertierung\]“ auf Seite 292](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 420 zurück:

```
SELECT HEXTOINT( '1A4' );
```

HOUR-Funktion [Datum und Uhrzeit]

Gibt die Stundenkomponente eines TIMESTAMP-Werts zurück.

Syntax

HOUR(*timestamp-expression*)

Parameter

- **Zeitstempelausdruck** Ein TIMESTAMP-Wert.

Rückgabe

SMALLINT

Bemerkungen

Der zurückgegebene Wert ist der Stundenteil des TIMESTAMP-Ausdrucks, ein SMALLINT-Wert zwischen 0 und 23.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung liefert den Wert 21:

```
SELECT HOUR( '1998-07-09 21:12:13' );
```

HOURS-Funktion [Datum und Uhrzeit]

Bearbeitet einen TIMESTAMP-Wert oder gibt die Anzahl von Stunden zwischen zwei TIMESTAMP-Werten zurück. Spezifische Details finden Sie unter dem Verwendungszweck dieser Funktion.

Syntax 1

HOURS (*timestamp-expression*)

Syntax 2

HOURS (*timestamp-expression*, *timestamp-expression*)

Syntax 3

HOURS (*time-or-timestamp-expression*, *integer-expression*)

Parameter

- **Zeit- oder Zeitstempelausdruck** Ein Wert vom Typ TIME oder TIMESTAMP.
- **Zeitstempelausdruck** Ein Wert vom Typ TIMESTAMP.

- **Ganzzahlausdruck** Die Anzahl der Stunden, die zu *Zeit-_oder_Zeitstempelausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl an Stunden von *Zeit-_oder_Zeitstempelausdruck* abgezogen.

Hinweise zum Casting von Datentypen finden Sie unter „[CAST-Funktion \[Datentypkonvertierung\]](#)“ auf Seite 186.

Rückgabe

INTEGER bei Syntax 1 und Syntax 2.

TIME oder TIMESTAMP bei Syntax 3.

Bemerkungen

Das Ergebnis der HOURS-Funktion hängt von deren Argumenten ab.

- **Syntax 1** Wenn Sie einen einzelnen *Zeitstempelausdruck* an die HOURS-Funktion übergeben, wird die Anzahl der Stunden zwischen Mitternacht am 0000-02-29 und *Zeitstempelausdruck* als INTEGER-Wert zurückgegeben.

Hinweis

"0000-02-29" gibt kein tatsächliches Datum wieder. Es ist der von der HOURS-Funktion standardmäßig verwendete TIMESTAMP-Wert.

- **Syntax 2** Wenn Sie zwei TIMESTAMP-Werte an die HOURS-Funktion übergeben, liefert die Funktion die als Ganzzahl ausgedrückte Anzahl der Stunden zwischen den beiden.
- **Syntax 3** Wenn Sie einen TIMESTAMP-Wert und einen INTEGER-Wert an die HOURS-Funktion übergeben, liefert die Funktion das TIMESTAMP-Ergebnis nach dem Addieren der durch die Ganzzahl ausgedrückten Anzahl an Stunden zum *Zeit-_oder_Zeitstempelausdruck*-Argument. Wenn Sie einen TIME-Wert als erstes Argument übergeben, wird entsprechend ein TIME-Wert als Ergebnis zurückgegeben. Syntax 3 unterstützt keine implizite Konvertierung des ersten Arguments. Es kann erforderlich sein, das erste Argument explizit in einen DATE-, TIME- oder TIMESTAMP-Wert zu konvertieren. Wenn das erste Argument ein DATE-Wert ist, wird Mitternacht für die Zeitangabe angenommen.

Verwenden Sie statt Syntax 2 die DATEDIFF-Funktion. Verwenden Sie statt Syntax 3 die DATEADD-Funktion.

Siehe auch

- „[DATEDIFF-Funktion \[Datum und Uhrzeit\]](#)“ auf Seite 218
- „[DATEADD-Funktion \[Datum und Uhrzeit\]](#)“ auf Seite 217

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgenden Anweisungen geben den Wert "4" zurück, womit angezeigt wird, dass der zweite TIMESTAMP-Wert vier Stunden nach dem ersten liegt. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEDIFF).

```
SELECT HOURS( '1999-07-13 06:07:12', '1999-07-13 10:07:12' );  
  
SELECT DATEDIFF( hour, '1999-07-13 06:07:12', '1999-07-13 10:07:12' );
```

Die folgende Anweisung gibt den Wert 17517342 zurück:

```
SELECT HOURS( '1998-07-13 06:07:12' );
```

Die folgenden Anweisungen geben den DATETIME-Wert 1999-05-13 02:05:07.000 zurück. Es wird empfohlen, dass Sie das zweite Beispiel (DATEADD) verwenden.

```
SELECT HOURS( CAST( '1999-05-12 21:05:07' AS DATETIME ), 5 );  
  
SELECT DATEADD( hour, 5, '1999-05-12 21:05:07' );
```

HTML_DECODE-Funktion [Verschiedene]

Dekodiert Sonderzeichen-Entitäten, die in literalen HTML-Zeichenfolgen auftreten.

Syntax

```
HTML_DECODE( string )
```

Parameter

- **string** Beliebige literale Zeichenfolge, die in einem HTML-Dokument verwendet wird

Rückgabe

LONG VARCHAR oder LONG NVARCHAR.

Bemerkungen

Diese Funktion gibt das Zeichenfolgenargument zurück, nachdem die entsprechenden Ersetzungen durchgeführt wurden. Die folgende Tabelle enthält Beispiele der zugelassenen Zeichen-Entitäten.

Zeichen	Ersetzt durch
"	"
'	'
&	&
<	<
>	>

Zeichen	Ersetzt durch
<code>&#xhexadezimale_Zahl;</code>	Unicode-Codepunkt, als eine hexadezimale Zahl angegeben. Beispiel: <code>&#x27;</code> gibt einen Apostroph zurück.
<code>&#Dezimalzahl;</code>	Unicode-Codepunkt, als eine Dezimalzahl angegeben. Beispiel: <code>&#8482;</code> gibt das Warenzeichensymbol zurück.

Wenn ein Unicode-Codepunkt angegeben ist und der Wert in ein Zeichen im Datenbank-Zeichensatz konvertiert werden kann, wird er in ein Zeichen konvertiert. Andernfalls wird er nicht interpretiert zurückgegeben.

SQL Anywhere unterstützt alle Zeichenentitätsreferenzen, die in der HTML 4.01-Spezifikation angegeben sind. Siehe <http://www.w3.org/TR/html4/> und <http://www.w3.org/TR/html4/sgml/entities.html#h-24.2>.

Siehe auch

- „HTML_ENCODE-Funktion [Verschiedene]“ auf Seite 279
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Die folgende Anweisung gibt die Zeichenfolge `<p>The piano was made for 'Steinway & Sons' .</p>` zurück:

```
SELECT HTML_DECODE('&lt;p&gt;The piano was made ' ||
  'by &lsquo;Steinway &amp; Sons&rsquo;.&lt;/p&gt;')
```

Die folgende Anweisung gibt die Zeichenfolge `<p>It cost €85.000,000. </p>` zurück:

```
SELECT HTML_DECODE('&lt;p&gt;It cost &euro;85.000,000.&lt;/p&gt;')
```

HTML_ENCODE-Funktion [Verschiedene]

Kodiert Sonderzeichen innerhalb von Zeichenfolgen, die in HTML-Dokumente eingefügt werden sollen.

Syntax

```
HTML_ENCODE( string )
```

Parameter

- **string** Beliebige Zeichenfolge, die in einem HTML-Dokument verwendet werden soll.

Rückgabe

LONG VARCHAR oder LONG NVARCHAR.

Bemerkungen

Diese Funktion gibt das Zeichenfolgenargument zurück, nachdem die folgenden Ersetzungen durchgeführt wurden:

Zeichen	Ersetzt durch
"	"
'	'
&	&
<	<
>	>
Codes <i>nn</i> weniger als 0x20	&#xnn;

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- [„HTML_DECODE-Funktion \[Verschiedene\]“ auf Seite 278](#)
- [„Webdienstfunktionen“ auf Seite 162](#)
- [„Webdienst-Systemprozeduren“ auf Seite 1162](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Im folgenden Beispiel wird die Zeichenfolge '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">' zurückgegeben:

```
SELECT HTML_ENCODE('<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">')
```

HTTP_BODY-Funktion [Webdienst]

Gibt den Hauptteil einer HTTP-Anforderung im Binärformat zurück. Beispiel: In einer POST-Anforderung sind dies die unbearbeiteten POST-Daten.

Syntax

HTTP_BODY()

Parameter

Keine

Rückgabe

LONG VARCHAR-Wert mit dem Hauptteil der HTTP Anforderung in binärer Form. Es wird keine Zeichensatzkonvertierung durchgeführt.

Bemerkungen

Ein NULL-Wert wird zurückgegeben, wenn der Hauptteil der Anforderung nicht existiert oder die Funktion nicht aus einem Webdienst aufgerufen wird.

Diese Funktion ist innerhalb der externen PHP-Umgebung nützlich.

Siehe auch

- „sa_http_php_page-Systemprozedur“ auf Seite 1236
- „sa_http_php_page_interpreted-Systemprozedur“ auf Seite 1237
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

HTTP_DECODE-Funktion [Webdienst]

Dekodiert HTTP-kodierte Zeichenfolgen. Dies wird auch als "URL-Dekodierung" bezeichnet.

Syntax

HTTP_DECODE(*string*)

Parameter

- **string** Beliebige Zeichenfolge, die einer URL oder einem URL-kodierten Anforderungshauptteil entnommen wurde.

Rückgabe

LONG VARCHAR oder LONG NVARCHAR

Bemerkungen

Diese Funktion gibt das Zeichenfolgenargument zurück, nachdem alle Zeichensequenzen im Format %*nn* durch den Zeichencode *nn* ersetzt wurden, wobei *nn* ein hexadezimaler Wert ist. Zusätzlich werden alle Pluszeichen (+) durch Leerstellen ersetzt.

Siehe auch

- „HTTP_ENCODE-Funktion [Webdienst]“ auf Seite 282
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Die folgende Anweisung gibt die Zeichenfolge `http://dcx.sybase.com` zurück:

```
SELECT HTTP_DECODE( 'http%3A%2F%2Fdcx.sybase.com' )
```

HTTP_ENCODE-Funktion [Webdienst]

Kodiert Zeichenfolgen für die Verwendung mit HTTP. Dies wird auch als "URL-Kodierung" bezeichnet.

Syntax

HTTP_ENCODE(*string*)

Parameter

- **string** Beliebige Zeichenfolge, die für den HTTP-Transport kodiert werden soll.

Rückgabe

LONG VARCHAR oder LONG NVARCHAR

Bemerkungen

Diese Funktion gibt das Zeichenfolgenargument zurück, nachdem die folgenden Ersetzungen durchgeführt wurden. Zusätzlich werden alle Zeichen mit hexadezimalen Codes kleiner als 20 oder größer als 7E durch `%nn` ersetzt, wobei `nn` der Zeichencode ist.

Zeichen	Ersetzt durch
Leerstelle	%20
"	%22
#	%23
%	%25
&	%26
,	%2C
;	%3B
<	%3C
>	%3E
[%5B

Zeichen	Ersetzt durch
\	%5C
]	%5D
`	%60
{	%7B
	%7C
}	%7D
Zeichencodes <i>nn</i> , die kleiner als 0x20 und größer als 0x7f sind	% <i>nn</i>

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- [„HTTP_DECODE-Funktion \[Webdienst\]“ auf Seite 281](#)
- [„Webdienstfunktionen“ auf Seite 162](#)
- [„Webdienst-Systemprozeduren“ auf Seite 1162](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Die folgende Anweisung gibt die Zeichenfolge `/opt%26id=123%26text='oid:c%09d%20ef'` zurück:

```
SELECT HTTP_ENCODE('/opt&id=123&text=''oid:c\x09d ef'')
```

HTTP_HEADER-Funktion [Webdienst]

Gibt den Wert eines HTTP-Anforderungsheaders zurück.

Syntax

```
HTTP_HEADER( header-field-name )
```

Parameter

- **Headerfeldname** Der Name eines HTTP-Anforderungsheaderfelds.

Rückgabe

LONG VARCHAR.

Hinweis

Der Datentyp des Ergebnisses ist LONG VARCHAR. Wenn Sie HTTP_HEADER in einer SELECT INTO-Anweisung verwenden, müssen Sie über eine Unstructured Data Analytics Option-Lizenz verfügen oder CAST verwenden und HTTP_HEADER auf den richtigen Datentyp und die richtige Größe setzen.

Bemerkungen

Diese Funktion gibt den Wert des angegebenen HTTP-Anforderungsheaderfelds zurück oder NULL, wenn es nicht vorhanden ist oder die Funktion nicht von einem HTTP-Dienst aufgerufen wird. Sie wird verwendet, wenn eine HTTP-Anforderung über einen Webdienst verarbeitet wird.

Einige Header, die beim Verarbeiten einer HTTP-Webdienstanforderung von Interesse sind, sind die folgenden:

- **Cookie** Die vom Client gespeicherten Cookie-Werte, die mit der angeforderten URI verknüpft sind (falls vorhanden)
- **Referer** Die URL der Seite (z.B. `http://documents.sample.com:80/index.html`), die den Link zum angeforderten URI enthält.
- **Host** Der Internet-Hostname oder IP-Adresse und Portnummer der angeforderten Ressource, wie aus dem ursprünglichen URI abgerufen, die vom Benutzer oder von der referenzierenden Ressource angegeben wurde, z.B. `webserver.sample.com:8082`.
- **User-Agent** Der Name der Clientanwendung, z.B. `Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0`.
- **Accept-Encoding** Eine Liste von Kodierungen für die Antwort, die für die Clientanwendung akzeptabel sind, z.B. `gzip, deflate`.

Weitere Hinweise zu diesen Headern finden Sie unter <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>.

Die folgenden speziellen Header ermöglichen den Zugriff auf die Elemente in der Anforderungszeile einer Clientanforderung.

- **@HttpMethod** Gibt den Typ der Anforderung zurück, die verarbeitet wird. Mögliche Werte sind DELETE, HEAD, GET, PUT oder POST.
- **@HttpURI** Die vollständige URI der Anforderung, wie sie in der HTTP-Anforderung angegeben wurde (z.B. `/myservice?id=-123&version=109&lang=en`).
- **@HttpVersion** Die HTTP-Version der Anforderung (z.B. `HTTP/1.0` oder `HTTP/1.1`).
- **@HttpQueryString** Gibt den Abfrageteil des angeforderten URIs an, sofern er existiert (z.B. `id=-123&version=109&lang=en`).

Siehe auch

- „NEXT_HTTP_HEADER-Funktion [Webdienst]“ auf Seite 325
- „sa_set_http_header-Systemprozedur“ auf Seite 1323
- „sa_http_header_info-Systemprozedur“ auf Seite 1235
- „Zugriff auf vom Client bereitgestellte HTTP-Variablen und -Header“ [*SQL Anywhere Server - Programmierung*]
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung ruft den Cookie-Headerwert ab, wenn sie innerhalb einer von einem HTTP-Webdienst aufgerufenen gespeicherten Prozedur verwendet wird:

```
SET cookie_value = HTTP_HEADER( 'Cookie' );
```

Die folgende Anweisung zeigt den Namen und die Werte der HTTP-Anforderungsheader im Meldungsfenster des Datenbankservers an, wenn sie innerhalb einer von einem HTTP-Webdienst aufgerufenen gespeicherten Prozedur verwendet wird:

```
BEGIN
  declare header_name long varchar;
  declare header_value long varchar;
  set header_name = NULL;
header_loop:
  LOOP
    SET header_name = NEXT_HTTP_HEADER( header_name );
    IF header_name IS NULL THEN
      LEAVE header_loop
    END IF;
    SET header_value = HTTP_HEADER( header_name );
    MESSAGE 'HEADER: ', header_name, '=',
            header_value TO CONSOLE;
  END LOOP;
END;
```

HTTP_RESPONSE_HEADER-Funktion [Webdienst]

Gibt den Wert eines HTTP-Antwortheaders zurück.

Syntax

```
HTTP_RESPONSE_HEADER( header-field-name )
```

Parameter

- **Headerfeldname** Der Name eines HTTP-Antwortheaderfelds.

Rückgabe

LONG VARCHAR

Bemerkungen

Diese Funktion gibt den Wert des angegebenen HTTP-Antwortheaderfelds zurück bzw. NULL, wenn *Headerfeldname* nicht vorhanden ist oder die Funktion nicht über einen HTTP-Dienst aufgerufen wird.

Einige Header, die beim Verarbeiten einer HTTP-Webdienstantwort von Interesse sind, sind die folgenden:

- **Verbindung** Mit dem Feld "Connection" kann der Absender Optionen angeben, die für die betreffende Verbindung gewünscht sind. In einer Antwort des HTTP-Servers von SQL Anywhere lautet die Option immer "close".
- **Content-Length** Das Content-Length-Feld gibt die Größe des Antworthauptteils in der dezimalen Anzahl der Achtbitzeichen an.
- **Content-Type** Das Content-Type-Feld gibt den Medientyp des an den Empfänger gesendeten Hauptteils an. Beispiel: **text/xml**
- **Date** Das Date-Feld steht für das Datum und die Uhrzeit, zu der die Antwort entstand.
- **Expires** Das Expires-Feld liefert das Datum und die Zeit, nach der die Antwort als veraltet eingestuft wird.
- **Speicherort** Das Location-Feld wird verwendet, um den Empfänger an einen Standort für den Abschluss der Anforderung oder die Identifizierung einer neuen Ressource weiterzuleiten.
- **Server** Das Server-Feld enthält Informationen über die Software, die vom ursprünglichen Server verwendet wird, um die Anforderung zu bearbeiten. In einer Antwort des SQL Anywhere-HTTP-Servers wird der Name des Webservers zusammen mit der Versionsnummer zurückgegeben.
- **Transfer-Encoding** Das Transfer-Encoding-Feld gibt an, welcher Umwandlungstyp (falls vorhanden) auf den Nachrichtenhauptteil angewendet wurde, um ihn sicher vom Absender zum Empfänger zu übertragen.
- **User-Agent** Das User-Agent-Feld enthält Informationen über den Benutzeragenten, von dem die Anforderung stammt. In einer Antwort des SQL Anywhere-HTTP-Servers wird der Name des Webservers zusammen mit der Versionsnummer zurückgegeben.
- **WWW-Authenticate** Das WWW-Authenticate-Feld ist in 401 (nicht autorisierten) Antwortnachrichten enthalten.

Weitere Hinweise zu diesen Headern finden Sie unter <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>.

Der folgende spezielle Header ermöglicht den Zugriff auf den Status innerhalb der Antwort einer Serverantwort.

- **@HttpStatus** Gibt den Statuscode der verarbeiteten Anforderung zurück.

Siehe auch

- „NEXT_HTTP_RESPONSE_HEADER-Funktion [Webdienst]“ auf Seite 326
- „sa_set_http_header-Systemprozedur“ auf Seite 1323
- „Zugriff auf vom Client bereitgestellte HTTP-Variablen und -Header“ [*SQL Anywhere Server - Programmierung*]
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung zeigt den Namen und die Werte der HTTP-Antwortheadere im Meldungsfenster des Datenbankservers an, wenn sie innerhalb einer von einem HTTP-Webdienst aufgerufenen gespeicherten Prozedur verwendet wird:

```
BEGIN
  declare header_name long varchar;
  declare header_value long varchar;
  set header_name = NULL;
header_loop:
  LOOP
    SET header_name = NEXT_HTTP_RESPONSE_HEADER( header_name );
    IF header_name IS NULL THEN
      LEAVE header_loop
    END IF;
    SET header_value = HTTP_RESPONSE_HEADER( header_name );
    MESSAGE 'RESPONSE HEADER: ', header_name, '=', header_value TO CONSOLE;
  END LOOP;
```

HTTP_VARIABLE-Funktion [Webdienst]

Gibt den Wert einer HTTP-Variablen zurück.

Syntax

HTTP_VARIABLE(*var-name* [, *instance* [, *attribute*]])

Parameter

- **Variablenname** Der Name einer HTTP-Variablen.
- **instance** Wenn mehr als eine Variable denselben Namen tragen, die Instanznummer der Feldinstanz oder NULL für die erste. Nützlich bei Auswahllisten, in denen mehrere Elemente gewählt werden können.
- **attribute** In einer mehrteiligen Anforderung kann das Attribut den Namen eines Headerfelds angeben, sodass der Wert des Headers für den mehrteiligen Namen zurückgegeben wird.

Wenn kein Attribut angegeben ist, wird der zurückgegebene Wert %-dekodiert und der Zeichensatz wird in den Zeichensatz der Datenbank konvertiert. UTF %-kodierte Daten werden in diesem Modus unterstützt.

Die Attribut kann außerdem einen der folgenden Modi haben:

- **'@BINARY'** Gibt einen binären x-www-form-urlencoded-Wert zurück. Dieser Modus gibt an, dass der zurückgegebene Wert %-dekodiert wird und daher nicht Zeichensatzkonvertiert sein sollte. UTF-8 %-Kodierung wird in diesem Modus nicht unterstützt, da %-kodierte Daten einfach in die jeweils entsprechende Byte-Darstellung dekodiert werden.
- **'@TRANSPORT'** Gibt die unformatierte HTTP-Transportform des Werts zurück, wobei %-Kodierungen beibehalten werden.

Rückgabe

LONG VARCHAR.

Bemerkungen

Diese Funktion gibt den Wert der benannten HTTP-Variablen zurück. Sie wird verwendet, wenn eine HTTP-Anforderung in einem Webdienst verarbeitet wird.

Wenn *Variablenname* nicht existiert, ist der Rückgabewert NULL.

Wenn die Webdienstanforderung vom Typ POST ist und die Variablendaten als Multipart/Formdaten übermittelt werden, empfängt der HTTP-Server HTTP-Header für jede einzelne Variable. Wenn der *attribute*-Parameter angegeben ist, gibt die HTTP_VARIABLE-Funktion den zugeordneten Multipart/Formdaten-Headerwert von der POST-Anforderung für die jeweilige Variable zurück. Eine Variable, die eine Datei, ein Attribut von Content-Disposition, Content-Type und @BINARY repräsentiert, würde jeweils den Datennamen, den Medientyp und die Dateiinhalte zurückgeben.

Alle Eingabedaten durchlaufen normalerweise eine Zeichensatzkonvertierung zwischen dem Zeichensatz des Clients (z.B. einem Browser) und dem Zeichensatz der Datenbank. Wenn allerdings @BINARY für *Attribut* angegeben ist, wird der Variablenwert ohne Zeichensatzkonvertierung oder %-Dekodierung zurückgegeben. Das ist nützlich beim Empfang von Binärdaten (z.B. Bilddaten) von einem Client.

Diese Funktion gibt NULL zurück, wenn die angegebene Instanz nicht vorhanden ist oder wenn die Funktion nicht von einem HTTP-Dienst aufgerufen wird.

Siehe auch

- „NEXT_HTTP_VARIABLE-Funktion [Webdienst]“ auf Seite 327
- „sa_http_variable_info-Systemprozedur“ auf Seite 1239
- „Zugriff auf vom Client bereitgestellte HTTP-Variablen und -Header“ [*SQL Anywhere Server - Programmierung*]
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Die folgende Anweisung ruft die Werte der in der Beispiel-URL angegebenen HTTP-Variablen ab, wenn sie innerhalb einer von einem HTTP-Webdienst aufgerufenen gespeicherten Prozedur verwendet wird:

```
-- http://sample.com/demo/ShowDetail?product_id=300&customer_id=101
BEGIN
  DECLARE v_customer_id LONG VARCHAR;
  DECLARE v_product_id LONG VARCHAR;
  SET v_customer_id = HTTP_VARIABLE( 'customer_id' );
  SET v_product_id = HTTP_VARIABLE( 'product_id' );
  CALL ShowSalesOrderDetail( v_customer_id, v_product_id );
END;
```

Die folgenden Anweisungen fordern die Header Content-Disposition und Content-Type für die Bildvariable an, wenn sie innerhalb einer von einem HTTP-Webdienst aufgerufenen gespeicherten Prozedur verwendet werden:

```
SET v_name = HTTP_VARIABLE( 'image', NULL, 'Content-Disposition' );
SET v_type = HTTP_VARIABLE( 'image', NULL, 'Content-Type' );
```

Die folgende Anweisung fordert den Wert der Bildvariablen in ihrem aktuellen Zeichensatz an, ohne eine Zeichensatzkonvertierung durchzuführen, wenn sie innerhalb einer von einem HTTP-Webdienst aufgerufenen gespeicherten Prozedur verwendet wird:

```
SET v_image = HTTP_VARIABLE( 'image', NULL, '@BINARY' );
```

IDENTITY-Funktion [Verschiedene]

Erstellt Ganzzahlwerte beginnend mit 1 für jede nachfolgende Zeile in einer Abfrage. Ihre Implementierung ist identisch mit der NUMBER-Funktion.

Syntax

IDENTITY(*expression*)

Parameter

- **expression** Ein Ausdruck. Der Ausdruck wird syntaktisch analysiert, aber bei der Durchführung der Funktion ignoriert.

Rückgabe

INT

Bemerkungen

Die Beschreibung der IDENTITY-Funktion ist die gleiche wie die Beschreibung der NUMBER-Funktion.

Siehe auch

- „[NUMBER-Funktion \[Verschiedene\]](#)“ auf Seite 331

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt eine fortlaufend nummerierte Liste der Mitarbeiter zurück:

```
SELECT IDENTITY( 10 ), Surname FROM GROUPO.Employees;
```

IFNULL-Funktion [Verschiedene]

Wenn der erste Ausdruck Null ist, wird der Wert des zweiten Ausdrucks zurückgegeben. Wenn der erste Ausdruck nicht NULL ist, wird der Wert des dritten Ausdrucks zurückgegeben. Wenn der erste Ausdruck nicht NULL ist und es keinen dritten Ausdruck gibt, wird NULL zurückgegeben.

Syntax

IFNULL(*expression-1*, *expression-2* [, *expression-3*])

Parameter

- ***expression-1*** Der zu untersuchende Ausdruck. Sein Wert bestimmt, ob *expression-2* oder *expression-3* zurückgegeben wird.
- ***expression-2*** Der Rückgabewert, wenn *expression-1* NULL ist
- ***expression-3*** Der Rückgabewert, wenn *expression-1* nicht NULL ist

Rückgabe

Welcher Datentyp zurückgegeben wird, hängt vom Datentyp von *expression-2* und *expression-3* ab.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert -66 zurück:

```
SELECT IFNULL( NULL, -66 );
```

Die folgende Anweisung gibt NULL zurück, weil der erste Ausdruck nicht NULL ist und es keinen dritten Ausdruck gibt:

```
SELECT IFNULL( -66, -66 );
```

INDEX_ESTIMATE-Funktion [Verschiedene]

Gibt Selektivitätsschätzungen aus dem Index als Prozentsatz zurück, der durch den Abfrageoptimierer basierend auf angegebenen Parametern berechnet wurde.

Syntax

INDEX_ESTIMATE(*column-name* [, *value* [, *relation-string*]])

Parameter

- ***Spaltenname*** Die Spalte, die in der Schätzung verwendet wird
- ***value*** Der Wert, mit dem die Spalte verglichen wird. Der Standardwert ist NULL.

- **relation-string** Der Vergleichsoperator, der für den Vergleich verwendet wird, in Apostrophe gesetzt. Mögliche Werte für diesen Parameter sind: '=', '>', '<', '>=', '<=', '<>', '!=', '!<', und '!>'. Der Standardwert ist '='.

Rückgabe

REAL

Bemerkungen

Diese Funktion gibt Selektivitätsschätzungen aus dem Index für das Prädikat `column-name relation-string value` zurück. Wenn *Wert* NULL ist und die Beziehungszeichenfolge '=', ist die Selektivität für das Prädikat `column-name IS NULL`. Wenn *value* NULL ist und die Relationszeichenfolge '!=' oder '<>', ist die Selektivität für das Prädikat `column-name IS NOT NULL`.

Siehe auch

- [„ESTIMATE-Funktion \[Verschiedene\]“ auf Seite 252](#)
- [„ESTIMATE_SOURCE-Funktion \[Verschiedene\]“ auf Seite 253](#)
- [„EXPERIENCE_ESTIMATE-Funktion \[Verschiedene\]“ auf Seite 259](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den geschätzten Prozentsatz an EmployeeID-Werten zurück, die höher sind als 200:

```
SELECT INDEX_ESTIMATE( EmployeeID, 200, '>' )
FROM GROUPO.Employees;
```

INSERTSTR-Funktion [Zeichenfolge]

Fügt an einer festgelegten Position eine Zeichenfolge in eine andere Zeichenfolge ein.

Syntax

INSERTSTR(*integer-expression*, *string-expression-1*, *string-expression-2*)

Parameter

- **Ganzzahlausdruck** Die Position, nach der die Zeichenfolge eingefügt werden soll. Verwenden Sie die Null, um eine Zeichenfolge am Beginn einzufügen.
- **Zeichenfolgenausdruck_1** Die Zeichenfolge, in die die andere Zeichenfolge eingefügt werden soll
- **Zeichenfolgenausdruck_2** Die einzufügende Zeichenfolge.

Rückgabe

LONG VARCHAR

Bemerkungen

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- [„STUFF-Funktion \[Zeichenfolge\]“ auf Seite 400](#)
- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert `backoffice` zurück:

```
SELECT INSERTSTR( 0, 'office ', 'back' );
```

INTTOHEX-Funktion [Datentypkonvertierung]

Gibt die Zeichenfolge zurück, die das hexadezimale Äquivalent einer Ganzzahl darstellt.

Syntax

INTTOHEX(*integer-expression*)

Parameter

- **Ganzzahlausdruck** Die Ganzzahl, die in das hexadezimale Format konvertiert wird

Rückgabe

VARCHAR

Bemerkungen

Die Funktionen CAST, CONVERT, HEXTOINT und INTTOHEX können benutzt werden, um in und aus hexadezimalen Werten zu konvertieren.

Siehe auch

- [„HEXTOINT-Funktion \[Datentypkonvertierung\]“ auf Seite 275](#)
- [Konvertierung in und aus hexadezimalen Werten auf Seite 7](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert `0000009c` zurück:

```
SELECT INTTOHEX( 156 );
```

ISDATE-Funktion [Datentypkonvertierung]

Testet, ob ein Zeichenfolgenargument in ein Datum konvertiert werden kann.

Syntax

ISDATE(*string*)

Parameter

- **string** Die Zeichenfolge, die analysiert wird, um herauszufinden, ob sie ein gültiges Datum darstellt.

Rückgabe

INT

Bemerkungen

Wenn eine Konvertierung möglich ist, gibt die Funktion 1 zurück, sonst 0. Wenn das Argument NULL ist, wird "0" zurückgegeben.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75
- „DATE-Datentyp“ auf Seite 123
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Datentyp“ auf Seite 124
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „Ausdrücke“ auf Seite 22
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 330
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „TIMESTAMP-Datentyp“ auf Seite 129
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel werden Daten aus einer externen Datei importiert, Zeilen mit ungültigen Werten exportiert und die restlichen Zeilen in eine dauerhafte Tabelle kopiert:

```
CREATE GLOBAL TEMPORARY TABLE MyData(  
    person VARCHAR(100),  
    birth_date VARCHAR(30),  
    height_in_cms VARCHAR(10)  
) ON COMMIT PRESERVE ROWS;  
LOAD TABLE MyData FROM 'exported.dat';  
UNLOAD  
    SELECT * FROM MyData  
    WHERE ISDATE( birth_date ) = 0  
OR ISNUMERIC( height_in_cms ) = 0  
TO 'badrows.dat';  
INSERT INTO PermData  
    SELECT person, birth_date, height_in_cms  
    FROM MyData  
    WHERE ISDATE( birth_date ) = 1  
AND ISNUMERIC( height_in_cms ) = 1;  
COMMIT;  
DROP TABLE MyData;
```

ISENCRYPTED Funktion [System]

Ermittelt, ob eine Zeichenfolge mithilfe der ENCRYPT-Funktion und des angegebenen Schlüssels verschlüsselt wurde.

Syntax

ISENCRYPTED(*string*, *key*[, *algorithm*])

Rückgabe

INT

Parameter

- **string** Die Zeichenfolge die analysiert werden soll, um zu ermitteln, ob sie verschlüsselt ist. Dieser Parameter beachtet die Groß- und Kleinschreibung, sogar in Datenbanken, die das nicht tun.
- **key** Der zum Verschlüsseln von *string* verwendete Chiffrierschlüssel. Dieser Parameter beachtet die Groß- und Kleinschreibung, sogar in Datenbanken, die das nicht tun.
- **algorithm** Dieser optionale Parameter legt den Algorithmus fest, der beim Verschlüsseln von *string* verwendet wurde. Zu den unterstützten Algorithmen gehören AES, AES256, AES_FIPS und AES256_FIPS.

Sie können einen der FIPS-zertifizierten Algorithmen als *Algorithmus* für jede Plattform festlegen, die FIPS-zertifizierte Verschlüsselung unterstützt.

Hinweis

RSA- und FIPS-zertifizierte Verschlüsselung ist nicht auf allen Plattformen verfügbar. Weitere Hinweise dazu, welche Plattformen welche Verschlüsselungsmethode unterstützen, finden Sie unter <http://www.sybase.com/detail?id=1061806>.

Wenn kein *algorithm* angegeben ist, wird standardmäßig AES verwendet. Wenn der Datenbankserver mit der Serveroption -fips gestartet wurde, lautet der Standardwert AES_FIPS.

Bemerkungen

ISENCRYPTED gibt 1 zurück, wenn die Eingabezeichenfolge mit dem angegebenen Schlüssel verschlüsselt wurde. Andernfalls wird 0 zurückgegeben.

Siehe auch

- [„ENCRYPT-Funktion \[Zeichenfolge\]“ auf Seite 241](#)
- [„DECRYPT-Funktion \[Zeichenfolge\]“ auf Seite 234](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 1 zurück, weil die Eingabezeichenfolge verschlüsselt ist:

```
SELECT ISENCRYPTED( ENCRYPT ('test_string', 'key' ), 'key');
```

ISNULL-Funktion [Verschiedene]

Gibt den ersten Nicht-NULL-Ausdruck in einer Liste zurück. Diese Funktion ist mit der COALESCE-Funktion identisch.

Syntax

ISNULL(*expression*, *expression* [, ...])

Parameter

- **expression** Ein Ausdruck, der auf NULL getestet wird.

Mindestens zwei Ausdrücke müssen an die Funktion übergeben werden und alle Ausdrücke müssen vergleichbar sein.

Rückgabe

Der Rückgabetyt dieser Funktion hängt von den angegebenen Ausdrücken ab. Das bedeutet, dass der Datenbankserver beim Auswerten der Funktion zuerst nach einem Datentyp sucht, mit dem alle Ausdrücke verglichen werden können. Wenn er gefunden wird, vergleicht der Datenbankserver die Ausdrücke und gibt das Ergebnis in dem für den Vergleich verwendeten Datentyp zurück. Wenn der Datenbankserver keinen gemeinsamen Vergleichsdattentyp findet, wird ein Fehler zurückgegeben.

Siehe auch

- [„COALESCE-Funktion \[Verschiedene\]“ auf Seite 191](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert -66 zurück:

```
SELECT ISNULL( NULL ,-66, 55, 45, NULL, 16 );
```

ISNUMERIC-Funktion [Verschiedene]

Bestimmt, ob das Zeichenfolgenargument eine gültige Zahl ist.

Syntax

ISNUMERIC(*string*)

Parameter

- **string** Die Zeichenfolge, die analysiert wird, um zu bestimmen, ob sie eine gültige Zahl darstellt

Rückgabe

INT

Bemerkungen

ISNUMERIC gibt 1 zurück, wenn sich die Eingabezeichenfolge zu einer gültigen Ganzzahl oder Gleitkommazahl auswerten lässt. Andernfalls wird 0 zurückgegeben. Die Funktion gibt außerdem 0 zurück, wenn die Zeichenfolge nur Leerzeichen enthält oder NULL ist.

Folgende Werte bewirken ebenfalls, dass die ISNUMERIC-Funktion "0" zurückgibt:

- Werte, die den Buchstaben "d" oder "D" als Exponenten-Trennzeichen verwenden. Zum Beispiel: 1d2.
- Spezialwerte wie NAN, 0x12, INF und INFINITY.
- NULL (z.B. `SELECT ISNUMERIC(NULL);`).

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel werden Daten von einer externen Datei importiert, Zeilen mit ungültigen Werten exportiert und die restlichen Zeilen in eine permanente Tabelle kopiert. In diesem Beispiel validiert die ISNUMERIC-Anweisung, dass die Werte in height_in_cms numerisch sind.

```
CREATE GLOBAL TEMPORARY TABLE MyData(
    person VARCHAR(100),
    birth_date VARCHAR(30),
    height_in_cms VARCHAR(10)
) ON COMMIT PRESERVE ROWS;
LOAD TABLE MyData FROM 'exported.dat';
UNLOAD
    SELECT *
    FROM MyData
    WHERE ISDATE( birth_date ) = 0
    OR ISNUMERIC( height_in_cms ) = 0
    TO 'badrows.dat';
INSERT INTO PermData
    SELECT person, birth_date, height_in_cms
    FROM MyData
```

```

WHERE ISDATE( birth_date ) = 1
AND ISNUMERIC( height_in_cms ) = 1;
COMMIT;
DROP TABLE MyData;

```

LAST_VALUE-Funktion [Aggregat]

Liefert Werte aus der letzten Zeile eines Fensters.

Syntax

```

LAST_VALUE( [ ALL ] expression[ { RESPECT | IGNORE } NULLS ] )
OVER ( window-spec )

```

Fensterspezifikation: Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Parameter

- **expression** Der auszuwertende Ausdruck. Z.B. ein Spaltenname.

Rückgabe

Datentyp des Arguments.

Bemerkungen

Mit der LAST_VALUE-Funktion können Sie den letzten Wert (entsprechend der Sortierung) in einer Tabelle auswählen, ohne einen Selbst-Join verwenden zu müssen. Dies ist nützlich, wenn Sie den letzten Wert als Basiswert in Berechnungen verwenden wollen.

Die LAST_VALUE-Funktion verwendet den letzten Datensatz aus der Partition, nachdem eine ORDER BY-Anweisung ausgeführt wurde. Anschließend wird der *expression* anhand des letzten Datensatzes berechnet und Ergebnisse werden zurückgegeben.

Wenn IGNORE NULLS angegeben ist, wird der letzte Nicht-NULL-Wert von *expression* zurückgegeben. Wenn RESPECT NULLS angegeben ist (Standardwert), wird der letzte Wert zurückgegeben, unabhängig davon, ob er NULL ist.

Die LAST_VALUE-Funktion unterscheidet sich von den meisten anderen Aggregatfunktionen, weil sie nur mit einer Fensterspezifikation verwendet werden kann.

Elemente von *Fensterspezifikation* können entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel. Siehe „[WINDOW-Klausel](#)“ auf Seite 1124.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „Fenster-Aggregatfunktionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „FIRST_VALUE-Funktion [Aggregat]“ auf Seite 263

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

SQL Anywhere unterstützt die SQL/2008 -Sprachenfunktion F441, "Extended set function support", die es zulässt, dass Operanden von Fensterfunktionen beliebige Ausdrücke sind, die keine Spaltenreferenzen darstellen.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der LAST_VALUE-Funktion als auch eine äußere Referenz enthalten. Ein Beispiel finden Sie unter „AVG-Funktion [Aggregat]“ [[UltraLite - Datenbankverwaltung](#)].

Beispiel

Das folgende Beispiel gibt das Gehalt jedes Mitarbeiters sowie den Namen des Mitarbeiters mit dem höchsten Gehalt in der Abteilung zurück:

```
SELECT GivenName + ' ' + Surname AS employee_name,  
       Salary, DepartmentID,  
       LAST_VALUE( employee_name ) OVER Salary_Window AS highest_paid  
FROM GROUPO.Employees  
WINDOW Salary_Window AS ( PARTITION BY DepartmentID ORDER BY Salary  
                           RANGE BETWEEN UNBOUNDED PRECEDING  
                           AND UNBOUNDED FOLLOWING );
```

employee_name	Salary	DepartmentID	highest_paid
Michael Lynch	24903	500	Jose Martinez
Joseph Barker	27290	500	Jose Martinez
Sheila Romero	27500	500	Jose Martinez
Felicia Kuo	28200	500	Jose Martinez
Jeannette Bertrand	29800	500	Jose Martinez
Jane Braun	34300	500	Jose Martinez
Anthony Rebeiro	34576	500	Jose Martinez
Charles Crowley	41700	500	Jose Martinez
Jose Martinez	55500.8	500	Jose Martinez
Doug Charlton	28300	400	Scott Evans

employee_name	Salary	DepartmentID	highest_paid
Elizabeth Lambert	29384	400	Scott Evans
Joyce Butterfield	34011	400	Scott Evans
Robert Nielsen	34889	400	Scott Evans
Alex Ahmed	34992	400	Scott Evans
Ruth Wetherby	35745	400	Scott Evans
...

Jose Martinez hat in der Abteilung 500 das höchste Gehalt und Scott Evans in der Abteilung 400.

LCASE-Funktion [Zeichenfolge]

Konvertiert alle Zeichen in einer Zeichenfolge in Kleinbuchstaben.

Syntax

LCASE(*string-expression*)

Parameter

- **Zeichenfolgenderausdruck** Die Zeichenfolge, die in Kleinbuchstaben konvertiert werden soll

Rückgabe

- CHAR
- NCHAR
- LONG VARCHAR
- VARCHAR
- NVARCHAR

Bemerkungen

Die LCASE-Funktion mit der LOWER-Funktion identisch.

Siehe auch

- „[LOWER-Funktion \[Zeichenfolge\]](#)“ auf Seite 308
- „[UCASE-Funktion \[Zeichenfolge\]](#)“ auf Seite 419
- „[UPPER-Funktion \[Zeichenfolge\]](#)“ auf Seite 422
- „[Zeichenfolgenfunktionen](#)“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Die gleichwertige LOWER-Funktion ist eine Kernfunktion des SQL/2008-Standards.

Beispiel

Die folgende Anweisung gibt den Wert `chocolate` zurück:

```
SELECT LCASE( 'ChoCoLatE' );
```

LEFT-Funktion [Zeichenfolge]

Gibt mehrere Zeichen ab dem Beginn einer Zeichenfolge zurück.

Syntax

```
LEFT( string-expression, integer-expression )
```

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge.
- **Ganzzahlausdruck** Die Anzahl der zurückzugebenden Zeichen

Rückgabe

- LONG VARCHAR
- LONG NVARCHAR

Bemerkungen

Wenn eine Zeichenfolge Mehrbytezeichen enthält und die korrekte Kollation verwendet wird, ist die Anzahl der zurückgegebenen Bytes möglicherweise größer als die festgelegte Anzahl von Zeichen.

Sie können einen *Ganzzahlausdruck* angeben, der größer ist als der Wert im Zeichenfolgenausdruck-Argument. In diesem Fall wird der gesamte Wert zurückgegeben.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben. Wenn in der Eingabezeichenfolge Zeichenlängensemantik verwendet wurde, wird der Rückgabewert soweit wie möglich mit Ausdrücken der Zeichenlängensemantik beschrieben.

Siehe auch

- [„RIGHT-Funktion \[Zeichenfolge\]“ auf Seite 370](#)
- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die ersten 5 Zeichen jedes Surname-Werts in der Tabelle "Customers" zurück:

```
SELECT LEFT( Surname, 5) FROM GROUPO.Customers;
```

LENGTH-Funktion [Zeichenfolge]

Gibt die Anzahl von Zeichen in der Zeichenfolge zurück.

Syntax

{ **LENGTH** | **LEN** }(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge.

Rückgabe

INT

Bemerkungen

Mit dieser Funktion bestimmen Sie die Länge einer Zeichenfolge. Geben Sie beispielsweise einen Spaltennamen für *Zeichenfolgenausdruck* an, um die Länge der Werte in der Spalte zu bestimmen.

Wenn eine Zeichenfolge Mehrbytezeichen enthält und die korrekte Kollation verwendet wird, gibt LENGTH die Anzahl der Zeichen und nicht die Byte-Anzahl zurück. Wenn der Zeichenfolgenausdruck ein BINARY-Datentyp ist, verhält sich die LENGTH-Funktion wie die BYTE_LENGTH-Funktion.

Hinweis

Sie können die LENGTH-Funktion und die CHAR_LENGTH-Funktion gleichermaßen für CHAR-, VARCHAR-, und LONG VARCHAR und NCHAR-Datentypen verwenden. Sie müssen allerdings die LENGTH-Funktion bei BINARY- und Bit-Array-Datentypen verwenden.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- „[BYTE_LENGTH-Funktion \[Zeichenfolge\]](#)“ auf Seite 183
- „[Internationale Sprachen und Zeichensätze](#)“ [*SQL Anywhere Server - Datenbankadministration*]
- „[Zeichenfolgenfunktionen](#)“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Die LENGTH-Funktion ist eine Erweiterung des Herstellers. Ihre Semantik ist jedoch identisch mit der für die CHAR_LENGTH-Funktion im SQL/2008-Standard. Die Verwendung von LENGTH über einen Zeichenfolgenausdruck vom Typ NCHAR umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion F421.

Beispiel

Die folgende Anweisung gibt den Wert 9 zurück:

```
SELECT LENGTH( 'chocolate' );
```

LESSER-Funktion [Verschiedene]

Gibt den kleineren von zwei Parameterwerten zurück.

Syntax

LESSER(*expression-1*, *expression-2*)

Parameter

- ***expression-1*** Der erste Parameterwert, der verglichen werden soll.
- ***expression-2*** Der zweite Parameterwert, der verglichen werden soll.

Rückgabe

Der Rückgabotyp dieser Funktion hängt von den angegebenen Ausdrücken ab. Das bedeutet, dass der Datenbankserver beim Auswerten der Funktion zuerst nach einem Datentyp sucht, mit dem alle Ausdrücke verglichen werden können. Wenn er gefunden wird, vergleicht der Datenbankserver die Ausdrücke und gibt das Ergebnis in dem für den Vergleich verwendeten Datentyp zurück. Wenn der Datenbankserver keinen gemeinsamen Vergleichsdattentyp findet, wird ein Fehler zurückgegeben.

Bemerkungen

Wenn die Parameter gleich sind, wird der erste Wert zurückgegeben.

Siehe auch

- [„GREATER-Funktion \[Verschiedene\]“ auf Seite 271](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 5 zurück:

```
SELECT LESSER( 10, 5 ) FROM dummy;
```

LIST-Funktion [Aggregat]

Gibt eine durch Begrenzer-Zeichenfolgen getrennte Liste von Werten für jede Zeile in einer Gruppe zurück.

Syntax

LIST(
[**ALL** | **DISTINCT**] *string-expression*
[, *delimiter-string*]
[**ORDER BY** *order-by-expression* [**ASC** | **DESC**], ...])

Parameter

- **Zeichenfolgenausdruck** Ein Zeichenfolgenausdruck, normalerweise ein Spaltenname. Wenn ALL angegeben ist (der Standardwert) wird für jede Zeile der Gruppe der Wert von *Zeichenfolgenausdruck* zur Ergebniszeichenfolge hinzugefügt, wobei die Werte durch *Trennzeichenfolge* getrennt werden. Wenn DISTINCT angegeben wurde, werden nur eindeutige *Zeichenfolgenausdruck*-Werte hinzugefügt.
- **delimiter-string** Eine Trennzeichenfolge für die Listeneinträge. Die Standardeinstellung ist ein Komma. Wenn NULL oder eine leere Zeichenfolge angegeben wird, gibt es keinen Begrenzer. *delimiter-string* muss eine Konstante sein.
- **Sortierausdruck** Die von der Funktion zurückgegebenen Elemente sortieren. Vor diesem Argument steht kein Komma, damit ist es einfacher einzusetzen, falls keine *Begrenzer-Zeichenfolge* angegeben wurde.

Sortierausdruck kann kein Ganzzahlliteral sein. Er kann aber eine Variable sein, die einen Ganzzahlenliteral enthält.

Wenn eine ORDER BY-Klausel Konstante enthält, werden sie vom Optimierer interpretiert und dann durch eine entsprechende ORDER BY-Klausel ersetzt. Der Optimierer interpretiert z.B. "ORDER BY 'a'" als ORDER BY-Ausdruck.

Ein Abfrageblock, der mehr als eine Aggregatfunktion mit gültigen ORDER BY-Klauseln enthält, kann ausgeführt werden, wenn die ORDER BY-Klauseln logisch in einer einzigen ORDER BY-Klausel kombiniert können werden. Zum Beispiel sind folgende Klauseln vorhanden:

```
ORDER BY expression1, 'a', expression2
ORDER BY expression1, 'b', expression2, 'c', expression3
```

Sie werden in der folgenden Klausel zusammengefasst:

```
ORDER BY expression1, expression2, expression3
```

Rückgabe

- LONG VARCHAR
- LONG NVARCHAR

Bemerkungen

Die LIST-Funktion gibt die Verkettung (mit Begrenzern) aller Nicht-NULL-Werte für X für jede Zeile der Gruppe zurück. Wenn in der Gruppe nicht wenigstens eine Zeile mit einem definierten X-Wert existiert, gibt LIST(X) eine leere Zeichenfolge zurück.

NULL-Werte und leere Zeichenfolgen werden von der LIST-Funktion ignoriert.

Eine LIST-Funktion kann nicht als Fensterfunktion verwendet werden, aber sie kann als Eingabe für eine Fensterfunktion verwendet werden.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

SQL Anywhere unterstützt die SQL/2008 -Sprachenfunktion F441, "Extended set function support", die es zulässt, dass Operanden von Aggregatfunktionen beliebige Ausdrücke sind, die keine Spaltenreferenzen darstellen.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der LIST-Funktion als auch eine äußere Referenz enthalten. Siehe „Fehlerbehandlung: Aggregatfunktionen und äußere Referenzen“ [[SQL Anywhere 16 - Änderungen und Upgrades](#)].

Ein Beispiel finden Sie unter „AVG-Funktion [Aggregat]“ [[UltraLite - Datenbankverwaltung](#)].

Siehe auch

- „sa_split_list-Systemprozedur“ auf Seite 1332
- „ARRAY_AGG-Funktion [Aggregat]“ auf Seite 170

Beispiele

Die folgende Anweisung gibt den Wert 487 Kennedy Court, 547 School Street zurück:

```
SELECT LIST( Street ) FROM GROUPO.Employees  
WHERE GivenName = 'Thomas';
```

In der folgenden Anweisung werden Mitarbeiter-IDs aufgeführt. Jede Zeile in der Ergebnismenge enthält eine Trennkomaliste mit Mitarbeiter-IDs für nur eine Abteilung.

```
SELECT LIST( EmployeeID )  
FROM GROUPO.Employees  
GROUP BY DepartmentID;
```

LIST(EmployeeID)
102,105,160,243,247,249,266,278,...
129,195,299,467,641,667,690,856,...
148,390,586,757,879,1293,1336,...
184,207,318,409,591,888,992,1062,...
191,703,750,868,921,1013,1570,...

Die folgende Anweisung sortiert die Mitarbeiter-IDs nach den Nachnamen der Mitarbeiter:

```
SELECT LIST( EmployeeID ORDER BY Surname ) AS "Sorted IDs"  
FROM GROUPO.Employees  
GROUP BY DepartmentID;
```

Sorted IDs
1013,191,750,921,868,1658,...
1751,591,1062,1191,992,888,318,...
1336,879,586,390,757,148,1483,...
1039,129,1142,195,667,1162,902,...
160,105,1250,247,266,249,445,...

Die folgende Anweisung gibt Listen mit Trennsemikolons zurück: Beachten Sie die Position der ORDER BY-Klausel und des Listentrennzeichens:

```
SELECT LIST( EmployeeID, ';' ORDER BY Surname ) AS "Sorted IDs"
FROM GROUPO.Employees
GROUP BY DepartmentID;
```

Sorted IDs
1013;191;750;921;868;1658;703;...
1751;591;1062;1191;992;888;318;...
1336;879;586;390;757;148;1483;...
1039;129;1142;195;667;1162;902; ...
160;105;1250;247;266;249;445;...

Achten Sie auf den Unterschied zwischen der vorigen und der folgenden Anweisung, die eine Liste mit Trennkommas der Mitarbeiter-IDs, sortiert nach einem zusammengesetzten Sortierschlüssel von (Surname, ';') zurückgibt:

```
SELECT LIST( EmployeeID ORDER BY Surname, ';' ) AS "Sorted IDs"
FROM GROUPO.Employees
GROUP BY DepartmentID;
```

LOCATE-Funktion [Zeichenfolge]

Gibt die Position einer Zeichenfolge in einer anderen zurück.

Syntax

LOCATE(*string-expression-1*, *string-expression-2* [, *integer-expression*])

Parameter

- **Zeichenfolgenausdruck_1** Die zu durchsuchende Zeichenfolge.

- **Zeichenfolgenausdruck_2** Die Zeichenfolge, nach der gesucht wird. Die Zeichenfolge ist auf 255 Byte beschränkt.
- **Ganzzahlausdruck** Die Zeichenposition in der Zeichenfolge, an der die Suche beginnen soll. Das erste Zeichen hat Position 1. Wenn das Start-Offset negativ ist, gibt die locate-Funktion anstatt des ersten den letzten passenden Zeichenfolgen-Offset zurück. Ein negativer Offset gibt an, wieviel vom Ende einer Zeichenfolge von der Suche auszunehmen ist. Die Anzahl der ausgeschlossenen Byte wird mit der folgenden Formel berechnet: $(-1 * \text{Offset}) - 1$.

Rückgabe

INT

Bemerkungen

Wenn *Ganzzahlausdruck* angegeben ist, beginnt die Suche an dieser Position der Zeichenfolge.

Die erste Zeichenfolge kann lang sein (länger als 255 Byte), die zweite ist aber auf 255 Byte begrenzt. Wenn eine lange Zeichenfolge als zweites Argument angegeben wird, gibt die Funktion NULL zurück. Wenn die Zeichenfolge nicht gefunden wird, wird "0" zurückgegeben. Die Suche nach einer Zeichenfolge mit Länge null gibt "1" zurück. Falls eines der Argumente NULL ist, ist das Ergebnis NULL.

Wenn Mehrbytezeichen mit der richtigen Kollation verwendet werden, können sich die Startposition und der Rückgabewert von den *Byte*-Positionen unterscheiden.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- „Zeichenfolgenfunktionen“ auf Seite 163
- „CHARINDEX-Funktion [Zeichenfolge]“ auf Seite 189

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung liefert den Wert 8:

```
SELECT LOCATE(  
    'office party this week - rsvp as soon as possible',  
    'party',  
    2 );
```

Die folgende Anweisung

```
BEGIN  
    DECLARE STR LONG VARCHAR;  
    DECLARE POS INT;  
    SET str = 'c:\test\functions\locate.sql';  
    SET pos = LOCATE( str, '\', -1 );  
    select str, pos,  
        SUBSTR( str, 1, pos -1 ) AS path,  
        SUBSTR( str, pos +1 ) AS filename;  
END;
```

Es wird diese Ausgabe ausgegeben:

str	pos	path	filename
c:\test\functions\locate.sql	18	c:\test\functions	locate.sql

LOG-Funktion [Nummerisch]

Gibt den natürlichen Logarithmus einer Zahl zurück.

Syntax

LOG(*numeric-expression*)

Parameter

- **Nummerischer_Ausdruck** Die Zahl.

Rückgabe

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück. Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Bemerkungen

Das Argument ist ein Ausdruck, der den Wert eines beliebigen integrierten numerischen Datentyps zurückgibt.

Siehe auch

- [„LOG10-Funktion \[Nummerisch\]“ auf Seite 307](#)

Standards und Kompatibilität

- **SQL/2008** Die SQL/2008-Standard definiert die Funktion für den natürlichen Logarithmus mit dem Schlüsselwort LN. Die Funktion für den natürlichen Logarithmus umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Die folgende Anweisung gibt den natürlichen Logarithmus von 50 zurück:

```
SELECT LOG( 50 );
```

LOG10-Funktion [Nummerisch]

Gibt den Logarithmus zur Basis 10 einer Zahl zurück.

Syntax

LOG10(*numeric-expression*)

Parameter

- **Nummerischer_Ausdruck** Die Zahl.

Rückgabe

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelte genaue Gleitkommazahl durch. Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Bemerkungen

Das Argument ist ein Ausdruck, der den Wert eines beliebigen integrierten numerischen Datentyps zurückgibt.

Siehe auch

- [„LOG-Funktion \[Numerisch\]“ auf Seite 307](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Logarithmus zur Basis 10 für 50 zurück:

```
SELECT LOG10( 50 );
```

LOWER-Funktion [Zeichenfolge]

Konvertiert alle Zeichen in einer Zeichenfolge in Kleinbuchstaben. Diese Funktion ist mit der LCASE-Funktion identisch.

Syntax

LOWER(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, die in Kleinbuchstaben konvertiert werden soll

Rückgabe

CHAR, VARCHAR, LONG VARCHAR, NCHAR, NVARCHAR, oder LONG NVARCHAR entsprechend dem Datentyp des Arguments.

Bemerkungen

Die LCASE-Funktion mit der LOWER-Funktion identisch.

Siehe auch

- [„LCASE-Funktion \[Zeichenfolge\]“ auf Seite 299](#)
- [„UCASE-Funktion \[Zeichenfolge\]“ auf Seite 419](#)
- [„UPPER-Funktion \[Zeichenfolge\]“ auf Seite 422](#)
- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Die LOWER-Funktion ist eine Kernfunktion des SQL/2008-Standards. Die Verwendung von LOWER über einen Ausdruck vom Datentyp NCHAR umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion F421.

Beispiel

Die folgende Anweisung gibt den Wert chocolate zurück:

```
SELECT LOWER( 'chOCOLate' );
```

LTRIM-Funktion [Zeichenfolge]

Entfernt führende Leerzeichen aus der Zeichenfolge.

Syntax

LTRIM(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die zu kürzende Zeichenfolge

Rückgabe

- VARCHAR
- NVARCHAR
- LONG VARCHAR
- LONG NVARCHAR

Bemerkungen

Die tatsächliche Länge des Ergebnisses ist die Länge des Ausdrucks minus der Anzahl der entfernten Zeichen. Wenn alle Zeichen entfernt werden, ist das Ergebnis eine leere Zeichenfolge.

Wenn der Parameter NULL sein kann, kann das Ergebnis NULL sein.

Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- „[RTRIM-Funktion \[Zeichenfolge\]](#)“ auf Seite 376
- „[TRIM-Funktion \[Zeichenfolge\]](#)“ auf Seite 415
- „[Zeichenfolgenfunktionen](#)“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Die TRIM-Angaben, die durch den SQL/2008-Standard (LEADING und TRAILING) definiert sind, werden von den SQL Anywhere-Funktionen LTRIM bzw. RTRIM geliefert.

Beispiel

Die folgende Anweisung gibt den Wert `Test Message` zurück, wobei alle führenden Leerzeichen entfernt werden:

```
SELECT LTRIM( '      Test Message' );
```

MAX-Funktion [Aggregat]

Gibt den maximalen Wert für den Ausdruck zurück, der in jeder Zeilengruppe gefunden wurde

Syntax 1

MAX([**ALL** | **DISTINCT**] *expression*)

Syntax 2

MAX([**ALL**] *expression*) **OVER** (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **[ALL] expression** Der Ausdruck, bei dem der Höchstwert berechnet werden soll. Das ist üblicherweise ein Spaltenname.
- **DISTINCT expression** Gibt denselben Wert wie **MAX**(*expression*) zurück, er ist nur der Vollständigkeit halber angeführt.

Rückgabe

Derselbe Datentyp wie das Argument.

Bemerkungen

Zeilen, bei denen *expression* NULL ist, werden ignoriert. Gibt NULL für eine Gruppe zurück, die keine Zeilen enthält.

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel. Siehe „[WINDOW-Klausel](#)“ auf Seite 1124.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Für einfache Vergleiche von zwei Ausdrücken können Sie auch die GREATER-Funktion verwenden.

Siehe auch

- „GREATER-Funktion [Verschiedene]“ auf Seite 271
- „MIN-Funktion [Aggregat]“ auf Seite 313
- „Fehlerbehandlung: Aggregatfunktionen und äußere Referenzen“ [*SQL Anywhere 16 - Änderungen und Upgrades*]

Standards und Kompatibilität

- **SQL/2008** Kernfunktion. Wenn MAX als Fensterfunktion verwendet wird (Syntax 2), umfasst sie einen Teil der optionalen SQL/2008-Sprachenfunktion T611, "Grundlegende OLAP-Vorgänge".

Die Möglichkeit, DISTINCT über einen Ausdruck anzugeben, der keine Spaltenreferenz ist, umfasst einen Teil der optionalen SQL-Sprachenfunktion F561, "Full value expressions". SQL Anywhere unterstützt auch die SQL/2008-Sprachenfunktion F441, "Extended set function support", die zulässt, dass die Operanden von Aggregatfunktionen beliebige Ausdrücke sein können, möglicherweise einschließlich äußerer Referenzen zu Ausdrücken in anderen Abfrageblöcken, die keine Spaltenreferenzen sind.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der MAX-Funktion als auch eine äußere Referenz enthalten.

Ein Beispiel finden Sie unter „AVG-Funktion [Aggregat]“ [*UltraLite - Datenbankverwaltung*].

Beispiel

Die folgende Anweisung gibt den Wert 138948,000 zurück, der das höchste Gehalt in der Tabelle "Employees" darstellt:

```
SELECT MAX( Salary )
FROM GROUPO.Employees;
```

MEDIAN-Funktion [Aggregat]

Berechnet den Median eines numerischen Ausdrucks für eine Zeilenmenge.

Syntax 1

```
MEDIAN( [ ALL | DISTINCT ] numeric-expression )
```

Syntax 2

```
MEDIAN( [ ALL ] numeric-expression ) OVER ( window-spec )
```

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Numerischer_Ausdruck** Der Ausdruck, dessen Median über eine Zeilenmenge berechnet wird.
- **DISTINCT-Klausel** Eliminiert mehrfach vorhandene Werte, bevor der Median aus den eindeutigen Werten in der Eingabe berechnet wird.

- **ALL-Klausel** Berechnet den Median aller Werte (auch Duplikatzeilen) in der Eingabe. Dies ist die Standardeinstellung.

Rückgabe

Der Datentyp des Rückgabewerts ist derselbe wie der des Eingabewerts.

NULL-Werte werden bei der Berechnung des Medianwerts ignoriert. Ein NULL-Wert wird jedoch zurückgegeben, wenn eine Gruppe keine Zeilen enthält.

Bemerkungen

Numerischer_Ausdruck-Werte können von jedem numerischen Datentyp sein, mit Ausnahme von BIT. Siehe „[Numerische Datentypen](#)“ auf Seite 103.

Der Median einer endlichen Liste von Zahlen kann gefunden werden, indem alle Beobachtungen vom niedrigsten Wert bis zum höchsten Wert geordnet werden und der mittlere Wert gewählt wird. Wenn eine geraden Anzahl von Beobachtungen vorliegt, ist der Median nicht eindeutig. Daher gibt MEDIAN den Mittelwert der beiden mittleren Werte zurück. Maximal hat die Hälfte der Beobachtungen Werte, die kleiner sind als der Median, und die Hälfte hat Werte, die größer sind als der Median. Wenn beide Gruppen weniger als die Hälfte der Beobachtungen enthalten, sind die Werte für einige der Beobachtungen genau gleich dem Median. Beispiel: Wenn $a < b < c$ ist, dann ist der Median der Liste $\{a, b, c\}$ gleich b . Wenn $a < b < c < d$ ist, dann ist der Median der Liste $\{a, b, c, d\}$ gleich dem Mittelwert aus b und c ($(b + c) / 2$).

Eventuelle Dezimalstellen im Ergebnis des Mittelwerts der beiden mittleren Elemente werden gekürzt, wenn der Eingabedatentyp sie nicht darstellen kann. Um diese Kürzung zu vermeiden, sollten Sie die Eingabe in einen numerischen Datentyp konvertieren, der Dezimalstellen zulässt.

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Siehe *window-spec*-Definition unter „[WINDOW-Klausel](#)“ auf Seite 1124.

window-spec kann nur über eine Partition angegeben werden. (Sie kann keine ROW- oder RANGE-Spezifikation enthalten.) DISTINCT wird nicht unterstützt, wenn eine WINDOW-Klausel verwendet wird. CUBE, ROLLUP und GROUPING SETS werden mit Syntax 1 unterstützt.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „[SUM-Funktion \[Aggregat\]](#)“ auf Seite 403
- „[COUNT-Funktion \[Aggregat\]](#)“ auf Seite 205

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Fensterfunktionen umfassen die optionale SQL/2008 Sprachenfunktion T611, "Grundlegende OLAP-Vorgänge".

SQL Anywhere unterstützt die SQL/2008 -Sprachenfunktion F441, "Extended set function support", die es zulässt, dass Operanden von Fensterfunktionen beliebige Ausdrücke sind, die keine Spaltenreferenzen darstellen.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der MEDIAN-Funktion als auch eine äußere Referenz enthalten. Ein Beispiel finden Sie unter „AVG-Funktion [Aggregat]“ [\[UltraLite - Datenbankverwaltung\]](#).

Beispiel

Die folgende Anweisung gibt das Median-Gehalt aus der Tabelle "Employees" zurück:

```
SELECT MEDIAN( Salary ) FROM GROUPO.Employees;
```

Die folgende Anweisung gibt das Median-Gehalt nach Bundesstaat aus der Tabelle "Employees" zurück:

```
SELECT EmployeeID, Surname, Salary, State,
       MEDIAN( Salary ) OVER Salary_Window
FROM GROUPO.Employees
WINDOW Salary_Window AS ( PARTITION BY State )
ORDER BY State, Surname;
```

MIN-Funktion [Aggregat]

Liefert den minimalen Wert für den Ausdruck, der in jeder Zeilengruppe gefunden wurde

Syntax 1

```
MIN( [ ALL | DISTINCT ] expression )
```

Syntax 2

```
MIN( [ ALL ] expression ) OVER ( window-spec )
```

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **[ALL] expression** Der Ausdruck, bei dem der minimale Wert berechnet werden soll. Das ist üblicherweise ein Spaltenname.
- **DISTINCT expression** Gibt denselben Wert wie MIN(expression) zurück, er ist nur der Vollständigkeit halber angeführt.

Rückgabe

Derselbe Datentyp wie das Argument.

Bemerkungen

Zeilen, bei denen *expression* NULL ist, werden ignoriert. Gibt NULL für eine Gruppe zurück, die keine Zeilen enthält.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie in der *Fensterspezifikation*-Definition der WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „Fensterfunktionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Für einfache Vergleiche von zwei Ausdrücken können Sie auch die LESSER-Funktion verwenden. Siehe „LESSER-Funktion [Verschiedene]“ auf Seite 302.

Siehe auch

- „MAX-Funktion [Aggregat]“ auf Seite 310
- „WINDOW-Klausel“ auf Seite 1124

Standards und Kompatibilität

- **SQL/2008** Kernfunktion. Wenn MIN als Fensterfunktion verwendet wird (Syntax 2), umfasst sie einen Teil der optionalen SQL/2008-Sprachenfunktion T611, "Grundlegende OLAP-Vorgänge".

Die Möglichkeit, DISTINCT über einen Ausdruck anzugeben, der keine Spaltenreferenz ist, umfasst einen Teil der optionalen SQL-Sprachenfunktion F561, "Full value expressions". SQL Anywhere unterstützt auch die SQL/2008-Sprachenfunktion F441, "Extended set function support", die zulässt, dass die Operanden von Aggregatfunktionen beliebige Ausdrücke sein können, möglicherweise einschließlich äußerer Referenzen zu Ausdrücken in anderen Abfrageblöcken, die keine Spaltenreferenzen sind.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der MIN-Funktion als auch eine äußere Referenz enthalten. Siehe „Fehlerbehandlung: Aggregatfunktionen und äußere Referenzen“ [[SQL Anywhere 16 - Änderungen und Upgrades](#)].

Ein Beispiel finden Sie unter „AVG-Funktion [Aggregat]“ [[UltraLite - Datenbankverwaltung](#)].

Beispiel

Die folgende Anweisung gibt den Wert 24903,000 zurück, der das niedrigste Gehalt in der Tabelle "Employees" darstellt:

```
SELECT MIN( Salary )  
FROM GROUPO.Employees;
```

MINUTE-Funktion [Datum und Uhrzeit]

Gibt die Minutenkomponente eines TIMESTAMP-Werts zurück.

Syntax

MINUTE(*timestamp-expression*)

Parameter

- **Zeitstempelausdruck** Der TIMESTAMP-Wert.

Rückgabe

SMALLINT

Bemerkungen

Der zurückgegebene Wert ist der Minutenteil des TIMESTAMP-Ausdrucks, ein SMALLINT-Wert zwischen 0 und 59.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 22 zurück:

```
SELECT MINUTE( '1998-07-13 12:22:34' );
```

MINUTES-Funktion [Datum und Uhrzeit]

Bearbeitet einen TIMESTAMP-Wert oder gibt die Anzahl von Minuten zwischen zwei TIMESTAMP-Werten zurück. Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Syntax 1

MINUTES(*timestamp-expression*)

Syntax 2

MINUTES(*timestamp-expression, timestamp-expression*)

Syntax 3

MINUTES(*timestamp-or-time-expression, integer-expression*)

Parameter

- **Zeitstempelausdruck** Ein Ausdruck vom Typ TIMESTAMP.

- **Zeitstempel-_oder_Zeitausdruck** Ein Ausdruck vom Typ TIME oder TIMESTAMP.
- **Ganzzahlausdruck** Die Anzahl der Minuten, die zu *Zeitstempel-_oder_Zeitausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl von Minuten von *Zeitstempel-_oder_Zeitausdruck* abgezogen.

Rückgabe

INTEGER bei Syntax 1 und Syntax 2.

TIME oder TIMESTAMP bei Syntax 3.

Bemerkungen

Das Ergebnis der MINUTES-Funktion hängt von deren Argumenten ab.

- **Syntax 1** Wenn Sie einen einzelnen *Zeitstempelausdruck* an die MINUTES-Funktion übergeben, wird die Anzahl der Minuten zwischen Mitternacht am 0000-02-29 und *Zeitstempelausdruck* als INTEGER-Wert zurückgegeben.

Hinweis

"0000-02-29" gibt kein tatsächliches Datum wieder. Es ist das von der MINUTES-Funktion verwendete Standarddatum.

- **Syntax 2** Wenn Sie zwei TIMESTAMP-Werte an die MINUTES-Funktion übergeben, liefert die Funktion die als Ganzzahl ausgedrückte Anzahl der Minuten zwischen den beiden.
- **Syntax 3** Wenn Sie einen TIMESTAMP-Wert und einen INTEGER-Wert an die MINUTES-Funktion übergeben, liefert die Funktion das TIMESTAMP-Ergebnis nach dem Addieren der durch die Ganzzahl ausgedrückten Anzahl an Minuten zum *Zeitstempelausdruck*-Argument. Entsprechend ist, wenn das erste Argument für MINUTES ein TIME-Wert ist, das Ergebnis ebenfalls ein TIME-Wert. Syntax 3 unterstützt keine implizite Konvertierung des ersten Arguments. Es kann erforderlich sein, das erste Argument explizit in einen DATE-, TIME- oder TIMESTAMP-Wert zu konvertieren. Wenn das erste Argument ein DATE-Wert ist, wird Mitternacht für die Zeitangabe angenommen.

Da die MINUTES-Funktion eine Ganzzahl zurückgibt, kann es zu einem Überlauf kommen, wenn Syntax 1 mit TIMESTAMP-Werten ab 4083-03-23 02:08:00 benutzt wird.

Verwenden Sie statt Syntax 2 die DATEDIFF-Funktion. Verwenden Sie statt Syntax 3 die DATEADD-Funktion.

Siehe auch

- „DATEDIFF-Funktion [Datum und Uhrzeit]“ auf Seite 218
- „DATEADD-Funktion [Datum und Uhrzeit]“ auf Seite 217
- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgenden Anweisungen geben den Wert "240" zurück, womit angezeigt wird, dass der zweite TIMESTAMP-Wert 240 Minuten nach dem ersten liegt. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEDIFF).

```
SELECT MINUTES( '1999-07-13 06:07:12',
                '1999-07-13 10:07:12' );

SELECT DATEDIFF( minute,
                '1999-07-13 06:07:12',
                '1999-07-13 10:07:12' );
```

Die folgende Anweisung gibt den Wert 1051040527 zurück:

```
SELECT MINUTES( '1998-07-13 06:07:12' );
```

Die folgenden Anweisungen geben den TIMESTAMP-Wert 1999-05-12 21:10:07.000 zurück. Die erste Anweisung erfordert eine explizite Umwandlung des literalen Zeichenfolgenparameters. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEADD).

```
SELECT MINUTES( CAST( '1999-05-12 21:05:07' AS TIMESTAMP ), 5 );

SELECT DATEADD( minute, 5, '1999-05-12 21:05:07' );
```

Die folgende Anweisung gibt 'TIME' zurück und zeigt, dass die MINUTES-Funktion einen TIME-Wert zurückgibt, wenn sie mit einem TIME-Argument aufgerufen wird:

```
SELECT EXPRTYPE('SELECT MINUTES( CAST( '13:45:00.000' AS TIME ), 16 )', 1);
```

MOD-Funktion [Numerisch]

Gibt den Rest zurück, wenn eine Ganzzahl durch eine andere dividiert wird.

Syntax

MOD(*dividend*, *divisor*)

Parameter

- **dividend** Der Dividend oder Zähler der Division.
- **divisor** Der Divisor oder Nenner der Division

Rückgabe

- SMALLINT
- INT
- NUMERIC

Bemerkungen

Eine Division mit einem negativen Dividenten ergibt eine negative Zahl oder eine Null. Das Vorzeichen des Divisors hat keine Auswirkung.

Siehe auch

- [„REMAINDER-Funktion \[Numerisch\]“](#) auf Seite 364

Standards und Kompatibilität

- **SQL/2008** Die MOD-Funktion ist Teil der optionalen SQL/2008-Sprachenfunktion T441.

Beispiel

Die folgende Anweisung gibt den Wert 2 zurück:

```
SELECT MOD( 5, 3 );
```

MONTH-Funktion [Datum und Uhrzeit]

Gibt den Monat des angegebenen Datums zurück.

Syntax

MONTH(*date-expression*)

Parameter

- **Datumsausdruck** Ein Wert vom Typ DATE.

Rückgabe

SMALLINT

Bemerkungen

Der zurückgegebene Wert ist eine Zahl zwischen 1 und 12, entsprechend dem Monat des angegebenen Datums.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 7 zurück:

```
SELECT MONTH( '1998-07-13' );
```

MONTHNAME-Funktion [Datum und Uhrzeit]

Gibt den Monatsnamen eines Datums zurück.

Syntax

MONTHNAME(*date-expression*)

Parameter

- **Zeitstempelausdruck** Ein TIMESTAMP-Wert.

Rückgabe

VARCHAR

Bemerkungen

Die MONTHNAME-Funktion gibt eine Zeichenfolge zurück, auch wenn das Ergebnis numerisch ist, wie z.B. "2" für den Monat Februar.

Siehe auch

- [„DATEPART-Funktion \[Datum und Uhrzeit\]“ auf Seite 221](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert September zurück:

```
SELECT MONTHNAME( '1998-09-05' );
```

MONTHS-Funktion [Datum und Uhrzeit]

Bearbeitet einen TIMESTAMP-Wert oder gibt die Anzahl von Monaten zwischen zwei TIMESTAMP-Werten zurück. Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Syntax 1

MONTHS(*timestamp-expression*)

Syntax 2

MONTHS(*timestamp-expression*, *timestamp-expression*)

Syntax 3

MONTHS(*timestamp-expression*, *integer-expression*)

Parameter

- **Zeitstempelausdruck** Ein Datums- und Uhrzeitwert vom Typ TIMESTAMP.
- **Ganzzahlausdruck** Die als Ganzzahl ausgedrückte Anzahl der Monate (vom Typ SMALLINT), die zu *Zeitstempelausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl an Monaten von *Zeitstempelausdruck* abgezogen. Wenn Sie *Ganzzahlausdruck* angeben, muss *Zeitstempelausdruck* explizit als TIME-, DATE- oder TIMESTAMP-Wert festgelegt sein. Wenn *Zeitstempelausdruck* ein TIME-Wert ist, wird der laufende Monat angenommen.

Hinweise zum Casting von Datentypen finden Sie unter [„CAST-Funktion \[Datentypkonvertierung\]“ auf Seite 186](#).

Rückgabe

INTEGER bei Syntax 1 und Syntax 2.

TIMESTAMP bei Syntax 3.

Bemerkungen

Das Ergebnis der MONTHS-Funktion hängt von deren Argumenten ab. Die MONTHS-Funktion ignoriert Stunden, Minuten, und Sekunden in ihren Argumenten.

- **Syntax 1** Wenn Sie einen einzelnen *Zeitstempelausdruck* an die MONTHS-Funktion übergeben, wird die Anzahl der Monate zwischen 0000-02 und *Zeitstempelausdruck* als INTEGER-Wert zurückgegeben.

Hinweis

"0000-02" gibt kein tatsächliches Datum wieder. Es ist das von der MONTHS-Funktion verwendete Standarddatum.

- **Syntax 2** Wenn Sie zwei TIMESTAMP-Werte an die MONTHS-Funktion übergeben, liefert die Funktion die als Ganzzahl ausgedrückte Anzahl der Monate zwischen den beiden.
- **Syntax 3** Wenn Sie einen TIMESTAMP-Wert und einen INTEGER-Wert an die MONTHS-Funktion übergeben, liefert die Funktion das TIMESTAMP-Ergebnis nach dem Addieren der durch die Ganzzahl ausgedrückten Anzahl an Monaten zu *Zeitstempelausdruck*.

Verwenden Sie statt Syntax 2 die DATEDIFF-Funktion. Verwenden Sie statt Syntax 3 die DATEADD-Funktion.

Der Wert von MONTHS wird anhand der Anzahl von Monatsersten zwischen zwei Datumsangaben berechnet.

Siehe auch

- „DATEDIFF-Funktion [Datum und Uhrzeit]“ auf Seite 218
- „DATEADD-Funktion [Datum und Uhrzeit]“ auf Seite 217

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 2 zurück, womit angezeigt wird, dass das zweite Datum zwei Monate nach dem ersten liegt. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEDIFF).

```
SELECT MONTHS( '1999-07-13 06:07:12', '1999-09-13 10:07:12' );

SELECT DATEDIFF( month,
  '1999-07-13 06:07:12',
  '1999-09-13 10:07:12' );
```

Die folgende Anweisung gibt den Wert 23981 zurück:

```
SELECT MONTHS( '1998-07-13 06:07:12' );
```

Die folgenden Anweisungen geben den TIMESTAMP-Wert 1999-10-12 21:05:07.000 zurück. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEADD):

```
SELECT MONTHS( CAST( '1999-05-12 21:05:07' AS DATETIME ), 5 );

SELECT DATEADD( month, 5, '1999-05-12 21:05:07' );
```

NCHAR-Funktion [Zeichenfolge]

Gibt eine NCHAR-Zeichenfolge zurück, die ein Zeichen enthält, dessen Unicode-Codepunkt im Parameter angegeben ist, oder NULL, wenn der Wert kein gültiger Codepunkt-Wert ist.

Syntax

NCHAR(*integer*)

Parameter

- **integer** Die Zahl, die in den entsprechenden Unicode-Codepunkt konvertiert werden soll.

Rückgabe

NVARCHAR

Siehe auch

- „CONNECTION_EXTENDED_PROPERTY-Funktion [Zeichenfolge]“ auf Seite 197
- „TO_NCHAR-Funktion [Zeichenfolge]“ auf Seite 410
- „TO_CHAR-Funktion [Zeichenfolge]“ auf Seite 409
- „UNICODE-Funktion [Zeichenfolge]“ auf Seite 420
- „UNISTR-Funktion [Zeichenfolge]“ auf Seite 421

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel gibt den arabischen Buchstaben ALEF zurück, der Unicode-Codepunkt U+627 ist:

```
SELECT NCHAR( 1575 );
```

NEWID-Funktion [Verschiedene]

Generiert einen UUID-Wert (universell eindeutiger Bezeichner). Ein UUID-Wert entspricht dem GUID-Wert (global eindeutiger Bezeichner).

Syntax

NEWID()

Parameter

Der NEWID-Funktion sind keine Parameter zugeordnet.

Rückgabe

UNIQUEIDENTIFIER

Bemerkungen

Die NEWID-Funktion kann in einer DEFAULT-Klausel für eine Spalte angewendet werden.

UUIDs können verwendet werden, um Zeilen in einer Tabelle eindeutig zu identifizieren. Ein auf einem Computer erzeugter Wert stimmt mit dem Wert, der auf einem anderen Computer erzeugt wurde, nicht überein, und kann daher als Schlüssel in Synchronisations- und Replikationsumgebungen verwendet werden.

UUID-Werte enthalten Bindestriche, um mit anderen RDBMS kompatibel zu sein. Sie können dies ändern, indem Sie die Option `uuid_has_hyphens` auf 'OFF' einstellen. Weitere Hinweise finden Sie unter „`uuid_has_hyphens`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)].

Die NEWID-Funktion ist eine nicht-deterministische Funktion. Aufeinanderfolgende Aufrufe geben unterschiedliche Werte zurück. Der Abfrageoptimierer behält die Ergebnisse der Funktion NEWID nicht im Cache.

Weitere Hinweise zu nicht-deterministischen Funktionen finden Sie unter [Caching von Funktionen](#) [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Siehe auch

- „Der Standardwert NEWID“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „STRTOUUID-Funktion [Zeichenfolge]“ auf Seite 399
- „UUIDTOSTR-Funktion [Zeichenfolge]“ auf Seite 424
- „UNIQUEIDENTIFIER-Datentyp“ auf Seite 134

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung erstellt eine Tabelle namens "mytab" mit zwei Spalten. Die Spalte pk hat einen eindeutig bezeichnenden Datentyp und ordnet die Funktion NEWID als den Standardwert zu. Die Spalte c1 hat einen Ganzzahl-Datentyp.

```
CREATE TABLE mytab(  
    pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),  
    c1 INT );
```

Die folgende Anweisung gibt einen eindeutigen Bezeichner als Zeichenfolge zurück:

```
SELECT UUIDTOSTR( NEWID() );
```

Beispiel: Der zurückgegebene Wert könnte "96603324-6FF6-49DE-BF7D-F44C1C7E6856" sein.

NEXT_CONNECTION-Funktion [System]

Gibt eine Kennnummer für die nächste Verbindung zurück.

Syntax

NEXT_CONNECTION(*connection-id* [, *database-id*])

Rückgabe

INT

Parameter

- **Verbindungs-ID** Eine Ganzzahl, gewöhnlich von einem früheren Aufruf von NEXT_CONNECTION zurückgegeben. Wenn *connection-id* NULL ist, gibt NEXT_CONNECTION die neueste Verbindungs-ID zurück.
- **Datenbank-ID** Eine Ganzzahl, die eine der Datenbanken auf dem aktuellen Server darstellt. Wenn Sie keine *database-id* angeben, wird die aktuelle Datenbank verwendet. Wenn Sie NULL angeben, gibt NEXT_CONNECTION die nächste Verbindung zurück, unabhängig von der Datenbank.

Bemerkungen

NEXT_CONNECTION wird verwendet, um die Verbindungen zu einer Datenbank aufzuzählen. Verbindungs-IDs werden üblicherweise in gleichmäßig ansteigender Reihenfolge erstellt. Diese Funktion gibt die nächste Verbindungs-ID in umgekehrter Reihenfolge zurück.

Um die Verbindungs-ID der letzten Verbindung zu erhalten, geben Sie NULL als *connection-id* an. Um die nächste Verbindung zu erhalten, geben Sie den vorherigen Rückgabewert ein. Die Funktion gibt NULL zurück, wenn es keine weiteren Verbindungen in der Reihe gibt.

NEXT_CONNECTION ist nützlich, wenn Sie alle Verbindungen trennen möchten, die vor einem bestimmten Zeitpunkt hergestellt wurden. Da NEXT_CONNECTION die Verbindungs-IDs in umgekehrter Reihenfolge zurückgibt, werden jedoch keine Verbindungen zurückgegeben, die nach dem Starten der Funktion hergestellt werden. Wenn Sie gewährleisten möchten, dass alle Verbindungen getrennt werden, verhindern Sie zuerst neue Verbindungen, bevor Sie NEXT_CONNECTION ausführen.

🔥 **Cloud-Hinweis:** Aufgrund von Isolationsregeln für Tenant-Datenbanken gibt diese Funktion bei einem Aufruf in der Cloud nur Informationen über die aktuelle Tenant-Datenbank zurück.

Privilegien

Keine Privilegien sind erforderlich, um mithilfe dieser Funktion die aktuelle Verbindungs-ID zurückzugeben. Um diese Funktion für andere Verbindungen ausführen zu können, benötigen Sie das SERVER OPERATOR-Systemprivileg, das MONITOR-Systemprivileg oder das DROP CONNECTION-Systemprivileg.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt einen Bezeichner (als Ganzzahlwert) für die nächste Verbindung an der aktuellen Datenbank zurück:

```
SELECT NEXT_CONNECTION( NULL );
```

Die folgende Anweisung gibt einen Ganzzahlwert wie 5 zurück.

```
SELECT NEXT_CONNECTION( 10 );
```

Der folgende Aufruf gibt die nächste Verbindungs-ID in umgekehrter Reihenfolge zur angegebenen *Verbindungs-ID* an der aktuellen Datenbank zurück:

```
SELECT NEXT_CONNECTION( connection-id );
```

Der folgende Aufruf gibt die nächste Verbindungs-ID in umgekehrter Reihenfolge zur angegebenen *Verbindungs-ID* zurück, unabhängig von der Datenbank:

```
SELECT NEXT_CONNECTION( connection-id, NULL );
```

Der folgende Aufruf gibt die nächste Verbindungs-ID in umgekehrter Reihenfolge zur angegebenen *Verbindungs-ID* an der angegebenen Datenbank zurück:

```
SELECT NEXT_CONNECTION( connection-id, database-id );
```

Der folgende Aufruf gibt die erste oder früheste Verbindung zurück, unabhängig von der Datenbank:

```
SELECT NEXT_CONNECTION( NULL, NULL );
```

Der folgende Aufruf gibt die erste oder früheste Verbindung an der angegebenen Datenbank zurück:

```
SELECT NEXT_CONNECTION( NULL, database-id );
```

NEXT_DATABASE-Funktion [System]

Gibt eine Kennnummer einer Datenbank zurück.

Syntax

```
NEXT_DATABASE( database-id )
```

Parameter

- **Datenbank-ID** Eine Ganzzahl, die die ID-Nummer einer Datenbank angibt

Rückgabe

INT

Bemerkungen

Die NEXT_DATABASE-Funktion wird verwendet, um die Datenbanken aufzuzählen, die auf einem Datenbankserver ausgeführt werden. Wenn Sie die erste Datenbank abrufen möchten, geben Sie NULL an. Wenn Sie die jeweils nachfolgende Datenbank abrufen möchten, geben Sie den vorherigen Rückgabewert an. Die Funktion gibt NULL zurück, wenn es keine weiteren Datenbanken gibt. Die Datenbank-ID-Nummern werden nicht in einer bestimmten Reihenfolge zurückgegeben, aber Sie können ihnen die Reihenfolge entnehmen, in der die Datenbanken mithilfe der Datenbank-ID auf dem Server gestartet wurden. Die erste Datenbank, die auf dem Server gestartet wurde, erhält den Wert 0, und bei den Datenbanken, die danach auf dem Server gestartet werden, werden die Datenbank-IDs jeweils um 1 erhöht.

🔥 **Cloud-Hinweis:** Aufgrund von Isolationsregeln für Tenant-Datenbanken gibt diese Funktion bei einem Aufruf in der Cloud nur Informationen über die aktuelle Tenant-Datenbank zurück.

Privilegien

Keine Privilegien sind erforderlich, um mithilfe dieser Funktion die aktuelle Datenbank zurückzugeben. Um diese Funktion für andere Datenbanken ausführen zu können, benötigen Sie das SERVER OPERATOR-Systemprivileg oder das MONITOR-Systemprivileg.

Siehe auch

- „DB_NAME-Funktion [System]“ auf Seite 231
- „sa_db_list-Systemprozedur“ auf Seite 1197

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 0 zurück, den ersten Datenbankwert:

```
SELECT NEXT_DATABASE( NULL );
```

Die folgende Anweisung gibt nur dann NULL zurück, wenn eine Datenbank gestartet wurde:

```
SELECT NEXT_DATABASE( 0 );
```

NEXT_HTTP_HEADER-Funktion [Webdienst]

Ruft den Namen des nächsten HTTP-Headers ab.

Syntax

```
NEXT_HTTP_HEADER( header-name )
```

Parameter

- **Headername** Der Name des vorherigen Anforderungsh-Headers. Wenn der Headername NULL ist, gibt diese Funktion den Namen des ersten HTTP-Anforderungsh-Headers zurück.

Rückgabe

LONG VARCHAR.

Hinweis

Der Datentyp des Ergebnisses ist LONG VARCHAR. Wenn Sie NEXT_HTTP_HEADER in einer SELECT INTO-Anweisung verwenden, müssen Sie über eine Unstructured Data Analytics Option-Lizenz verfügen oder CAST verwenden und NEXT_HTTP_HEADER auf den richtigen Datentyp und die richtige Größe setzen.

Bemerkungen

Diese Funktion wird verwendet, um wiederholt die HTTP-Anforderungsheader zu durchlaufen und den Namen des nächsten HTTP-Headers zurückzugeben. Sie mit NULL aufzurufen führt dazu, dass sie den Namen des ersten Headers zurückgibt. Nachfolgende Header werden abgerufen, indem der Funktion der Name des vorigen Headers weitergegeben wird. Diese Funktion gibt NULL zurück, wenn sie mit dem Namen des letzten Headers oder nicht von einem Webdienst aufgerufen wird.

Wenn diese Funktion mehrfach aufgerufen wird, werden alle Headerfelder genau einmal zurückgegeben, jedoch nicht unbedingt in der Reihenfolge, in der sie in der HTTP-Anforderung erscheinen.

Siehe auch

- „HTTP_HEADER-Funktion [Webdienst]“ auf Seite 283
- „sa_http_header_info-Systemprozedur“ auf Seite 1235
- „Zugriff auf vom Client bereitgestellte HTTP-Variablen und -Header“ [*SQL Anywhere Server - Programmierung*]
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung zeigt den Namen und die Werte der HTTP-Anforderungsheader im Meldungsfenster des Datenbankservers an, wenn sie innerhalb einer von einem HTTP-Webdienst aufgerufenen gespeicherten Prozedur verwendet wird:

```
BEGIN
  declare header_name long varchar;
  declare header_value long varchar;
  set header_name = NULL;
header_loop:
  LOOP
    SET header_name = NEXT_HTTP_HEADER( header_name );
    IF header_name IS NULL THEN
      LEAVE header_loop
    END IF;
    SET header_value = HTTP_HEADER( header_name );
    MESSAGE 'HEADER: ', header_name, '=',
           header_value TO CONSOLE;
  END LOOP;
END;
```

NEXT_HTTP_RESPONSE_HEADER-Funktion [Webdienst]

Ruft den Namen des nächsten HTTP-Antwortheaders ab.

Syntax

NEXT_HTTP_RESPONSE_HEADER(*header-name*)

Parameter

- **Headername** Der Name des vorherigen Antwortheaders. Wenn der Headername NULL ist, gibt diese Funktion den Namen des ersten HTTP-Antwortheaders zurück.

Rückgabe

LONG VARCHAR

Bemerkungen

Diese Funktion wird verwendet, um wiederholt die HTTP-Antwortheader zu durchlaufen und den Namen des nächsten HTTP-Antwortheaders zurückzugeben. Sie mit NULL aufzurufen führt dazu, dass sie den Namen des ersten Antwortheaders zurückgibt. Nachfolgende Antwortheader werden abgerufen, indem der Funktion der Name des vorigen Antwortheaders übergeben wird. Diese Funktion gibt NULL zurück, wenn sie mit dem Namen des letzten Antwortheaders oder nicht von einem Webdienst aufgerufen wird.

Wenn diese Funktion mehrfach aufgerufen wird, werden alle Antwortheaderfelder genau einmal zurückgegeben, jedoch nicht unbedingt in der Reihenfolge, in der sie in der HTTP-Antwort erscheinen.

Siehe auch

- „HTTP_RESPONSE_HEADER-Funktion [Webdienst]“ auf Seite 285
- „HTTP-Anforderungsheader verwalten“ [*SQL Anywhere Server - Programmierung*]
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung zeigt den Namen und die Werte der HTTP-Antwortheader im Meldungsfenster des Datenbankservers an, wenn sie innerhalb einer von einem HTTP-Webdienst aufgerufenen gespeicherten Prozedur verwendet wird:

```
BEGIN
  declare header_name long varchar;
  declare header_value long varchar;
  set header_name = NULL;
header_loop:
  LOOP
    SET header_name = NEXT_HTTP_RESPONSE_HEADER( header_name );
    IF header_name IS NULL THEN
      LEAVE header_loop
    END IF;
    SET header_value = HTTP_RESPONSE_HEADER( header_name );
    MESSAGE 'RESPONSE HEADER: ', header_name, '=', header_value TO CONSOLE;
  END LOOP;
```

NEXT_HTTP_VARIABLE-Funktion [Webdienst]

Gibt den Namen der nächsten HTTP-Variablen zurück.

Syntax

NEXT_HTTP_VARIABLE(*var-name*)

Parameter

- **Variablenname** Der Name der vorigen Variablen. Wenn *Variablenname* NULL ist, gibt diese Funktion den Namen der ersten HTTP-Variablen zurück.

Rückgabe

LONG VARCHAR.

Bemerkungen

Diese Funktion wird über die HTTP-Variablen wiederholt, die in eine Anforderung einbezogen sind. Sie mit NULL aufzurufen führt dazu, dass sie den Namen der ersten Variablen zurückgibt. Nachfolgende Variable werden abgerufen, indem der Funktion der Name der vorigen Variablen weitergegeben wird. Diese Funktion gibt NULL zurück, wenn sie mit dem Namen der abschließenden Variablen oder nicht von einem Webdienst aufgerufen wird.

Wenn diese Funktion mehrfach aufgerufen wird, werden alle Variablen genau einmal zurückgegeben, jedoch nicht unbedingt in der Reihenfolge, in der sie in der HTTP-Anfrage erscheinen. Die Variablen url oder url1, url2, ... , url10 werden einbezogen, wenn URL PATH auf ON bzw. ELEMENTS gesetzt wurde.

Siehe auch

- „HTTP_VARIABLE-Funktion [Webdienst]“ auf Seite 287
- „NEXT_HTTP_HEADER-Funktion [Webdienst]“ auf Seite 325
- „sa_http_variable_info-Systemprozedur“ auf Seite 1239
- „Zugriff auf vom Client bereitgestellte HTTP-Variablen und -Header“ [*SQL Anywhere Server - Programmierung*]
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Namen der ersten HTTP-Variablen zurück, wenn sie innerhalb einer von einem HTTP-Webdienst aufgerufenen gespeicherten Prozedur verwendet wird:

```
BEGIN
DECLARE variable_name LONG VARCHAR;
DECLARE variable_value LONG VARCHAR;
SET variable_name = NULL;
SET variable_name = NEXT_HTTP_VARIABLE( variable_name );
SET variable_value = HTTP_VARIABLE( variable_name );
END;
```

NEXT_SOAP_HEADER-Funktion [SOAP]

Gibt den nächsten Headerschlüssel in einem Header der SOAP-Anforderung zurück.

Syntax

NEXT_SOAP_HEADER(*header-key*)

Parameter

- **Headerschlüssel** Der lokale XML-Name des obersten XML-Elements für den angegebenen Headereintrag

Rückgabe

LONG VARCHAR

Bemerkungen

Wenn Sie NULL für den *Headerschlüssel* angeben, gibt die Funktion den Headerschlüssel für den ersten Headereintrag zurück, der im SOAP-Header gefunden wird.

Diese Funktion gibt NULL zurück, wenn Sie mit dem letzten *Headerschlüssel* aufgerufen wird.

Siehe auch

- „SOAP_HEADER-Funktion [SOAP]“ auf Seite 384
- „Praktische Einführung: Verwenden von SQL Anywhere für den Zugriff auf einen SOAP/DISH-Dienst“ [*SQL Anywhere Server - Programmierung*]
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung verarbeitet alle Schlüssel im SOAP-Anforderungsheader, wenn sie innerhalb einer von einem HTTP-Webdienst aufgerufenen gespeicherten Prozedur verwendet wird. Wenn die Funktion den **Authentication**-Schlüssel verarbeitet, ruft sie auch den Wert des Schlüssels ab.

```
BEGIN
  DECLARE hd_key LONG VARCHAR;
  DECLARE hd_entry LONG VARCHAR;
header_loop:
  LOOP
    SET hd_key = NEXT_SOAP_HEADER( hd_key );
    IF hd_key IS NULL THEN
      -- no more header entries
      LEAVE header_loop;
    END IF;
    IF hd_key = 'Authentication' THEN
      SET hd_entry = SOAP_HEADER( hd_key );
    END IF;
  END LOOP header_loop;
END;
```

NOW-Funktion [Datum und Uhrzeit]

Gibt das aktuelle Datum und die aktuelle Zeit als TIMESTAMP Wert zurück. Die Genauigkeit ist durch die Genauigkeit der Systemuhr begrenzt.

Syntax

NOW([*])

Rückgabe

TIMESTAMP

Bemerkungen

NOW ist gleichwertig mit der GETDATE-Funktion und dem CURRENT TIMESTAMP-Spezialwert. NOW(*) und NOW() sind gleichwertige Konstruktionen.

Jede Instanz der NOW-Funktion in einer Anforderung wird höchstens einmal ausgewertet. Mehrere Instanzen von NOW in derselben Anforderung liefern möglicherweise identische oder auch unterschiedliche TIMESTAMP-Werte.

Siehe auch

- „CURRENT TIME-Spezialwert“ auf Seite 72
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 73
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 75
- „DATE-Datentyp“ auf Seite 123
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 216
- „DATETIME-Datentyp“ auf Seite 124
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 222
- „DATETIMEOFFSET-Datentyp“ auf Seite 125
- „Ausdrücke“ auf Seite 22
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 268
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 293
- „SMALLDATETIME-Datentyp“ auf Seite 127
- „TIME-Datentyp“ auf Seite 128
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „TIMESTAMP-Datentyp“ auf Seite 129
- „UTC TIMESTAMP-Spezialwert“ auf Seite 84

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt das aktuelle Datum und die aktuelle Uhrzeit zurück:

```
SELECT NOW( * );
```

NULLIF-Funktion [Verschiedene]

Liefert einen verkürzten CASE-Ausdruck, indem Ausdrücke verglichen werden.

Syntax

NULLIF(*expression-1*, *expression-2*)

Parameter

- ***expression-1*** Ein zu vergleichender Ausdruck.
- ***expression-2*** Ein zu vergleichender Ausdruck

Rückgabe

Datentyp des ersten Arguments.

Bemerkungen

NULLIF vergleicht die Werte der zwei Ausdrücke.

Wenn der erste Ausdruck gleich dem zweiten Ausdruck ist, gibt NULLIF den Wert NULL zurück.

Wenn der erste Ausdruck nicht gleich dem zweiten Ausdruck oder der zweite Ausdruck NULL ist, gibt NULLIF den ersten Ausdruck zurück.

Die NULLIF-Funktion bietet eine verkürzte Methode zum Schreiben bestimmter CASE-Ausdrücke.

Siehe auch

- [„CASE-Ausdrücke“ auf Seite 25](#)

Standards und Kompatibilität

- **SQL/2008** Kernfunktion.

Beispiel

Die folgende Anweisung gibt den Wert a zurück:

```
SELECT NULLIF( 'a', 'b' );
```

Die folgende Anweisung gibt NULL zurück:

```
SELECT NULLIF( 'a', 'a' );
```

NUMBER-Funktion [Verschiedene]

Erzeugt Zahlen, beginnend mit 1 für jede folgende Zeile in den Ergebnissen der Abfrage. Die NUMBER-Funktion ist hauptsächlich für den Einsatz in Auswahllisten (SELECT-Listen) vorgesehen.

Aufgrund der Einschränkungen, die von der NUMBER-Funktion vorgegeben werden (wie im untenstehenden Abschnitt 'Anmerkungen' beschrieben), sollten Sie stattdessen die ROW_NUMBER-Funktion verwenden. Die ROW_NUMBER-Funktion bietet dieselben Funktionalitäten wie die

NUMBER-Funktion, aber ohne ihre Einschränkungen. Siehe: „[ROW_NUMBER-Funktion \[Verschiedene\]](#)“ auf Seite 373.

Syntax

NUMBER([*])

Rückgabe

INT

Bemerkungen

Sie können die Funktion NUMBER(*) in einer Auswahlliste verwenden, um eine aufsteigende Nummerierung der Zeilen in der Ergebnismenge zu bewirken. NUMBER(*) gibt den Wert der ANSI-Zeilenummer jeder Ergebniszeile zurück. Die NUMBER-Funktion kann positive oder negative Werte zurückgeben, je nachdem, wie die Anwendung durch die Ergebnismenge blättert. Für unempfindliche Cursor ist der Wert von NUMBER(*) immer positiv, weil die gesamte Ergebnismenge bei OPEN materialisiert wird.

Außerdem kann sich die Zeilennummer bei bestimmten Cursortypen ändern. Der Wert ist für unempfindliche Cursor und Scroll-Cursor fix. Wenn gleichzeitige Aktualisierungen vorkommen, kann sich der Wert für dynamische und empfindliche Cursor ändern.

Ein Syntaxfehler wird generiert, wenn Sie die NUMBER-Funktion in folgenden Zusammenhängen verwenden: in einer DELETE-Anweisung, einer WHERE-Klausel, einer HAVING-Klausel, einer ORDER BY-Klausel, einer Unterabfrage, einer Abfrage mit Aggregaten, einer Integritätsregel, einer GROUP BY-Klausel, einer DISTINCT-Klausel, einem Mengenoperator (UNION, EXCEPT, INTERSECT) oder einer abgeleiteten Tabelle.

NUMBER(*) kann (unter Berücksichtigung der oben genannten Einschränkungen) in einer Ansicht verwendet werden, die Ansichtsspalte, die dem Ausdruck entspricht, in dem NUMBER(*) enthalten ist, kann aber in einer Abfrage oder äußeren Ansicht nur einmal referenziert werden, und die Ansicht kann nicht als Nullwert-liefernde Tabelle in einem linken Outer-Join oder einem vollen Outer-Join benutzt werden.

In Embedded SQL sollte ein Cursor mit Vorsicht behandelt werden, der sich auf eine Abfrage bezieht, die eine NUMBER(*)-Funktion enthält. Diese Funktion gibt insbesondere dann negative Zahlen zurück, wenn ein Datenbankcursor mithilfe von "relativ" zum Ende des Cursors positioniert wird (eine absolute Position mit einem negativen Ausgangspunkt).

Sie können NUMBER im rechten Teil einer Zuordnung in der SET-Klausel einer UPDATE-Anweisung verwenden. z. B. SET x = NUMBER(*).

Die NUMBER-Funktion kann auch verwendet werden, um Primärschlüssel zu generieren, wenn INSERT kombiniert mit einer SELECT-Anweisung verwendet wird, auch wenn normalerweise eine AUTOINCREMENT-Klausel benutzt wird, um sequenzielle Primärschlüssel zu erzeugen.

Hinweise zur AUTOINCREMENT-Klausel finden Sie unter „[CREATE TABLE-Anweisung](#)“ auf Seite 737.

NUMBER(*) und NUMBER() sind semantisch gleichwertig.

Siehe auch

- „INSERT-Anweisung“ auf Seite 917

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt eine fortlaufend nummerierte Liste der Abteilungen zurück:

```
SELECT NUMBER( * ), DepartmentName
FROM GROUP0.Departments
WHERE DepartmentID > 5
ORDER BY DepartmentName;
```

PATINDEX-Funktion [Zeichenfolge]

Gibt eine Ganzzahl zurück, die die Startposition des ersten Auftretens eines Musters in einer Zeichenfolge darstellt.

Syntax

PATINDEX('%pattern%', string-expression)

Parameter

- **pattern** Das Muster, nach dem gesucht wird. Wenn der führende Prozent-Platzhalter weggelassen wird, gibt die PATINDEX-Funktion "1" zurück, falls das Muster am Anfang der Zeichenfolge auftritt, und "0", falls nicht.

Das Muster verwendet dieselben Platzhalter wie der LIKE-Vergleich. Das sind die folgenden:

Platzhalterzeichen	Gefunden
_ (Unterstrich)	Ein Zeichen
% (Prozent)	Eine Zeichenfolge mit null oder mehr Zeichen
[]	Ein einzelnes Zeichen im angegebenen Bereich oder Menge
[^]	Ein einzelnes Zeichen außerhalb des angegebenen Bereichs oder Menge

- **Zeichenfolgenderausdruck** Die Zeichenfolge, die nach dem Muster durchsucht werden soll

Rückgabe

INT

Bemerkungen

Die PATINDEX-Funktion gibt die Startposition des ersten Auftretens des Musters zurück. Wenn das Muster nicht gefunden wird, wird "0" zurückgegeben.

Siehe auch

- „LIKE-Suchbedingung“ auf Seite 51
- „LOCATE-Funktion [Zeichenfolge]“ auf Seite 305
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 2 zurück:

```
SELECT PATINDEX( '%hoco%', 'chocolate' );
```

Die folgende Anweisung gibt den Wert 11 zurück:

```
SELECT PATINDEX( '%4_5_', '0a1A 2a3A 4a5A' );
```

Die folgende Anweisung gibt 14 zurück, also das erste, nicht alphanumerische Zeichen im Zeichenfolgenausdruck. Das Muster '%[^a-z0-9]%' kann anstelle von '%[^a-zA-Z0-9]%' verwendet werden, wenn die Datenbank die Groß- und Kleinschreibung nicht berücksichtigt.

```
SELECT PATINDEX( '%[^a-zA-Z0-9]%', 'SQLAnywhere16 has many new features' );
```

Die folgende Anweisung kann verwendet werden, um alle Zeichen bis einschließlich zum ersten nicht alphanumerischen Zeichen in einer Zeichenfolge abzurufen.

```
SELECT LEFT( @string, PATINDEX( '%[^a-zA-Z0-9]%', @string ) );
```

Die folgenden Anweisungen erstellen die Tabelle myTable und füllen sie mit verschiedenen Zeichenfolgen mit alphanumerischen Zeichen, Leerzeichen und nicht-alphanumerischen Zeichen. Danach zeigen die SELECT-Anweisung und die folgenden Ergebnisse, wie Sie PATINDEX verwenden können, um die Startposition von Leerstellen und nicht-alphanumerischen Zeichen in den Zeichenfolgen zu finden:

```
CREATE TABLE myTable( coll LONG VARCHAR );

INSERT INTO myTable (coll) VALUES( 'the quick brown fox jumped over the lazy dog' ),
( 'the quick brown fox $$$$ jumped over the lazy dog' ),
( 'the quick brown fox 0999 jumped over the lazy dog' ),
( 'the quick brown fox ** jumped over the lazy dog' ),
( 'thequickbrownfoxjumpedoverthelazydog' ),
( 'thequickbrownfoxjum999pedoverthelazydog' ),
( 'thequick$$$$brownfox' ),
( 'the quick brown fox$$ jumped over the lazy dog' );

SELECT coll,
  //position of first non-alphanumeric character or space:
  PATINDEX( '%[^a-z0-9]%', coll) AS blank_posn,
  //position of first non-alphanumeric char that isn't a space:
  PATINDEX( '%[^ a-z0-9]%', coll) AS non_alpha_char,
  //everything up to and including first non-alphanumeric char that isn't a space:
  LEFT ( coll, PATINDEX( '%[^ a-zA-Z0-9]%', coll) ) AS left_str,
  //first non-alphanumeric char that isn't a space, and everything to the
```

```

right:
  SUBSTRING ( col1, PATINDEX( '%[^ a-zA-Z0-9]%', col1) ) AS sub_str
FROM myTable;

```

col1	blank_posn	non_alpha_char	left_str	sub_str
the quick brown fox jumped over the lazy dog	4	0		the quick brown fox jumped over the lazy dog
the quick brown fox \$\$\$\$ jumped over the lazy dog	4	21	the quick brown fox \$	\$\$\$\$ jumped over the lazy dog
the quick brown fox 0999 jumped over the lazy dog	4	0		the quick brown fox 0999 jumped over the lazy dog
the quick brown fox ** jumped over the lazy dog	4	21	the quick brown fox *	** jumped over the lazy dog
thequickbrownfoxjumpedoverthelazydog	0	0		thequickbrownfoxjumpedoverthelazydog
thequickbrownfox-jum999pedoverthelazydog	0	0		thequickbrownfox-jum999pedoverthelazydog
thequick\$\$\$\$brownfox	9	9	thequick\$	\$\$\$\$brownfox
the quick brown fox\$\$ jumped over the lazy dog	4	20	the quick brown fox \$	\$\$ jumped over the lazy dog

PERCENT_RANK-Funktion [Rangfolge]

Bei jeder Zeile X, die durch die Argumente und ORDER BY-Spezifizierungen der Funktion festgelegt ist, bestimmt die PERCENT_RANK-Funktion den Rang der Zeile X - 1, dividiert durch die Anzahl der Zeilen in der Gruppe.

Syntax

PERCENT_RANK() OVER (window-spec)

Fensterspezifikation: Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Rückgabe

Die PERCENT_RANK-Funktion gibt einen DOUBLE-Wert zwischen 0 und 1 zurück.

Bemerkungen

Elemente von *Fensterspezifikation* können entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Bei Verwendung als Fensterfunktion müssen Sie eine ORDER BY-Klausel angeben und dürfen eine PARTITION BY-Klausel angeben, können jedoch keine ROWS- oder RANGE-Klausel angeben. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „Fensterfunktionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Siehe auch

- „WINDOW-Klausel“ auf Seite 1124
- „CUME_DIST-Funktion [Rangfolge]“ auf Seite 213
- „DENSE_RANK-Funktion [Rangfolge]“ auf Seite 237
- „RANK-Funktion [Rangfolge]“ auf Seite 346

Standards und Kompatibilität

- **SQL/2008** PERCENT_RANK ist Teil der optionalen SQL/2008-Sprachenfunktion T612, "Erweiterte OLAP-Vorgänge".

Beispiel

Im folgenden Beispiel wird eine Ergebnismenge zurückgegeben, die die Rangfolge der Gehälter von Mitarbeitern in New York in absteigender Reihenfolge nach Geschlecht zeigt:

```
SELECT DepartmentID, Surname, Salary, Sex,  
       PERCENT_RANK() OVER (PARTITION BY Sex  
       ORDER BY Salary DESC) "Rank"  
FROM GROUP0.Employees  
WHERE State IN ('NY');
```

DepartmentID	Surname	Salary	Sex	Rank
200	Martel	55700.000	M	0
100	Guevara	42998.000	M	0.333333333
100	Soo	39075.000	M	0.666666667
400	Ahmed	34992.000	M	1
300	Davidson	57090.000	F	0
400	Blaikie	54900.000	F	0.333333333
100	Whitney	45700.000	F	0.666666667

DepartmentID	Surname	Salary	Sex	Rank
400	Wetherby	35745.000	F	1

PI-Funktion [Nummerisch]

Gibt den numerischen Wert PI zurück.

Syntax

PI([*])

Rückgabe

DOUBLE

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Bemerkungen

Die Funktion gibt einen DOUBLE-Wert zurück.

PI(*) und PI() sind semantisch gleichwertig.

Beispiel

Die folgende Anweisung gibt den Wert 3,141592653(...) zurück:

```
SELECT PI( * );
```

PLAN-Funktion [Verschiedene]

Gibt die lange Planoptimierungsstrategie einer als Zeichenfolge angegebenen SQL-Anweisung zurück.

Syntax

PLAN(*string-expression*, [*cursor-type* [*update-status*]])

Parameter

- **Zeichenfolgenerausdruck** Die SQL-Anweisung, die gewöhnlich eine SELECT-Anweisung ist, aber auch eine UPDATE-, MERGE- oder DELETE-Anweisung sein kann.
- **Cursortyp** Eine Zeichenfolge. *Cursortyp* kann "asensitive" (Standardwert), "insensitive", "sensitive" oder "keyset-driven" sein.
- **update-status** Ein Zeichenfolgenparameter, der einen der folgenden Werte akzeptiert, die angeben, wie der Optimierer die existierenden Cursor behandeln soll:

Wert	Beschreibung
READ-ONLY	Der Cursor ist schreibgeschützt.
READ-WRITE (Standardwert)	Der Cursor kann gelesen oder beschrieben werden.
FOR UPDATE	Der Cursor kann gelesen oder beschrieben werden. Dieser Wert ist exakt derselbe wie READ-WRITE.

Rückgabe

LONG VARCHAR

Siehe auch

- „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „EXPLANATION-Funktion [Verschiedene]“ auf Seite 260
- „GRAPHICAL_PLAN-Funktion [Verschiedene]“ auf Seite 269

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung übergibt eine SELECT-Anweisung als Zeichenfolgenparameter und gibt den Ausführungsplan für die Abfrage zurück:

```
SELECT PLAN(  
    'SELECT * FROM GROUPO.Departments WHERE DepartmentID > 100' );
```

Anhand dieser Informationen können Sie entscheiden, ob Sie Indizes hinzufügen oder wie Sie Ihre Datenbank zur Steigerung der Performance strukturieren sollen.

Die folgende Anweisung gibt eine Zeichenfolge zurück, die einen Textplan für einen INSENSITIVE-Cursor über die Abfrage `SELECT * FROM Departments WHERE DepartmentID > 100` enthält:

```
SELECT PLAN(  
    'SELECT * FROM GROUPO.Departments WHERE DepartmentID > 100',  
    'insensitive',  
    'read-only' );
```

POWER-Funktion [Nummerisch]

Berechnet die Potenz einer Zahl zur Basis einer anderen Zahl.

Syntax

POWER(*numeric-expression-1*, *numeric-expression-2*)

Parameter

- **Nummerischer_Ausdruck_1** Die Basis.
- **Nummerischer_Ausdruck_2** Der Exponent.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch. Wenn ein Argument NULL ist, ist das Ergebnis NULL.

Standards und Kompatibilität

- **SQL/2008** Die POWER-Funktion umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Die folgende Anweisung gibt den Wert 64 zurück:

```
SELECT POWER( 2, 6 );
```

PROPERTY-Funktion [System]

Gibt den Wert der angegebenen Datenbankserver-Eigenschaft als Zeichenfolge zurück.

Syntax

```
PROPERTY( { property-id | property-name } [, second-parameter ] )
```

Parameter

- **property-id** Eine Ganzzahl, die die Eigenschaftsnummer der Datenbankserver-Eigenschaft darstellt. Diese Zahl kann mit der PROPERTY_NUMBER-Funktion ermittelt werden. Die *property-id* wird üblicherweise bei einem Schleifendurchlauf bei einer Reihe von Eigenschaften verwendet.
- **Eigenschaftsname** Eine Zeichenfolge, die den Namen der Datenbankeigenschaft darstellt.
- **second-parameter** Sie können für einige Eigenschaften einen zweiten Parameter angeben, wie folgt:

Eigenschaft	Zweiter Parameter	Beschreibung
EventTypeDesc	Positive_Ganzzahl	Geben Sie eine Ereignis-ID an, um die Ereignistyp-Beschreibung zurückzugeben. Siehe EventTypeDesc-Servereigenschaft [SQL Anywhere Server - Datenbankadministration] .

Eigenschaft	Zweiter Parameter	Beschreibung
EventTypeName	<i>Positive_Ganzzahl</i>	Geben Sie eine Ereignis-ID an, um den Ereignistyp-Namen zurückzugeben. Siehe EventTypeName-Servereigenschaft [SQL Anywhere Server - Datenbankadministration] .
FunctionMaxParms	<i>Positive_Ganzzahl</i>	Geben Sie eine Funktionsnummer an, um die maximale Anzahl von Parametern zurückzugeben, die für die Funktion festgelegt werden kann. Siehe FunctionMaxParms-Servereigenschaft [SQL Anywhere Server - Datenbankadministration] .
FunctionMinParms	<i>Positive_Ganzzahl</i>	Geben Sie eine Funktionsnummer an, um die minimale Anzahl von Parametern zurückzugeben, die für die Funktion festgelegt werden muss. Siehe FunctionMinParms-Servereigenschaft [SQL Anywhere Server - Datenbankadministration] .
FunctionName	<i>Positive_Ganzzahl</i>	Geben Sie eine Funktionsnummer an, um den Namen der Funktion zurückzugeben. Siehe FunctionName-Servereigenschaft [SQL Anywhere Server - Datenbankadministration] .
Message	<i>Positive_Ganzzahl</i>	Geben Sie eine Zeilennummer an, um den Inhalt der entsprechenden Zeilen im Meldungsfenster des Datenbankservers zurückzugeben sowie als Präfix das Datum und die Uhrzeit der Meldung. Siehe Message-Servereigenschaft [SQL Anywhere Server - Datenbankadministration] .
MessageText	<i>Positive_Ganzzahl</i>	Geben Sie eine Zeilennummer an, um den Text ohne Datum- und Uhrzeit-Präfix zurückzugeben, der im Meldungsfenster des Datenbankservers der angegebenen Zeilennummer zugeordnet ist. Siehe MessageText-Servereigenschaft [SQL Anywhere Server - Datenbankadministration] .
MessageTime	<i>Positive_Ganzzahl</i>	Geben Sie eine Zeilennummer an, um das Datum und die Uhrzeit zurückzugeben, die im Meldungsfenster des Datenbankservers der angegebenen Zeilennummer zugeordnet sind. Siehe MessageTime-Servereigenschaft [SQL Anywhere Server - Datenbankadministration] .
RemoteCapability	<i>Positive_Ganzzahl</i>	Geben Sie die ID einer entfernten Fähigkeit ein, um den Namen der dieser ID zugeordneten entfernten Fähigkeit zurückzugeben. Siehe RemoteCapability-Servereigenschaft [SQL Anywhere Server - Datenbankadministration] .

Rückgabe

VARCHAR, LONG VARCHAR

Bemerkungen

Jede Eigenschaft hat sowohl eine Nummer als auch einen Namen, doch die Nummer kann sich zwischen den Versionen ändern und sollte daher nicht als zuverlässiger Bezeichner für eine bestimmte Eigenschaft verwendet werden.

Siehe auch

- „Liste der Datenbankseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „DB_PROPERTY-Funktion [System]“ auf Seite 232

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Namen des aktuellen Datenbankservers zurück.

```
SELECT PROPERTY( 'Name' );
```

PROPERTY_DESCRIPTION-Funktion [System]

Gibt die Beschreibung einer Eigenschaft zurück.

Syntax

```
PROPERTY_DESCRIPTION( { property-id | property-name } )
```

Parameter

- ***property-id*** Eine Ganzzahl, die die Eigenschaftsnummer der Datenbankeigenschaft darstellt. Diese Zahl kann mit der PROPERTY_NUMBER-Funktion ermittelt werden. Die *property-id* wird üblicherweise bei einem Schleifendurchlauf bei einer Reihe von Eigenschaften verwendet.
- ***Eigenschaftsname*** Eine Zeichenfolge, die den Namen der Datenbankeigenschaft darstellt.

Rückgabe

VARCHAR

Bemerkungen

Jede Eigenschaft hat sowohl eine Nummer als auch einen Namen, doch die Nummer kann sich zwischen den Versionen ändern und sollte daher nicht als zuverlässiger Bezeichner für eine bestimmte Eigenschaft verwendet werden.

Siehe auch

- „Verbindungs-, Datenbank- und Datenbankseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt `Number of index insertions` zurück, die Beschreibung der IndAdd-Eigenschaft:

```
SELECT PROPERTY_DESCRIPTION( 'IndAdd' );
```

PROPERTY_NAME-Funktion [System]

Die Funktion gibt den Namen der Eigenschaft mit der Eigenschafts-ID für die festgelegte Verbindungsstufe zurück.

Syntax

```
PROPERTY_NAME( property-id [, property-scope ] )
```

property-scope:

NULL

'server'

'database'

'db'

'connection'

'conn'

Parameter

- ***property-id*** Die Eigenschafts-ID der Datenbankeigenschaft.
- ***property-scope*** Der Bereich der Eigenschaft oder NULL.

Rückgabe

VARCHAR

Siehe auch

- „Liste der Verbindungseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Liste der Datenbankserveigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Liste der Datenbankeigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die Eigenschaft auf Serverebene mit der Eigenschafts-ID 102 zurück:

```
SELECT PROPERTY_NAME( 102, 'server' );
```

PROPERTY_NUMBER-Funktion [System]

Gibt die Eigenschaftsnummer der Eigenschaft mit dem angegebenen Eigenschaftsnamen zurück

Syntax

PROPERTY_NUMBER(*property-name*)

Parameter

- **Eigenschaftsname** Ein Eigenschaftsname

Rückgabe

INT

Bemerkungen

Jede Eigenschaft hat sowohl eine Nummer als auch einen Namen, doch die Nummer kann sich zwischen den Versionen ändern und sollte daher nicht als zuverlässiger Bezeichner für eine bestimmte Eigenschaft verwendet werden. In Fällen, in denen entweder die Eigenschaftsnummer oder der Eigenschaftsname verwendet werden kann, wird empfohlen, den Eigenschaftsnamen zu verwenden. Verwenden Sie immer die PROPERTY_NUMBER-Funktion um sicherzugehen, dass die Eigenschaftsnummer die aktuelle für den verwendeten Server ist.

Siehe auch

- „Verbindungs-, Datenbank- und Datenbankservereigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die Eigenschaftsnummer der PAGESIZE-Eigenschaft als Ganzzahl zurück:

```
SELECT PROPERTY_NUMBER( ' PAGESIZE' );
```

QUARTER-Funktion [Datum und Uhrzeit]

Gibt eine Zahl zurück, die das Quartal des Jahres im angegebenen TIMESTAMP-Ausdruck darstellt.

Syntax

QUARTER(*timestamp-expression*)

Parameter

- **Zeitstempelausdruck** Das Datum für das Quartal.

Rückgabe

INTEGER

Bemerkungen

Die Quartale sind die folgenden:

Quarter	Zeitraum (inklusive)
1	Januar bis 31. März
2	April bis 30. Juni
3	Juli bis 30. September
4	Oktober bis 31. Dezember

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 2 zurück:

```
SELECT QUARTER( '1987/05/02' );
```

RADIANS-Funktion [Numerisch]

Konvertiert eine Zahl von Grad in Bogenmaß.

Syntax

RADIANS(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Eine Zahl, in Grad. Dieser Winkel wird in Bogenmaß konvertiert.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt einen Wert von ungefähr 0,5236 zurück:

```
SELECT RADIANS( 30 );
```

RAND-Funktion [Numerisch]

Returns a random number in the interval 0 to 1, with an optional seed.

Syntax

RAND([*integer-expression*])

Parameter

- **Ganzzahlausdruck** Ein optionaler Initialwert zur Erstellung einer Zufallszahl. Dieses Argument ermöglicht Ihnen die Erstellung einer Zufallszahlensequenz.

Rückgabe

DOUBLE

Bemerkungen

Die RAND-Funktion ist ein multiplikativ-linear-kongruenter Zufallszahlengenerator. Siehe Park and Miller (1988), CACM 31(10), S. 1192-1201 and Press et al. (1992), Numerical Recipes in C (2. Ausgabe, Kapitel 7, S. 279). Das Ergebnis eines Aufrufs der RAND-Funktion ist eine Pseudo-Zufallszahl n . Dabei gilt: $0 < n < 1$ (weder 0,0 noch 1,0 kann das Ergebnis sein).

Wenn eine Verbindung zum Server hergestellt wird, setzt der Zufallszahlengenerator einen Anfangswert (Initialwert). Jede Verbindung erhält einen eindeutigen Initialwert und sieht daher eine andere Zufallssequenz von anderen Verbindungen. Sie können auch einen Initialwert (*Ganzzahlausdruck*) als Argument angeben. Normalerweise sollten Sie dies nur einmal durchführen, bevor Sie eine Sequenz von Zufallszahlen durch nachfolgende Aufrufe der RAND-Funktion anfordern. Wenn Sie den Initialwert mehr als einmal initialisieren, wird die Sequenz neu gestartet. Wenn Sie denselben Initialwert angeben, wird dieselbe Sequenz generiert. Initialwerte, deren Werte nahe beieinander liegen, generieren ähnliche Anfangssequenzen, deren Divergenzen im Laufe der Sequenz zunehmen.

Kombinieren Sie bei dem Versuch, statistische Zufallsergebnisse zu erhalten, nicht die von einem Initialwert generierte Sequenz mit einem von einem zweiten Initialwert generierten Sequenz. Anders gesagt: Sie dürfen den Initialwert während der Generierung einer Sequenz von Zufallswerten nicht zurücksetzen.

Die RAND-Funktion wird als eine nicht-deterministische Funktion angesehen. Der Abfrageoptimierer behält die Ergebnisse der Funktion RAND nicht im Cache.

Weitere Hinweise zu nicht-deterministischen Funktionen finden Sie unter [Caching von Funktionen \[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#).

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgenden Anweisungen ergeben elf Zufallsergebnisse. Jeder nachfolgende Aufruf der RAND-Funktion, bei dem kein Initialwert angegeben ist, erzeugt weiterhin unterschiedliche Ergebnisse:

```
SELECT RAND( 1 );  
SELECT RAND( ), RAND( ), RAND( ), RAND( ), RAND( );  
SELECT RAND( ), RAND( ), RAND( ), RAND( ), RAND( );
```

Die folgende Anweisung erzeugt zwei Ergebnismengen mit identischen Sequenzen, da der Initialwert zweimal angegeben wird:

```
SELECT RAND( 1 ), RAND( ), RAND( ), RAND( ), RAND( );  
SELECT RAND( 1 ), RAND( ), RAND( ), RAND( ), RAND( );
```

Das folgende Beispiel erzeugt fünf Ergebnisse, die ähnliche Werte enthalten und keine zufällige Verteilung aufweisen. Aus diesem Grund ist es nicht empfehlenswert, die RAND-Funktion mehr als einmal mit ähnlichen Initialwerten aufzurufen:

```
SELECT RAND( 1 ), RAND( 2 ), RAND( 3 ), RAND( 4 ), RAND( 5 );
```

Das folgende Beispiel erzeugt fünf identische Ergebnisse und sollte vermieden werden:

```
SELECT RAND( 1 ), RAND( 1 ), RAND( 1 ), RAND( 1 ), RAND( 1 );
```

RANK-Funktion [Rangfolge]

Berechnet den Rang eines Wertes in einer Gruppe von Werten. Im Fall von Gleichwertigkeit bewirkt die RANK-Funktion eine Lücke in der Rangfolgesequenz.

Syntax

RANK() OVER (window-spec)

window-spec : see the Remarks section below

Rückgabe

INTEGER

Bemerkungen

Elemente von *Fensterspezifikation* können entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Bei Verwendung als Fensterfunktion müssen Sie eine ORDER BY-Klausel angeben und dürfen eine PARTITION BY-Klausel angeben, können jedoch keine ROWS- oder RANGE-Klausel angeben. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel. Siehe „[WINDOW-Klausel](#)“ auf Seite 1124.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „CUME_DIST-Funktion [Rangfolge]“ auf Seite 213
- „DENSE_RANK-Funktion [Rangfolge]“ auf Seite 237
- „ROW_NUMBER-Funktion [Verschiedene]“ auf Seite 373
- „PERCENT_RANK-Funktion [Rangfolge]“ auf Seite 335

Standards und Kompatibilität

- **SQL/2008** Die RANK-Funktion ist Teil der optionalen SQL/2008-Sprachenfunktion T612, "Erweiterte OLAP-Vorgänge".

Beispiel

Das folgende Beispiel liefert eine Rangordnung der Gehälter von Mitarbeitern in Utah und New York in absteigender Reihenfolge. Beachten Sie, dass der 7. und 8. Mitarbeiter das gleiche Gehalt beziehen und sich daher den 7. Rang teilen. Der darauf folgende Mitarbeiter erhält den 9. Rang, wodurch eine Lücke in der Rangfolgensequenz entsteht (kein 8. Rang).

```
SELECT Surname, Salary, State,
       RANK() OVER (ORDER BY Salary DESC) "Rank"
FROM GROUP0.Employees WHERE State IN ('NY','UT');
```

Surname	Salary	State	Rank
Shishov	72995.000	UT	1
Wang	68400.000	UT	2
Cobb	62000.000	UT	3
Morris	61300.000	UT	4
Davidson	57090.000	NY	5
Martel	55700.000	NY	6
Blaikie	54900.000	NY	7
Diaz	54900.000	NY	7
Driscoll	48023.690	UT	9
Hildebrand	45829.000	UT	10
Whitney	45700.000	NY	11
...
Lynch	24903.000	UT	19

READ_CLIENT_FILE-Funktion

Liest Daten aus einer angegebenen Datei auf dem Clientcomputer.

Syntax

READ_CLIENT_FILE(*client-filename-expression*)

Parameter

- **Clientdateiname-Ausdruck** CHAR-Wert, der den Namen der Datei auf dem Clientcomputer angibt. Der Pfad wird auf dem Clientcomputer relativ zum aktuellen Arbeitsverzeichnis der Clientanwendung aufgelöst.

Rückgabe

LONG BINARY

Bemerkungen

Der von der READ_CLIENT_FILE-Funktion zurückgegebene Wert entspricht dem Inhalt der angegebenen Clientdatei. Sie können die Funktion in der Syntax überall dort verwenden, wo ein BINARY-Ausdruck zulässig ist.

Da die Daten als Binärzeichenfolge zurückgegeben werden, gilt: Wenn die Daten in einem anderen Zeichensatz, komprimiert oder verschlüsselt sind, müssen Sie gegebenenfalls auch eine Zeichensatzkonvertierung, Dekomprimierung oder Entschlüsselung durchführen.

Während der Auswertung der READ_CLIENT_FILE-Funktion beginnt der Datenbankserver mit der Übermittlung der angegebenen Datei vom Client. Nach Empfang der Übertragungsanforderungen setzt der Client eine gemeinsame Sperre für die Datei auf dem Client und hält die Sperre, bis der Datenbankserver den Client auffordert, die Anforderung abzuschließen.

Das Lesen der Datei erfolgt durch die Client-Softwarebibliothek und die Übertragung der Daten erfolgt mit dem Befehlsfolgen-Kommunikationsprotokoll.

Privilegien

Beim Lesen einer Datei auf einem Clientcomputer gilt Folgendes:

- Sie müssen das READ FILE-Systemprivileg haben.
- Leseberechtigungen sind in dem Verzeichnis erforderlich, aus dem gelesen werden soll.
- Die Datenbankoption allow_read_client_file muss aktiviert sein.
- Die gesicherte Funktion read_client_file muss aktiviert sein.

Siehe auch

- „Datenbankserveroption -sf“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „allow_read_client_file-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Zugriff auf Daten auf Clientcomputern“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „DECOMPRESS-Funktion [Zeichenfolge]“ auf Seite 233
- „DECRYPT-Funktion [Zeichenfolge]“ auf Seite 234
- „CSCONVERT-Funktion [Zeichenfolge]“ auf Seite 211

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

REGEXP_SUBSTR-Funktion [Zeichenfolge]

Extrahiert Teilzeichenfolgen aus Zeichenfolgen mithilfe von regulären Ausdrücken.

Syntax

```
REGEXP_SUBSTR( expression,
regular-expression
[, start-offset [, occurrence-number [, escape-expression ] ] ] )
```

Parameter

- **expression** Die zu durchsuchende Zeichenfolge.
- **Regulärer Ausdruck** Das Suchmuster. Weitere Hinweise zur Syntax regulärer Ausdrücke finden Sie unter „[Überblick über reguläre Ausdrücke](#)“ auf Seite 27.
- **start-offset** Ausgangspunkt in *expression*, an dem begonnen werden soll. *start-offset* wird als positive Ganzzahl ausgedrückt und stellt die Anzahl der Zeichen dar, die gezählt werden müssen, wenn auf der linken Seite der Zeichenfolge begonnen wird. Standardwert ist 1 (Beginn der Zeichenfolge).
- **occurrence-number** Bei mehreren Fundstellen in *expression* geben Sie mit einer Ganzzahl an, welches Vorkommen ermittelt werden soll. Beispiel: Wenn dieser Parameter 3 lautet, wird das dritte Vorkommen gefunden. Standardwert ist "1".
- **Escape-Ausdruck** Das Escapezeichen, das für *Regulärer Ausdruck* verwendet werden soll. Das Standardzeichen ist der Backslash (\).

Rückgabe

LONG VARCHAR

Bemerkungen

REGEXP_SUBSTR gibt NULL zurück, wenn *Regulärer Ausdruck* nicht gefunden wird.

Ähnlich wie bei der Suchbedingung REGEXP benutzt die REGEXP_SUBSTR-Funktion Codepunkte für die Übereinstimmungs- und Bereichsüberprüfung. Die Berücksichtigung von Groß- und Kleinschreibung durch die Datenbank hat keine Auswirkungen auf die Ergebnisse. Weitere Hinweise zur Suche nach

Übereinstimmungen mit REGEXP_SUBSTR und die Auswertung von Gruppen finden Sie unter [Unterschiede im Zeichenvergleich von LIKE, REGEXP und SIMILAR TO auf Seite 49](#).

Wenn Übereinstimmungen mit einer Zeichenklasse gesucht werden, die nur eine Teilzeichenklasse enthält, verwenden Sie äußere eckige Klammern und eckige Klammern für die Teilzeichenklasse (Beispiel: REGEXP_SUBSTR (expression, '[:digit:]')). Weitere Hinweise zur Suche nach Übereinstimmungen in Teilzeichenklassen finden Sie unter [Reguläre Ausdrücke: Spezielle Teilzeichenklassen auf Seite 31](#).

Wenn *start-offset* angegeben ist, legt dieser Wert fest, wo der zu suchende Ausdruck beginnt. Speziell sind folgende Punkte zu beachten: "^" findet den Anfang des Ausdruck, der am *start-offset*-Wert beginnt.

Siehe auch

- „Syntax für reguläre Ausdrücke“ auf Seite 28
- „REGEXP-Suchbedingung“ auf Seite 55

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Die entsprechende Funktion im SQL/2008-Standard ist die SUBSTRING_REGEX Funktion, die ähnliche Parameter hat. SUBSTRING_REGEX ist Teil der optionalen SQL/2008-Sprachenfunktion T844.

Beispiel

Die folgende Anweisung teilt Werte in der Employees.Street-Spalte in Hausnummer und Straßennamen auf:

```
SELECT REGEXP_SUBSTR( Street, '^S+' ) as street_num,  
       REGEXP_SUBSTR( Street, '(?<=^S+\s+).*S+' ) AS street_name  
FROM GROUPO.Employees;
```

street_num	street_name
9	East Washington Street
7	Pleasant Street
539	Pond Street
1244	Great Plain Avenue
...	...

Um zu ermitteln, ob die IP-Adresse der aktuellen Verbindung in einem bestimmten Bereich von IP-Adressen liegt (in diesem Fall 10.25.101.xxx oder 10.25.102.xxx), können Sie folgende Anweisung ausführen:

```
IF REGEXP_SUBSTR( CONNECTION_PROPERTY( 'NodeAddress' ), '\\d+\\.\\d+\\.\\d+' )  
IN ( '10.25.101' , '10.25.102' ) THEN  
    MESSAGE 'In range' TO CLIENT;  
ELSE
```

```
MESSAGE 'Out of range' TO CLIENT;
END IF;
```

REGR_AVGX-Funktion [Aggregat]

Berechnet den Mittelwert der unabhängigen Variablen von der Regressionszeile.

Syntax 1

REGR_AVGX(*dependent-expression* , *independent-expression*)

Syntax 2

REGR_AVGX(*dependent-expression* , *independent-expression*)
OVER (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Abhängiger_Ausdruck** Die Variable, auf die sich der unabhängige Ausdruck auswirkt.
- **Unabhängiger_Ausdruck** Die Variable, die das Ergebnis beeinflusst

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelte genaue Gleitkommazahl durch. Wenn die Funktion bei einer leeren Menge angewendet wird, gibt sie NULL zurück.

Die Funktion wird auf die Menge von (*Abhängiger_Ausdruck* und *Unabhängiger_Ausdruck*)-Paaren angewendet, nachdem alle Paare eliminiert wurden, bei denen *Abhängiger_Ausdruck* oder *Unabhängiger_Ausdruck* NULL ist. Die Funktion wird gleichzeitig während eines einzigen Durchlaufes der Daten berechnet. Nach dem Entfernen von NULL-Werten wird die folgende Berechnung durchgeführt, wobei *x* den *unabhängigen_Ausdruck* darstellt:

```
AVG( x )
```

Weitere Hinweise zur durchgeführten statistischen Berechnung finden Sie unter „[Mathematische Formeln für die Aggregatfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „AVG-Funktion [Aggregat]“ auf Seite 175
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_INTERCEPT-Funktion [Aggregat]“ auf Seite 355
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_SLOPE-Funktion [Aggregat]“ auf Seite 358
- „REGR_SXX-Funktion [Aggregat]“ auf Seite 360
- „REGR_SXY-Funktion [Aggregat]“ auf Seite 361
- „REGR_SYY-Funktion [Aggregat]“ auf Seite 362
- „REGR_AVGY-Funktion [Aggregat]“ auf Seite 352
- „WINDOW-Klausel“ auf Seite 1124

Standards und Kompatibilität

- **SQL/2008** REGR_AVGX ist Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Die folgende Anweisung berechnet den Mittelwert der abhängigen Variablen, des Mitarbeiteralters:

```
SELECT REGR_AVGX( Salary, ( 2008 - YEAR( BirthDate ) ) )  
FROM GROUPO.Employees;
```

REGR_AVGY-Funktion [Aggregat]

Berechnet den Mittelwert der abhängigen Variablen von der Regressionszeile.

Syntax 1

REGR_AVGY(*dependent-expression* , *independent-expression*)

Syntax 2

REGR_AVGY(*dependent-expression* , *independent-expression*)
OVER (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Abhängiger_Ausdruck** Die Variable, auf die sich der unabhängige Ausdruck auswirkt.
- **Unabhängiger_Ausdruck** Die Variable, die das Ergebnis beeinflusst

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch. Wenn die Funktion bei einer leeren Menge angewendet wird, gibt sie NULL zurück.

Die Funktion wird auf die Menge von (*Abhängiger_Ausdruck* und *Unabhängiger_Ausdruck*)-Paaren angewendet, nachdem alle Paare eliminiert wurden, bei denen *Abhängiger_Ausdruck* oder *Unabhängiger_Ausdruck* NULL ist. Die Funktion wird gleichzeitig während eines einzigen Durchlaufes der Daten berechnet. Nach dem Entfernen von NULL-Werten wird die folgende Berechnung durchgeführt, wobei *y* *Abhängiger_Ausdruck* darstellt:

`AVG(y)`

Weitere Hinweise zur durchgeführten statistischen Berechnung finden Sie unter „[Mathematische Formeln für die Aggregatfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_INTERCEPT-Funktion [Aggregat]“ auf Seite 355
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_SLOPE-Funktion [Aggregat]“ auf Seite 358
- „REGR_SXX-Funktion [Aggregat]“ auf Seite 360
- „REGR_SXY-Funktion [Aggregat]“ auf Seite 361
- „REGR_SYY-Funktion [Aggregat]“ auf Seite 362
- „REGR_AVGX-Funktion [Aggregat]“ auf Seite 351
- „AVG-Funktion [Aggregat]“ auf Seite 175
- „WINDOW-Klausel“ auf Seite 1124

Standards und Kompatibilität

- **SQL/2008** REGR_AVGY ist Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Die folgende Anweisung berechnet den Mittelwert der unabhängigen Variablen, des Mitarbeitergehalts:

```
SELECT REGR_AVGY( Salary, ( YEAR( NOW( )) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

REGR_COUNT-Funktion [Aggregat]

Gibt eine Ganzzahl zurück, die die Anzahl von Nicht-NULL-Zahlenpaaren darstellt, die in der Regressionszeile verwendet werden.

Syntax 1

REGR_COUNT(*dependent-expression* , *independent-expression*)

Syntax 2

REGR_COUNT(*dependent-expression* , *independent-expression*)
OVER (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Abhängiger_Ausdruck** Die Variable, auf die sich der unabhängige Ausdruck auswirkt.
- **Unabhängiger_Ausdruck** Die Variable, die das Ergebnis beeinflusst

Rückgabe

INTEGER

Bemerkungen

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „Fensterfunktionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Weitere Hinweise zur durchgeführten statistischen Berechnung finden Sie unter „Mathematische Formeln für die Aggregatfunktionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Siehe auch

- „WINDOW-Klausel“ auf Seite 1124
- „REGR_INTERCEPT-Funktion [Aggregat]“ auf Seite 355
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_SLOPE-Funktion [Aggregat]“ auf Seite 358
- „REGR_SXX-Funktion [Aggregat]“ auf Seite 360
- „REGR_SXY-Funktion [Aggregat]“ auf Seite 361
- „REGR_SYY-Funktion [Aggregat]“ auf Seite 362
- „REGR_AVGY-Funktion [Aggregat]“ auf Seite 352
- „REGR_AVGX-Funktion [Aggregat]“ auf Seite 351
- „COUNT-Funktion [Aggregat]“ auf Seite 205
- „AVG-Funktion [Aggregat]“ auf Seite 175
- „SUM-Funktion [Aggregat]“ auf Seite 403

Standards und Kompatibilität

- **SQL/2008** REGR_COUNT ist Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Das folgende Beispiel gibt die Anzahl der in der Regressionszeile verwendeten Nicht-NULL-Paare zurück:

```
SELECT REGR_COUNT( Salary, ( YEAR( NOW() ) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

REGR_INTERCEPT-Funktion [Aggregat]

Berechnet den y-Abschnitt der linearen Regressionszeile, der am besten zu den abhängigen und unabhängigen Variablen passt.

Syntax 1

REGR_INTERCEPT(*dependent-expression* , *independent-expression*)

Syntax 2

REGR_INTERCEPT(*dependent-expression* , *independent-expression*)
OVER (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Abhängiger_Ausdruck** Die Variable, auf die sich der unabhängige Ausdruck auswirkt.
- **Unabhängiger_Ausdruck** Die Variable, die das Ergebnis beeinflusst

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelte genaue Gleitkommazahl durch. Wenn die Funktion bei einer leeren Menge angewendet wird, gibt sie NULL zurück.

Die Funktion wird auf die Menge von (*Abhängiger_Ausdruck* und *Unabhängiger_Ausdruck*)-Paaren angewendet, nachdem alle Paare eliminiert wurden, bei denen *Abhängiger_Ausdruck* oder *Unabhängiger_Ausdruck* NULL ist. Die Funktion wird gleichzeitig während eines einzigen Durchlaufes der Daten berechnet. Nach dem Entfernen von NULL-Werten wird die folgende Berechnung durchgeführt, wobei *y* *Abhängiger_Ausdruck* und *x* *Unabhängiger_Ausdruck* darstellt:

$$\text{AVG}(y) - \text{REGR_SLOPE}(y, x) * \text{AVG}(x)$$

Weitere Hinweise zur durchgeführten statistischen Berechnung finden Sie unter „[Mathematische Formeln für die Aggregatfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „WINDOW-Klausel“ auf Seite 1124
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_SLOPE-Funktion [Aggregat]“ auf Seite 358
- „REGR_SXX-Funktion [Aggregat]“ auf Seite 360
- „REGR_SXY-Funktion [Aggregat]“ auf Seite 361
- „REGR_SYY-Funktion [Aggregat]“ auf Seite 362
- „REGR_AVGY-Funktion [Aggregat]“ auf Seite 352
- „REGR_AVGX-Funktion [Aggregat]“ auf Seite 351
- „REGR_SLOPE-Funktion [Aggregat]“ auf Seite 358
- „AVG-Funktion [Aggregat]“ auf Seite 175

Standards und Kompatibilität

- **SQL/2008** REGR_INTERCEPT ist Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Die folgende Anweisung gibt den y-Achsenabschnitt der linearen Regressionszeile zurück:

```
SELECT REGR_INTERCEPT( Salary, ( YEAR( NOW( )) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

REGR_R2-Funktion [Aggregat]

Berechnet den Koeffizienten der Bestimmtheit (auch *R-Quadrat*-- bzw. *Goodness of fit-Test* genannt) für die Regressionszeile.

Syntax 1

REGR_R2(*dependent-expression* , *independent-expression*)

Syntax 2

REGR_R2(*dependent-expression* , *independent-expression*)
OVER (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Abhängiger_Ausdruck** Die Variable, auf die sich der unabhängige Ausdruck auswirkt.
- **Unabhängiger_Ausdruck** Die Variable, die das Ergebnis beeinflusst

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch. Wenn die Funktion bei einer leeren Menge angewendet wird, gibt sie NULL zurück.

Die Funktion wird auf die Menge von (*Abhängiger_Ausdruck* und *Unabhängiger_Ausdruck*)-Paaren angewendet, nachdem alle Paare eliminiert wurden, bei denen *Abhängiger_Ausdruck* oder *Unabhängiger_Ausdruck* NULL ist.

Weitere Hinweise zur durchgeführten statistischen Berechnung finden Sie unter „[Mathematische Formeln für die Aggregatfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „WINDOW-Klausel“ auf Seite 1124
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_INTERCEPT-Funktion [Aggregat]“ auf Seite 355
- „REGR_SLOPE-Funktion [Aggregat]“ auf Seite 358
- „REGR_SXX-Funktion [Aggregat]“ auf Seite 360
- „REGR_SXY-Funktion [Aggregat]“ auf Seite 361
- „REGR_SYY-Funktion [Aggregat]“ auf Seite 362
- „REGR_AVGX-Funktion [Aggregat]“ auf Seite 351
- „REGR_AVGY-Funktion [Aggregat]“ auf Seite 352

Standards und Kompatibilität

- **SQL/2008** REGR_R2 ist Teil der optionalen SQL/2008-Sprachenfunktion T/621, "Enhanced numeric functions".

Beispiel

Das folgende Beispiel gibt das Bestimmtheitsmaß für die Regressionszeile zurück:

```
SELECT REGR_R2( Salary, ( YEAR( NOW( )) - YEAR( BirthDate ) ) )  
FROM GROUPO.Employees;
```

REGR_SLOPE-Funktion [Aggregat]

Berechnet den Richtungskoeffizienten der linearen Regressionszeile für Nicht-NULL-Paare.

Syntax 1

REGR_SLOPE(*dependent-expression* , *independent-expression*)

Syntax 2

REGR_SLOPE(*dependent-expression* , *independent-expression*)
OVER (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Abhängiger_Ausdruck** Die Variable, auf die sich der unabhängige Ausdruck auswirkt.
- **Unabhängiger_Ausdruck** Die Variable, die das Ergebnis beeinflusst

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelte genaue Gleitkommazahl durch. Wenn die Funktion bei einer leeren Menge angewendet wird, gibt sie NULL zurück.

Die Funktion wird auf die Menge von (*Abhängiger_Ausdruck* und *Unabhängiger_Ausdruck*)-Paaren angewendet, nachdem alle Paare eliminiert wurden, bei denen *Abhängiger_Ausdruck* oder *Unabhängiger_Ausdruck* NULL ist. Die Funktion wird gleichzeitig während eines einzigen Durchlaufes der Daten berechnet. Nach dem Entfernen von NULL-Werten wird die folgende Berechnung durchgeführt, wobei *y* *Abhängiger_Ausdruck* und *x* *Unabhängiger_Ausdruck* darstellt:

$$\text{COVAR_POP}(y, x) / \text{VAR_POP}(x)$$

Weitere Hinweise zur durchgeführten statistischen Berechnung finden Sie unter „[Mathematische Formeln für die Aggregatfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „WINDOW-Klausel“ auf Seite 1124
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_INTERCEPT-Funktion [Aggregat]“ auf Seite 355
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_SXX-Funktion [Aggregat]“ auf Seite 360
- „REGR_SXY-Funktion [Aggregat]“ auf Seite 361
- „REGR_SYY-Funktion [Aggregat]“ auf Seite 362
- „REGR_AVGX-Funktion [Aggregat]“ auf Seite 351
- „REGR_AVGY-Funktion [Aggregat]“ auf Seite 352
- „COVAR_POP-Funktion [Aggregat]“ auf Seite 209
- „VAR_POP-Funktion [Aggregat]“ auf Seite 425

Standards und Kompatibilität

- **SQL/2008** REGR_SLOPE ist Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Die folgende Anweisung gibt den Wert 935,3429749445614 zurück:

```
SELECT REGR_SLOPE( Salary, ( YEAR( NOW() ) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

REGR_SXX-Funktion [Aggregat]

Gibt die Summe der Quadrate von unabhängigen Ausdrücken zurück, die in einem linearen Regressionsmodell verwendet werden. Die REGR_SXX-Funktion kann verwendet werden, um die statistische Gültigkeit eines Regressionsmodells zu bewerten.

Syntax 1

REGR_SXX(*dependent-expression* , *independent-expression*)

Syntax 2

REGR_SXX(*dependent-expression* , *independent-expression*)
OVER (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Abhängiger_Ausdruck** Die Variable, auf die sich der unabhängige Ausdruck auswirkt.
- **Unabhängiger_Ausdruck** Die Variable, die das Ergebnis beeinflusst

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppeltgenaue Gleitkommazahl durch. Wenn die Funktion bei einer leeren Menge angewendet wird, gibt sie NULL zurück.

Die Funktion wird auf die Menge von (*Abhängiger_Ausdruck* und *Unabhängiger_Ausdruck*)-Paaren angewendet, nachdem alle Paare eliminiert wurden, bei denen *Abhängiger_Ausdruck* oder *Unabhängiger_Ausdruck* NULL ist. Die Funktion wird gleichzeitig während eines einzigen Durchlaufes der Daten berechnet. Nach dem Entfernen von NULL-Werten wird die folgende Berechnung durchgeführt, wobei *y* *Abhängiger_Ausdruck* und *x* *Unabhängiger_Ausdruck* darstellt:

```
REGR_COUNT( y , x ) * VAR_POP( x )
```

Weitere Hinweise zur durchgeführten statistischen Berechnung finden Sie unter „[Mathematische Formeln für die Aggregatfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „WINDOW-Klausel“ auf Seite 1124
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_INTERCEPT-Funktion [Aggregat]“ auf Seite 355
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_AVGX-Funktion [Aggregat]“ auf Seite 351
- „REGR_AVGY-Funktion [Aggregat]“ auf Seite 352
- „REGR_SXY-Funktion [Aggregat]“ auf Seite 361
- „REGR_SYY-Funktion [Aggregat]“ auf Seite 362
- „VAR_POP-Funktion [Aggregat]“ auf Seite 425

Standards und Kompatibilität

- **SQL/2008** REGR_SXX ist Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Die folgende Anweisung gibt den Wert 5916,4800000000105 zurück:

```
SELECT REGR_SXX( Salary, ( YEAR( NOW() ) - YEAR( BirthDate ) ) )
FROM GROUPO.Employees;
```

REGR_SXY-Funktion [Aggregat]

Gibt die Summe der Produkte der abhängigen und unabhängigen Variablen zurück. Die REGR_SXY-Funktion kann verwendet werden, um die statistische Gültigkeit eines Regressionsmodells zu bewerten.

Syntax 1

REGR_SXY(*dependent-expression* , *independent-expression*)

Syntax 2

REGR_SXY(*dependent-expression* , *independent-expression*)
OVER (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Abhängiger_Ausdruck** Die Variable, auf die sich der unabhängige Ausdruck auswirkt.
- **Unabhängiger_Ausdruck** Die Variable, die das Ergebnis beeinflusst

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltgenauem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück. Wenn die Funktion bei einer leeren Menge angewendet wird, gibt sie NULL zurück.

Die Funktion wird auf die Menge von (*Abhängiger_Ausdruck* und *Unabhängiger_Ausdruck*)-Paaren angewendet, nachdem alle Paare eliminiert wurden, bei denen *Abhängiger_Ausdruck* oder *Unabhängiger_Ausdruck* NULL ist. Die Funktion wird gleichzeitig während eines einzigen Durchlaufes der Daten berechnet. Nach dem Entfernen von NULL-Werten wird die folgende Berechnung durchgeführt, wobei *y* *Abhängiger_Ausdruck* und *x* *Unabhängiger_Ausdruck* darstellt:

```
REGR_COUNT( y, x ) * COVAR_POP( y, x )
```

Weitere Hinweise zur durchgeführten statistischen Berechnung finden Sie unter „[Mathematische Formeln für die Aggregatfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „WINDOW-Klausel“ auf Seite 1124
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_INTERCEPT-Funktion [Aggregat]“ auf Seite 355
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_SLOPE-Funktion [Aggregat]“ auf Seite 358
- „REGR_AVGX-Funktion [Aggregat]“ auf Seite 351
- „REGR_AVGY-Funktion [Aggregat]“ auf Seite 352
- „REGR_SXX-Funktion [Aggregat]“ auf Seite 360
- „REGR_SYY-Funktion [Aggregat]“ auf Seite 362

Standards und Kompatibilität

- **SQL/2008** REGR_SXY ist Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Das folgende Beispiel gibt die Summe der Produkte aus abhängigen und unabhängigen Variablen zurück:

```
SELECT REGR_SXY( Salary, ( YEAR( NOW( )) - YEAR( BirthDate ) ) )  
FROM GROUP0.Employees;
```

REGR_SYY-Funktion [Aggregat]

Gibt Werte zurück, die die statistische Gültigkeit eines Regressionsmodells bewerten.

Syntax 1

REGR_SYY(*dependent-expression* , *independent-expression*)

Syntax 2

REGR_SYY(*dependent-expression* , *independent-expression*)
OVER (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Abhängiger_Ausdruck** Die Variable, auf die sich der unabhängige Ausdruck auswirkt.
- **Unabhängiger_Ausdruck** Die Variable, die das Ergebnis beeinflusst

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch. Wenn die Funktion bei einer leeren Menge angewendet wird, gibt sie NULL zurück.

Die Funktion wird auf die Menge von (*Abhängiger_Ausdruck* und *Unabhängiger_Ausdruck*)-Paaren angewendet, nachdem alle Paare eliminiert wurden, bei denen *Abhängiger_Ausdruck* oder *Unabhängiger_Ausdruck* NULL ist. Die Funktion wird gleichzeitig während eines einzigen Durchlaufes der Daten berechnet. Nach dem Entfernen von NULL-Werten wird die folgende Berechnung durchgeführt, wobei *y* *Abhängiger_Ausdruck* und *x* *Unabhängiger_Ausdruck* darstellt:

```
REGR_COUNT( y, x ) * VAR_POP( y )
```

Weitere Hinweise zur durchgeführten statistischen Berechnung finden Sie unter „[Mathematische Formeln für die Aggregatfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „WINDOW-Klausel“ auf Seite 1124
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_INTERCEPT-Funktion [Aggregat]“ auf Seite 355
- „REGR_COUNT-Funktion [Aggregat]“ auf Seite 354
- „REGR_AVGX-Funktion [Aggregat]“ auf Seite 351
- „REGR_AVGY-Funktion [Aggregat]“ auf Seite 352
- „REGR_SLOPE-Funktion [Aggregat]“ auf Seite 358
- „REGR_SXX-Funktion [Aggregat]“ auf Seite 360
- „REGR_SXY-Funktion [Aggregat]“ auf Seite 361

Standards und Kompatibilität

- **SQL/2008** REGR_SYY ist Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Die folgende Anweisung gibt den Wert 672.843,3002 zurück:

```
SELECT REGR_SYY( Salary, ( YEAR( NOW( )) - YEAR( BirthDate ) ) )  
FROM GROUPO.Employees;
```

REMAINDER-Funktion [Nummerisch]

Gibt den Rest zurück, wenn eine Ganzzahl durch eine andere dividiert wird.

Syntax

REMAINDER(*dividend*, *divisor*)

Parameter

- ***dividend*** Der Dividend oder Zähler der Division.
- ***divisor*** Der Divisor oder Nenner der Division

Rückgabe

- INTEGER
- NUMERIC

Bemerkungen

Sie können auch die MOD-Funktion verwenden, um den Rest zurückzugeben.

Siehe auch

- „MOD-Funktion [Nummerisch]“ auf Seite 317

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 2 zurück:

```
SELECT REMAINDER( 5, 3 );
```

REPEAT-Funktion [Zeichenfolge]

Verkettet eine Zeichenfolge in der angegebenen Häufigkeit.

Syntax

REPEAT(*string-expression*, *integer-expression*)

Parameter

- **Zeichenfolgenderausdruck** Die zu wiederholende Zeichenfolge.
- **Ganzzahlausdruck** Die Anzahl, wie oft eine Zeichenfolge wiederholt werden soll. Wenn *Ganzzahlausdruck* negativ ist, wird eine leere Zeichenfolge zurückgegeben.

Rückgabe

- LONG VARCHAR
- LONG NVARCHAR

Bemerkungen

Wenn die tatsächliche Länge der Ergebniszeichenfolge das Maximum für den Rückgabebetyp überschreitet, tritt ein Fehler auf. Das Ergebnis wird auf die maximal zulässige Zeichenfolgenreihe gekürzt.

Das Verhalten dieser Funktion ist identisch mit dem der REPLICATE-Funktion.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- „REPLICATE-Funktion [Zeichenfolge]“ auf Seite 367
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert repeatrepeatrepeat zurück:

```
SELECT REPEAT( 'repeat', 3 );
```

REPLACE-Funktion [Zeichenfolge]

Ersetzt eine Zeichenfolge mit einer anderen Zeichenfolge und gibt das neue Ergebnis zurück.

Syntax

REPLACE(*original-string*, *search-string*, *replace-string*)

Parameter

Wenn ein Argument NULL ist, gibt die Funktion NULL zurück.

- ***original-string*** Die zu durchsuchende Zeichenfolge. Sie kann eine beliebige Länge haben.
- ***search-string*** Die Zeichenfolge, nach der gesucht und die von *replace-string* ersetzt werden soll. Die Zeichenfolge ist auf 255 Byte beschränkt. Wenn *search-string* eine leere Zeichenfolge ist, wird die ursprüngliche Zeichenfolge zurückgegeben.
- ***replace-string*** Die Zeichenfolge, die *search-string* ersetzt. Sie kann eine beliebige Länge haben. Wenn *replace-string* eine leere Zeichenfolge ist, wird jedes Auftreten von *search-string* gelöscht.

Rückgabe

- LONG VARCHAR
- LONG NVARCHAR

Bemerkungen

Diese Funktion ersetzt alle gefundenen Zeichenfolgen.

Vergleiche berücksichtigen die Groß-/Kleinschreibung, wenn die Datenbank auf die Berücksichtigung von Groß-/Kleinschreibung eingestellt ist.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- „SUBSTRING-Funktion [Zeichenfolge]“ auf Seite 401
- „CHARINDEX-Funktion [Zeichenfolge]“ auf Seite 189
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert `xx.def.xx.ghi` zurück:

```
SELECT REPLACE( 'abc.def.abc.ghi', 'abc', 'xx' );
```

Die folgende Anweisung generiert eine Ergebnismenge, die ALTER PROCEDURE-Anweisungen enthält, welche bei der Ausführung gespeicherte Prozeduren ausbessern würde, die sich wiederum auf eine Tabelle beziehen, die umbenannt wurde. (Um nutzbar zu sein, muss der Tabellename eindeutig sein.)

```
SELECT REPLACE(
    REPLACE( proc_defn, 'OldTableName', 'NewTableName' ),
    'CREATE PROCEDURE',
    'ALTER PROCEDURE' )
```

```
FROM SYS.SYSPROCEDURE  
WHERE proc_defn LIKE '%OldTableName%';
```

REPLICATE-Funktion [Zeichenfolge]

Verkettet eine Zeichenfolge in der angegebenen Häufigkeit.

Syntax

REPLICATE(*string-expression*, *integer-expression*)

Parameter

- **Zeichenfolgenausdruck** Die zu wiederholende Zeichenfolge.
- **Ganzzahlausdruck** Die Anzahl, wie oft eine Zeichenfolge wiederholt werden soll.

Rückgabe

- LONG VARCHAR
- LONG NVARCHAR

Bemerkungen

Wenn die tatsächliche Länge der Ergebniszeichenfolge das Maximum für den Rückgabebetyp überschreitet, tritt ein Fehler auf. Das Ergebnis wird auf die maximal zulässige Zeichenfolgenreihe gekürzt.

Das Verhalten dieser Funktion ist identisch mit dem der REPEAT-Funktion.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- „REPEAT-Funktion [Zeichenfolge]“ auf Seite 365
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert repeatrepeatrepeat zurück:

```
SELECT REPLICATE( 'repeat', 3 );
```

REVERSE-Funktion [Zeichenfolge]

Gibt die Umkehrung eines Zeichenausdrucks zurück.

Syntax

REVERSE(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die umzukehrende Zeichenfolge

Rückgabe

- LONG VARCHAR
- LONG NVARCHAR

Bemerkungen

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert cba zurück:

```
SELECT REVERSE( 'abc' );
```

REWRITE-Funktion [Verschiedene]

Gibt eine neu geschriebene SELECT-, DELETE- oder UPDATE-SQL-Anweisung aus.

Syntax

```
REWRITE( select-statement [, 'ANSI' ] )
```

Parameter

- ***select-statement*** Die SQL-Anweisung, an der die Neuschreibungsoptimierungen angewendet werden, um die Ergebnisse der Funktion zu generieren

Rückgabe

LONG VARCHAR

Bemerkungen

Sie können die REWRITE-Funktion ohne ANSI-Argument verwenden, um besser zu verstehen, wie der Optimierer den Zugriffsplan für eine gegebene Abfrage erstellt hat. Insbesondere können Sie herausfinden, wie SQL Anywhere die Bedingungen in den Klauseln WHERE, ON und HAVING der Anweisung neu geschrieben wurden, und anschließend ermitteln, ob anwendbare Indizes vorhanden sind, die zur Verbesserung der Anforderungsausführungszeit genutzt werden können.

Die Anweisung, die von REWRITE zurückgegeben wird, passt unter Umständen nicht zu der Semantik der ursprünglichen Anweisung. Dies ist darauf zurückzuführen, dass einige Neuschreibungsoptimierungen interne Mechanismen einbauen, die nicht direkt in SQL übersetzt werden

können. Beispielsweise kann die vom Server erfolgte Verwendung der Zeilenbezeichner zur Eliminierung von Duplikaten nicht in SQL übersetzt werden.

Die überarbeitete Abfrage der REWRITE-Funktion ist nicht für eine Ausführung ausgelegt. Es handelt sich um ein Tool zur Analyse von Performance-Fragen durch Aufzeigen der Vorkommnisse, die nach der Neuschreibungsphase an den Optimierer weitergeleitet werden.

Es gibt einige Neuschreibungsoptimierungen, die in der Ausgabe von REWRITE nicht wiedergegeben werden. Diese umfassen LIKE-Optimierungen, Optimierungen für Mindest- und Maximalfunktionen, Eliminierung von Ober- und Untergrenzwerten sowie die Einordnung von Prädikaten.

Wenn ANSI angegeben wird, gibt REWRITE das ANSI-Äquivalent der Anweisung zurück. In diesem Fall werden nur die folgenden Neuschreibungsoptimierungen angewendet:

- Transact-SQL Outer-Joins werden als ANSI SQL Outer-Joins neu geschrieben.
- Doppelte Korrelationsnamen werden gelöscht.
- KEY- und NATURAL-Outer-Joins werden als ANSI SQL-Joins neu geschrieben.

Siehe auch

- „Optimierungen während der Abfrageverarbeitung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „extended_join_syntax-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Transact-SQL-Outer-Joins (*= oder =*)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Schlüssel-Joins“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Natürliche Joins“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Doppelte Korrelationsnamen in Joins (Stern-Joins)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In der folgenden Anweisung werden zwei REWRITE-Optimierungen für eine Abfrage ausgeführt. Die erste ist die Entschachtelung der Unterabfrage in einen Join zwischen den Tabellen "Employees" und "SalesOrders". Die zweite Optimierung vereinfacht die Abfrage durch Löschen des Primärschlüssel- und Fremdschlüssel-Joins zwischen den Tabellen "Employees" und "SalesOrders". Teil der Neuschreibungsoptimierung ist das Ersetzen des Join-Prädikats e.EmployeeID=s.SalesRepresentative durch s.SalesRepresentative IS NOT NULL.

```
SELECT REWRITE( 'SELECT s.ID, s.OrderDate
FROM GROUP0.SalesOrders s
WHERE EXISTS ( SELECT *
FROM GROUP0.Employees e
WHERE e.EmployeeID = s.SalesRepresentative)' ) FROM dummy;
```

Die Abfrage gibt eine einzige Spalten-Ergebnismenge zurück, die die neugeschriebene Abfrage enthält:

```
'SELECT s.ID, s.OrderDate FROM GROUP0.SalesOrders s WHERE
s.SalesRepresentative IS NOT NULL'
```

Die nächste REWRITE-Anweisung verwendet das ANSI-Argument:

```
SELECT REWRITE( 'SELECT DISTINCT s.ID, s.OrderDate, e.GivenName, e.EmployeeID
FROM GROUPO.SalesOrders s, GROUPO.Employees e
WHERE e.EmployeeID != s.SalesRepresentative', 'ANSI' ) FROM dummy;
```

Das Ergebnis entspricht dem ANSI-Äquivalent der Anweisung. In diesem Fall wird der Transact-SQL Outer-Join in einen ANSI-Outer-Join konvertiert. Die Abfrage gibt eine einzige Spalten-Ergebnismenge zurück (zur besseren Lesbarkeit aufgeteilt in mehrere Zeilen):

```
'SELECT DISTINCT s.ID, s.OrderDate, e.GivenName, e.EmployeeID
FROM GROUPO.Employees as e
LEFT OUTER JOIN GROUPO.SalesOrders as s
ON e.EmployeeID = s.SalesRepresentative';
```

RIGHT-Funktion [Zeichenfolge]

Gibt die äußersten rechten Zeichen einer Zeichenfolge zurück.

Syntax

RIGHT(*string-expression*, *integer-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, für die die äußersten rechten Zeichen zurückgegeben werden.
- **Ganzzahlausdruck** Die Anzahl der Zeichen am Ende der Zeichenfolge, die zurückgegeben werden sollen

Rückgabe

- LONG VARCHAR
- LONG NVARCHAR

Bemerkungen

Wenn eine Zeichenfolge Mehrbytezeichen enthält, ist die Anzahl der zurückgegebenen Bytes möglicherweise größer als die angegebene Anzahl von Zeichen.

Sie können einen *Ganzzahlausdruck* angeben, der größer ist als der Wert in der Spalte. In diesem Fall wird der gesamte Wert zurückgegeben.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben. Wenn in der Eingabezeichenfolge Zeichenlängensemantik verwendet wurde, wird der Rückgabewert soweit wie möglich mit Ausdrücken der Zeichenlängensemantik beschrieben.

Siehe auch

- „LEFT-Funktion [Zeichenfolge]“ auf Seite 300
- „Internationale Sprachen und Zeichensätze“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die letzten 5 Zeichen jedes Surname-Werts in der Tabelle "Customers" zurück:

```
SELECT RIGHT( Surname, 5 ) FROM GROUPO.Customers;
```

ROUND-Funktion [Numerisch]

Rundet *Numerischer_Ausdruck* auf die Anzahl von Stellen hinter dem Komma, die in *Ganzzahlausdruck* angegeben ist.

Syntax

ROUND(*numeric-expression*, *integer-expression*)

Parameter

- **Numerischer_Ausdruck** Die zu rundende Zahl, die an die Funktion übergeben wurde.
- **Ganzzahlausdruck** Eine positive Ganzzahl bestimmt die Anzahl von signifikanten Stellen rechts vom Dezimalzeichen für die Rundung. Eine negative Ganzzahl bestimmt die Anzahl von signifikanten Stellen links vom Dezimalzeichen für die Rundung.

Rückgabe

NUMERIC

Bemerkungen

Das Ergebnis dieser Funktion ist entweder NUMERIC oder DOUBLE. Wenn es ein numerisches Ergebnis gibt und *Ganzzahlausdruck* ein negativer Wert ist, wird die Gesamtstellenzahl um 1 erhöht.

Siehe auch

- „[TRUNCNUM-Funktion \[Numerisch\]](#)“ auf Seite 417

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 123,200 zurück:

```
SELECT ROUND( 123.234, 1 );
```

ROW-Konstruktor [zusammengesetzt]

Gibt eine Sequenz von (*Feldname Datentyp*,...)-Paaren namens **Felder** zurück.

Syntax 1

ROW(*expression* [, *expression* ...])

Syntax 2

ROW(*single-row-query-expression*)

Parameter

- ***expression*** Ein Ausdruck, der ein einzelnes Feld darstellt.
- ***Einzeiliger_Abfrageausdruck*** Eine Abfrage, die ein einzelnes Feld zurückgibt.

Rückgabe

Feldwert

Bemerkungen

Alle Felder werden auf NULL initialisiert und bleiben NULL, bis ein Wert implizit oder explizit innerhalb eines bestimmten Felds platziert wird.

ROW-Typen können weder in der äußersten SELECT-Liste einer Ansichtsdefinition angegeben werden noch in SELECT-Anweisungen oder Abfrageausdrücken der obersten Ebene, die an den Client zurückgegeben werden. ROW-Typen können nicht als Spalten in einer Basistabelle oder temporären Tabelle gespeichert werden.

Ein Zeilentyp kann beliebige Verschachtelungsebenen enthalten, sodass sich komplexe strukturierte Typen ergeben. Jeder Zeilensubtyp erhält einen Namen. Diese Namen können mit Punktnotation referenziert werden, um bestimmte Werte zu referenzieren. Beispiel:

```
DECLARE sample ROW( x INT, w ROW( y INT, z INT ) );
SET Sample = ROW( 3, ROW( 6,7 ) );
SELECT (Sample).w.y FROM dummy;
```

Punktnotation lässt zu, dass andere Ausdrücke in demselben oder einem verschachtelten Abfrageblock einen Zeilentyp oder Teile davon referenzieren, um Vergleiche auszuführen oder andere ROW-Typen zu initialisieren.

Wenn Sie Felder in Spalten mit einem qualifizierten Namen referenzieren, setzen Sie Spaltennamen in Klammern. Im folgenden Beispiel wird die myrowcolumn-Spalte durch den Namen der abgeleiteten Tabelle qualifiziert:

```
SELECT ( myderivedtable.myrowcolumn ).id FROM
  (SELECT ROW( id, name )
   FROM GROUPO.Product )
  AS myderivedtable( myrowcolumn );
```

Siehe auch

- „Zusammengesetzte Datentypen“ auf Seite 136
- „Vergleiche von zusammengesetzten Typen“ auf Seite 145
- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung veranschaulicht, wie Sie einen ROW-Typ konstruieren können, der Strukturtypen von Produktdaten für jedes Produkt in der Tabelle "Products" enthält.

```
SELECT ROW( ID, NAME, DESCRIPTION ) AS pInfo FROM GROUP0.Products;
```

Der pInfo-Zeilentyp enthält Elemente namens ID, NAME und DESCRIPTION, die er aus den Attributnamen in der Tabelle "Products" entnimmt.

Die folgende Anweisung veranschaulicht, wie Sie die CAST-Funktion verwenden können, um explizite Namen einem impliziten ROW-Typ zuzuordnen:

```
SELECT CAST( ROW( 9, 'Tee Shirt', 'My tee shirt' )
  AS ROW(ProductID INTEGER,
        ProductName CHAR(25),
        ProductDescription CHAR(35)
  )
  AS pInfo FROM dummy;
```

SQL-Ausdrücke, die dieses Abfrageergebnis als Eingabe verwenden, können die Komponenten der pInfo-Zeile mit Ausdrücken in Punktnotation nach Namen referenzieren. ROW-Typen können auch mit einem einzeiligen Abfrageausdruck konstruiert werden.

Die folgende Anweisung veranschaulicht eine alternative Möglichkeit zum Konstruieren eines ROW-Objekts:

```
SELECT ROW( SELECT Name, Quantity FROM Products WHERE ID = 300 );
```

Die folgende Anweisung veranschaulicht, wie Sie ein Feld, last_name, in einer Zeile, student, nach Name festlegen können:

```
SET student.last_name = 'Johnson';
```

ROW_NUMBER-Funktion [Verschiedene]

Ordnet jeder Zeile eine eindeutige Nummer zu. Verwenden Sie diese Funktion statt der NUMBER-Funktion.

Syntax

```
ROW_NUMBER( ) OVER ( window-spec )
```

Fensterspezifikation: Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Rückgabe

INTEGER

Bemerkungen

Elemente von *Fensterspezifikation* können entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Bei Verwendung als Fensterfunktion müssen Sie eine ORDER BY-Klausel angeben und dürfen eine PARTITION BY-Klausel angeben, können jedoch keine ROWS- oder RANGE-Klausel angeben. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „Fensterfunktionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Siehe auch

- „WINDOW-Klausel“ auf Seite 1124
- „NUMBER-Funktion [Verschiedene]“ auf Seite 331
- „RANK-Funktion [Rangfolge]“ auf Seite 346
- „ROWID-Funktion [Verschiedene]“ auf Seite 375

Standards und Kompatibilität

- **SQL/2008** ROW_NUMBER ist Teil der optionalen SQL/2008-Sprachenfunktion T611, "Elementare OLAP-Vorgänge".

Beispiel

Die folgende Anweisung gibt eine Ergebnismenge zurück, die für jeden Mitarbeiter in New York und Utah eine eindeutige Zeilennummer liefert. Da die Abfrage anhand des Gehalts (salary) in absteigender Sortierfolge sortiert ist, erhält der Mitarbeiter mit dem höchsten Gehalt in der Datenmenge die erste Zeilennummer. Obwohl zwei Mitarbeiter die gleichen Gehälter beziehen, entsteht keine Gleichwertigkeit, weil den beiden Mitarbeitern eindeutige Zeilennummern zugeordnet sind.

```
SELECT Surname, Salary, State,  
       ROW_NUMBER() OVER (ORDER BY Salary DESC) "Rank"  
FROM GROUPO.Employees WHERE State IN ('NY','UT');
```

Surname	Salary	State	Rank
Shishov	72995.000	UT	1
Wang	68400.000	UT	2
Cobb	62000.000	UT	3
Morris	61300.000	UT	4
Davidson	57090.000	NY	5
Martel	55700.000	NY	6

Surname	Salary	State	Rank
Blaikie	54900.000	NY	7
Diaz	54900.000	NY	8
Driscoll	48023.690	UT	9
Hildebrand	45829.000	UT	10
...
Lynch	24903.000	UT	19

ROWID-Funktion [Verschiedene]

Gibt einen UNSIGNED BIGINT-Wert zurück, der eine Zeile in einer Tabelle eindeutig identifiziert.

Syntax

ROWID(*correlation-name*)

Parameter

- **Korrelationsname** Der Korrelationsname einer in der Abfrage verwendeten Tabelle. Der Korrelationsname sollte sich auf eine Basistabelle, eine temporäre Tabelle, eine globale temporäre Tabelle oder eine Proxytabelle (nur zulässig, wenn der Proxyserver eine ähnliche Funktion unterstützt) beziehen. Das Argument der ROWID-Funktion sollte sich nicht auf eine Ansicht, eine abgeleitete Tabelle, einen allgemeinen Tabellenausdruck oder eine Prozedur beziehen.

Rückgabe

UNSIGNED BIGINT

Bemerkungen

Gibt den Zeilenbezeichner der Zeile in der Tabelle zurück, der dem angegebenen Korrelationsnamen entspricht.

Der von der Funktion zurückgegebene Wert ist zwischen Abfragen nicht notwendigerweise konstant, weil auf der Datenbank durchgeführte Vorgänge zu Änderungen bei den Zeilenbezeichnern einer Tabelle führen können. Besonders die REORGANIZE TABLE-Anweisung kann zu Änderungen bei Zeilenbezeichnern führen. Zusätzlich können Zeilenbezeichner erneut verwendet werden, wenn eine Zeile gelöscht wurde. Daher sollten es Benutzer vermeiden, die ROWID-Funktion im Normalbetrieb zu verwenden, und stattdessen den Primärschlüsselwert zum Abruf verwenden. Es wird empfohlen, ROWID nur für Diagnosezwecke einzusetzen.

Auch wenn das Ergebnis dieser Funktion ein UNSIGNED BIGINT-Typ ist, haben die Ergebnisse von arithmetischen Operationen keine besondere Bedeutung. So dürfen Sie beispielsweise nicht erwarten, wenn Sie einem Zeilenbezeichner 1 hinzufügen, dass Sie den Zeilenbezeichner der nächsten Zeile

erhalten. Überdies sind im Zusammenhang mit ROWID nur IN- und Gleichheitsprädikate als Suchargument nutzbar. Nötigenfalls werden ROWID betreffende Prädikate, wie z.B. ROWID(T) = *literal* dazu verwendet, um in einen 64-Bit-Wert vom Typ UNSIGNED INTEGER konvertiert zu werden. Wenn die Konvertierung nicht durchgeführt werden kann, tritt eine Datenausnahme auf. Wenn der Wert von *Literal* ein ungültiger Zeilenbezeichner ist, ergibt das Vergleichsprädikat FALSE.

Die ROWID-Funktion kann nicht in einer CHECK-Integritätsregel auf einer Tabelle oder Spalte verwendet werden, auch kann sie nicht im COMPUTE-Ausdruck für eine berechnete Spalte verwendet werden.

Siehe auch

- „ROW_NUMBER-Funktion [Verschiedene]“ auf Seite 373

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Zeilenbezeichner der Zeile in der Employees-Tabelle zurück, bei der 'ID' gleich 105 ist:

```
SELECT ROWID( Employees ) FROM GROUPO.Employees WHERE Employees.EmployeeID = 105;
```

Die folgende Anweisung gibt eine Liste der Sperren auf Zeilen in der Employees-Tabelle zusammen mit dem Inhalt dieser Zeilen zurück:

```
SELECT *  
FROM sa_locks() S JOIN GROUPO.Employees WITH( NOLOCK )  
ON ROWID( Employees ) = S.row_identifier  
WHERE S.table_name = 'Employees';
```

RTRIM-Funktion [Zeichenfolge]

Entfernt nachgestellte Leerzeichen aus der Zeichenfolge.

Syntax

RTRIM(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die zu kürzende Zeichenfolge

Rückgabe

- VARCHAR
- NVARCHAR
- LONG VARCHAR
- LONG NVARCHAR

Bemerkungen

Die tatsächliche Länge des Ergebnisses ist die Länge des Ausdrucks minus der Anzahl der entfernten Zeichen. Wenn alle Zeichen entfernt werden, ist das Ergebnis eine leere Zeichenfolge.

Wenn das Argument NULL ist, ist das Ergebnis NULL.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- „TRIM-Funktion [Zeichenfolge]“ auf Seite 415
- „LTRIM-Funktion [Zeichenfolge]“ auf Seite 309
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Die TRIM-Angaben, die durch den SQL/2008-Standard (LEADING und TRAILING) definiert sind, werden von den SQL Anywhere-Funktionen LTRIM bzw. RTRIM geliefert.

Beispiel

Die folgende Anweisung gibt den Wert `Test Message` zurück, wobei alle nachgestellten Leerzeichen entfernt werden:

```
SELECT RTRIM( 'Test Message      ' );
```

SECOND-Funktion [Datum und Uhrzeit]

Gibt den Sekundenwert des TIMESTAMP-Arguments zurück.

Syntax

SECOND(*timestamp-expression*)

Parameter

- **Zeitstempelausdruck** Der TIMESTAMP-Wert.

Rückgabe

SMALLINT

Bemerkungen

Gibt eine Zahl von 0 bis 59 zurück, die der Sekundenkomponente des angegebenen TIMESTAMP-Arguments entspricht.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung liefert den Wert 25.

```
SELECT SECOND( '1998-07-13 21:21:25' );
```

SECONDS-Funktion [Datum und Uhrzeit]

Bearbeitet einen TIMESTAMP-Wert oder gibt die Anzahl von Sekunden zwischen zwei TIMESTAMP-Werten zurück. Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Syntax 1

SECONDS(*timestamp-expression*)

Syntax 2

SECONDS(*timestamp-expression*, *timestamp-expression*)

Syntax 3

SECONDS(*time-or-timestamp-expression*, *integer-expression*)

Parameter

- **Zeitstempelausdruck** Ein TIMESTAMP-Wert.
- **Zeit-_oder_Zeitstempelausdruck** Ein Wert vom Typ TIME oder TIMESTAMP.
- **Ganzzahlausdruck** Die Anzahl der Sekunden, die zu *Zeit-_oder_Zeitstempelausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl an Sekunden von *Zeit-_oder_Zeitstempelausdruck* abgezogen. Wenn Sie einen Ganzzahlausdruck angeben, muss *Zeitstempelausdruck* explizit als Datentyp TIME, DATE oder TIMESTAMP festgelegt sein. Wenn *Zeit-_oder_Zeitstempelausdruck* vom Typ DATE ist, wird die Zeitangabe als Mitternacht angenommen.

Rückgabe

UNSIGNED BIGINT bei Syntax 1.

SIGNED BIGINT bei Syntax 2.

TIME oder TIMESTAMP bei Syntax 3.

Bemerkungen

Das Ergebnis der SECONDS-Funktion hängt von deren Argumenten ab.

- **Syntax 1** Wenn Sie einen einzelnen *Zeitstempelausdruck* an die SECONDS-Funktion übergeben, wird die Anzahl der Sekunden zwischen Mitternacht am 0000-02-29 und *Zeitstempelausdruck* als UNSIGNED BIGINT-Wert zurückgegeben.

Hinweis

"0000-02" gibt kein tatsächliches Datum wieder. Es ist das von der SECONDS-Funktion verwendete Standarddatum.

- **Syntax 2** Wenn Sie zwei TIMESTAMP-Werte an die SECONDS-Funktion übergeben, liefert die Funktion die als Ganzzahl ausgedrückte Anzahl der Sekunden zwischen den beiden.
- **Syntax 3** Wenn Sie einen TIMESTAMP-Wert und eine Ganzzahl an die SECONDS-Funktion übergeben, liefert die Funktion das TIMESTAMP-Ergebnis nach dem Addieren der durch die Ganzzahl ausgedrückten Anzahl an Sekunden zum *Zeit- oder Zeitstempelausdruck*-Argument. Wenn Sie einen TIME-Wert an die SECONDS-Funktion übergeben, liefert die Funktion entsprechend einen Wert vom Typ TIME.

Verwenden Sie statt Syntax 2 die DATEDIFF-Funktion. Verwenden Sie statt Syntax 3 die DATEADD-Funktion.

Siehe auch

- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186
- „DATEADD-Funktion [Datum und Uhrzeit]“ auf Seite 217
- „DATEDIFF-Funktion [Datum und Uhrzeit]“ auf Seite 218

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgenden Anweisungen geben den Wert "14400" zurück, womit angezeigt wird, dass der zweite TIMESTAMP-Wert 14400 Sekunden nach dem ersten liegt.

```
SELECT SECONDS( '1999-07-13 06:07:12',
               '1999-07-13 10:07:12' );
SELECT DATEDIFF( second,
               '1999-07-13 06:07:12',
               '1999-07-13 10:07:12' );
```

Die folgende Anweisung liefert den Wert 63062431632.

```
SELECT SECONDS( '1998-07-13 06:07:12' );
```

Die folgenden Anweisungen geben den TIMESTAMP-Wert "1999-05-12 21:05:12.000" zurück.

```
SELECT SECONDS( CAST( '1999-05-12 21:05:07' AS TIMESTAMP ), 5);
SELECT DATEADD( second, 5, '1999-05-12 21:05:07' );
```

SET_BIT-Funktion [Bit-Array]

Legt den Wert eines bestimmten Bits in einem Bit-Array fest.

Syntax

```
SET_BIT( [ bit-expression, ]bit-position [, value ] )
```

Parameter

- **Bit-Ausdruck** Das Bit-Array, in dem das Bit geändert werden soll.
- **bit-position** Die Position des zu setzenden Bits. Es muss sich um eine Ganzzahl ohne Vorzeichen handeln.
- **value** Der Wert, auf den das Bit gesetzt wird

Rückgabe

LONG VARBIT

Bemerkungen

Der Standardwert von *Bit-Ausdruck* ist ein Bit-Array der Länge *Bit-Position*, das alle auf 0 (FALSE) gesetzten Bits enthält.

Der Standardwert von *value* ist '1' (TRUE).

Das Ergebnis ist NULL, wenn ein Parameter NULL ist.

Die Positionen im Array werden von links mit 1 beginnend abgezählt.

Siehe auch

- [„GET_BIT-Funktion \[Bit-Array\]“ auf Seite 266](#)
- [„SET_BITS-Funktion \[Aggregat\]“ auf Seite 381](#)
- [„INTEGER-Datentyp“ auf Seite 109](#)
- [„Bit-Operatoren“ auf Seite 20](#)
- [„sa_get_bits-Systemprozedur“ auf Seite 1221](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert "00100011" zurück:

```
SELECT SET_BIT( '00110011', 4 , 0);
```

Die folgende Anweisung gibt den Wert "00111011" zurück:

```
SELECT SET_BIT( '00110011', 5 , 1);
```

Die folgende Anweisung gibt den Wert "00111011" zurück:

```
SELECT SET_BIT( '00110011', 5 );
```

Die folgende Anweisung gibt den Wert "00001" zurück:

```
SELECT SET_BIT( 5 );
```

SET_BITS-Funktion [Aggregat]

Erstellt ein Bit-Array, bei dem spezifische Bits, die Werten einer Reihe von Zeilen entsprechen, auf "1" (TRUE) gesetzt werden.

Syntax

SET_BITS(*expression*)

Parameter

- **expression** Der Ausdruck, der verwendet wird, um zu bestimmen, welches Bit auf "1" zu setzen ist. Das ist üblicherweise ein Spaltenname.

Rückgabe

LONG VARBIT

Bemerkungen

Zeilen, bei denen die angegebenen Werte NULL sind, werden ignoriert.

Wenn es keine Zeilen gibt, wird NULL zurückgegeben.

Die Länge des Ergebnisses ist die höchste Position, die auf "1" gesetzt wurde.

Die SET_BITS-Funktion ist äquivalent zu, aber schneller als die folgende Anweisung:

```
SELECT BIT_OR( SET_BIT( expression ) )
FROM table;
```

Siehe auch

- „Bit-Operatoren“ auf Seite 20
- „GET_BIT-Funktion [Bit-Array]“ auf Seite 266
- „SET_BIT-Funktion [Bit-Array]“ auf Seite 379
- „sa_get_bits-Systemprozedur“ auf Seite 1221

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt ein Bit-Array zurück, bei dem das 2., das 5. und das 10. Bit auf "1" (oder "0100100001") gesetzt sind:

```
CREATE TABLE t( r INTEGER );
INSERT INTO t values( 2 );
INSERT INTO t values( 5 );
INSERT INTO t values(10 );
SELECT SET_BITS( r ) FROM t;
```

SIGN-Funktion [Numerisch]

Gibt das Vorzeichen (positiv und negativ) für die angegebene Zahl.

Syntax

SIGN(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Die Zahl, deren Vorzeichen zurückgegeben werden soll. *Numerischer_Ausdruck* kann vom Typ INTEGER, DOUBLE oder NUMERIC sein.

Rückgabe

SMALLINT

Bemerkungen

Bei negativen Zahlen gibt die SIGN-Funktion "-1" zurück.

Bei Null gibt die SIGN-Funktion "0" zurück.

Bei positiven Zahlen gibt die SIGN-Funktion "1" zurück.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert "-1" zurück:

```
SELECT SIGN( -550 );
```

SIMILAR-Funktion [Zeichenfolge]

Gibt eine Zahl zurück, die die Ähnlichkeit zwischen zwei Zeichenfolgen angibt.

Syntax

SIMILAR(*string-expression-1*, *string-expression-2*)

Parameter

- **Zeichenfolgenausdruck_1** Die erste Zeichenfolge, die verglichen werden soll.
- **Zeichenfolgenausdruck_2** Die zweite Zeichenfolge, die verglichen werden soll

Rückgabe

SMALLINT

Bemerkungen

Die Funktion gibt eine Ganzzahl zwischen 0 und 100 zurück, die die Ähnlichkeit zwischen den beiden Zeichenfolgen repräsentiert. Das Ergebnis kann als Prozent der Zeichen interpretiert werden, die bei beiden Zeichenfolgen übereinstimmen. Ein Wert von 100 Prozent bedeutet, dass die beiden Zeichenfolgen identisch sind.

Diese Funktion kann zum Korrigieren einer Namensliste verwendet werden (wie zum Beispiel Kunden). Einige Kunden könnten zu der Liste mehrmals mit leicht unterschiedlichen Namen hinzugefügt worden sein. Sie können mit der SIMILAR-Funktion nach ähnlichen Kundennamen suchen, indem Sie die customer-Tabelle mit sich selbst verknüpfen und einen Bericht über alle Ähnlichkeiten erzeugen, die größer sind als 90 Prozent, aber kleiner als 100 Prozent.

Die bei der SIMILAR-Funktion durchgeführte Berechnung ist komplexer als nur die Anzahl der übereinstimmenden Zeichen.

Siehe auch

- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert "75" zurück, was angibt, dass zwei Werte einander zu 75% ähnlich sind.

```
SELECT SIMILAR( 'toast', 'coast' );
```

SIN-Funktion [Nummerisch]

Gibt den Sinus einer Zahl zurück.

Syntax

SIN(*numeric-expression*)

Parameter

- **Nummerischer_Ausdruck** Der Winkel im Bogenmaß.

Rückgabe

DOUBLE

Bemerkungen

Die SIN-Funktion gibt den Sinus des Arguments zurück, wobei das Argument ein Winkel im Bogenmaß ist. Die SIN- und ASIN-Funktionen sind inverse Vorgänge.

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück.

Siehe auch

- [„ASIN-Funktion \[Nummerisch\]“ auf Seite 173](#)
- [„COS-Funktion \[Nummerisch\]“ auf Seite 204](#)
- [„COT-Funktion \[Nummerisch\]“ auf Seite 204](#)
- [„TAN-Funktion \[Nummerisch\]“ auf Seite 407](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den SIN-Wert "0,52" zurück:

```
SELECT SIN( 0.52 );
```

SOAP_HEADER-Funktion [SOAP]

Gibt einen SOAP-Headereintrag oder einen Attributwert für einen Headereintrag der SOAP-Anforderung zurück.

Syntax

SOAP_HEADER(*header-key* [, *index*, *header-attribute*])

Parameter

- **Headerschlüssel** Dieser VARCHAR-Parameter gibt den lokalen XML-Namen des obersten XML-Elements für den angegebenen SOAP-Headereintrag an.
- **index** Dieser optionale INTEGER-Parameter unterscheidet zwischen SOAP-Headerfeldern, die identische Namen haben. Dies kann auftreten, wenn mehrere Headereinträge oberste XML-Elemente mit demselben localname-Eintrag haben. Üblicherweise haben solche Elemente eindeutige Namespaces.
- **Headerattribut** Dieser optionale VARCHAR-Parameter kann jeden Attributknoten in einem Element des Headereintrags angeben, und zwar:
 - **@namespace** Ein spezielles SQL Anywhere-Attribut, das zum Zugriff auf den Namespace des angegebenen Headereintrags verwendet wird.
 - **mustUnderstand** Ein Attribut des SOAP 1.1 Headereintrags, das angibt, ob ein Headereintrag obligatorisch oder optional für die Verarbeitung durch den Empfänger ist.
 - **encodingStyle** Ein Attribut des SOAP 1.1 Headereintrags, das den Kodierungsstil angibt. Auf dieses Attribut kann zugegriffen werden, es wird aber von SQL Anywhere nicht intern verwendet.
 - **actor** Ein Attribut des SOAP 1.1 Headereintrags, das den vorhergesehenen Empfänger eines Headereintrags angibt, indem der URL des Empfängers angegeben wird

Rückgabe

LONG VARCHAR

Bemerkungen

Diese Funktion kann mit einem einzigen Parameter *Headerschlüssel* verwendet werden, um einen Headereintrag zurückzugeben. Ein Headereintrag ist eine XML-Zeichenfolgendarstellung eines Elements und seiner Unterelemente, die in einem SOAP-Header enthalten sind.

Diese Funktion kann auch verwendet werden, um ein Attribut eines Headereintrags zu extrahieren, indem die optionalen Parameter *Index* und *Headerattribut* angegeben werden.

Diese Funktion gibt den Wert des benannten SOAP-Headerfelds zurück, oder NULL, wenn sie nicht von einem SOAP-Dienst aufgerufen wurde. Sie wird verwendet, wenn eine SOAP-Anforderung über einen Webdienst verarbeitet wird.

Wenn ein Header für einen angegebenen *Headerschlüssel* nicht existiert, ist der Rückgabewert NULL.

Siehe auch

- „NEXT_SOAP_HEADER-Funktion [SOAP]“ auf Seite 328
- „sa_set_soap_header-Systemprozedur“ auf Seite 1329
- „Praktische Einführung: Verwenden von SQL Anywhere für den Zugriff auf einen SOAP/DISH-Dienst“ [*SQL Anywhere Server - Programmierung*]
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Bei Verwendung des folgenden Beispiels in einer gespeicherten Prozedur, die von einem HTTP-Webdienst aufgerufen wird, werden alle im SOAP-Anforderungsheader enthaltenen Schlüssel verarbeitet. Wenn die Funktion den **Authentication**-Schlüssel verarbeitet, ruft sie auch den Wert des Schlüssels ab.

```
BEGIN
  DECLARE hd_key LONG VARCHAR;
  DECLARE hd_entry LONG VARCHAR;
  header_loop:
  LOOP
    SET hd_key = NEXT_SOAP_HEADER( hd_key );
    IF hd_key IS NULL THEN
      -- no more header entries
      LEAVE header_loop;
    END IF;
    IF hd_key = 'Authentication' THEN
      SET hd_entry = SOAP_HEADER( hd_key );
    END IF;
  END LOOP header_loop;
END;
```

SORTKEY-Funktion [Zeichenfolge]

Generiert Sortierschlüsselwerte. Das sind Werte, die benutzt werden können, um Zeichenfolgen basierend auf anderen Kollationsregeln zu sortieren.

Syntax

```
SORTKEY( string-expression
[, { collation-id
| collation-name [ ( collation-tailoring-string ) ] } ]
)
```

Parameter

- **Zeichenfolgenausdruck** Der Zeichenfolgenausdruck muss Zeichen enthalten, die im Zeichensatz der Datenbank kodiert sind.

Wenn *Zeichenfolgenausdruck* eine leere Zeichenfolge ist, gibt die SORTKEY-Funktion einen Binärwert mit Nulllänge zurück. Wenn *Zeichenfolgenausdruck* NULL ist, gibt die SORTKEY-Funktion einen NULL-Wert zurück. Eine leere Zeichenfolge hat einen anderen Sortierfolgenwert als eine NULL-Zeichenfolge aus einer Datenbankspalte.

Die Maximallänge einer Zeichenfolge, die von der SORTKEY-Funktion verarbeitet werden kann, ist 254 Byte. Danach kommende Zeichen werden ignoriert.

- **Kollationsname** Eine Zeichenfolge oder eine Zeichenvariable, die den Namen der zu verwenden Sortierfolge angibt. Sie können auch den Alias `char_collation` bzw. das Äquivalent `db_collation` angeben, um Sortierschlüssel zu generieren, wie sie die von der Datenbank verwendete CHAR-Kollation verwendet. Gleichermäßen können Sie den Alias `nchar_collation` angeben, um Sortierschlüssel zu generieren, wie sie die von der Datenbank verwendete NCHAR-Kollation verwendet.
- **Kollations-ID** Eine Variable, Ganzzahlkonstante oder Zeichenfolge, die den ID-Wert der zu verwendenden Sortierfolge angibt. Wenn Sie keine *Kollations-ID* angeben, ist der Standardwert "Default Unicode Multilingual".
- **Zeichenfolge_Kollationsanpassung** Optional können Sie Optionen der Kollationsanpassung (*Zeichenfolge_Kollationsanpassung*) für eine zusätzliche Steuerung bei Zeichensortierungen und -vergleichen angeben. Diese Optionen werden in der Form von Schlüsselwort=Wert-Paaren, die in Klammern gesetzt werden, hinter dem Kollationsnamen angegeben. Zum Beispiel: `'UCA(locale=es;case=LowerFirst;accent=respect)'`. Die Syntax zur Angabe dieser Optionen ist identisch mit der für die COLLATION-Klausel der CREATE DATABASE-Anweisung definierten Syntax.

Hinweis

Alle Optionen der Kollationsanpassung werden unterstützt, wenn die UCA-Kollation angegeben ist. Bei allen anderen Kollationen wird nur die Kollationsanpassungsoption für Groß-/Kleinschreibung unterstützt.

Rückgabe

BINARY

Bemerkungen

Die SORTKEY-Funktion generiert Werte, die benutzt werden können, um Ergebnisse nach einem vordefinierten Sortierreihenfolgenverhalten zu sortieren. Damit können Sie mit einem Sortierfolgeverhalten arbeiten, das möglicherweise der Datenbankskollation nicht zur Verfügung steht. Der zurückgegebene Wert ist ein Binärwert, der kodierte Sortierfolgeninformationen für die Eingabezeichenfolge aus der SORTKEY-Funktion enthält. Beispiel: Sie können die von der SORTKEY-Funktion zurückgegebenen Werte in einer Spalte mit der Quell-Zeichenfolge speichern. Wenn Sie die Zeichendaten in der gewünschten Reihenfolge abrufen wollen, braucht die SELECT-Anweisung nur eine ORDER BY-Klausel für die Spalten, in denen die Ergebnisse der SORTKEY-Funktion gespeichert sind.

Die SORTKEY-Funktion gewährleistet, dass die von ihr für gegebene Sortierfolgenkriterien zurückgegebenen Werte für die binären Vergleiche herangezogen werden können, die bei VARBINARY-Datentypen durchgeführt werden.

Das Generieren von Sortierschlüsseln für Abfragen kann kostenträchtig sein. Sie können alternativ für häufig angeforderte Sortierschlüssel eine berechnete Spalte zum Speichern des Sortierschlüssels erstellen, um anschließend diese Spalte in der ORDER BY-Klausel der Abfrage zu referenzieren.

Die Eingabedaten der SORTKEY-Funktion können bis zu sechs Byte von Sortierinformationen für jedes Eingabezeichen erzeugen. Die Ausgabedaten der SORTKEY-Funktion haben den Datentyp VARBINARY und eine Maximallänge von 1024 Byte.

Wenn Sie während der Erstellung des Sortierungsschlüssels UCA für die Kollation angeben, verwendet die Kollationsanpassung die Berücksichtigung von Akzenten und der Groß- und Kleinschreibung.

Beispiel: Wenn UCA selbst angegeben wird, entspricht die Anpassung der Kollatierung 'UCA(case=UpperFirst;accent=Respect;punct=Primary)'.

Wenn im zweiten Parameter von SORTKEY andere Kollatierungsanpassungen angegeben werden, heben diese Einstellungen die Standardeinstellungen auf. Die folgenden Anweisungen sind beispielsweise gleichwertig:

```
SELECT SORTKEY( 'abc', 'UCA(accent=Ignore)' );
SELECT SORTKEY( 'abc', 'UCA(case=UpperFirst;accent=Ignore;punct=Primary)' );
```

Wenn eine Nicht-UCA-Kollation angegeben wird, entspricht die Kollationsanpassung ebenfalls der Berücksichtigung von Akzenten und der Groß- und Kleinschreibung. Bei Nicht-UCA-Kollationen kann aber nur die Berücksichtigung von Groß- und Kleinschreibung durch Kollationsanpassung außer Kraft gesetzt werden. Beispiel:

```
SELECT SORTKEY( 'abc', '1252LATIN1(case=Respect)' );
```

Wenn die Datenbank ohne die Angabe von Optionen der Kollationsanpassung erstellt wurde (z.B. dbinit -c -zn uca -dba DBA,sql mydb.db), generieren die folgenden zwei Klauseln möglicherweise unterschiedliche Sortierreihenfolgen, auch wenn der Name der Datenbankkollation in der SORTKEY-Funktion angegeben wird:

```
ORDER BY string-expression
ORDER BY SORTKEY( string-expression, database-collation-name )
```

Dies liegt daran, dass sich die Standardkollatierungseinstellungen, die für die Datenbankerstellung und die SORTKEY-Funktion verwendet wurden, voneinander unterscheiden. Um von SORTKEY dasselbe Verhalten wie bei der Datenbankkollation zu erhalten, geben Sie entweder eine Kollationsanpassungssyntax für *Zeichenfolge_Kollationsanpassung* an, die den Einstellungen für die Datenbankkollation entspricht, oder geben Sie db_collation für *Kollationsname* an. Beispiel:

```
SORTKEY( expression, 'db_collation' )
```

Hinweis

Sortierungsschlüsselwerte werden je nach der Version von SQL Anywhere auf verschiedene Weise generiert. Dies kann zu Problemen mit der Sortierung führen, wenn Sortierungsschlüsselwerte, die von einer Version von SQL Anywhere erstellt wurden, in einer Datenbank verwendet werden, die von einer anderen Version von SQL Anywhere erstellt wurde. Sie sollten Sortierschlüsselwerte neu generieren, wenn Probleme mit der Sortierung auftreten.

Außerdem sollten Sie Sortierschlüsselwerte neu generieren, wenn Sie ein Upgrade Ihrer Datenbank durch Entladen und Neuladen ausführen.

Siehe auch

- „sort_collation-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Alternative Kollationen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Optionen der Kollationsanpassung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Empfohlene Zeichensätze und Kollationen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „COMPARE-Funktion [Zeichenfolge]“ auf Seite 192
- „CREATE DATABASE-Anweisung“ auf Seite 583
- „Internationale Sprachen und Zeichensätze“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgenden Anweisungen fragen die Tabelle "Employees" ab und geben den Vornamen (FirstName) und Nachnamen (Surname) von allen Mitarbeitern zurück, und zwar sortiert anhand der Sortierschlüsselwerte für die Spalte "Surname", wobei die dict-Kollation (Latin-1, Englisch, Französisch, Deutsch lexikalisch) verwendet wird.

```
SELECT Surname, GivenName FROM GROUPO.Employees ORDER BY SORTKEY( Surname, 'dict' );
```

Das folgende Beispiel gibt den Sortierschlüsselwert für abc zurück, wobei die UCA-Kollation und Optionen der Kollationsanpassung verwendet werden.

```
SELECT SORTKEY( 'abc', 'UCA(locale=es;case=LowerFirst;accent=respect)' );
```

SOUNDEX-Funktion [Zeichenfolge]

Gibt eine Zahl zurück, die den Klang der Zeichenfolge darstellt.

Syntax

SOUNDEX(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die auszuwertende Zeichenfolge

Rückgabe

SMALLINT

Bemerkungen

Der Wert der SOUNDEX-Funktion für eine Zeichenfolge basiert auf dem ersten Buchstaben und den nächsten drei Konsonanten außer H, Y und W. Vokale in *Zeichenfolgenausdruck* werden ignoriert, sofern sie nicht der erste Buchstabe der Zeichenfolge sind. Doppelte Buchstaben werden als ein Buchstabe gezählt. Beispiel: Das Wort "Apples" basiert auf den Buchstaben A, P, L und S.

Mehrbytezeichen werden von der SOUNDEX-Funktion ignoriert.

Wenn auch nicht ganz fehlerfrei, so gibt die SOUNDEX-Funktion üblicherweise dieselbe Zahl für Wörter zurück, die ähnlich klingen und die mit demselben Buchstaben beginnen.

Die SOUNDEX-Funktion funktioniert grundsätzlich nur mit englischen Wörtern. Für andere Sprachen ist sie nur beschränkt sinnvoll.

Siehe auch

- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt zwei identische Zahlen (3827) zurück, die den Klang der einzelnen Namen darstellen.

```
SELECT SOUNDEX( 'Smith' ), SOUNDEX( 'Smythe' );
```

SPACE-Funktion [Zeichenfolge]

Gibt eine angegebene Anzahl von Leerstellen zurück.

Syntax

SPACE(*integer-expression*)

Parameter

- **Ganzzahlausdruck** Die Anzahl von Leerstellen, die zurückgegeben werden sollen

Rückgabe

LONG VARCHAR

Bemerkungen

Wenn *Ganzzahlausdruck* negativ ist, wird eine Nullzeichenfolge zurückgegeben.

Siehe auch

- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt eine Zeichenfolge zurück, die 10 Leerstellen enthält.

```
SELECT SPACE( 10 );
```

SQLDIALECT-Funktion [Verschiedene]

Gibt "Watcom-SQL" oder "Transact-SQL" zurück und gibt damit an, in welchem Dialekt eine SQL-Anweisung geschrieben ist.

Syntax

SQLDIALECT(*sql-statement-string*)

Parameter

- ***sql-statement-string*** Die SQL-Anweisung, die neugeschrieben werden soll.

Rückgabe

LONG VARCHAR

Siehe auch

- [„TRANSACTSQL-Funktion \[Verschiedene\]“ auf Seite 413](#)
- [„WATCOMSQL-Funktion \[Verschiedene\]“ auf Seite 430](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die Zeichenfolge Transact-SQL zurück.

```
SELECT  
  SQLDIALECT( 'SELECT EmployeeName = Surname FROM GROUPO.Employees' )  
FROM dummy;
```

SQLFLAGGER-Funktion [Verschiedene]

Gibt die Übereinstimmung einer gegebenen SQL-Anweisung mit einem angegebenen Standard zurück.

Syntax

SQLFLAGGER(*sql-standard-string*, *sql-statement-string*)

Parameter

- **sql-standard-string** Die Standardebene, anhand derer die Übereinstimmung getestet wird. Die möglichen Werte sind dieselben wie bei der Datenbankoption `sql_flagger_error_level`:
 - **SQL:2008/Core** Test auf Übereinstimmung mit Kern-SQL/2008-Syntax
 - **SQL:2008/Package** Test auf Übereinstimmung mit vollständiger SQL/2008-Syntax
 - **SQL:2003/Core** Test auf Übereinstimmung mit Kern-SQL/2003-Syntax
 - **SQL:2003/Package** Test auf Übereinstimmung mit vollständiger SQL/2003-Syntax
 - **SQL:1999/Core** Test auf Übereinstimmung mit Kern-SQL/1999-Syntax
 - **SQL:1999/Package** Test auf Übereinstimmung mit vollständiger SQL/1999-Syntax
 - **SQL:1992/Entry** Test auf Übereinstimmung mit Entry-Level-SQL/1992-Syntax
 - **SQL:1992/Intermediate** Test auf Übereinstimmung mit Intermediate-Level-SQL/1992-Syntax
 - **SQL:1992/Full** Test auf Übereinstimmung mit vollständiger SQL/1992-Syntax
 - **UltraLite** Test auf Übereinstimmung mit UltraLite
- **SQL-Anweisung-Zeichenfolge** Die auf Übereinstimmung zu prüfende SQL-Anweisung

Rückgabe

LONG VARCHAR

Siehe auch

- „sql_flagger_error_level-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Der SQL-Präprozessor“ [[SQL Anywhere Server - Programmierung](#)]
- „sa_ansi_standard_packages-Systemprozedur“ auf Seite 1168
- „Testen der SQL-Konformität mit dem SQL Flagger“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung zeigt ein Beispiel für die Meldung, die zurückgegeben wird, wenn eine unzulässige Erweiterung gefunden wird:

```
SELECT SQLFLAGGER(
  'SQL:2003/Package', 'SELECT top 1 dummy_col FROM sys.dummy ORDER BY
  dummy_col' );
```

Diese Anweisung gibt folgende Meldung zurück: '0AW03 Unzulässige Spracherweiterung in Syntax bei 'top' auf Zeile 1 erkannt'.

Die folgende Anweisung gibt '00000' zurück, weil sie keine unzulässigen Erweiterungen enthält:

```
SELECT SQLFLAGGER( 'SQL:2003/Package', 'SELECT dummy_col FROM sys.dummy' );
```

SQRT-Funktion [Numerisch]

Gibt die Quadratwurzel einer Zahl zurück.

Syntax

SQRT(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Die Zahl, für die die Quadratwurzel berechnet werden soll

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltgenauem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück.

Standards und Kompatibilität

- **SQL/2008** Die SQRT-Funktion umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions".

Beispiel

Die folgende Anweisung liefert den Wert 3.

```
SELECT SQRT( 9 );
```

STACK_TRACE-Funktion [Verschiedene]

Gibt Informationen zum Stack-Trace für die aktuelle Anweisung zurück.

Syntax

STACK_TRACE()

Rückgabe

LONG VARCHAR-Wert, der das Stack-Trace für die aktuelle Anweisung darstellt.

Bemerkungen

Jede Zeile des Rückgabewerts enthält den qualifizierten Prozedurnamen oder Batch-Typ, gefolgt von der Zeilennummer der Anweisung. Die letzte Zeile des zurückgegebenen Werts endet nicht mit einer Zeilenendmarke. Die erste Zeile des Stack-Trace stellt die Zeile dar, in der die Funktion aufgerufen

wurde. Prozeduren, die aus verborgene Prozeduren aufgerufen wurden, geben im Stack-Trace **<hidden>** zurück und nicht den Prozedurnamen.

Diese Funktion gibt dieselben Informationen zurück wie die `sa_stack_trace`-Systemprozedur.

Siehe auch

- „TRY-Anweisung“ auf Seite 1093
- „BEGIN-Anweisung“ auf Seite 557
- „ERROR_LINE-Funktion [Verschiedene]“ auf Seite 244
- „ERROR_MESSAGE-Funktion [Verschiedene]“ auf Seite 245
- „ERROR_PROCEDURE-Funktion [Funktionstyp]“ auf Seite 246
- „ERROR_SQLCODE-Funktion [Verschiedene]“ auf Seite 247
- „ERROR_SQLSTATE-Funktion [Verschiedene]“ auf Seite 248
- „sa_error_stack_trace-Systemprozedur“ auf Seite 1216
- „sa_stack_trace-Systemprozedur“ auf Seite 1335
- „Verschachtelte zusammengesetzte Anweisungen und Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

STDDEV-Funktion [Aggregat]

Ein Alias für `STDDEV_SAMP`.

Siehe auch

- „STDDEV_SAMP-Funktion [Aggregat]“ auf Seite 395

STDDEV_POP-Funktion [Aggregat]

Berechnet die Standardabweichung einer Population, die aus einem numerischen Ausdruck besteht, als `DOUBLE`.

Syntax 1

STDDEV_POP(*numeric-expression*)

Syntax 2

STDDEV_POP(*numeric-expression*) **OVER** (*window-spec*)

window-spec : see Syntax 2 instructions in the Remarks section below

Parameter

- **Numerischer_Ausdruck** Der Ausdruck, dessen Standardabweichung auf Populationsbasis über eine Zeilenmenge berechnet wird. Der Ausdruck ist normalerweise ein Spaltenname.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelte genaue Gleitkommazahl durch.

Die Standardabweichung auf Populationsbasis (s) wird nach der folgenden Formel berechnet:

$$s = [(1/N) * \text{SUM}(x_i - \text{mean}(x))^2]^{1/2}$$

Diese Standardabweichung umfasst keine Zeilen, in denen *Nummerischer_Ausdruck* NULL ist. Er gibt NULL zurück für eine Gruppe, die keine Zeilen enthält.

Weitere Hinweise zur durchgeführten statistischen Berechnung finden Sie unter „[Mathematische Formeln für die Aggregatfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „[Fensterfunktionen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „[Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „[WINDOW-Klausel](#)“ auf Seite 1124
- „[Aggregatfunktionen](#)“ auf Seite 153

Standards und Kompatibilität

- **SQL/2008** Die STDDEV_POP-Funktion umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions". Wenn STDDEV_POP als Fensterfunktion verwendet wird, umfasst sie einen Teil der optionalen SQL-Sprachenfunktion T611, "Elementare OLAP-Vorgänge".

Die Möglichkeit, DISTINCT über einen Ausdruck anzugeben, der keine Spaltenreferenz ist, umfasst einen Teil der optionalen SQL-Sprachenfunktion F561, "Full value expressions". SQL Anywhere unterstützt auch die SQL/2008-Sprachenfunktion F441, "Extended set function support", die zulässt, dass die Operanden von Aggregatfunktionen beliebige Ausdrücke sein können, möglicherweise einschließlich äußerer Referenzen zu Ausdrücken in anderen Abfrageblöcken, die keine Spaltenreferenzen sind.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der STDDEV_POP-

Funktion als auch eine äußere Referenz enthalten. Ein Beispiel finden Sie unter „AVG-Funktion [Aggregat]“ [\[UltraLite - Datenbankverwaltung\]](#).

Beispiel

Die folgende Anweisung listet den Durchschnitt und die Varianz in der Anzahl der Elemente pro Auftrag in verschiedenen Zeitabschnitten auf:

```
SELECT YEAR( ShipDate ) AS Year,
       QUARTER( ShipDate ) AS Quarter,
       AVG( Quantity ) AS Average,
       STDDEV_POP( quantity ) AS Variance
FROM GROUPO.SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

Year	Quarter	Average	Variance
2000	1	25.775148	14.2794...
2000	2	27.050847	15.0270...
...

STDDEV_SAMP-Funktion [Aggregat]

Berechnet die Standardabweichung eines Musters, das aus einem numerischen Ausdruck besteht, als DOUBLE.

Syntax 1

STDDEV_SAMP(*numeric-expression*)

Syntax 2

STDDEV_SAMP(*numeric-expression*) **OVER** (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Numerischer_Ausdruck** Der Ausdruck, dessen Standardabweichung auf Musterbasis über eine Zeilenmenge berechnet wird. Der Ausdruck ist normalerweise ein Spaltenname.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelte genaue Gleitkommazahl durch.

Die Standardabweichung (s) wird nach der folgenden Formel berechnet, die von einer Normalverteilung ausgeht:

$$s = [(1/(N - 1)) * \text{SUM}(x_i - \text{mean}(x))^2]^{1/2}$$

Diese Standardabweichung umfasst keine Zeilen, in denen *Nummerischer_Ausdruck* NULL ist. Es wird NULL für eine Gruppe zurückgegeben, die entweder '0' oder '1' enthält.

Weitere Hinweise zur durchgeführten statistischen Berechnung finden Sie unter [„Mathematische Formeln für die Aggregatfunktionen“ \[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#).

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter [„Fensterfunktionen“ \[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#).

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter [„Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW“ \[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#).

Siehe auch

- [„WINDOW-Klausel“ auf Seite 1124](#)
- [„Aggregatfunktionen“ auf Seite 153](#)

Standards und Kompatibilität

- **SQL/2008** Die STDDEV_SAMP-Funktion umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions". Wenn STDDEV_SAMP als Fensterfunktion verwendet wird, umfasst sie einen Teil der optionalen SQL-Sprachenfunktion T611, "Elementare OLAP-Vorgänge".

Die Möglichkeit, DISTINCT über einen Ausdruck anzugeben, der keine Spaltenreferenz ist, umfasst einen Teil der optionalen SQL-Sprachenfunktion F561, "Full value expressions". SQL Anywhere unterstützt auch die SQL/2008-Sprachenfunktion F441, "Extended set function support", die zulässt, dass die Operanden von Aggregatfunktionen beliebige Ausdrücke sein können, möglicherweise einschließlich äußerer Referenzen zu Ausdrücken in anderen Abfrageblöcken, die keine Spaltenreferenzen sind.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der STDDEV_SAMP-Funktion als auch eine äußere Referenz enthalten. Ein Beispiel finden Sie unter [„AVG-Funktion \[Aggregat\]“ auf Seite 175](#).

Beispiel

Die folgende Anweisung listet den Durchschnitt und die Varianz in der Anzahl der Elemente pro Auftrag in verschiedenen Zeitabschnitten auf:

```

SELECT YEAR( ShipDate ) AS Year,
       QUARTER( ShipDate ) AS Quarter,
       AVG( Quantity ) AS Average,
       STDDEV_SAMP( quantity ) AS Variance
FROM GROUPO.SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;

```

Year	Quarter	Average	Variance
2000	1	25.775148	14.3218...
2000	2	27.050847	15.0696...
...

STR-Funktion [Zeichenfolge]

Gibt die einer Zahl entsprechende Zeichenfolge zurück.

Syntax

STR(*numeric-expression* [, *length* [, *decimal*]])

Parameter

- **Nummerischer_Ausdruck** Jeder angenähert numerische Ausdruck (float, real, double precision) zwischen -1E126 und 1E127.
- **length** Die Anzahl der zurückzugebenden Zeichen (einschließlich dem Dezimalzeichen, aller Stellen rechts und links vom Dezimalzeichen sowie Leerzeichen). Standardwert ist "10".
- **decimal** Die Anzahl der zurückzugebenden Dezimalstellen. Standardwert ist "0".

Rückgabe

VARCHAR

Bemerkungen

Wenn der Ganzzahlteil der Zahl nicht in die angegebene Länge passt, dann ist das Ergebnis eine Zeichenfolge mit der angegebenen Länge, die nur Sternchen enthält. Die folgende Anweisung liefert beispielsweise "***".

```
SELECT STR( 1234.56, 3 );
```

Hinweis

Die maximale unterstützte Länge ist 128. Jede Länge, die nicht zwischen 1 und 128 liegt, ergibt ein Ergebnis von NULL.

Siehe auch

- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt eine Zeichenfolge von sechs Leerstellen, gefolgt von "1235" zurück, bei einer Gesamtsumme von zehn Zeichen:

```
SELECT STR( 1234.56 );
```

Die folgende Anweisung liefert das Ergebnis "1234,6":

```
SELECT STR( 1234.56, 6, 1 );
```

STRING-Funktion [Zeichenfolge]

Verkettet eine oder mehrere Zeichenfolgen in eine große Zeichenfolge.

Syntax

STRING(*string-expression* [, ...])

Parameter

- **Zeichenfolgenderausdruck** Die auszuwertende Zeichenfolge

Wenn nur ein Argument angegeben ist, wird es in einen einzelnen Ausdruck konvertiert. Wenn mehrere Argumente übergeben werden, werden sie zu einer einzelnen Zeichenfolge verkettet.

Rückgabe

- LONG VARCHAR
- LONG NVARCHAR
- LONG BINARY

Bemerkungen

Nummerische oder Datumsparameter werden vor der Verkettung in Zeichenfolgen konvertiert. Mit der STRING-Funktion kann auch ein einzelner Ausdruck in eine Zeichenfolge konvertiert werden, indem dieser Ausdruck als einziger Parameter angegeben wird.

Wenn alle Parameter NULL sind, gibt STRING NULL zurück. Wenn irgend welche Parameter nicht NULL sind, werden alle NULL-Parameter als leere Zeichenfolgen behandelt.

Siehe auch

- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung liefert den Wert "testing123".

```
SELECT STRING( 'testing', NULL, 123 );
```

STRTOUUID-Funktion [Zeichenfolge]

Konvertiert einen Zeichenfolgenwert in den Wert eines eindeutigen Bezeichners (UUID oder GUID).

Hinweis

In einer Datenbank, die vor Version 9.0.2 erstellt wurde, ist der UNIQUEIDENTIFIER-Datentyp als ein benutzerdefinierter Datentyp festgelegt, und es werden die STRTOUUID- und UUIDTOSTR-Funktionen benötigt, um zwischen binären und Zeichenfolgen-Darstellungen von UUID-Werten zu konvertieren.

In Datenbanken, die mit Version 9.0.2 oder später erstellt wurden, ist der UNIQUEIDENTIFIER-Datentyp ein nativer Datentyp und SQL Anywhere führt Konvertierungen je nach Bedarf durch. Verwenden Sie bei diesen Versionen STRTOUUID und UUIDTOSTR nicht.

Weitere Hinweise finden Sie unter „[UNIQUEIDENTIFIER-Datentyp](#)“ auf Seite 134.

Syntax

STRTOUUID(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Eine Zeichenfolge im Format `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`

Rückgabe

UNIQUEIDENTIFIER

Bemerkungen

Konvertiert eine Zeichenfolge des Formats `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`, wobei hier *x* für eine hexadezimale Ziffer steht, in einen eindeutig identifizierenden Wert.

Wenn die Zeichenfolge keine gültige UUID-Zeichenfolge ist, wird ein Konvertierungsfehler zurückgegeben, außer die `conversion_error`-Option ist auf 'OFF' gesetzt. In diesem Fall wird NULL zurückgegeben.

Diese Funktion ist nützlich für das Einfügen von UUID-Werten in eine Datenbank.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Geschweifte Klammern können als erstes und letztes Zeichen in *Zeichenfolgenausdruck* verwendet werden.

Siehe auch

- „[UUIDTOSTR-Funktion \[Zeichenfolge\]](#)“ auf Seite 424
- „[NEWID-Funktion \[Verschiedene\]](#)“ auf Seite 321
- „[Zeichenfolgenfunktionen](#)“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgenden Anweisungen sind gleichwertig und liefern das Ergebnis 0x6c2b64a93c6f47dc901536b9ed49fec2.

```
SELECT STRTOUUID( '6c2b64a9-3c6f-47dc-9015-36b9ed49fec2' );  
SELECT STRTOUUID( '{6c2b64a9-3c6f-47dc-9015-36b9ed49fec2}' );
```

STUFF-Funktion [Zeichenfolge]

Löscht mehrere Zeichen aus einer Zeichenfolge und ersetzt sie durch eine andere Zeichenfolge.

Syntax

STUFF(*string-expression-1*, *start*, *length*, *string-expression-2*)

Parameter

- **Zeichenfolgenausdruck_1** Die Zeichenfolge, die durch die STUFF-Funktion geändert wird.
- **start** Die Zeichenposition, an der mit dem Löschen der Zeichen begonnen wird. Das erste Zeichen in der Zeichenfolge hat die Position 1.
- **length** Die Anzahl der zu löschenden Zeichen.
- **Zeichenfolgenausdruck_2** Die einzufügende Zeichenfolge. Zum Löschen eines Teils einer Zeichenfolge mithilfe der STUFF-Funktion verwenden Sie eine Ersetzungszeichenfolge von NULL.

Rückgabe

LONG NVARCHAR

Bemerkungen

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- „[INSERTSTR-Funktion \[Zeichenfolge\]](#)“ auf Seite 291
- „[Zeichenfolgenfunktionen](#)“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert "chocolate pie" zurück:

```
SELECT STUFF( 'chocolate cake', 11, 4, 'pie' );
```

SUBSTRING-Funktion [Zeichenfolge]

Gibt eine Teilzeichenfolge einer Zeichenfolge zurück.

Syntax

```
{ SUBSTRING | SUBSTR } ( string-expression, start  
[, length] )
```

Parameter

- **Zeichenfolgenderausdruck** Die Zeichenfolge, aus der eine Teilzeichenfolge zurückgegeben werden soll.
- **start** Die Anfangsposition der zurückzugebenden Teilzeichenfolge, in Zeichen.
- **length** Die Länge der zurückzugebenden Teilzeichenfolge, in Zeichen. Wenn *length* angegeben ist, ist die Teilzeichenfolge auf diese Länge begrenzt.

Rückgabe

- LONG BINARY
- LONG VARCHAR
- LONG NVARCHAR

Bemerkungen

Das Verhalten der Funktion hängt von der Einstellung der ansi_substring-Datenbankoption ab. Wenn die ansi_substring-Option auf ON gesetzt ist (Standardwert), entspricht das Verhalten der SUBSTRING-Funktion dem ANSI/ISO SQL/2008-Verhalten. Das Verhalten ist wie folgt:

ansi_substring-Optionseinstellung	Startwert	Längenwert
On	Das erste Zeichen in der Zeichenfolge ist auf Position 1. Ein negatives oder ein Null-Start-Offset wird so behandelt, als ob die Zeichenfolge links mit Nicht-Zeichen angefüllt ist.	Eine positive <i>length</i> gibt an, dass die Teilzeichenfolge <i>length</i> Zeichen rechts von der Startposition endet. Eine negative <i>length</i> gibt einen Fehler zurück.

ansi_substring-Optionseinstellung	Startwert	Längenwert
Off	<p>Das erste Zeichen in der Zeichenfolge hat die Position 1. Eine negative Startposition legt eine Anzahl von Zeichen vom Ende anstatt vom Anfang der Zeichenfolge aus fest.</p> <p>Wenn <i>start</i> Null und die Länge nicht-negativ ist, wird ein Start-Wert von "1" verwendet. Wenn <i>Start</i> Null und <i>length</i> negativ ist, wird ein Start-Wert von "-1" verwendet.</p>	<p>Eine positive <i>length</i> gibt an, dass die Teilzeichenfolge <i>length</i> Zeichen rechts von der Startposition endet.</p> <p>Eine negative <i>length</i> gibt maximal <i>length</i> Zeichen links von der Startposition bis zur und einschließlich der Startposition zurück.</p>

Wenn *Zeichenfolgendruck* ein binärer Datentyp ist, verhält sich die SUBSTRING-Funktion wie BYTE_SUBSTR.

Um Zeichen am Ende einer Zeichenfolge zu erhalten, benutzen Sie die Funktion RIGHT.

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben. Wenn in der Eingabezeichenfolge Zeichenlängensemantik verwendet wurde, wird der Rückgabewert soweit wie möglich mit Ausdrücken der Zeichenlängensemantik beschrieben.

Siehe auch

- „[BYTE_SUBSTR-Funktion \[Zeichenfolge\]](#)“ auf Seite 184
- „[LEFT-Funktion \[Zeichenfolge\]](#)“ auf Seite 300
- „[RIGHT-Funktion \[Zeichenfolge\]](#)“ auf Seite 370
- „[CHARINDEX-Funktion \[Zeichenfolge\]](#)“ auf Seite 189
- „[Zeichenfolgenfunktionen](#)“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** SUBSTRING ist eine Kernfunktion des SQL/2008-Standards. Die Implementierung im Standard unterscheidet sich etwas von der SQL Anywhere-Implementierung: Im Standard ist SUBSTRING mit drei Parametern unter Verwendung der Schlüsselwörter FROM und FOR definiert, die beide in SQL Anywhere nicht erforderlich sind.

Beispiel

Die folgende Tabelle zeigt die Werte, die von der SUBSTRING-Funktion zurückgegeben werden.

Beispiel	Ergebnis
SUBSTRING('front yard', 1, 4)	fron
SUBSTRING('back yard', 6, 4)	yard
SUBSTR('abcdefgh', 0, -2)	Gibt einen Fehler zurück, wenn ansi_substring "On" ist

Beispiel	Ergebnis
SUBSTR('abcdefgh', -2, 2)	Gibt eine leere Zeichenfolge zurück, wenn ansi_substring "On" ist

SUM-Funktion [Aggregat]

Gibt die Gesamtsumme des angegebenen Ausdrucks für jede Zeilengruppe zurück

Syntax 1

SUM([ALL | DISTINCT] *expression*)

Syntax 2

SUM([ALL] *expression*) **OVER** (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **[ALL] *expression*** Der Name des Ausdrucks, dessen Summe gebildet werden soll. Das ist üblicherweise ein Spaltenname.
- **DISTINCT *expression*** Berechnet die Summe der eindeutigen Werte von *expression* in jeder Gruppe.

Rückgabe

- INTEGER
- DOUBLE
- NUMERIC

Bemerkungen

Zeilen, in denen der angegebene Ausdruck NULL ist, werden nicht mit eingeschlossen.

Gibt NULL für eine Gruppe zurück, die keine Zeilen enthält.

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „Fensterfunktionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Diese Funktion kann einen Überlauffehler erzeugen, was dazu führt, dass ein Fehler zurückgegeben wird. Sie können die CAST-Funktion auf *Nummerischer_Ausdruck* anwenden, um den Überlauffehler zu vermeiden. Siehe „[CAST-Funktion \[Datentypkonvertierung\]](#)“ [*UltraLite - Datenbankverwaltung*].

Siehe auch

- „[WINDOW-Klausel](#)“ auf Seite 1124
- „[COUNT-Funktion \[Aggregat\]](#)“ auf Seite 205
- „[AVG-Funktion \[Aggregat\]](#)“ auf Seite 175

Standards und Kompatibilität

- **SQL/2008** Kernfunktion. Wenn SUM als Fensterfunktion verwendet wird (Syntax 2), umfasst sie einen Teil der optionalen SQL/2008-Sprachenfunktion T611, "Grundlegende OLAP-Vorgänge".

Die Möglichkeit, DISTINCT über einen Ausdruck anzugeben, der keine Spaltenreferenz ist, umfasst einen Teil der optionalen SQL-Sprachenfunktion F561, "Full value expressions". SQL Anywhere unterstützt auch die SQL/2008-Sprachenfunktion F441, "Extended set function support", die zulässt, dass die Operanden von Aggregatfunktionen beliebige Ausdrücke sein können, möglicherweise einschließlich äußerer Referenzen zu Ausdrücken in anderen Abfrageblöcken, die keine Spaltenreferenzen sind.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der SUM-Funktion als auch eine äußere Referenz enthalten. Siehe „[Fehlerbehandlung: Aggregatfunktionen und äußere Referenzen](#)“ [*SQL Anywhere 16 - Änderungen und Upgrades*].

Ein Beispiel finden Sie unter „[AVG-Funktion \[Aggregat\]](#)“ [*UltraLite - Datenbankverwaltung*].

Beispiel

Die folgende Anweisung liefert den Wert 3749146.740.

```
SELECT SUM( Salary )  
FROM GROUPO.Employees;
```

SUSER_ID-Funktion [System]

Gibt die numerische Benutzer-ID für den angegebenen Benutzernamen zurück.

Syntax

```
SUSER_ID( [ user-name ] )
```

Parameter

- **Benutzername** Der Benutzername für die Benutzer-ID, die Sie suchen.

Rückgabe

VARCHAR

Bemerkungen

Wenn Sie *Benutzername* nicht angeben, wird die ID des aktuellen Benutzers zurückgegeben.

Diese Funktion wird aus Gründen der Kompatibilität mit anderen Herstellern bereitgestellt. Sie können auch die `USER_ID`-Funktion verwenden, die genau dasselbe bewirkt.

Siehe auch

- „`SUSER_NAME`-Funktion [System]“ auf Seite 405
- „`USER_ID`-Funktion [System]“ auf Seite 423

Standards und Kompatibilität

- **SQL/2008** Transact-SQL-Erweiterung.

Beispiel

Die folgende Anweisung gibt die ID für den GROUPO-Nutzer zurück.

```
SELECT SUSER_ID( 'GROUPO' );
```

SUSER_NAME-Funktion [System]

Gibt den Benutzernamen für die angegebene Benutzer-ID zurück.

Syntax

```
SUSER_NAME( [ user-id ] )
```

Parameter

- ***user-id*** Die Benutzer-ID des Benutzers, nach dem Sie suchen.

Rückgabe

VARCHAR

Bemerkungen

Wenn Sie keine *user-id* angeben, wird der Name des aktuellen Benutzers zurückgegeben.

Diese Funktion wird aus Gründen der Kompatibilität mit anderen Herstellern bereitgestellt. Sie können auch die `USER_NAME`-Funktion verwenden, die genau dasselbe bewirkt.

Siehe auch

- „`SUSER_ID`-Funktion [System]“ auf Seite 404
- „`USER_NAME`-Funktion [System]“ auf Seite 423

Standards und Kompatibilität

- **SQL/2008** Transact-SQL-Erweiterung.

Beispiel

Die folgende Anweisung gibt den Benutzernamen mit der ID 101 zurück.

```
SELECT SUSER_NAME( 101 );
```

SWITCHOFFSET-Funktion [Datum und Zeit]

Gibt einen TIMESTAMP WITH TIME ZONE-Wert zurück, der vom ursprünglichen Zeitzone-Offset in den angegebenen Zeitzone-Offset umgewandelt wurde.

Syntax

SWITCHOFFSET(*tmz-expression*, *time-zone-offset*)

Parameter

- **TMZ-Ausdruck** Der TIMESTAMP WITH TIME ZONE-Wert, der konvertiert werden soll.
- **time-zone-offset** Der Zeitzone-Offset des Ergebnisses. Der Wert kann eine Ganzzahl sein, welche die Minuten vor oder nach der Coordinated Universal Time (UTC) darstellt, eine Zeichenfolge im Format { + | - } hh:nn oder "Z" für die Zulu-Zeitzone. Die Zulu-Zeitzone ist dieselbe Zeitzone wie UTC.

Rückgabe

TIMESTAMP WITH TIME ZONE

Siehe auch

- „TIMESTAMP WITH TIME ZONE-Datentyp“ auf Seite 131
- „SYSDATETIMEOFFSET-Funktion [Datum und Zeit]“ auf Seite 406

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird eine Zeitzone-Offsetwert von -04:00 Stunden in -07:00 Stunden geändert. Der zurückgegebene Wert ist 2009-04-03 11:45:12.123-07:00.

```
SELECT CAST ( '2009-04-03 14:45:12.123-04:00' AS datetimeoffset ) AS EDT,  
SWITCHOFFSET( EDT, '-07:00' ) AS PDT;
```

SYSDATETIMEOFFSET-Funktion [Datum und Zeit]

Gibt anhand der Systemuhr die aktuellen Werte für Datum, Zeit und Zeitzone-Offset des Datenbankservers zurück.

Syntax

SYSDATETIMEOFFSET ()

Rückgabe

TIMESTAMP WITH TIME ZONE

Siehe auch

- „TIMESTAMP WITH TIME ZONE-Datentyp“ auf Seite 131
- „SWITCHOFFSET-Funktion [Datum und Zeit]“ auf Seite 406

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel werden die aktuellen Werte für Datum, Zeit und Zeitzonen-Offset des Datenbankservers zurückgegeben.

```
SELECT SYSDATETIMEOFFSET ( );
```

Im folgenden Beispiel wird der SYSDATETIMEOFFSET-Wert in die Zeitzone des Clientcomputers konvertiert.

```
SELECT SWITCHOFFSET ( SYSDATETIMEOFFSET ( ),  
CAST( connection_property ( 'TimeZoneAdjustment' ) AS INT ) );
```

TAN-Funktion [Numerisch]

Gibt den Tangens einer Zahl zurück.

Syntax

TAN(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Ein Winkel im Bogenmaß.

Rückgabe

DOUBLE

Bemerkungen

Die ATAN- und TAN-Funktionen sind inverse Vorgänge.

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltgenauem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück.

Siehe auch

- „COS-Funktion [Numerisch]“ auf Seite 204
- „SIN-Funktion [Numerisch]“ auf Seite 383

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Tangenswert für "0,52" zurück:

```
SELECT TAN( 0.52 );
```

TEXTPTR-Funktion [Text und Bild]

Gibt einen 16-Byte-Binärzeiger auf die angegebene Spalte zurück. Diese Funktion wird nur für die Kompatibilität mit Transact-SQL verfügbar gemacht und ihre Verwendung wird nicht empfohlen.

Syntax

```
TEXTPTR( column-name )
```

Parameter

- **Spaltenname** Der Name einer Spalte mit CHAR-, NCHAR- oder BINARY-Daten.

Rückgabe

BINARY

Bemerkungen

Diese Funktion ist zur Wahrung der Transact-SQL-Kompatibilität mit eingeschlossen.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Embedded SQL-Beispiel verwendet TEXTPTR, um die Spalte Description zu finden, die mit ProductID 500 in der Tabelle MarketingInformation verknüpft ist.

Der Textzeiger wird in der Variablen **txtptr** gespeichert und als Parameter an die READTEXT-Anweisung übergeben, die 55 Byte, beginnend beim Spaltenoffset 181, zurückgibt.

```
EXEC SQL BEGIN DECLARE SECTION;
char          hostvar[100];
EXEC SQL END DECLARE SECTION;

EXEC SQL create variable txtptr binary(16);
EXEC SQL set txtptr =
    ( SELECT textptr(Description)
      FROM GROUPO.MarketingInformation
      WHERE ProductID = '500' );
EXEC SQL PREPARE S1 FROM
    'READTEXT GROUPO.MarketingInformation.Description txtptr 181 55';
EXEC SQL EXECUTE S1 INTO :hostvar;
printf( "hostvar: %s\n", hostvar );
```

READTEXT gibt die folgende Zeichenfolge zurück.

Lightweight 100% organically grown cotton construction.

TO_CHAR-Funktion [Zeichenfolge]

Konvertiert Zeichendaten aus einem unterstützten Zeichensatz in den bei der Datenbank eingestellten CHAR-Zeichensatz.

Syntax

```
TO_CHAR( string-expression [, source-charset-name ] )
```

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, die konvertiert werden soll.
- **Quellzeichensatzname** Der Zeichensatz der Zeichenfolge.

Rückgabe

LONG VARCHAR

Bemerkungen

Wenn *Quellzeichensatzname* angegeben wird, ist diese Funktion äquivalent zu:

```
CAST( CSCONVERT( CAST( string-expression AS BINARY ),
  'db_charset', source-charset-name )
  AS CHAR );
```

Weitere Hinweise zu db_charset finden Sie unter „[CSCONVERT-Funktion \[Zeichenfolge\]](#)“ auf Seite 211.

Wenn *Quellzeichensatzname* nicht angegeben wird, ist diese Funktion äquivalent zu:

```
CAST( string-expression AS CHAR );
```

Siehe auch

- „Empfohlene Zeichensätze und Kollationen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „[CONNECTION_EXTENDED_PROPERTY-Funktion \[Zeichenfolge\]](#)“ auf Seite 197
- „[CSCONVERT-Funktion \[Zeichenfolge\]](#)“ auf Seite 211
- „[NCHAR-Funktion \[Zeichenfolge\]](#)“ auf Seite 321
- „[TO_NCHAR-Funktion \[Zeichenfolge\]](#)“ auf Seite 410
- „[UNICODE-Funktion \[Zeichenfolge\]](#)“ auf Seite 420
- „[UNISTR-Funktion \[Zeichenfolge\]](#)“ auf Seite 421

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Wenn Sie einen BINARY-Wert haben, der Daten im cp850-Zeichensatz enthält, konvertiert die folgende Anweisung die Daten in den CHAR-Zeichensatz und Datentyp:

```
SELECT TO_CHAR( 'cp850_data', 'cp850' );
```

TO_NCHAR-Funktion [Zeichenfolge]

Konvertiert Zeichendaten aus einem unterstützten Zeichensatz in den NCHAR-Zeichensatz.

Syntax

```
TO_NCHAR( string-expression [, source-charset-name ] )
```

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, die konvertiert werden soll.
- **Quellzeichensatzname** Der Zeichensatz der Zeichenfolge.

Rückgabe

LONG NVARCHAR

Bemerkungen

Wenn *Quellzeichensatzname* angegeben wird, ist diese Funktion äquivalent zu:

```
CAST( CSCONVERT( CAST( string-expression AS BINARY ),  
  'nchar_charset', source-charset-name )  
  AS NCHAR );
```

Weitere Hinweise zu `nchar_charset` finden Sie unter „[CSCONVERT-Funktion \[Zeichenfolge\]](#)“ auf Seite 211.

Wenn *Quellzeichensatzname* nicht angegeben wird, ist diese Funktion äquivalent zu:

```
CAST( string-expression AS NCHAR );
```

Siehe auch

- „Empfohlene Zeichensätze und Kollationen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „[CONNECTION_EXTENDED_PROPERTY-Funktion \[Zeichenfolge\]](#)“ auf Seite 197
- „[CSCONVERT-Funktion \[Zeichenfolge\]](#)“ auf Seite 211
- „[NCHAR-Funktion \[Zeichenfolge\]](#)“ auf Seite 321
- „[TO_CHAR-Funktion \[Zeichenfolge\]](#)“ auf Seite 409
- „[UNICODE-Funktion \[Zeichenfolge\]](#)“ auf Seite 420
- „[UNISTR-Funktion \[Zeichenfolge\]](#)“ auf Seite 421

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Wenn Sie einen BINARY-Wert haben, der Daten im cp850-Zeichensatz enthält, konvertiert das folgende Beispiel die Daten in den NCHAR-Zeichensatz und Datentyp:

```
SELECT TO_NCHAR( 'cp850_data', 'cp850' );
```

TODATETIMEOFFSET-Funktion [Datum und Zeit]

Verwendet den angegebenen Zeitzonen-Offset, um einen TIMESTAMP-Wert in einen TIME STAMP WITH TIME ZONE-Wert umzuwandeln.

Syntax

TODATETIMEOFFSET(*timestamp-expression*, *time-zone-offset*)

Parameter

- **Zeitstempelausdruck** Der zu konvertierende TIMESTAMP-Ausdruck.
- **time-zone-offset** Der Zeitzonen-Offset. Der Wert kann eine Ganzzahl sein, welche die Minuten vor oder nach der Coordinated Universal Time (UTC) darstellt, eine VARCHAR-Zeichenfolge in der Form { + | - }hh:nn oder die Zeichenfolge "Z" für die Zulu Time Zone. Die Zulu-Zeitzone ist dieselbe Zeitzone wie UTC.

Rückgabe

TIMESTAMP WITH TIME ZONE

Siehe auch

- „TIMESTAMP WITH TIME ZONE-Datentyp“ auf Seite 131

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird TIMESTAMP-Wert in einen TIMESTAMP WITH TIME ZONE-Wert konvertiert.

```
SELECT CAST('2009-04-03 14:45:12.123' AS TIMESTAMP) AS orig,
       TODATETIMEOFFSET (orig, '+11:00');
```

TODAY-Funktion [Datum und Uhrzeit]

Gibt das aktuelle Datum als DATE-Wert zurück.

Syntax

TODAY([*])

Rückgabe

DATE

Bemerkungen

TODAY(*) und TODAY() sind semantisch gleichwertig. TODAY ist gleichwertig mit dem CURRENT DATE-Spezialwert.

Jede Instanz der TODAY-Funktion in einer Anforderung wird höchstens einmal ausgewertet. Mehrere Instanzen von TODAY in derselben Anforderung liefern möglicherweise identische oder auch unterschiedliche DATE-Werte.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgenden Anweisungen geben den aktuellen Tag gemäß der Systemuhr zurück.

```
SELECT TODAY( * );  
SELECT CURRENT DATE;
```

TRACEBACK-Funktion [Verschiedene]

Gibt Anweisungen im Stack der letzten Ausnahmebedingung (des letzten Fehlers) zurück, die beim Ausführen einer gespeicherten Prozedur, eines Triggers oder einer benutzerdefinierten Funktion aufgetreten ist.

Syntax

TRACEBACK([*])

Rückgabe

LONG VARCHAR

Bemerkungen

Dem zurückgegebenen Aufruf-Stack werden Objektnamen und Zeilennummern hinzugefügt. Anweisungen aus gespeicherten Prozeduren mit HIDDEN-Definitionen werden nicht in den Stack geschrieben.

Die Anweisungen im TRACEBACK-Wert werden aus der Datenbankserver-Darstellung der Anweisungen in den gespeicherten Prozeduren generiert, die möglicherweise nicht genau mit dem Text in der Prozedurdefinition übereinstimmt.

TRACEBACK(*) und TRACEBACK() sind semantisch gleichwertig.

Diese Funktion erleichtert die Fehlersuche in Prozeduren und Triggern, vor allem solche, die im Transact-SQL-Dialekt geschrieben wurden.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Um die TRACEBACK-Funktion zu verwenden, führen Sie die folgende Anweisung aus, nachdem beim Ausführen einer Prozedur ein Fehler aufgetreten ist:

```
SELECT TRACEBACK( * );
```

Eine Ausgabe ähnlich der folgenden wird zurückgegeben, nachdem die TRACEBACK-Funktion verwendet wurde:

```
"user1"."proc1" : 10 : set ret_val = in_val / (in_val - in_val)
"user2"."proc2" : 5 : <hidden>
"user3"."proc1" : 7 : call user2.proc2( 10 )
```

TRACED_PLAN-Funktion [Verschiedene]

Generiert einen grafischen Plan für eine Abfrage unter Verwendung von Protokollierungsdaten in Sybase Central.

Syntax

TRACED_PLAN(*logging_session_id*, *query_id*)

Parameter

- **logging_session_id** Zusammen mit *query_id* identifiziert dieser INTEGER-Parameter eine Zeile aus der sa_diagnostic_query-Tabelle, für die ein Plan erstellt werden soll.
- **query_id** Zusammen mit *logging_session_id* identifiziert dieser INTEGER-Parameter eine Zeile aus der sa_diagnostic_query-Tabelle, für die ein Plan erstellt werden soll.

Rückgabe

LONG VARCHAR

Bemerkungen

Diese Funktion ist für Sybase Central vorgesehen.

Siehe auch

- „sa_diagnostic_query-Tabelle“ auf Seite 1151

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

TRANSACTSQL-Funktion [Verschiedene]

In diesem Beispiel wird eine Watcom-SQL-Anweisung im Transact-SQL-Dialekt neu geschrieben.

Syntax

TRANSACTSQL(*sql-statement-string*)

Parameter

- **sql-statement-string** Die SQL-Anweisung, die in Transact-SQL neu geschrieben werden soll.

Rückgabe

LONG VARCHAR

Siehe auch

- „SQLDIALECT-Funktion [Verschiedene]“ auf Seite 390
- „WATCOMSQL-Funktion [Verschiedene]“ auf Seite 430

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die Zeichenfolge 'SELECT EmployeeName=empl_name FROM Employees' zurück.

```
SELECT TRANSACTSQL( 'SELECT empl_name as EmployeeName FROM  
GROUPO.Employees' ) FROM dummy;
```

TREAT-Funktion [Datentypkonvertierung]

Mit der TREAT-Funktion können Sie den deklarierten Typ eines Geometrieausdrucks in einen Subtyp ändern. Diese Funktion ist für die Verwendung mit räumlichen Daten vorgesehen.

Syntax

TREAT (*Geometrieausdruck AS Subtyp*)

Parameter

- **Geometrieausdruck** Der zu konvertierende Ausdruck.
- **Subtyp** Der Zielsubtyp, in den *Geometrieausdruck* konvertiert werden soll.

Rückgabe

Abhängig vom angeforderten Datentyp.

Bemerkungen

Die TREAT-Funktion kann nur für Geometrien verwendet werden.

Wenn der dynamische Typ des Ausdrucks kein Subtyp des Zieldatentyps ist, wird ein Fehler zurückgegeben. Die CAST-Funktion kann ebenfalls verwendet werden, um den deklarierten Typ eines Geometrieausdrucks zu ändern. Die CAST-Funktion ermöglicht jedoch Änderungen außerhalb der Subtyphierarchie. Zum Beispiel kann CAST dazu verwendet werden, einen Punkt in eine Mehrpunktangabe zu konvertieren. Diese Arten von Konvertierungen können den dynamischen Typ eines Ausdrucks in einer unerwarteten Weise ändern. Daher wird TREAT empfohlen, wenn Sie von einem Obertyp zu einem Subtyp übergehen möchten. Außerdem wird die TREAT-Funktion effizienter ausgeführt als die CAST-Funktion.

Siehe auch

- Ausdruck TREAT für Subtypen verwenden [[SQL Anywhere Server - Unterstützung für räumliche Daten](#)]
- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 186

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Führen Sie Folgendes in Interactive SQL aus, um eine Tabelle zu erstellen und zwei Werte in diese zu laden:

```
DROP TABLE IF EXISTS treatExample;
CREATE TABLE treatExample( pk INT PRIMARY KEY, geo ST_Geometry );
INSERT INTO treatExample VALUES(0, NEW ST_Point(3,4) );
INSERT INTO treatExample VALUES(1, NEW ST_MultiPoint( new ST_Point( 5,
6 ) ) );
```

Die folgende Abfrage gibt einen Fehler zurück. Siehe „Typ '%1' hat keinen Methodennamen '%2' (bei '%3')“ [[Fehlermeldungen](#)].

```
SELECT geo.ST_X() FROM treatExample T WHERE pk = 0;
```

Die folgende Abfrage ist erfolgreich:

```
SELECT TREAT( geo AS ST_Point ).ST_X() FROM treatExample WHERE pk = 0;
```

Die folgende Abfrage gibt einen Fehler zurück. Siehe „Typ '%1' hat keinen Methodennamen '%2' (bei '%3')“ [[Fehlermeldungen](#)].

```
SELECT TREAT( geo AS ST_Point ).ST_X() FROM treatExample T WHERE pk = 0;
```

Die folgende Abfrage ist erfolgreich, weil statt einer TREAT-Anweisung eine CAST-Anweisung verwendet wird:

```
SELECT CAST( geo AS ST_Point ) FROM treatExample WHERE pk = 1;
```

TRIM-Funktion [Zeichenfolge]

Entfernt führende und nachgestellte Leerzeichen aus einer Zeichenfolge.

Syntax

TRIM(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die zu kürzende Zeichenfolge

Rückgabe

- VARCHAR
- NVARCHAR
- LONG VARCHAR
- LONG NVARCHAR

Bemerkungen

Diese Funktion unterstützt NCHAR-Eingaben bzw. Ausgaben.

Siehe auch

- [„LTRIM-Funktion \[Zeichenfolge\]“ auf Seite 309](#)
- [„RTRIM-Funktion \[Zeichenfolge\]“ auf Seite 376](#)
- [„Zeichenfolgenfunktionen“ auf Seite 163](#)

Standards und Kompatibilität

- **SQL/2008** Die TRIM-Funktion ist eine SQL/2008 Kernfunktion.

SQL Anywhere unterstützt nicht die zusätzlichen Parameter *Trim_Specification* und *Trim_Character*, wie in SQL/2008 definiert. Die SQL Anywhere-Implementierung von TRIM entspricht der TRIM-Angabe BOTH.

Für die anderen TRIM-Angaben, die durch den SQL/2008-Standard (LEADING und TRAILING) definiert sind, stellt SQL Anywhere die Funktionen LTRIM bzw. RTRIM zur Verfügung.

Beispiel

Die folgende Anweisung gibt den Wert "chocolate" ohne führende oder nachgestellte Leerzeichen zurück.

```
SELECT TRIM( '   chocolate   ' );
```

TRIM_ARRAY-Funktion [zusammengesetzt]

Gibt ein implizit begrenztes Array zurück, das aus einer angegebenen Anzahl von Elementen in einem Array besteht.

Syntax

```
TRIM_ARRAY( array-expression, integer-expression )
```

Parameter

- **Array-Ausdruck** Das zu kürzende Array.
- **Ganzzahlausdruck** Die Anzahl der Elemente, die das resultierende Array enthalten soll. Das resultierende Array enthält die durch *Ganzzahlausdruck* angegebene Anzahl von Elementen ab dem Anfang von *Array-Ausdruck*.

Wenn *Ganzzahlausdruck* gleich Null ist, ist das resultierende Array leer. Wenn *Ganzzahlausdruck* kleiner als Null ist oder größer als die Kardinalität des Arrays, wird ein Fehler generiert.

Rückgabe

ARRAY

Bemerkungen

Wenn eines der Argumente für die TRIM_ARRAY-Funktion NULL ist, ist auch das Ergebnis NULL.

Siehe auch

- [„ARRAY_AGG-Funktion \[Aggregat\]“ auf Seite 170](#)
- [„ARRAY_MAX_CARDINALITY-Funktion \[zusammengesetzt\]“ auf Seite 171](#)
- [„CARDINALITY-Funktion \[zusammengesetzt\]“ auf Seite 185](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel veranschaulicht, wie Sie mithilfe der TRIM_ARRAY-Funktion ein Array erstellen können, das die ersten zwei Elemente aus einem Array mit vier Elementen enthält:

```
DECLARE UntrimmedArray ARRAY( 4 ) OF INT;
DECLARE TrimmedArray ARRAY( 2 ) OF INT;
SET UntrimmedArray = ARRAY( 1, 2, 3, 4 );
SET TrimmedArray = TRIM_ARRAY( UntrimmedArray, 2 );
```

TRUNCNUM-Funktion [Numerisch]

Kürzt eine Zahl an der angegebenen Anzahl von Dezimalstellen.

Syntax

{ TRUNCNUM | TRUNCATE }(*numeric-expression*, *integer-expression*)

Parameter

- **Numerischer Ausdruck** Die zu kürzende Zahl. Dieses Argument kann vom Typ NUMERIC oder DOUBLE sein.
- **Ganzzahlausdruck** Eine positive Ganzzahl bestimmt die Anzahl von signifikanten Stellen rechts vom Dezimalzeichen für die Rundung. Eine negative Ganzzahl bestimmt die Anzahl von signifikanten Stellen links vom Dezimalzeichen für die Rundung.

Rückgabe

NUMERIC oder DOUBLE

Bemerkungen

Sie sollten die TRUNCNUM-Funktion und nicht die TRUNCATE-Funktion zum Kürzen von Zahlen verwenden.

Die Verwendung der TRUNCATE-Funktion wird nicht empfohlen, weil das Wort "truncate" ein Schlüsselwort ist, was bedeutet, dass Sie entweder die quoted_identifier-Option auf OFF setzen oder das Wort "TRUNCATE" in Anführungszeichen setzen müssen.

Siehe auch

- „ROUND-Funktion [Nummerisch]“ auf Seite 371
- „quoted_identifier-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert "600" zurück:

```
SELECT TRUNCNUM( 655, -2 );
```

Die folgende Anweisung liefert den Wert 655.340.

```
SELECT TRUNCNUM( 655.348, 2 );
```

TSEQUAL-Funktion [System] (nicht mehr empfohlen)

Vergleicht zwei TIMESTAMP-Werte und gibt zurück, ob sie identisch sind.

Syntax

```
TSEQUAL ( timestamp-expression-1, timestamp-expression-2 )
```

Parameter

- **Zeitstempelausdruck_1** Ein TIMESTAMP-Wert.
- **Zeitstempelausdruck_2** Ein TIMESTAMP-Wert.

Rückgabe

BIT

Bemerkungen

Die TSEQUAL-Funktion kann nur in einer WHERE-Klausel verwendet werden und wird meist als Teil einer UPDATE-Anweisung eingesetzt.

Obwohl die TSEQUAL-Funktion auch verwendet werden kann, um zwei gewöhnliche TIMESTAMP-Werte zu vergleichen, besteht der Zweck von TSEQUAL darin zu ermitteln, ob eine Zeile von einer anderen Verbindung geändert wurde. Dazu werden zwei spezielle Transact-SQL-TIMESTAMP-Werte verglichen.

Wenn in einer einzeiligen UPDATE-Anweisung mit TSEQUAL *Zeitstempelausdruck_1* gleich *Zeitstempelausdruck_2* ist, wobei einer dieser Werte eine mit DEFAULT TIMESTAMP deklarierte Spalte referenziert und der andere den Wert der Spalte beim letzten Abruf der Zeile, dann wurde die Zeile

seit dem letzten Abruf nicht geändert und TSEQUAL gibt TRUE zurück. Wenn die Zeile von einem anderen Benutzer geändert wurde, hat sich auch ihr Zeitstempel geändert und die Funktion TSEQUAL gibt FALSE zurück. Wenn die TSEQUAL Funktion in dieser Situation FALSE zurückgibt, wird der UPDATE-Vorgang nicht ausgeführt. Die Anwendung kann ermitteln, dass keine Zeilen aktualisiert wurden, indem sie die Anzahl der betroffenen Zeilen überprüft, z.B. mit @@rowcount. Wenn keine Zeilen betroffen sind, kann die Anwendung davon ausgehen, dass die Zeile von einem anderen Benutzer geändert wurde und dass sie nochmals abgerufen werden muss.

Siehe auch

- Der Datentyp einer TIMESTAMP-Spalte [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „TIMESTAMP-Spezialwert“ auf Seite 82
- „Spezielle TIMESTAMP-Spalte und TIMESTAMP-Datentyp in Transact-SQL“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „UPDATE-Anweisung“ auf Seite 1109

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Angenommen, Sie erstellen eine TIMESTAMP-Spalte Products.LastUpdated, um den Zeitstempel für den Zeitpunkt der letzten Aktualisierung der Zeile zu speichern. Im folgenden Beispiel wird die TSEQUAL-Funktion verwendet, um einen Zeilenwert zu ändern. Eine Aktualisierung wird nur dann in die Zeile übernommen, wenn die Zeile seit ihrem letzten Abruf nicht geändert wurde.

```
SELECT LastUpdated into old_ts_value
FROM GROUPO.Products
WHERE ID = '300';

UPDATE GROUPO.Products
SET Color = 'Yellow'
WHERE ID = '300'
AND TSEQUAL( LastUpdated, old_ts_value );
```

UCASE-Funktion [Zeichenfolge]

Konvertiert alle Zeichen in einer Zeichenfolge in Großbuchstaben.

Syntax

UCASE(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, die in Großbuchstaben konvertiert werden soll.

Rückgabe

CHAR, VARCHAR, LONG VARCHAR, NCHAR, NVARCHAR, oder LONG NVARCHAR entsprechend dem Datentyp des Arguments.

Bemerkungen

Diese Funktion ist mit der UPPER-Funktion identisch.

Siehe auch

- „UPPER-Funktion [Zeichenfolge]“ auf Seite 422
- „LCASE-Funktion [Zeichenfolge]“ auf Seite 299
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Die UPPER-Funktion entspricht SQL/2008.

Beispiel

Die folgende Anweisung gibt den Wert "CHOCOLATE" zurück:

```
SELECT UCASE( 'ChocoLate' );
```

UNICODE-Funktion [Zeichenfolge]

Gibt eine Ganzzahl zurück, die den Unicode-Codepunkt des ersten Zeichens in der Zeichenfolge enthält, oder NULL, wenn das erste Zeichen keine gültige Kodierung ist.

Syntax

UNICODE(*nchar-string-expression*)

Parameter

- **NCHAR-Zeichenfolgenderausdruck** Die NCHAR-Zeichenfolge, deren erstes Zeichen in eine Ganzzahl konvertiert werden soll.

Rückgabe

INT

Siehe auch

- „CONNECTION_EXTENDED_PROPERTY-Funktion [Zeichenfolge]“ auf Seite 197
- „NCHAR-Funktion [Zeichenfolge]“ auf Seite 321
- „TO_CHAR-Funktion [Zeichenfolge]“ auf Seite 409
- „TO_NCHAR-Funktion [Zeichenfolge]“ auf Seite 410
- „UNISTR-Funktion [Zeichenfolge]“ auf Seite 421

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel gibt die Ganzzahl "65536" zurück.

```
SELECT UNICODE(UNISTR( '\u010000data' ));
```

UNISTR-Funktion [Zeichenfolge]

Konvertiert eine Zeichenfolge, die Zeichen und Unicode-Escapesequenzen enthält, in eine NCHAR-Zeichenfolge.

Syntax

UNISTR(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, die konvertiert werden soll.

Rückgabe

- NVARCHAR
- LONG NVARCHAR

Bemerkungen

Die UNISTR-Funktion ermöglicht die Verwendung von Unicode-Zeichen, die nicht im von der SQL-Anweisung verwendeten CHAR-Zeichensatz dargestellt werden können. In einer englischsprachigen Umgebung beispielsweise kann die UNISTR-Funktion verwendet werden, um chinesische Schriftzeichen aufzunehmen.

Die UNISTR-Funktion bietet eine ähnliche Funktionalität wie die N"-Konstante, nur dass die UNISTR-Funktion Unicode-Zeichen und Zeichen aus dem CHAR-Zeichensatz zulässt, während die N"-Konstante nur Zeichen aus dem CHAR-Zeichensatz zulässt.

Zeichenfolgenausdruck enthält Zeichen und Unicode-Escapesequenzen. Die Unicode-Escapesequenzen haben die Form \uXXXX oder \uXXXXXX, wobei jedes X eine hexadezimale Ziffer ist. Die UNISTR-Funktion konvertiert jedes Zeichen und jede Unicode-Escapesequenz in das entsprechende Unicode-Zeichen.

Wenn eine 6-stellige Unicode-Escapesequenz verwendet wird, darf ihr Wert nicht '10FFFF' überschreiten, den größten Unicode-Codepunkt. Eine Sequenz wie '\u234567' ist keine 6-stellige Unicode-Escapesequenz. Es ist die 4-stellige Sequenz '\u2345', gefolgt von den Zeichen 6 und 7.

Wenn zwei benachbarte Unicode-Escapesequenzen ein UTF-16-Ersatzpaar bilden, werden sie in der Ausgabe zu einem Unicode-Zeichen zusammengefasst.

Siehe auch

- „[CONNECTION_EXTENDED_PROPERTY-Funktion \[Zeichenfolge\]](#)“ auf Seite 197
- „[NCHAR-Funktion \[Zeichenfolge\]](#)“ auf Seite 321
- „[TO_CHAR-Funktion \[Zeichenfolge\]](#)“ auf Seite 409
- „[TO_NCHAR-Funktion \[Zeichenfolge\]](#)“ auf Seite 410
- „[UNICODE-Funktion \[Zeichenfolge\]](#)“ auf Seite 420
- „[Zeichenfolgen](#)“ auf Seite 6

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Das folgende Beispiel gibt die Zeichenfolge "Hello" zurück.

```
SELECT UNISTR( 'Hel\u006c\u006F' );
```

Das folgende Beispiel fasst das UTF-16-Ersatzpaar D800-DF02 zum Unicode-Codepunkt 10302 zusammen.

```
SELECT UNISTR( '\uD800\uDF02' );
```

Dieses Beispiel ist gleichwertig mit dem vorhergehenden:

```
SELECT UNISTR( '\u010302' );
```

UPPER-Funktion [Zeichenfolge]

Konvertiert alle Zeichen in einer Zeichenfolge in Großbuchstaben.

Syntax

UPPER(*string-expression*)

Parameter

- **Zeichenfolgendruck** Die Zeichenfolge, die in Großbuchstaben konvertiert werden soll.

Rückgabe

CHAR, VARCHAR, LONG VARCHAR, NCHAR, NVARCHAR, oder LONG NVARCHAR entsprechend dem Datentyp des Arguments.

Bemerkungen

Diese Funktion ist mit der UCASE-Funktion identisch.

Siehe auch

- „UCASE-Funktion [Zeichenfolge]“ auf Seite 419
- „LCASE-Funktion [Zeichenfolge]“ auf Seite 299
- „LOWER-Funktion [Zeichenfolge]“ auf Seite 308
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Die UPPER-Funktion ist eine Kernfunktion des SQL/2008-Standards.

Beispiel

Die folgende Anweisung gibt den Wert "CHOCOLATE" zurück:

```
SELECT UPPER( 'ChocoLate' );
```

USER_ID-Funktion [System]

Gibt die numerische Benutzer-ID für den angegebenen Benutzernamen zurück.

Syntax

`USER_ID([user-name])`

Parameter

- **Benutzername** Der Benutzername für die numerische Benutzer-ID, nach der Sie suchen.

Rückgabe

INTEGER

Bemerkungen

Wenn Sie *Benutzername* nicht angeben, wird die numerische Benutzer-ID des aktuellen Benutzers zurückgegeben.

Siehe auch

- „USER_NAME-Funktion [System]“ auf Seite 423
- „SUSER_ID-Funktion [System]“ auf Seite 404

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die numerische Benutzer-ID für den Benutzernamen GROUPO zurück.

```
SELECT USER_ID( 'GROUPO' );
```

USER_NAME-Funktion [System]

Gibt den Benutzernamen für die angegebene Benutzer-ID zurück.

Syntax

`USER_NAME([user-id])`

Parameter

- **user-id** Die Benutzer-ID des Benutzers, nach dem Sie suchen.

Rückgabe

VARCHAR

Bemerkungen

Wenn Sie keine *user-id* angeben, wird der Name des aktuellen Benutzers zurückgegeben.

Siehe auch

- „USER_ID-Funktion [System]“ auf Seite 423
- „SUSER_NAME-Funktion [System]“ auf Seite 405

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Benutzernamen mit der ID 101 zurück.

```
SELECT USER_NAME( 101 );
```

UUIDTOSTR-Funktion [Zeichenfolge]

Konvertiert den Wert eines eindeutigen Bezeichners (UUID, auch bekannt als GUID) in einen Zeichenfolgenwert.

Hinweis

In einer Datenbank, die vor Version 9.0.2 erstellt wurde, ist der UNIQUEIDENTIFIER-Datentyp als ein benutzerdefinierter Datentyp festgelegt, und es werden die STRTOUUID- und UUIDTOSTR-Funktionen benötigt, um zwischen binären und Zeichenfolgen-Darstellungen von UUID-Werten zu konvertieren.

In Datenbanken, die mit Version 9.0.2 oder später erstellt wurden, ist der UNIQUEIDENTIFIER-Datentyp ein nativer Datentyp und SQL Anywhere führt Konvertierungen je nach Bedarf durch. Verwenden Sie bei diesen Versionen STRTOUUID und UUIDTOSTR nicht.

Weitere Hinweise finden Sie unter „[UNIQUEIDENTIFIER-Datentyp](#)“ auf Seite 134.

Syntax

UUIDTOSTR(*uuid-expression*)

Parameter

- **UUID-Ausdruck** Ein eindeutig bezeichnender Wert.

Rückgabe

VARCHAR

Bemerkungen

Konvertiert einen eindeutig bezeichnenden Wert in eine Zeichenfolge des Formats *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx*, wobei hier x für eine hexadezimale Ziffer steht. Sollte der BINARY-Wert kein gültiger eindeutiger Bezeichner sein, wird NULL zurückgegeben.

Diese Funktion ist nützlich, um einen UUID-Wert anzuzeigen.

Siehe auch

- „NEWID-Funktion [Verschiedene]“ auf Seite 321
- „STRTOUUID-Funktion [Zeichenfolge]“ auf Seite 399
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung erstellt eine Tabelle namens "mytab" mit zwei Spalten. Spalte pk verfügt über einen eindeutig identifizierenden Datentyp, die Spalte c1 über einen Ganzzahl-Datentyp. Die Anweisung fügt dann zwei Zeilen mit den Werten "1" und "2" entsprechend in die Spalte c1 ein.

```
CREATE TABLE mytab(
    pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    c1 INT );
INSERT INTO mytab( c1 ) values ( 1 );
INSERT INTO mytab( c1 ) values ( 2 );
```

Die Ausführung der folgenden SELECT-Anweisung gibt alle Daten der neu erstellten Tabelle zurück.

```
SELECT * FROM mytab;
```

Sie werden eine Tabelle sehen, die aus zwei Spalten und zwei Zeilen besteht. Der Wert für die Spalte pk wird binary-Werte darstellen.

Wenn Sie die eindeutigen Bezeichnerwerte in ein lesbares Format konvertieren möchten, führen Sie die folgende Anweisung aus:

```
SELECT UUIDTOSTR(pk), c1 FROM mytab;
```

Die UUIDTOSTR-Funktion wird nicht für Datenbanken benötigt, die mit Version 9.0.2 oder höher erstellt wurden.

VAR_POP-Funktion [Aggregat]

Berechnet die statistische Varianz einer Population, die aus einem numerischen Ausdruck besteht, als DOUBLE.

Syntax 1

```
VAR_POP( numeric-expression )
```

Syntax 2

```
VAR_POP( numeric-expression ) OVER ( window-spec )
```

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Numerischer_Ausdruck** Der Ausdruck, dessen Varianz auf Populationsbasis über eine Zeilenmenge berechnet wird. Der Ausdruck ist normalerweise ein Spaltenname.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltgenauem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück.

Die Varianz auf Populationsbasis (s^2) von *Nummerischer_Ausdruck* (x) wird nach der folgenden Formel berechnet:

$$s^2 = (1/N) * \text{SUM}(x_i - \text{mean}(x))^2$$

Diese Varianz umfasst keine Zeilen, in denen *Nummerischer_Ausdruck* gleich NULL ist. Er gibt NULL zurück für eine Gruppe, die keine Zeilen enthält.

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „Fensterfunktionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Siehe auch

- „Aggregatfunktionen“ auf Seite 153
- „WINDOW-Klausel“ auf Seite 1124

Standards und Kompatibilität

- **SQL/2008** Die VAR_POP-Funktion umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions". Wenn VAR_POP als Fensterfunktion verwendet wird, umfasst sie einen Teil der optionalen SQL-Sprachenfunktion T611, "Elementare OLAP-Vorgänge".

Die Möglichkeit, DISTINCT über einen Ausdruck anzugeben, der keine Spaltenreferenz ist, umfasst einen Teil der optionalen SQL-Sprachenfunktion F561, "Full value expressions". SQL Anywhere unterstützt auch die SQL/2008-Sprachenfunktion F441, "Extended set function support", die zulässt, dass die Operanden von Aggregatfunktionen beliebige Ausdrücke sein können, möglicherweise einschließlich äußerer Referenzen zu Ausdrücken in anderen Abfrageblöcken, die keine Spaltenreferenzen sind.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der VAR_POP-Funktion als auch eine äußere Referenz enthalten. Ein Beispiel finden Sie unter „AVG-Funktion [Aggregat]“ [[UltraLite - Datenbankverwaltung](#)].

Beispiel

Die folgende Anweisung listet den Durchschnitt und die Varianz in der Anzahl der Elemente pro Auftrag in verschiedenen Zeitabschnitten auf:

```
SELECT YEAR( ShipDate ) AS Year,
       QUARTER( ShipDate ) AS Quarter,
       AVG( Quantity ) AS Average,
       VAR_POP( quantity ) AS Variance
FROM GROUPO.SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

Year	Quarter	Average	Variance
2000	1	25.775148	203.9021...
2000	2	27.050847	225.8109...
...

VAR_SAMP-Funktion [Aggregat]

Berechnet die statistische Varianz eines Musters, das aus einem numerischen Ausdruck besteht, als DOUBLE.

Syntax 1

VAR_SAMP(*numeric-expression*)

Syntax 2

VAR_SAMP(*numeric-expression*) **OVER** (*window-spec*)

Fensterspezifikation: Siehe untenstehende Anweisung "Syntax 2" im Abschnitt Bemerkungen.

Parameter

- **Numerischer_Ausdruck** Der Ausdruck, dessen Varianz auf Musterbasis über eine Zeilenmenge berechnet wird. Der Ausdruck ist normalerweise ein Spaltenname.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück.

Die Varianz (s^2) von *Numerischer_Ausdruck* (x) wird nach der folgenden Formel berechnet, die von einer Normalverteilung ausgeht:

$$s^2 = (1 / (N - 1)) * \text{SUM}(x_i - \text{mean}(x))^2$$

Diese Varianz umfasst keine Zeilen, in denen *Nummerischer_Ausdruck* gleich NULL ist. Es wird NULL für eine Gruppe zurückgegeben, die entweder '0' oder '1' enthält.

Syntax 2 stellt die Verwendung als Fensterfunktion in einer SELECT-Anweisung dar. Deshalb können Elemente von *Fensterspezifikation* entweder in der Funktionssyntax (inline) oder mit einer WINDOW-Klausel in der SELECT-Anweisung angegeben werden. Weitere Hinweise finden Sie unter der *Fensterspezifikation*-Definition für die WINDOW-Klausel.

Weitere Hinweise zur Verwendung von Fensterfunktionen in SELECT-Anweisungen mit Arbeitsbeispielen finden Sie unter „Fensterfunktionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Weitere Hinweise zum Angeben einer Fensterspezifikation in einer OVER-Klausel finden Sie unter „Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Siehe auch

- „Aggregatfunktionen“ auf Seite 153
- „VARIANCE-Funktion [Aggregat]“ auf Seite 429
- „WINDOW-Klausel“ auf Seite 1124

Standards und Kompatibilität

- **SQL/2008** Die VAR_SAMP-Funktion umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion T621, "Enhanced numeric functions". Wenn VAR_SAMP als Fensterfunktion verwendet wird, umfasst sie einen Teil der optionalen SQL-Sprachenfunktion T611, "Elementare OLAP-Vorgänge". Die VARIANCE-Syntax ist eine Erweiterung des Herstellers.

Die Möglichkeit, DISTINCT über einen Ausdruck anzugeben, der keine Spaltenreferenz ist, umfasst einen Teil der optionalen SQL-Sprachenfunktion F561, "Full value expressions". SQL Anywhere unterstützt auch die SQL/2008-Sprachenfunktion F441, "Extended set function support", die zulässt, dass die Operanden von Aggregatfunktionen beliebige Ausdrücke sein können, möglicherweise einschließlich äußerer Referenzen zu Ausdrücken in anderen Abfrageblöcken, die keine Spaltenreferenzen sind.

SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Funktion F442, "Gemischte Spaltenreferenzen in Gruppenfunktionen". SQL Anywhere lässt es nicht zu, dass die Argumente einer Aggregatfunktion sowohl eine Spaltenreferenz aus dem Abfrageblock mit der VAR_SAMP-Funktion als auch eine äußere Referenz enthalten. Ein Beispiel finden Sie unter „AVG-Funktion [Aggregat]“ [[UltraLite - Datenbankverwaltung](#)].

Beispiel

Die folgende Anweisung listet den Durchschnitt und die Varianz in der Anzahl der Elemente pro Auftrag in verschiedenen Zeitabschnitten auf:

```
SELECT YEAR( ShipDate ) AS Year,
       QUARTER( ShipDate ) AS Quarter,
       AVG( Quantity ) AS Average,
       VAR_SAMP( quantity ) AS Variance
FROM GROUPO.SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

Year	Quarter	Average	Variance
2000	1	25.775148	205.1158...
2000	2	27.050847	227.0939...
...

VAREXISTS-Funktion [Verschiedene]

Gibt "1" zurück, wenn eine benutzerdefinierte Variable mit dem angegebenen Namen erstellt oder deklariert wurde. Gibt "0" zurück, wenn keine dieser Variablen erstellt wurde.

Syntax

VAREXISTS(*variable-name-string*)

Parameter

- **Variablenname-Zeichenfolge** Der zu testende Variablenname als eine Zeichenfolge.

Rückgabe

INT

Siehe auch

- „CREATE VARIABLE-Anweisung“ auf Seite 771
- „DECLARE-Anweisung“ auf Seite 786
- „IF-Anweisung“ auf Seite 906

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende IF-Anweisung erstellt eine Variable mit dem Namen start_time, wenn noch keine erstellt oder deklariert wurde. Die Variable kann dann problemlos verwendet werden.

```
IF VAREXISTS( 'start_time' ) = 0 THEN
    CREATE VARIABLE start_time TIMESTAMP;
END IF;
SET start_time = current timestamp;
```

VARIANCE-Funktion [Aggregat]

Ein Alias für VAR_SAMP.

Siehe auch

- „VAR_SAMP-Funktion [Aggregat]“ auf Seite 427

WATCOMSQL-Funktion [Verschiedene]

Schreibt eine Transact-SQL-Anweisung im Watcom-SQL-Dialekt neu. Dies kann für die Konvertierung bestehender gespeicherter Prozeduren in Adaptive Server Enterprise auf die Watcom SQL-Syntax benutzt werden.

Syntax

WATCOMSQL(*sql-statement-string*)

Parameter

- **sql-statement-string** Die SQL-Anweisung, die in den Watcom-SQL-Dialekt umgeschrieben werden soll.

Rückgabe

LONG VARCHAR

Siehe auch

- „SQLDIALECT-Funktion [Verschiedene]“ auf Seite 390
- „TRANSACTSQL-Funktion [Verschiedene]“ auf Seite 413

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die Zeichenfolge 'SELECT Surname as last_name FROM Employees' zurück:

```
SELECT WATCOMSQL( 'SELECT last_name = Surname FROM GROUPO.Employees' ) FROM dummy;
```

WEEKS-Funktion [Datum und Uhrzeit]

Bearbeitet einen TIMESTAMP-Wert oder gibt die Anzahl von Wochen zwischen zwei TIMESTAMP-Werten zurück. Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Syntax 1

WEEKS(*timestamp-expression*)

Syntax 2

WEEKS(*timestamp-expression*, *timestamp-expression*)

Syntax 3

WEEKS(*timestamp-expression*, *integer-expression*)

Parameter

- **Zeitstempelausdruck** Ein Datums- und Uhrzeitwert vom Typ TIMESTAMP.

- **Ganzzahlausdruck** Die Anzahl der Wochen, die zu *Zeitstempelausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl an Wochen von *Zeitstempelausdruck* abgezogen. Wenn Sie einen *Ganzzahlausdruck* angeben, muss *Zeitstempelausdruck* explizit als TIME-, DATE- oder TIMESTAMP-Wert festgelegt sein.

Rückgabe

INTEGER bei Syntax 1 und Syntax 2.

TIMESTAMP bei Syntax 3.

Bemerkungen

Wenn ein einzelnes Datum eingegeben wird (Syntax 1), liefert die WEEKS-Funktion die Anzahl der Wochen seit 0000-02-29.

Wenn zwei Daten eingegeben werden (Syntax 2), liefert die WEEKS-Funktion die Anzahl der Wochen zwischen den beiden Daten. Die WEEKS-Funktion ähnelt der DATEDIFF-Funktion, allerdings ist die Methode zur Berechnung der Anzahl der Wochen zwischen den Daten nicht identisch und kann ein anderes Ergebnis bringen. Der Rückgabewert für WEEKS wird festgelegt, indem die Anzahl von Tagen zwischen den beiden Daten durch 7 geteilt und abgerundet wird. DATEDIFF verwendet jedoch in seiner Berechnung die Anzahl der überschrittenen Wochengrenzen. Daher können die beiden Funktionen unterschiedliche Werte zurückgeben. Beispiel: Wenn das erste Datum ein Freitag ist und das zweite der nachfolgende Montag, gibt die WEEKS-Funktion den Unterschied 0 zurück, die DATEDIFF-Funktion dagegen den Unterschied 1. Obwohl keine Methode besser ist als die andere, sollten Sie den Unterschied im Auge behalten, wenn Sie sich für WEEKS oder DATEDIFF entscheiden.

Weitere Hinweise zur DATEDIFF-Funktion finden Sie unter „[DATEDIFF-Funktion \[Datum und Uhrzeit\]](#)“ auf Seite 218.

Wenn ein Datum und eine Ganzzahl eingegeben werden (Syntax 3), addiert die WEEKS-Funktion die mit der Ganzzahl ausgedrückte Anzahl von Wochen zum angegebenen *Zeitstempelausdruck*. Bei Syntax 3 müssen Sie *Zeitstempelausdruck* explizit als Datentyp TIME, DATE oder TIMESTAMP festlegen. Wenn *Zeitstempelausdruck* ein TIME-Wert ist, wird das aktuelle Datum angenommen. Verwenden Sie statt Syntax 3 die DATEADD-Funktion.

Weitere Hinweise zur DATEADD-Funktion finden Sie unter „[DATEADD-Funktion \[Datum und Uhrzeit\]](#)“ auf Seite 217.

Siehe auch

Hinweise zum Casting von Datentypen finden Sie unter „[CAST-Funktion \[Datentypkonvertierung\]](#)“ auf Seite 186.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt den Wert 8 zurück. Das bedeutet, dass 2008-09-13 10:07:12 acht Wochen später ist als 2008-07-13 06:07:12.

```
SELECT WEEKS( '2008-07-13 06:07:12', '2008-09-13 10:07:12' );
```

Die folgende Anweisung gibt den Wert 104792 zurück. Das bedeutet, dass das Datum 104792 nach 0000-02-29 liegt.

```
SELECT WEEKS( '2008-07-13 06:07:12' );
```

Die folgende Anweisung gibt den TIMESTAMP-Wert 2008-06-16 21:05:07.0 zurück. Dies steht für das Datum und die Uhrzeit fünf Wochen nach 2008-05-12 21:05:07.

```
SELECT WEEKS( CAST( '2008-05-12 21:05:07' AS TIMESTAMP ), 5 );
```

WRITE_CLIENT_FILE-Funktion [Zeichenfolge]

Erstellt eine Datei auf dem Clientcomputer und schreibt Daten hinein.

Syntax

```
WRITE_CLIENT_FILE( filename, blob-expression [, 'A' ] )
```

Parameter

- ***filename*** Der Name der Datei auf dem Clientcomputer. Der Name wird auf dem Clientcomputer relativ zum aktuellen Arbeitsverzeichnis der Clientanwendung aufgelöst.
- ***BLOB-Ausdruck*** Eine Binärzeichenfolge, die in *filename* auf dem Clientcomputer geschrieben werden soll.
- **A** Standardmäßig wird die Datei überschrieben, wenn sie bereits existiert. Wenn die Dateien an bestehende Daten angehängt werden sollen, geben Sie 'A' an. Wenn die Datei noch nicht existiert und Sie 'A' angeben, wird die Datei dennoch erstellt.

Rückgabe

INT

Bemerkungen

Der Datenbankserver konvertiert *filename* aus dem Zeichensatz des Datenbankservers in den Zeichensatz des Clients. Auf dem Clientcomputer wird dann *filename* in den Zeichensatz des Betriebssystems konvertiert.

Da die Daten als Binärzeichenfolge übermittelt werden, müssen Sie, falls die Daten in einem bestimmten Zeichensatz, komprimiert oder verschlüsselt werden sollen, diese Vorgänge mit den Daten durchführen, bevor Sie sie an die WRITE_CLIENT_FILE-Funktion übergeben.

Das Lesen der Datei erfolgt durch die Client-Softwarebibliothek und die Übertragung der Daten erfolgt mit dem Befehlsfolgen-Kommunikationsprotokoll.

Privilegien

Beim Schreiben in eine Datei auf einem Clientcomputer gilt Folgendes:

- Sie müssen das WRITE FILE-Systemprivileg haben.
- Die Clientanwendung muss Schreibberechtigungen auf dem Computer haben, auf dem der Schreibvorgang erfolgt.
- Die Datenbankoption `allow_write_client_file` muss aktiviert sein.
- Die gesicherte Funktion `write_client_file` muss aktiviert sein.

Siehe auch

- „`allow_write_client_file`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption `-sf`“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Zugriff auf Daten auf Clientcomputern“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „UNLOAD-Anweisung“ auf Seite 1098
- „CSCONVERT-Funktion [Zeichenfolge]“ auf Seite 211
- „DECOMPRESS-Funktion [Zeichenfolge]“ auf Seite 233
- „DECRYPT-Funktion [Zeichenfolge]“ auf Seite 234

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

XMLAGG-Funktion [Aggregat]

Erzeugt einen Wald von XML-Elementen aus einer Sammlung von XML-Werten.

Syntax

XMLAGG(*expression* [**ORDER BY** *order-by-expression*])

Parameter

- **expression** Ein XML-Wert. Der Inhalt wird in Escapezeichen gesetzt, es sei denn, der Datentyp ist XML. Der *order_by_expression* sortiert die von der Funktion zurückgegebenen Elemente.
- **Sortierausdruck** Ein Ausdruck, der zum Sortieren der XML-Elemente anhand des Ausdruckswerts verwendet wird.

Wenn eine ORDER BY-Klausel Konstante enthält, werden sie vom Optimierer interpretiert und dann durch eine entsprechende ORDER BY-Klausel ersetzt. Der Optimierer interpretiert z.B. "ORDER BY 'a'" als ORDER BY-Ausdruck.

Ein Abfrageblock, der mehr als eine Aggregatfunktion mit gültigen ORDER BY-Klauseln enthält, kann ausgeführt werden, wenn die ORDER BY-Klauseln logisch in einer einzigen ORDER BY-Klausel kombiniert können werden. Zum Beispiel sind folgende Klauseln vorhanden:

```
ORDER BY expression1, 'a', expression2
```

```
ORDER BY expression1, 'b', expression2, 'c', expression3
```

Sie werden in der folgenden Klausel zusammengefasst:

`ORDER BY expression1, expression2, expression3`

Rückgabe

XML

Bemerkungen

NULL wird nicht in das Ergebnis einbezogen. Wenn alle Eingaben gleich NULL sind bzw. keine Zeilen vorhanden sind, ist das Ergebnis gleich NULL. Wenn ein korrekt aufgebautes XML-Dokument gefordert ist, müssen Sie sicherstellen, dass Ihre Abfrage so geschrieben ist, dass die generierte XML nur ein Wurzelement hat.

Daten in den Spalten BINARY, LONG BINARY, IMAGE und VARBINARY werden automatisch im Base64-kodierten Format zurückgegeben, wenn Sie eine Abfrage ausführen, die XMLAGG enthält.

Ein Beispiel, das die XMLAGG-Funktion mit einer ORDER BY-Klausel verwendet, finden Sie unter „[Verwendung der XMLAGG-Funktion](#)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Siehe auch

- „[Verwendung der XMLAGG-Funktion](#)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** XMLAGG ist Teil der optionalen SQL/2008-Sprachenfunktion X034. Die optionale ORDER BY-Klausel für die XMLAGG-Funktion umfasst die SQL/2008 Sprachenfunktion X035.

Beispiel

Die folgende Anweisung erzeugt ein XML-Dokument, aus dem die von den einzelnen Kunden erteilten Aufträge ersichtlich sind.

```
SELECT XMLELEMENT( NAME "order",
                   XMLATTRIBUTES( ID AS order_id ),
                   ( SELECT XMLAGG(
                       XMLELEMENT(
                           NAME "Products",
                           XMLATTRIBUTES( ProductID, Quantity AS
"quantity_shipped" ) ) )
                   FROM SalesOrderItems soi
                   WHERE soi.ID = so.ID
                   )
                   ) AS products_ordered
FROM GROUPO.SalesOrders so
ORDER BY so.ID;
```

XMLCONCAT-Funktion [Zeichenfolge]

Produziert einen Wald von XML-Elementen.

Syntax

`XMLCONCAT(xml-value [, ...])`

Parameter

- **xml-value** Die zu verkettenden XML-Werte.

Rückgabe

XML

Bemerkungen

Erzeugt einen Wald von XML-Elementen. In einem nicht syntaktisch analysierten Dokument bezieht sich "Wald" auf die Vielzahl der Stammknoten innerhalb des Dokumentes. NULL wird nicht in das Ergebnis einbezogen. Wenn alle Werte NULL sind, wird NULL zurückgegeben. Die XMLCONCAT-Funktion prüft nicht, ob das Argument einen Prolog aufweist. Wenn ein korrekt aufgebautes XML-Dokument gefordert ist, müssen Sie sicherstellen, dass Ihre Abfrage so geschrieben ist, dass nur ein Wurzelement erzeugt wird.

Der Elementeninhalt wird immer in Escapezeichen gesetzt, es sei denn, der Datentyp ist XML. Daten in den Spalten BINARY, LONG BINARY, IMAGE und VARBINARY werden automatisch im Base64-kodierten Format zurückgegeben, wenn Sie eine Abfrage ausführen, die eine XMLCONCAT-Funktion enthält.

Siehe auch

- „Verwendung der XMLCONCAT-Funktion“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „XMLELEMENT-Funktion [Zeichenfolge]“ auf Seite 435
- „XMLFOREST-Funktion [Zeichenfolge]“ auf Seite 438
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** XMLCONCAT umfasst einen Teil der optionalen SQL/2008-Sprachenfunktion X020.

Beispiel

Die folgende Abfrage erzeugt die Elemente <CustomerID>, <cust_fname> und <cust_lname> für jeden einzelnen Kunden.

```
SELECT XMLCONCAT( XMLELEMENT ( NAME CustomerID, ID ),
                  XMLELEMENT( NAME cust_fname, GivenName ),
                  XMLELEMENT( NAME cust_lname, Surname )
                ) AS "Customer Information"
FROM GROUPO.Customers
WHERE ID < 120;
```

XMLELEMENT-Funktion [Zeichenfolge]

Erzeugt ein XML-Element innerhalb einer Abfrage.

Syntax

```
XMLELEMENT( { NAME element-name-expression } | string-expression
            [, XMLATTRIBUTES ( attribute-value-expression [ AS attribute-name ],... ) ]
            [, element-content-expression,... ]
          )
```

Parameter

- **Elementnamen Ausdruck** Ein Bezeichner. Für jede Zeile wird ein XML-Element mit demselben Namen wie der Bezeichner erzeugt.
- **Attributwert Ausdruck** Ein Attribut des Elementes. Mit diesem optionalen Argument können Sie einen Attributwert für das erzeugte Element angeben. Dieses Argument legt den Attributnamen und Inhalt fest. Wenn *Attributwert Ausdruck* ein Spaltenname ist, nimmt der Attributname den Spaltennamen an. Sie können den Attributnamen ändern, indem Sie das *Attributname-Argument* angeben.
- **Elementinhalt Ausdruck** Der Inhalt des Elementes. Dies kann ein beliebiger Zeichenfolgenausdruck sein. Sie können eine unbegrenzte Anzahl von *Elementinhalt Ausdruck*-Argumenten angeben, die dann verkettet werden. Beispiel: Die folgende SELECT-Anweisung gibt den Wert '<x>abcdef</x>' zurück:

```
SELECT XMLELEMENT( NAME x, 'abc', 'def' );
```

Rückgabe

XML

Bemerkungen

NULL-Elementwerte und NULL-Attributwerte werden nicht in das Ergebnis einbezogen. Die Groß- und Kleinschreibung für Element- und Attributnamen wird aus der Abfrage entnommen.

Der Elementeninhalt wird immer in Escapezeichen gesetzt, es sei denn, der Datentyp ist XML. Ungültige Element- und Attributnamen werden ebenfalls in Anführungszeichen gesetzt. Nehmen wir als Beispiel folgende Anweisung:

```
SELECT XMLELEMENT('H1', f_get_page_heading() );
```

Wenn die Funktion `f_get_page_heading` als `RETURNS LONG VARCHAR` oder `RETURNS VARCHAR(1000)` definiert ist, ist das Ergebnis HTML-kodiert:

```
CREATE FUNCTION f_get_page_heading() RETURNS LONG VARCHAR
BEGIN
    RETURN ( '<B>My Heading</B>' );
END;
```

Die obenstehende SELECT-Anweisung gibt Folgendes zurück:

```
<H1>&lt;B&gt;My Heading&lt;/B&gt;</H1>
```

Wenn die Funktion als `RETURNS XML` deklariert ist, gibt die obenstehende SELECT-Anweisung Folgendes zurück:

```
<H1><B>My Heading</B></H1>
```

Weitere Hinweise zum Zitieren und zur XMLELEMENT-Funktion finden Sie unter [Ungültige Namen und SQL/XML \[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#).

XMLEMENT-Funktionen können ineinander verschachtelt werden und eine Hierarchie bilden. Wenn verschiedene Elemente auf derselben Ebene der Dokumenthierarchie zurückgegeben werden sollen, verwenden Sie die XMLFOREST-Funktion.

Weitere Hinweise finden Sie unter „XMLFOREST-Funktion [Zeichenfolge]“ auf Seite 438.

Daten in den Spalten BINARY, LONG BINARY, IMAGE und VARBINARY werden automatisch im Base64-kodierten Format zurückgegeben, wenn Sie eine Abfrage ausführen, die eine XMLEMENT-Funktion enthält.

Siehe auch

- „Verwendung der XMLEMENT-Funktion“ [SQL Anywhere Server - SQL-Benutzerhandbuch]
- „XMLCONCAT-Funktion [Zeichenfolge]“ auf Seite 434
- „XMLFOREST-Funktion [Zeichenfolge]“ auf Seite 438
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** XMLEMENT ist Teil der optionalen SQL/2008-Sprachenfunktion X031. Das Weglassen des NAME-Schlüsselworts und die Verwendung eines Zeichenfolgenausdrucks als erstes Argument ist eine Erweiterung des Herstellers. SQL Anywhere bietet keine Unterstützung für die optionale Klausel OPTION mit der XMLEMENT-Funktion.

Beispiel

Das folgende Beispiel erzeugt ein Element <item_name> für jedes Produkt in der Ergebnismenge, wobei der Produktname der Inhalt des Elementes ist.

```
SELECT ID, XMLEMENT( NAME item_name, p.Name )
FROM Products p
WHERE ID > 400;
```

Das folgende Beispiel gibt iAnywhere web site zurück:

```
SELECT XMLEMENT(
  'A',
  XMLATTRIBUTES( 'http://www.iAnywhere.com/'
    AS "HREF", '_top' AS "TARGET"),
  'iAnywhere web site'
);
```

Das folgende Beispiel gibt <table><tbody><tr align="center" valign="top"><td>Cell 1 info</td><td>Cell 2 info</td></tr></tbody></table> zurück:

```
SELECT XMLEMENT( name "table",
  XMLEMENT( name "tbody",
    XMLEMENT( name "tr",
      XMLATTRIBUTES('center' AS "align", 'top' AS "valign"),
      XMLEMENT( name "td", 'Cell 1 info' ),
      XMLEMENT( name "td", 'Cell 2 info' )
    )
  )
);
```

Das folgende Beispiel gibt '<x>abcdef</x>', '<custom_element>abcdef</custom_element>' zurück:

```
CREATE VARIABLE @my_element_name VARCHAR(200);
SET @my_element_name = 'custom_element';
SELECT XMLELEMENT( NAME x, 'abc', 'def' ),
       XMLELEMENT( @my_element_name, 'abc', 'def' );
```

XMLFOREST-Funktion [Zeichenfolge]

Erzeugt einen Wald von XML-Elementen.

Syntax

XMLFOREST(*element-content-expression* [**AS** *element-name*],...)

Parameter

- **Elementinhaltsausdruck** Eine Zeichenfolge. Für jedes angegebene *Elementinhaltsausdruck*-Argument wird ein Element generiert. Der Wert von *Elementinhaltsausdruck* wird der Inhalt des Elements. Wenn Sie z.B. die Spalte EmployeeID aus der Tabelle Employees für dieses Argument angeben, wird für jeden Wert in der Tabelle ein <EmployeeID>-Element erzeugt, das einen EmployeeID-Wert enthält.

Geben Sie das *Elementname*-Argument an, wenn Sie dem Element einen anderen Namen zuordnen möchten als *Elementinhaltsausdruck*. Andernfalls wird standardmäßig der Name *Elementinhaltsausdruck* verwendet.

Rückgabe

XML

Bemerkungen

Produziert einen Wald von XML-Elementen. In einem nicht syntaktisch analysierten Dokument bezieht sich "Wald" auf die Vielzahl der Stammknoten innerhalb des Dokumentes. Wenn alle Argumente für die XMLFOREST-Funktion gleich NULL sind, wird NULL zurückgegeben. Wenn nur einige Werte NULL sind, wird NULL nicht in das Ergebnis einbezogen. Der Elementeninhalt wird in Anführungszeichen gesetzt, es sei denn, der Datentyp ist XML. Sie können mit der XMLFOREST-Funktion keine Attribute angeben. Verwenden Sie die XMLELEMENT-Funktion, um Attribute für generierte Elemente anzugeben.

Weitere Hinweise zur Funktion XMLELEMENT finden Sie unter „[XMLELEMENT-Funktion \[Zeichenfolge\]](#)“ auf Seite 435.

Elementnamen werden in Escapezeichen gesetzt, es sei denn, der Datentyp ist XML.

Wenn ein korrekt aufgebautes XML-Dokument gefordert ist, müssen Sie sicherstellen, dass Ihre Abfrage so geschrieben ist, dass nur ein Wurzelement erzeugt wird.

Daten in den Spalten BINARY, LONG BINARY, IMAGE und VARBINARY werden automatisch im Base64-kodierten Format zurückgegeben, wenn Sie eine Abfrage ausführen, die XMLFOREST enthält.

Siehe auch

- „Verwendung der XMLFOREST-Funktion“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „XMLELEMENT-Funktion [Zeichenfolge]“ auf Seite 435
- „XMLCONCAT-Funktion [Zeichenfolge]“ auf Seite 434
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** XMLFOREST ist Teil der optionalen SQL/2008-Sprachenfunktion X032. SQL Anywhere bietet keine Unterstützung für die optionale XMLNAMESPACES-Klausel oder die OPTION-Klausel mit der XMLFOREST-Funktion.

Beispiel

Die folgende Anweisung produziert ein XML-Element für den Vor- und Nachnamen eines jeden Mitarbeiters.

```
SELECT EmployeeID,
       XMLFOREST( GivenName, Surname )
       AS "Employee Name"
FROM Employees;
```

XMLGEN-Funktion [Zeichenfolge]

Erzeugt einen XML-Wert auf der Grundlage eines XQuery-Konstruktors.

Syntax

XMLGEN(*xquery-constructor*, *content-expression* [**AS** *variable-name*],...)

Parameter

- ***xquery-constructor*** Ein XQuery-Konstruktor. Der XQuery-Konstruktor ist ein Element, das in der XQuery-Sprache definiert ist. Es bietet eine Syntax zum Aufbau von XML-Elementen auf der Grundlage von XQuery-Ausdrücken. Das Argument *xquery-constructor* muss ein korrekt aufgebautes XML-Dokument mit einer oder mehreren Variablenreferenzen sein. Eine Variablenreferenz wird in geschweifte Klammern eingeschlossen und erhält das Präfix \$ ohne vor- bzw. nachgestellte Leerstellen. Beispiel:

```
SELECT XMLGEN( '<a>{$x}</a>', 1 AS x );
```

- ***Inhaltsausdruck*** Eine Variable. Sie können mehrere *Inhaltsausdruck*-Argumente angeben. Das optionale Argument *Variablenname* wird zum Benennen der Variablen verwendet. Beispiel:

```
SELECT XMLGEN( '<emp EmployeeID="{ $EmployeeID }"><StartDate>{$x}</StartDate></emp>',
              EmployeeID, StartDate
              AS x )
FROM GROUPO.Employees;
```

Rückgabe

XML

Bemerkungen

Berechnete Konstruktoren, wie sie in der XQuery-Spezifikation beschrieben sind, werden von der XMLGEN-Funktion nicht unterstützt.

Wenn Sie eine Abfrage mit der XMLGEN-Funktion ausführen, werden Daten in den Spalten BINARY, LONG BINARY, IMAGE und VARBINARY automatisch im Base64-kodierten Format zurückgegeben.

Der Elementeninhalt wird immer in Escapezeichen gesetzt, es sei denn, der Datentyp ist XML. Ungültige XML-Element- und Attributnamen werden ebenfalls in Escapezeichen gesetzt.

Weitere Hinweise zu Escapezeichen und zur XMLGEN-Funktion finden Sie unter [Ungültige Namen und SQL/XML \[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#).

Siehe auch

- „Verwendung der XMLGEN-Funktion“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Zeichenfolgenfunktionen“ auf Seite 163

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. XMLGEN stellt ähnliche Funktionen bereit wie die SQL/2008-Funktion XMLDOCUMENT.

Beispiel

Das folgende Beispiel generiert die Tags <emp>, <Surname>, <GivenName> und <StartDate> für jeden Mitarbeiter.

```
SELECT XMLGEN( ' <emp EmployeeID="{ $EmployeeID} ">
               <Surname>="{ $Surname} "</Surname>
               <GivenName>="{ $GivenName} "</GivenName>
               <StartDate>="{ $StartDate} "</StartDate>
               </emp>',
               EmployeeID,
               Surname,
               GivenName,
               StartDate
             ) AS employee_list
FROM GROUPO.Employees;
```

YEAR-Funktion [Datum und Uhrzeit]

Gibt die Jahreskomponente des TIMESTAMP-Arguments zurück.

Syntax

YEAR(*timestamp-expression*)

Parameter

- **Zeitstempelausdruck** Ein TIMESTAMP-Wert.

Rückgabe

SMALLINT

Bemerkungen

Der zurückgegebene Wert ist die Jahreskomponente des angegebenen TIMESTAMP-Werts, in Form eines SMALLINT-Werts.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel gibt den Wert "2001" zurück.

```
SELECT YEAR( '2001-09-12' );
```

YEARS-Funktion [Datum und Uhrzeit]

Bearbeitet einen TIMESTAMP-Wert oder gibt die Anzahl von Jahren zwischen zwei TIMESTAMP-Werten zurück. Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Syntax 1

YEARS(*timestamp-expression*)

Syntax 2

YEARS(*timestamp-expression*, *timestamp-expression*)

Syntax 3

YEARS(*timestamp-expression*, *integer-expression*)

Parameter

- **Zeitstempelausdruck** Ein Datums- und Uhrzeitwert vom Typ TIMESTAMP.
- **Ganzzahlausdruck** Die Anzahl der Jahre (als SMALLINT-Wert), die zu *Zeitstempelausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl an Jahren von *Zeitstempelausdruck* abgezogen. Wenn Sie einen *Ganzzahlausdruck* angeben, muss *Zeitstempelausdruck* explizit als DATE-, TIME- oder TIMESTAMP-Wert festgelegt sein. Wenn *Zeitstempelausdruck* ein TIME-Wert ist, wird das laufende Jahr angenommen.

Hinweise zum Casting von Datentypen finden Sie unter „[CAST-Funktion \[Datentypkonvertierung\]](#)“ auf Seite 186.

Rückgabe

SMALLINT bei Syntax 1 und Syntax 2.

TIMESTAMP bei Syntax 3.

Bemerkungen

Der Wert von YEARS wird anhand der Anzahl der Neujahrstage zwischen zwei Datumsangaben berechnet.

Siehe auch

- „DATEDIFF-Funktion [Datum und Uhrzeit]“ auf Seite 218
- „DATEADD-Funktion [Datum und Uhrzeit]“ auf Seite 217

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die nachstehenden Anweisungen geben beide -4 zurück.

```
SELECT YEARS( '1998-07-13 06:07:12',  
              '1994-03-13 08:07:13' );  
  
SELECT DATEDIFF( year,  
                '1998-07-13 06:07:12',  
                '1994-03-13 08:07:13' );
```

Die folgenden Anweisungen geben "1998" zurück:

```
SELECT YEARS( '1998-07-13 06:07:12' )  
SELECT DATEPART( year, '1998-07-13 06:07:12' );
```

Die folgenden Anweisungen geben das eingegebene Datum plus 300 Jahre zurück:

```
SELECT YEARS( CAST( '1998-07-13 06:07:12' AS TIMESTAMP ), 300 )  
  
SELECT DATEADD( year, 300, '1998-07-13 06:07:12' );
```

YMD-Funktion [Datum und Uhrzeit]

Gibt einen Datumswert zurück, der dem Jahr, Monat und Monatstag entspricht. Argumente sind SMALLINT-Werte von -32768 bis 32767.

Syntax

YMD(*smallint-expression1*, *smallint-expression2*, *smallint-expression3*)

Parameter

- **SMALLINT-Ausdruck_1** Die Jahreszahl.
- **SMALLINT-Ausdruck_2** Die Monatszahl. Das Jahr wird angepasst, wenn der Monat außerhalb des Bereichs 1-12 liegt.
- **SMALLINT-Ausdruck_3** Die Tagesnummer. Der Tag kann jede Ganzzahl sein, das Datum wird entsprechend angepasst.

Rückgabe

DATE

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung liefert den Wert 1998-06-12.

```
SELECT YMD( 1998, 06, 12 );
```

Wenn die Werte außerhalb des normalen Bereichs liegen, wird das Datum entsprechend angepasst. Die folgende Anweisung gibt beispielsweise den DATE-Wert "2000-03-01" zurück.

```
SELECT YMD( 1999, 15, 1 );
```

SQL-Anweisungen

In diesem Abschnitt werden die in der SQL-Anweisungsdokumentation verwendeten Konventionen beschrieben.

Allgemeine Elemente der SQL-Syntax

In diesem Abschnitt finden Sie eine Liste der Sprachelemente, die in der Syntax vieler SQL-Anweisungen zu finden sind.

- **column-name** Ein Bezeichner, der den Namen einer Spalte repräsentiert.
- **condition** Ein Ausdruck, der als TRUE, FALSE oder UNKNOWN bewertet werden kann.
- **connection-name** Eine Zeichenfolge, die den Namen einer aktiven Verbindung repräsentiert.
- **data-type** Ein Speicherdatentyp.
- **expression** Ein Ausdruck. Ein allgemeines Beispiel für einen Ausdruck in der SQL-Syntax ist ein Spaltenname.
- **filename** Eine Zeichenfolge, die einen Dateinamen enthält.
- **hostvar** Eine Variable der Sprache C, die als Hostvariable mit vorangestelltem Doppelpunkt deklariert ist.
- **materialized-view-name** Ein Bezeichner, der den Namen einer materialisierten Ansicht repräsentiert.
- **number** Eine Ziffernfolge, gefolgt von einem optionalen dezimalen Teil und mit einem optionalen negativen Vorzeichen. Wahlweise kann nach der Zahl ein E und dann ein Exponent stehen. Beispiel:

```
42  
-4.038  
.001  
3.4e10  
1e-10
```

- **owner** Ein Bezeichner für die Benutzer-ID, die Eigentümer eines Datenbankobjektes ist.
- **query-block** Ein Abfrageblock ist ein einfacher Abfrageausdruck oder ein Abfrageausdruck mit einer ORDER BY-Klausel.
- **query-expression** Ein Abfrageausdruck kann ein SELECT-, UNION-, INTERSECT- oder EXCEPT-Block sein (d.h., eine Anweisung, die keine ORDER BY-, WITH-, FOR-, FOR XML- oder OPTION-Klausel enthält), oder eine Kombination solcher Blöcke.
- **role-name** Ein Bezeichner für den Rollennamen eines Fremdschlüssels. Bei der konzeptuellen Datenbankmodellierung sind dies Verben oder Sätze, die die Beziehung von einem Gesichtspunkt aus

beschreiben. Sie können jede Beziehung mit zwei Rollen beschreiben. Beispiele für Rollen sind "enthält" und "ist ein Mitglied von".

- **savepoint-name** Ein Bezeichner für den Namen eines Savepoints.
- **search-condition** Eine Bedingung, die als TRUE, FALSE oder UNKNOWN bewertet werden kann.
- **special-value** Ein Spezialwert.
- **statement-label** Ein Bezeichner für das Label einer Schleife oder einer zusammengesetzten Anweisung.
- **statement-list** Eine Liste der SQL-Anweisungen, die jeweils mit einem Semikolon enden.
- **string-expression** Ein Ausdruck, der in eine Zeichenfolge aufgelöst wird.
- **table-list** Eine Liste von Tabellennamen, die Korrelationsnamen einschließen können.
- **table-name** Ein Bezeichner für den Namen einer Tabelle.
- **userid** Ein Bezeichner für einen Benutzernamen.
- **variable-name** Ein Bezeichner für einen Variablennamen.
- **window-name** Ein Bezeichner, der einen Fensternamen darstellt. Wird in der Syntax verwendet, die mit der Fensterdefinition zusammenhängt (z.B. die WINDOW-Klausel und Fensterfunktionen wie RANK).

Siehe auch

- „Savepoints innerhalb von Transaktionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Variablen“ auf Seite 85
- „Wahrwert-Suchbedingungen“ auf Seite 67
- „Materialisierte Ansichten“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Hostvariablen in Embedded SQL“ [[SQL Anywhere Server - Programmierung](#)]
- „Ausdrücke“ auf Seite 22
- „SQL-Datentypen“ auf Seite 95
- „Zeichenfolgen“ auf Seite 6
- „Datenbankverbindungen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Steueranweisungen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Suchbedingungen“ auf Seite 42
- „Spezialwerte“ auf Seite 70
- „FROM-Klausel“ auf Seite 863
- „Schlüssel-Joins“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Bezeichner“ auf Seite 4

Syntaxkonventionen

Folgende Konventionen werden bei SQL-Syntaxbeispielen verwendet:

- **Schlüsselwörter** Alle SQL-Schlüsselwörter werden wie die SQL-Anweisung ALTER TABLE im folgenden Beispiel in Großschreibung angezeigt:

ALTER TABLE [*owner.*]*table-name*

- **Platzhalter** Elemente, die durch entsprechende Bezeichner oder Ausdrücke ersetzt werden müssen, werden wie die Wörter *owner* und *table-name* im folgenden Beispiel kursiv angezeigt:

ALTER TABLE [*owner.*]*table-name*

- **Klauselreihenfolge** Wenn die Reihenfolge der optionalen Klauseln in einer SQL-Anweisungssyntax signifikant ist, werden die Klauseln im Hauptteil der Syntax in der Reihenfolge aufgeführt, in der sie angegeben werden müssen, ähnlich wie im folgenden Beispiel:

CREATE SYNCHRONIZATION SUBSCRIPTION [*subscription-name*]
TO *publication-name*
 [**FOR** *ml-username*, ...]
 ...

Wenn die Reihenfolge der optionalen Klauseln in einer SQL-Anweisungssyntax nicht signifikant ist, werden die Klauseln wie in einer Optionenliste separat aufgeführt, ähnlich wie im folgenden Beispiel:

CREATE [OR REPLACE] SPATIAL REFERENCE SYSTEM
srs-name
 [*srs-attribute*] [*srs-attribute* ...]

srs-attribute :
IDENTIFIED BY *srs-id*
 | **DEFINITION** { *definition-string* | **NULL** }
 ...

- **Optionale Bestandteile** Optionale Bestandteile einer Anweisung werden in eckige Klammern gesetzt. Beispiel:

RELEASE SAVEPOINT [*savepoint-name*]

Diese eckigen Klammern zeigen an, dass der *savepoint-name* optional ist. Die eckigen Klammern werden nicht eingegeben.

Bestandteile von Schlüsselwörtern können auch in eckige Klammern gesetzt werden. Die folgende Syntax z.B. zeigt an, dass Sie entweder COMMIT TRAN oder COMMIT TRANSACTION verwenden können:

COMMIT TRAN[SACTION] ...

Nach dem gleichen Prinzip zeigt die folgende Syntax an, dass Sie entweder COMMIT oder COMMIT WORK verwenden können:

COMMIT [WORK]

- **Wiederholungen** Wenn ein Element wiederholt werden kann, wird das entsprechende Listentrennzeichen nachgestellt, dem wiederum drei Punkte folgen, wie im folgenden Beispiel mit *column-constraint*:

ADD *column-definition* [*column-constraint*, ...]

In diesem Fall können Sie keine, eine oder mehrere Spalten-Integritätsregeln angeben. Wenn mehr als ein Element angegeben wird, muss eine Trennung der Elemente durch Kommas erfolgen.

- **Optionen** Wenn aus einer Liste kein oder nur ein Element ausgewählt werden kann, werden die Elemente durch Senkrechtstriche voneinander getrennt, und die komplette Liste wird in eckige Klammern gesetzt.

[**ASC** | **DESC**]

Sie können z.B. entweder ASC, DESC oder keines wählen. Die eckigen Klammern werden nicht eingegeben.

- **Alternativen** Wenn nur eine der vorhandenen Optionen gewählt werden darf, werden die Alternativen in geschweifte Klammern gesetzt.

[**QUOTES** { **ON** | **OFF** }]

Wenn in diesem Fall die Option QUOTES ausgewählt wird, muss entweder ON oder OFF angegeben werden. Eckige und geschweifte Klammern sind nicht einzugeben.

Indikatoren der Anweisungsanwendbarkeit

Auf einige Anweisungstitel folgt ein Indikator in eckigen Klammern, der angibt, wo die Anweisung benutzt werden kann. Es gibt die folgenden Indikatoren:

- **[ESQL]** Diese Anweisung kann in Embedded SQL benutzt werden.
- **[Interactive SQL]** Die Anweisung kann nur in Interactive SQL benutzt werden.
- **[SP]** Diese Anweisung kann in gespeicherten Prozeduren, Triggern oder Batches verwendet werden.
- **[T-SQL]** Diese Anweisung wird für die Kompatibilität mit Adaptive Server Enterprise implementiert. In einigen Fällen kann die Anweisung nicht in gespeicherten Prozeduren verwendet werden, die nicht im Transact-SQL-Format sind. In anderen Fällen wird eine alternative Anweisung empfohlen, die dem SQL/2008-Standard näher kommt, es sei denn, Transact-SQL-Kompatibilität ist erforderlich.
- **[Externer Aufruf]** Die Anweisung wird in aufrufenden externen Funktionen und Prozeduren verwendet.
- **[MobiLink]** Die Anweisung kann nur bei MobiLink-Clients verwendet werden.
- **[SQL Remote]** Die Anweisung kann nur in SQL Remote benutzt werden.
- **[Webdienst]** Die Anweisung wird in Webdienstclients verwendet.

Wenn zweimal eckige Klammern verwendet werden, kann die Anweisung in beiden Umgebungen benutzt werden. Zum Beispiel bedeutet [ESQL][SP], dass eine Anweisung sowohl in Embedded SQL als auch in gespeicherten Prozeduren verwendet werden kann.

SQL-Anweisungen

In den folgenden Abschnitten werden die Syntaxinformationen für alle unterstützten SQL-Anweisungen definiert.

ALLOCATE DESCRIPTOR-Anweisung [ESQL]

Reserviert Speicherplatz für einen SQL-Deskriptorbereich (SQLDA).

Syntax

```
ALLOCATE DESCRIPTOR descriptor-name  
[ WITH MAX { integer | hostvar } ]
```

descriptor-name : *identifizier*

Parameter

WITH MAX-Klausel Damit können Sie die Anzahl der Variablen innerhalb des Deskriptorbereichs angeben. Die Standardgröße ist eins. Trotzdem müssen Sie `fill_sqlda` aufrufen, um den eigentlichen Datenelementen Speicherplatz zuzuweisen, bevor Sie einen Aufruf oder eine Anweisung ausführen können, die auf die Daten innerhalb eines Deskriptorbereichs zugreift.

Bemerkungen

Reserviert Speicherplatz für einen Deskriptorbereich (SQLDA). In Ihrem C-Code müssen Sie vor der Verwendung dieser Anweisung Folgendes angeben:

```
struct sqlda * descriptor_name
```

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „[DEALLOCATE DESCRIPTOR-Anweisung \[ESQL\]](#)“ auf Seite 777
- „[Der SQL-Deskriptor-Bereich \(SQLDA\)](#)“ [[SQL Anywhere Server - Programmierung](#)]

Standards und Kompatibilität

- **SQL/2008** `ALLOCATE DESCRIPTOR` ist Teil der optionalen SQL-Sprachenfunktion B031 (Basic Dynamic SQL) des SQL/2008-Standards.

Beispiel

Das folgende Beispielprogramm enthält ein Beispiel für die Benutzung von ALLOCATE DESCRIPTOR-Anweisungen.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
EXEC SQL INCLUDE SQLCA;
#include "sqldef.h"
EXEC SQL BEGIN DECLARE SECTION;
int      x;
short    type;
int      numcols;
char     string[100];
a_SQL_statement_number stmt = 0;
EXEC SQL END DECLARE SECTION;
int main(int argc, char * argv[]){
    struct sqllda *    sqlda1;
    if( !db_init( &sqlca ) ) {
        return 1;
    }
    db_string_connect( &sqlca,
        "UID=DBA;PWD=sql;DBF=d:\\DB Files\\sample.db");
    EXEC SQL ALLOCATE DESCRIPTOR sqlda1 WITH MAX 25;
    EXEC SQL PREPARE :stmt FROM
        'SELECT * FROM Employees';
    EXEC SQL DECLARE curs CURSOR FOR :stmt;
    EXEC SQL OPEN curs;
    EXEC SQL DESCRIBE :stmt into sqlda1;
    EXEC SQL GET DESCRIPTOR sqlda1 :numcols=COUNT;
    // how many columns?
    if( numcols > 25 ) {
        // reallocate if necessary
        EXEC SQL DEALLOCATE DESCRIPTOR sqlda1;
        EXEC SQL ALLOCATE DESCRIPTOR sqlda1
            WITH MAX :numcols;
        EXEC SQL DESCRIBE :stmt into sqlda1;
    }
    type = DT_STRING; // change the type to string
    EXEC SQL SET DESCRIPTOR sqlda1 VALUE 2 TYPE = :type;
    fill_sqlda( sqlda1 );
    // allocate space for the variables
    EXEC SQL FETCH ABSOLUTE 1 curs
        USING DESCRIPTOR sqlda1;
    EXEC SQL GET DESCRIPTOR sqlda1
        VALUE 2 :string = DATA;
    printf("name = %s", string );
    EXEC SQL DEALLOCATE DESCRIPTOR sqlda1;
    EXEC SQL CLOSE curs;
    EXEC SQL DROP STATEMENT :stmt;
    db_string_disconnect( &sqlca, "" );
    db_fini( &sqlca );
    return 0;
}
```

ALTER DATABASE-Anweisung

Führt ein Upgrade der Datenbank durch, schaltet jConnect-Unterstützung für eine Datenbank ein und aus, kalibriert die Datenbank, ändert die Namen der Transaktions- und Spiegellogdateien oder veranlasst einen Spiegelserver, Eigentümer einer Datenbank zu werden.

Syntax 1 - Upgrade von Komponenten durchführen oder Objekte wiederherstellen

```
ALTER DATABASE UPGRADE
[ PROCEDURE ON ]
[ JCONNECT { ON | OFF } ]
[ RESTART { ON | OFF } ]
[ SYSTEM PROCEDURE AS DEFINER { ON | OFF } ]
```

Syntax 2 - Kalibrierung durchführen

```
ALTER DATABASE {
  CALIBRATE [ SERVER ]
  | CALIBRATE DBSPACE dbspace-name
  | CALIBRATE DBSPACE TEMPORARY
  | CALIBRATE GROUP READ
  | CALIBRATE PARALLEL READ
  | RESTORE DEFAULT CALIBRATION
}
```

Syntax 3 - Namen von Transaktionslog und Transaktionslogspiegel ändern

```
ALTER DATABASE dbfile
ALTER [ TRANSACTION ] LOG
{ ON [ log-name ] [ MIRROR mirror-name ] | OFF }
[ KEY key ]
```

Syntax 4 - Eigentümer einer Datenbank ändern

```
ALTER DATABASE
{ dbname FORCE START
| SET PARTNER FAILOVER }
```

Syntax 5 - Prüfsummeneinstellungen ändern

```
ALTER DATABASE dbfile
CHECKSUM OFF
```

Parameter

PROCEDURE-Klausel Mit dieser Klausel können Sie alle Prozeduren in der Datenbank löschen und neu erstellen, deren Eigentümer dbo oder SYS ist.

JCONNECT-Klausel Um dem jConnect JDBC-Treiber den Zugriff auf Systemkataloginformationen zu ermöglichen, geben Sie JCONNECT ON an. Diese Klausel installiert die Systemobjekte, die jConnect-Unterstützung zur Verfügung stellen. Geben Sie JCONNECT OFF an, wenn Sie die jConnect-Systemobjekte ausschließen möchten. Sie können trotzdem JDBC verwenden, sofern Sie nicht auf Systemdaten zugreifen. JCONNECT ist standardmäßig ON.

RESTART-Klausel RESTART ist standardmäßig ON. Wenn RESTART ON angegeben und der AutoStop-Verbindungsparameter auf "No" gesetzt ist, wird die Datenbank nach einem Upgrade neu gestartet. Andernfalls wird die Datenbank nach einem Upgrade gestoppt.

Klausel SYSTEM PROCEDURE AS DEFINER {ON | OFF} Die SYSTEM PROCEDURE AS DEFINER-Klausel gibt an, ob Systemprozeduren vor Version 16.0, die mit Privilegien verbundene Aufgaben ausführen, mit den Privilegien des Aufrufers oder des Definierers (Eigentümers) ausgeführt werden sollen. ON bedeutet, dass diese Systemprozeduren mit den Privilegien des Definierers (Eigentümers) ausgeführt werden. OFF bedeutet, dass diese Systemprozeduren mit den Privilegien des Aufrufers ausgeführt werden.

Wenn diese Klausel nicht angegeben ist, wird standardmäßig das aktuelle Verhalten der Datenbank beibehalten, für die ein Upgrade durchgeführt wird. Beim Upgrade einer Datenbank vor Version 16.0 bedeutet dies das Ausführen der Prozeduren als Definierer.

Diese Einstellung betrifft nicht benutzerdefinierte Prozeduren oder Prozeduren, die in Version 16.0 oder später eingeführt wurden. Hinweise dazu, welche Systemprozeduren betroffen sind und welche Auswirkungen die Einstellung hat, finden Sie unter „[Systemprozeduren vor Version 16.0 als Aufrufer oder Definierer ausführen](#)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

CALIBRATE [SERVER]-Klausel Damit kalibrieren Sie alle DBSpaces außer den temporären DBSpace. Diese Klausel führt auch die von CALIBRATE PARALLEL READ ausgeführte Arbeit aus.

CALIBRATE DBSPACE-Klausel Mit dieser Klausel kalibrieren Sie den angegebenen DBSpace.

CALIBRATE DBSPACE TEMPORARY-Klausel Mit dieser Klausel kalibrieren Sie den temporären DBSpace.

CALIBRATE GROUP READ-Klausel Mit dieser Klausel kalibrieren Sie Gruppenlesevorgänge für den temporären DBSpace. Sie schreibt große Arbeitstabellen in den temporären DBSpace und benutzt verschiedene Gruppenlesegrößen zur zeitlichen Abstimmung des Lesens der Dateien. Wenn das Hinzufügen von Speicherplatz zur temporären Tabelle das Limit für die Verbindung überschreitet oder der Cache nicht groß genug ist, um eine Kalibrierung mit dem größten Speicherwert zu ermöglichen, schlägt die Kalibrierung fehl und eine Fehlermeldung wird angezeigt.

CALIBRATE PARALLEL READ-Klausel Damit kalibrieren Sie die parallelen I/O-Fähigkeiten von Devices für alle DBSpace-Dateien. Die Klausel CALIBRATE [SERVER] führt ebenfalls diese Kalibrierung durch.

RESTORE DEFAULT CALIBRATION-Klausel Damit setzen Sie das DTT-Modell (Disk Transfer Time) auf die integrierten Standardwerte zurück, die auf typischen Hardware- und Konfigurationseinstellungen basieren.

ALTER [TRANSACTION] LOG-Klausel Damit ändern Sie den Dateinamen des Transaktionslogs oder des Transaktionslogsiegels. Wenn MIRROR *mirror-name* nicht angegeben ist, legt die Klausel den Namen für ein neues Transaktionslog fest. Wenn die Datenbank gerade kein Transaktionslog benutzt, verwendet sie ab jetzt eines. Wenn die Datenbank bereits ein Transaktionslog verwendet, fungiert nun die neue Datei als Transaktionslog.

Wenn MIRROR *mirror-name* angegeben ist, legt die Klausel einen Dateinamen für einen neuen Transaktionslogsiegel fest. Wenn die Datenbank gerade keinen Transaktionslogsiegel benutzt,

verwendet sie ab jetzt einen. Wenn die Datenbank bereits einen Transaktionslog-Spiegel verwendet, fungiert nun die neue Datei als Transaktionslog-Spiegel.

Sie können diese Klausel auch verwenden, um das Transaktionslog bzw. den Transaktionslog-Spiegel zu deaktivieren. Beispiel: ALTER DATABASE ALTER LOG OFF.

KEY-Klausel Damit geben Sie den Chiffrierschlüssel an, der für das Transaktionslog oder den Transaktionslogspiegel verwendet werden soll. Der Chiffrierschlüssel kann entweder eine Zeichenfolge oder ein Variablenname sein. Wenn Sie die ALTER [TRANSACTION] LOG-Klausel für eine stark verschlüsselte Datenbank verwenden, müssen Sie den Chiffrierschlüssel angeben.

dbname FORCE START-Klausel Zwingt einen Datenbankserver, der derzeit als Spiegelserver fungiert, Eigentümer der Datenbank zu werden.

Vorsicht

Die Verwendung der FORCE START-Klausel kann zum Verlust von Transaktionen führen, wenn der Primärserver Transaktionen enthält, die der Spiegelserver nicht hat.

Es wird empfohlen, dass Sie den Primärserver neu starten und ALTER DATABASE mit der SET PARTNER FAILOVER-Klausel ausführen, um einen Failover ohne Verlust von Transaktionen zu erzwingen. Verwenden Sie die FORCE START-Klausel nur als letztes Mittel, wenn der Primärserver nicht neu gestartet werden kann. Siehe „[Fehlerbehandlung: Primärserver kann nicht neu gestartet werden](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Diese Klausel kann von innerhalb einer Prozedur oder eines Ereignisses ausgeführt werden. Sie muss ausgeführt werden, während eine Verbindung zur Dienstprogrammdateiabank auf dem Spiegelserver besteht.

SET PARTNER FAILOVER-Klausel Initiiert einen Datenbankspiegelungs-Failover vom Primärserver auf dem Spiegelserver, ohne den Server zu stoppen. Diese Anweisung muss ausgeführt werden, während eine Verbindung mit der Datenbank auf dem Primärserver besteht, und sie kann aus einer Prozedur oder einem Ereignis heraus ausgeführt werden. Beim Ausführen dieser Anweisung gilt Folgendes:

1. Der Datenbankserver schließt alle Verbindungen mit der Datenbank, einschließlich der Verbindung, die die Anweisung ausgeführt hat.
2. Die Datenbank wird gestoppt und anschließend in der Rolle des Spiegelservers neu gestartet.
3. Wenn die Anweisung in einer Prozedur oder einem Ereignis enthalten ist, werden nachfolgende Anweisungen möglicherweise nicht ausgeführt.

CHECKSUM-Klausel Deaktiviert globale Prüfsummen für die Datenbank. Standardmäßig haben neue Datenbanken aktivierte globale Prüfsummen, während Datenbanken der Version 11 und früher keine globale Prüfsummen aktiviert haben.

Ungeachtet der Einstellung dieser Klausel aktiviert der Datenbankserver das Schreiben von Prüfsummen immer für Datenbanken, die auf Speichermedien wie beispielsweise Wechseldatenträgern laufen, sowie für Datenbanken, die unter Windows Mobile ausgeführt werden, damit eine frühe Erkennung von Beschädigungen der Datenbankdatei möglich ist. Außerdem berechnet der Datenbankserver während der Validierung Prüfsummen für entscheidende Seiten.

Bei Datenbanken, für die keine globalen Prüfsummen aktiviert wurden, können Sie das Schreiben von Prüfsummen unter Verwendung der Optionen für -wc aktivieren.

Bemerkungen

- **Syntax 1** Sie können die ALTER DATABASE UPGRADE-Anweisung als Alternative zum Dienstprogramm zum Upgrade (dbupgrad) verwenden, um ein Upgrade oder eine Aktualisierung einer Datenbank durchzuführen. Standardmäßig wird die Datenbank nach dem Upgrade gestoppt und neu gestartet. Das Transaktionslog wird während des Upgrades archiviert und ein neues Transaktionslog wird erstellt, bevor die Datenbank gestoppt oder neu gestartet wird.

In der Regel beschränken sich Änderungen in Datenbanken zwischen Minor Releases auf zusätzliche Datenbankoptionen sowie kleinere Änderungen an Systemtabellen und Systemprozeduren. Die ALTER DATABASE UPGRADE-Anweisung führt ein Upgrade der Systemtabellen auf die aktuelle Version durch und fügt etwaige neue Datenbankoptionen hinzu. Falls erforderlich werden auch alle Systemprozeduren gelöscht und neu erstellt. Sie können eine Neuerstellung der Systemprozeduren erzwingen, indem Sie die PROCEDURE ON-Klausel angeben.

Eine Fehlermeldung wird zurückgegeben, wenn Sie eine ALTER DATABASE UPGRADE-Anweisung auf einem Datenbankserver ausführen, der derzeit gespiegelt wird.

Sie können mit der ALTER DATABASE UPGRADE-Anweisung auch den Originalzustand (Installationsstatus) von Einstellungen und Systemobjekten wiederherstellen.

Funktionen, die eine physische Neuorganisation der Datenbankdatei erfordern, werden durch das Ausführen einer ALTER DATABASE UPGRADE-Anweisung nicht verfügbar gemacht. Solche Funktionen sind Index-Verbesserungen und Änderungen in der Datenspeicherung. Damit Sie diese Verbesserungen nützen können, müssen Sie Ihre Datenbank entladen und neu laden.

Vorsicht

Sichern Sie Ihre Datenbankdateien, bevor Sie ein Upgrade der Datenbank versuchen.

Wenn Sie den jConnect JDBC-Treiber für den Zugriff auf Systemkataloginformationen verwenden möchten, geben Sie JCONNECT ON (Standardwert) an. Wenn Sie die jConnect-Systemobjekte ausschließen möchten, geben Sie JCONNECT OFF an. Eine Einstellung auf JCONNECT OFF entfernt die Javaunterstützung nicht aus der Datenbank. Sie können trotzdem JDBC verwenden, solange Sie nicht auf Systemkataloginformationen zugreifen. Wenn Sie danach eine neuere Version von jConnect herunterladen, können Sie ein Upgrade der Version in der Datenbank durchführen, indem Sie die ALTER DATABASE UPGRADE JCONNECT ON-Anweisung (erneut) ausführen.

- **Syntax 2** Verwenden Sie Syntax 2, um das vom Optimierer verwendete I/O-Kostenmodell neu zu kalibrieren. Damit wird das Modell "Disk Transfer Time" (DTT) aktualisiert, ein mathematisches Modell der vom Kostenmodell verwendeten I/O-Vorgänge. Wenn Sie das I/O-Kostenmodell neu kalibrieren, ist der Datenbankserver nicht für andere Aufgaben verfügbar. Außerdem ist es von wesentlicher Bedeutung, dass alle anderen Aktivitäten auf dem Computer ruhen. Das Neukalibrieren des Datenbankservers ist ein kostenträchtiger Vorgang und kann eine gewisse Zeit in Anspruch nehmen. Es wird empfohlen, die Standardwerte nicht zu ändern.

Wenn Sie die CALIBRATE PARALLEL READ-Klausel verwenden, wird keine parallele Kalibrierung auf DBSpace-Dateien mit weniger als 10.000 Seiten durchgeführt. Auch wenn der

Datenbankserver während Kalibrierungsvorgängen automatisch alle seine Aktivitäten unterbricht, sollte eine parallele Kalibrierung nur ausgeführt werden, wenn es keine Prozesse auf dem Computer gibt, die signifikant Ressourcen verbrauchen. Nach der Kalibrierung können Sie mithilfe der erweiterten Datenbankeigenschaft IOParallelism die geschätzte maximale Anzahl von in einer DBSpace-Datei zulässigen parallelen I/O-Vorgängen abrufen.

Wenn Sie mehrere ähnliche Geräte installiert haben, können Sie wiederholte, zeitaufwendige Neukalibrierungsaktivitäten vermeiden und eine vorhandene Kalibrierung wiederverwenden, indem Sie sie mit den Systemprozeduren `sa_unload_cost_model` bzw. `sa_load_cost_model` entladen und in eine andere Datenbank übernehmen (laden).

- **Syntax 3** Verwenden Sie die ALTER DATABASE-Anweisung, um die Namen des Transaktionslogs und des Transaktionslog-Spiegels zu ändern, die mit einer Datenbankdatei verbunden sind. Die Datenbank darf nicht laufen, wenn diese Änderungen vorgenommen werden. Das sind die gleichen Veränderungen, die auch mit dem Transaktionslog-Dienstprogramm (dblog) durchgeführt werden können. Sie können diese Anweisung ausführen, während Sie mit der Dienstprogrammdatei oder einer anderen Datenbank verbunden sind, je nach Einstellung der Option `-gu`.

Wenn Sie das Transaktionslog oder den Transaktionslogspiegel einer verschlüsselten Datenbank verändern, müssen Sie den Schlüssel angeben. Sie können die Verwendung des Transaktionslogs nicht beenden, wenn die Datenbank Auditing verwendet. Sobald Sie Auditing ausgeschaltet haben, können Sie die Verwendung des Transaktionslogs beenden. Diese Syntax wird in Prozeduren, Triggern, Ereignissen oder Anwendungsfolgen nicht unterstützt.

Verwenden Sie die BACKUP DATABASE-Anweisung, um das Transaktionslog für eine laufende Datenbank umzubenennen. Beispiel:

```
BACKUP DATABASE DIRECTORY 'directory-name'
TRANSACTION LOG ONLY
TRANSACTION LOG RENAME;
```

- **Syntax 4** ALTER DATABASE...FORCE START muss auf dem Spiegelserver ausgeführt werden, nicht auf dem Primärserver.
- **Syntax 5** Diese Klausel kann nur verwendet werden, um Prüfsummen bei einer Datenbank zu deaktivieren.

ALTER DATABASE UPGRADE wird unter Windows Mobile nicht unterstützt.

Privilegien

- Syntax 1 und 2: Sie müssen das ALTER DATABASE-Systemprivileg haben und als einziger Benutzer mit der Datenbank verbunden sein.
- Syntax 3: Sie müssen das SERVER OPERATOR-Systemprivileg haben sowie die Dateiberechtigungen für die Verzeichnisse, in denen sich das Transaktionslog befindet. Außerdem darf die Datenbank nicht laufen.

Ob Sie die Anweisung ALTER DATABASE *dbfile* ALTER TRANSACTION LOG ausführen können, hängt von der Einstellung für die Datenbankoption `-gu` ab und davon, ob Sie das SERVER OPERATOR-Systemprivileg haben.

- Syntax 4: Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Die zum Ausführen der Anweisung ALTER DATABASE Datenbankname FORCE START erforderlichen Privilegien können durch die Datenbankserveroption -gd geändert werden.

- Syntax 5: Sie müssen das ALTER DATABASE-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Syntax 1: Das Transaktionslog wird während des Upgrades archiviert. Ein neues Transaktionslog wird erstellt, wenn die Datenbank nach dem Upgrade neu gestartet wird.

Syntax 1: Die Datenbank wird am Ende des Upgrades gestoppt und standardmäßig neu gestartet.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** Die ALTER DATABASE-Anweisung wird von Adaptive Server Enterprise unterstützt. Die von Adaptive Server Enterprise unterstützten Klauseln der Anweisung sind jedoch unabhängig von den Klauseln, die von SQL Anywhere unterstützt werden.

Beispiel

Das folgende Beispiel deaktiviert die jConnect-Unterstützung:

```
ALTER DATABASE UPGRADE JCONNECT OFF;
```

Das folgende Beispiel setzt den Dateinamen des Transaktionslogs, das zur Datenbank *demo.db* gehört, auf *mynewdemo.log*:

```
ALTER DATABASE 'C:\\Users\\Public\\Documents\\SQL Anywhere 16\\Samples\\  
demo.db'  
ALTER LOG ON 'mynewdemo.log';
```

Siehe auch

- „CREATE DATABASE-Anweisung“ auf Seite 583
- „CREATE STATISTICS-Anweisung“ auf Seite 729
- „BACKUP-Anweisung“ auf Seite 548
- „Dienstprogramm zum Upgrade (dbupgrad)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Transaktionslog-Dienstprogramm (dblog)“ [*SQL Anywhere Server - Datenbankadministration*]
- „DB_EXTENDED_PROPERTY-Funktion [System]“ auf Seite 226
- „Datenbankserveroption -gk“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -gu“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -wc“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankoption -wc“ [*SQL Anywhere Server - Datenbankadministration*]
- „sa_validate-Systemprozedur“ auf Seite 1352
- „sa_unload_cost_model-Systemprozedur“ auf Seite 1348
- „sa_load_cost_model-Systemprozedur“ auf Seite 1249
- „Fehlerbehandlung: Primärserver kann nicht neu gestartet werden“ [*SQL Anywhere Server - Datenbankadministration*]
- „Benutzerinitiiert Rollentausch (Failover)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankspiegelung“ [*SQL Anywhere Server - Datenbankadministration*]
- „Neuaufbau von Datenbanken“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Upgrade und Neuaufbau in einem Datenbankspiegelungssystem“ [*SQL Anywhere 16 - Änderungen und Upgrades*]
- „DB_EXTENDED_PROPERTY-Funktion [System]“ auf Seite 226
- „jConnect-Systemobjekte in einer Datenbank installieren“ [*SQL Anywhere Server - Programmierung*]
- „Erkennung von Beschädigungen mithilfe von Prüfsummen“ [*SQL Anywhere Server - Datenbankadministration*]
- „jConnect unter Windows Mobile“ [*SQL Anywhere Server - Datenbankadministration*]
- „Validierungs-Dienstprogramm (dbvalid)“ [*SQL Anywhere Server - Datenbankadministration*]
- „VALIDATE-Anweisung“ auf Seite 1118

ALTER DBSPACE-Anweisung

Weist Speicherplatz für einen DBSpace oder das Transaktionslog vorab zu oder aktualisiert den Katalog, wenn eine DBSpace-Datei umbenannt oder verschoben wird.

Syntax

```
ALTER DBSPACE { dbspace-name | TRANSLOG | TEMPORARY }
{ ADD number [ add-unit ]
  | RENAME filename }
```

add-unit :

```
PAGES
| KB
| MB
| GB
| TB
```

Parameter

TRANSLOG-Klausel Sie geben den speziellen DBSpace-Namen TRANSLOG an, um Plattenspeicherplatz für das Transaktionslog vorab zuzuweisen. Die Vorabzuweisung verbessert die Performance, wenn ein rasches Anwachsen des Transaktionslogs zu erwarten ist. Diese Funktion ist sinnvoll, wenn Sie beispielsweise viele BLOBs verarbeiten, wie etwa Bitmaps.

Die Syntax **ALTER DBSPACE** *dbspace-name* **TRANSLOG RENAME** *filename* wird nicht unterstützt.

TEMPORARY-Klausel Sie geben den speziellen DBSpace-Namen TEMPORARY an, um temporären DBSpaces Speicher hinzuzufügen. Wenn einem temporären DBSpace Speicher hinzugefügt wird, materialisiert sich der zusätzliche Speicher in der entsprechenden temporären Datei sofort. Die Vorabzuweisung von Speicherplatz für den temporären DBSpace einer Datenbank kann die Performance bei Abfragen mit komplexen Ausführungsprozessen, die umfangreiche Arbeitstabellen benutzen, deutlich verbessern.

ADD-Klausel Eine ALTER DBSPACE-Anweisung mit der ADD-Klausel wird für die Vorabzuweisung von Plattenspeicher für einen DBSpace verwendet. Sie erweitert die zugehörige Datenbankdatei um die angegebene Größe, und zwar in Einheiten von Seiten, Kilobyte (kB), Megabyte (MB), Gigabyte (GB) oder Terabyte (TB). Wenn Sie keine Einheit angeben, wird PAGES (Seiten) als Standard verwendet. Die Seitengröße einer Datenbank wird bei ihrer Erstellung festgelegt.

Wenn der Platzbedarf nicht vorab zugewiesen wurde, werden Datenbankdateien jedes Mal, wenn Platz benötigt wird, um 256 kB für die Seitengrößen 2 kB, 4 kB und 8 kB erweitert, sowie um rund 32 Seiten für andere Seitengrößen. Die Vorabzuweisung von Speicherplatz kann die Performance zum Laden großer Datenmengen verbessern und sorgt außerdem dafür, dass die Datenbankdateien im Dateisystem zusammenhängend abgelegt werden.

Mit dieser Klausel können Sie den vordefinierten DBSpaces (SYSTEM, TEMPORARY, TEMP, TRANSLOG und TRANSLOGMIRROR) Speicherplatz zuweisen.

RENAME-Klausel Wenn Sie nicht die Stammdatei, sondern eine andere Datenbankdatei umbenennen oder in ein anderes Verzeichnis bzw. Device verschieben, müssen Sie ALTER DBSPACE in Verbindung mit der RENAME-Klausel verwenden, um zu gewährleisten, dass SQL Anywhere die Datei finden kann, wenn die Datenbank gestartet wird. Der Parameter *filename* kann ein Zeichenfolgenliteral oder eine Variable sein.

🔥 **Cloud-Hinweis:** Wenn Sie für Tenant-Datenbanken in einer Cloud den Standort eines DBSpace festlegen, können nur einen Dateinamen angeben. Sie können keinen Verzeichnispfad angeben.

Die Namensänderung wirkt sich folgendermaßen aus:

- Wenn der DBSpace bereits geöffnet war, bevor die Anweisung ausgeführt wurde (d.h., Sie haben die Datei noch nicht umbenannt), bleibt er zugreifbar. Der im Katalog gespeicherte Name wird allerdings aktualisiert. Nachdem die Datenbank gestoppt wird, müssen Sie die Datei umbenennen, damit ihr Name mit dem in der RENAME-Klausel verwendeten Namen übereinstimmt. Andernfalls würde der Name nicht dem DBSpace-Namen im Katalog entsprechen und der Datenbankserver wäre nicht in der Lage, den DBSpace beim nächsten Start der Datenbank zu öffnen.
- Wenn der DBSpace nicht geöffnet war, als die Anweisung ausgeführt wurde, versucht der Datenbankserver, ihn zu öffnen, nachdem er den Katalog aktualisiert hat. Wenn der DBSpace geöffnet

werden kann, wird er zugreifbar. Es wird kein Fehler zurückgegeben, wenn der DBSpace nicht geöffnet werden kann.

Um zu bestimmen, ob ein DBSpace geöffnet ist, führen Sie die nachfolgende Anweisung aus. Wenn das Ergebnis NULL ist, ist der DBSpace nicht geöffnet.

```
SELECT DB_EXTENDED_PROPERTY('FileSize','dbspace-name');
```

Die Verwendung von ALTER DBSPACE mit RENAME ist für den Stamm-DBSpace SYSTEM wirkungslos. Die RENAME-Klausel wird zum Ändern des Namens der Transaktionslogdatei nicht unterstützt. Sie können die BACKUP DATABASE-Anweisung verwenden, um das Transaktionslog für eine laufende Datenbank umzubenennen. Beispiel:

```
BACKUP DATABASE DIRECTORY 'directory-name'  
TRANSACTION LOG ONLY  
TRANSACTION LOG RENAME;
```

Bemerkungen

Jede Datenbank ist in einer oder mehreren Dateien enthalten. Ein DBSpace ist eine zusätzliche Datei mit einem logischen Namen für jede einzelne Datenbankdatei, die mehr Daten enthalten kann als die Hauptdatenbankdatei allein. ALTER DBSPACE ändert den Stamm-DBSpace (auch Stammdatei genannt) oder eine zusätzliche DBSpace. Die DBSpace-Namen für die Datenbank befinden sich in der Systemansicht SYSDBSPACE. Der DBSpace-Name der Stammdatenbankdatei ist SYSTEM.

Wenn eine Mehrdateien-Datenbank gestartet wird, teilt die Startzeile oder die ODBC-Datenquellen-Beschreibung SQL Anywhere mit, wo die Stammdatenbankdatei zu finden ist. Die Systemtabellen befinden sich in der Hauptdatenbankdatei. SQL Anywhere durchsucht diese Systemtabellen nach der Position der anderen DBSpaces, und öffnet diese dann jeweils. Sie können angeben, in welchem DBSpace neue Tabellen erstellt werden sollen, indem Sie die default_DBSpace-Option einstellen.

Privilegien

Sie müssen das MANAGE ANY DBSPACE-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE DBSPACE-Anweisung“ auf Seite 593
- „BACKUP-Anweisung“ auf Seite 548
- „SYSDBSPACE-Systemansicht“ auf Seite 1444
- „default_dbspace-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Vordefinierte DBSpaces“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankdateitypen“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Mit dem folgenden Beispiel wird die Größe des SYSTEM-DBSpaces um 200 Seiten erhöht:

```
ALTER DBSPACE system  
ADD 200;
```

Mit dem folgenden Beispiel wird die Größe des SYSTEM-DBSpaces um 400 MB erhöht:

```
ALTER DBSPACE system  
ADD 400 MB;
```

Im folgenden Beispiel wird der Dateiname für den fiktiven DBSpace system_2 geändert:

```
ALTER DBSPACE system_2  
RENAME 'e:\db\dbspace2.db';
```

ALTER DOMAIN-Anweisung

Benennt eine benutzerdefinierte Domäne bzw. einen benutzerdefinierten Datentyp um.

Syntax

```
ALTER { DOMAIN | DATATYPE } user-type  
RENAME new-name
```

Bemerkungen

Wenn Sie diese Anweisung ausführen, wird der Name der benutzerdefinierten Domäne bzw. des Datentyps in der Systemtabelle ISYSUSERTYPE aktualisiert.

Hinweis

Alle Prozeduren, Trigger, Ansichten oder Ereignisse, die sich auf die benutzerdefinierte Domäne bzw. den Datentyp beziehen, müssen neu erstellt werden, weil sie sich sonst weiterhin auf den alten Namen beziehen würden.

Privilegien

Sie müssen Eigentümer der Domäne sein oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Domäne
- ALTER DATATYPE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „SYSUSERTYPE-Systemansicht“ auf Seite 1510
- „CREATE DOMAIN-Anweisung“ auf Seite 598
- „Domänen“ auf Seite 138
- „Domänen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Die ALTER DOMAIN-Anweisung ist die optionale SQL-Funktion F711 des SQL/2008-Standards. Im Standard kann ALTER DOMAIN jedoch geänderte DEFAULT- oder CHECK-Integritätsregeln für eine vorhandene Domäne angeben. Keiner dieser beiden Vorgänge wird in SQL Anywhere unterstützt. Funktion F711 bietet keine Unterstützung für das Umbenennen einer Domäne.

Beispiel

Im folgenden Beispiel wird die fiktive Domäne "Address" in "MailingAddress" umbenannt:

```
ALTER DOMAIN Address RENAME MailingAddress;
```

ALTER EVENT-Anweisung

Ändert die Definition eines Ereignisses oder seiner zugehörigen Verarbeitungsroutine zum Automatisieren vordefinierter Aktionen bzw. die Definition von geplanten Aktionen. Sie können diese Anweisung auch verwenden, um die Definition eines Event-Handlers zu verbergen.

Syntax 1 - Ereignis ändern

```
ALTER EVENT [ owner.]event-name
[ AT { CONSOLIDATED | REMOTE | ALL } ]
[ FOR { PRIMARY | ALL } ]
[ { DELETE TYPE
  | TYPE event-type
  | WHERE { trigger-condition | NULL }
  | { ADD | ALTER | DELETE } SCHEDULE schedule-spec } ]
[ ENABLE | DISABLE ]
[ [ ALTER ] HANDLER compound-statement | DELETE HANDLER ]
```

event-type :

```
BackupEnd
Connect
ConnectFailed
DatabaseStart
DBDiskSpace
Deadlock
"Disconnect"
GlobalAutoincrement
GrowDB
GrowLog
GrowTemp
LogDiskSpace
RAISERROR
ServerIdle
TempDiskSpace
```

trigger-condition :

```
event_condition( condition-name ) { = | < | > | != | <= | >= } value
```

schedule-spec :

```
[ schedule-name ]
{ START TIME start-time | BETWEEN start-time AND end-time }
```

```
[ EVERY period { HOURS | MINUTES | SECONDS } ]  
[ ON { ( day-of-week, ... ) | ( day-of-month, ... ) } ]  
[ START DATE start-date ]
```

event-name | *schedule-name* : *identifier*

day-of-week : *string*

value | *period* | *day-of-month* : *integer*

start-time | *end-time* : *time*

start-date : *date*

Syntax 2 - Definition eines Event-Handlers verbergen

```
ALTER EVENT event-name SET HIDDEN
```

Parameter

AT-Klausel Verwenden Sie diese Klausel, um die Angabe in Bezug zur Datenbank zu ändern, bei der das Ereignis verarbeitet wird.

FOR-Klausel Verwenden Sie diese Klausel in einem Datenbankspiegelungssystem oder einem Scale-Out-System mit Schreibschutz, um die Datenbanken einzuschränken, in denen das Ereignis verarbeitet wird.

DELETE TYPE-Klausel Verwenden Sie diese Klausel, um eine Zuordnung des Ereignisses zu einem Ereignistyp zu entfernen.

ADD | ALTER | DELETE SCHEDULE-Klausel Verwenden Sie diese Klausel, um die Definition eines Abfolgeplans zu ändern. Es kann nur ein Abfolgeplan in einer einzigen ALTER EVENT-Anweisung geändert werden.

WHERE-Klausel Verwenden Sie diese Klausel, um die Triggerbedingung, unter der ein Ereignis ausgelöst wird, zu ändern. Die Option WHERE NULL löscht eine Bedingung.

START TIME-Klausel Mit dieser Klausel geben Sie die Startzeit und optional die Endzeit für das Ereignis ein. Die Parameter *start-time* und *end-time* sind Zeichenfolgen (z.B. '12:34:56'). Variablen und Ausdrücke sind nicht zulässig (z.B. NOW()).

START DATE-Klausel Mit dieser Klausel geben Sie das Startdatum für das Ereignis ein. Der Parameter *start-date* ist eine Zeichenfolge. Variablen und Ausdrücke sind nicht zulässig (z.B. TODAY()).

SET HIDDEN-Klausel Mit dieser Klausel blenden Sie die Definition eines Event-Handlers aus. Die Angabe der SET HIDDEN-Klausel führt zur permanenten Verschleierung der Event-Handler-Definition, die in der Aktionsspalte der Systemtabelle ISYSEVENT gespeichert ist.

Bemerkungen

Mit dieser Anweisung können Sie eine Ereignisdefinition ändern, die mit CREATE EVENT erstellt wurde. Sie können die Anweisung folgendermaßen verwenden:

- Definition eines Event-Handlers verbergen
- Einen Event-Handler ohne Triggerbedingung oder Abfolgeplan während der Entwicklungsphase definieren und testen und anschließend dann die Bedingungen zum Ausführen mithilfe von ALTER EVENT hinzufügen, sobald der Event-Handler fertiggestellt ist

Wenn Sie ein Ereignis ändern müssen, können Sie es deaktivieren, während es läuft, indem Sie die Anweisung ALTER EVENT...DISABLE ausführen. Um ein Element in Sybase Central zu deaktivieren, rechtsklicken Sie auf das Ereignis und deaktivieren die Option **Aktiviert**. Die Deaktivierung des Ereignisses unterbricht die Ausführung des aktuellen Event-Handlers nicht. Der Event-Handler läuft bis zum Abschluss weiter. Wenn der Event-Handler abgeschlossen ist, wird er erst wieder gestartet, wenn Sie ihn wieder aktivieren. Sie können die Definition ändern und wieder aktivieren. Um zu ermitteln, welche Ereignisse laufen, führen Sie folgende Anweisung aus:

```
SELECT *
FROM dbo.sa_conn_info()
WHERE CONNECTION_PROPERTY( 'EventName', Number ) = 'event-name';
```

Privilegien

Sie müssen entweder das ALTER ANY OBJECT-Systemprivileg oder das MANAGE ANY EVENT-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Siehe auch

- „SYSEVENT-Systemansicht“ auf Seite 1446
- „BEGIN-Anweisung“ auf Seite 557
- „CREATE EVENT-Anweisung“ auf Seite 606
- „Systemereignisse“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Ausblenden eines Event-Handlers“ [[SQL Anywhere Server - Datenbankadministration](#)]

ALTER EXTERNAL ENVIRONMENT-Anweisung

Gibt den Speicherort einer externen Umgebung wie Java, PHP oder Perl an.

Syntax

```
ALTER EXTERNAL ENVIRONMENT environment-name
LOCATION location-string
```

environment-name :

```
JAVA
| PERL
| PHP
| CLR
| C_ESQL32
```

C_ESQL64
C_ODBC32
C_ODBC64
DBMLSYNC

Parameter

environment-name Mit *environment-name* legen Sie die externe Umgebung fest, die Sie ändern möchten.

LOCATION-Klausel Mit der LOCATION-Klausel geben Sie den Speicherort auf dem Computer des Datenbankservers an, wo die Programmdatei für die externe Umgebung gefunden werden kann. Sie enthält den Namen des Programms oder der Binärdatei. Der Pfad kann voll qualifiziert oder relativ sein. Wenn der Pfad relativ ist, muss sich das Programm oder die Binärdatei an einem Speicherort befinden, an dem sie der Server finden kann.

Bemerkungen

Der Speicherort des Dienstprogramms dbmlsync kann mithilfe dieser Anweisung festgelegt werden. Wenn die ausführbare dbmlsync-Datei während der Synchronisation nicht mit der PATH-Umgebungsvariablen des Datenbankservers gefunden werden kann, verwenden Sie diese Anweisung, um den Speicherort des Programms aufzuzeichnen.

Privilegien

Sie müssen das MANAGE ANY EXTERNAL ENVIRONMENT-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Unterstützung für externe Umgebungen in SQL Anywhere“ [[SQL Anywhere Server - Programmierung](#)]
- „START EXTERNAL ENVIRONMENT-Anweisung“ auf Seite 1066
- „STOP EXTERNAL ENVIRONMENT-Anweisung“ auf Seite 1075
- „INSTALL EXTERNAL OBJECT-Anweisung“ auf Seite 923
- „REMOVE EXTERNAL OBJECT-Anweisung“ auf Seite 996
- „SYSEXTNENV-Systemansicht“ auf Seite 1448
- „SYSEXTNENVOBJECT-Systemansicht“ auf Seite 1450

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird der Speicherort des Perl-Programms angegeben, das bei Verwendung von Perl als externe Umgebung eingesetzt werden soll.

```
ALTER EXTERNAL ENVIRONMENT PERL  
LOCATION 'c:\\Perl64\\bin\\perl.exe';
```

ALTER FUNCTION-Anweisung

Ändert eine Funktion.

Syntax 1 - Ändert die Definition einer Funktion

ALTER FUNCTION [*owner.*]*function-name* *function-definition*

function-definition : *CREATE FUNCTION*-Syntax

Syntax 2 - Verschleiert eine Funktionsdefinition

ALTER FUNCTION [*owner.*]*function-name*
SET HIDDEN

Syntax 3 - Kompiliert eine Funktion neu

ALTER FUNCTION [*owner.*]*function-name*
RECOMPILE

Bemerkungen

Nehmen Sie die vollständige neue Funktion in die ALTER FUNCTION-Anweisung auf.

- **Syntax 1** Die ALTER FUNCTION-Anweisung unterscheidet sich in der Syntax nur durch das erste Wort von der CREATE FUNCTION-Anweisung.

Mit ALTER FUNCTION bleiben bestehende Privilegien für die betreffende Funktion unverändert. Im Gegensatz dazu werden beim Ausführen von DROP FUNCTION, gefolgt von CREATE FUNCTION, EXECUTE-Privilegien neu zugewiesen.

- **Syntax 2** Sie können SET HIDDEN verwenden, um die Definition der zugehörigen Funktion zu verschleiern und somit unlesbar zu machen. Die Funktion kann aus Ihrer Datenbank entladen und in andere Datenbanken geladen werden.

Wenn SET HIDDEN verwendet wird, ist die Definition der Funktion bei der Fehlerbehandlung mit dem Debugger nicht zu sehen und sie wird auch nicht in den Prozedurprofilen angezeigt.

Hinweis

Diese Einstellung ist irreversibel. Es wird dringend empfohlen, die ursprüngliche Funktionsdefinition außerhalb der Datenbank aufzubewahren.

- **Syntax 3** Verwenden Sie die RECOMPILE-Syntax, um eine benutzerdefinierte SQL-Funktion neu zu kompilieren. Wenn Sie eine Funktion neu kompilieren, wird die im Katalog gespeicherte Definition neuerlich syntaktisch analysiert und die Syntax wird geprüft. Die Quelle des Funktionscodes wird durch die Neukompilierung nicht verändert. Wenn Sie eine Funktion neu kompilieren, bleibt die durch die SET HIDDEN-Klausel verschleierte Definition weiterhin verschleiert und unlesbar.

Privilegien

Sie müssen Eigentümer der Funktion sein oder eines der folgenden Privilegien haben:

- ALTER ANY PROCEDURE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Um eine Funktion als extern definieren zu können, benötigen Sie das CREATE EXTERNAL REFERENCE-Systemprivileg.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE FUNCTION-Anweisung“ auf Seite 633
- „CREATE FUNCTION-Anweisung [externer Aufruf]“ auf Seite 617
- „CREATE FUNCTION-Anweisung [Webdienst]“ auf Seite 624
- „ALTER PROCEDURE-Anweisung“ auf Seite 483
- „DROP FUNCTION-Anweisung“ auf Seite 809
- „Verbergen des Inhalts von Prozeduren, Funktionen, Triggern, Ereignissen oder Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. ALTER FUNCTION ist die optionale SQL-Sprachenfunktion F381 des SQL/2008-Standards. Im SQL-Standard kann ALTER FUNCTION jedoch nicht dazu verwendet werden, eine PSM-Funktionsdefinition (Persistent Stored Module, beständiges gespeichertes Modul) neu festzulegen. SQL/2008 umfasst keine Unterstützung für SET HIDDEN oder RECOMPILE.

Beispiel

In diesem Beispiel wird MyFunction erstellt und geändert. Mit der SET HIDDEN-Klausel wird die Funktionsdefinition verschleiert und unlesbar. Um dieses Beispiel ausführen zu können, benötigen Sie auch das CREATE PROCEDURE-Systemprivileg, da eine Funktion erstellt wird, bevor sie geändert wird.

```
CREATE FUNCTION MyFunction(  
    firstname CHAR(30),  
    lastname CHAR(30) )  
RETURNS CHAR(61)  
BEGIN  
    DECLARE name CHAR(61);  
    SET name = firstname || ' ' || lastname;  
    RETURN (name);  
ALTER FUNCTION MyFunction SET HIDDEN;  
END;
```

ALTER INDEX-Anweisung

Benennt einen Index, einen Primärschlüssel oder einen Fremdschlüssel um oder ändert das Clustered-Merkmal eines Indexes.

Syntax

```
ALTER { INDEX index-name
| [ INDEX ] FOREIGN KEY role-name
| [ INDEX ] PRIMARY KEY }
ON [ owner. ] object-name { REBUILD | rename-clause | cluster-clause }
```

object-name : *table-name* | *materialized-view-name*

rename-clause : RENAME { AS | TO } *new-index-name*

cluster-clause : CLUSTERED | NONCLUSTERED

Parameter

rename-clause Geben Sie den neuen Namen für den Index, Primärschlüssel oder Fremdschlüssel an.

Wenn Sie den zugrunde liegenden Index für einen Fremdschlüssel oder Primärschlüssel umbenennen, wird der Name der entsprechenden RI-Integritätsregel für den Index nicht geändert. Der Rollenname des Fremdschlüssels, soweit er mit dem Indexnamen übereinstimmt, wird geändert. Verwenden Sie die ALTER TABLE-Anweisung zum Umbenennen der RI-Integritätsregel, falls erforderlich.

cluster-clause Geben Sie an, ob der Index in CLUSTERED oder NONCLUSTERED geändert werden soll. Nur ein Index der Tabelle kann "Clustered" sein.

REBUILD-Klausel Verwenden Sie diese Klausel, um einen Index neu aufzusetzen, anstatt ihn zu löschen und neu zu erstellen.

Bemerkungen

Die ALTER INDEX-Anweisung führt zwei Aufgaben aus:

- Sie kann verwendet werden, um einen Index, einen Primärschlüssel oder einen Fremdschlüssel umzubenennen.
- Sie kann eingesetzt werden, um den Indextyp von Nonclustered in Clustered bzw. umgekehrt zu ändern.

Die ALTER INDEX-Anweisung kann verwendet werden, um die Clustered-Angabe des Indexes zu ändern, jedoch nicht, um die Daten neu zu organisieren. Nur ein Index der Tabelle oder materialisierten Ansicht kann Clustered sein.

ALTER INDEX kann nicht zum Ändern eines Indexes für eine lokale, temporäre Tabelle verwendet werden. Wenn Sie dies versuchen, erhalten Sie die Fehlermeldung "Index nicht gefunden".

Diese Anweisung kann nicht ausgeführt werden, wenn mit der WITH HOLD-Klausel geöffnete Cursor vorhanden sind, die entweder Anweisungs- oder Transaktions-Snapshots verwenden.

Privilegien

Wenn Sie einen Index für eine Tabelle ändern möchten, müssen Sie Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- REFERENCES-Privileg für die Tabelle
- ALTER ANY INDEX-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Wenn Sie einen Index für eine materialisierte Ansicht ändern möchten, müssen Sie Eigentümer der materialisierten Ansicht sein oder eines der folgenden Privilegien haben:

- ALTER ANY INDEX-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Löscht den Inhalt der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** in Interactive SQL. Schließt alle Cursor für die aktuelle Verbindung. Wenn ALTER INDEX REBUILD angegeben ist, wird ein Checkpoint gesetzt.

Siehe auch

- „CREATE INDEX-Anweisung“ auf Seite 638
- „ALTER TABLE-Anweisung“ auf Seite 516
- „Snapshot-Isolation“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung ändert IX_product_name in einen Clustered-Index:

```
ALTER INDEX IX_product_name ON GROUP0.Products  
CLUSTERED;
```

Die folgende Anweisung benennt den index IX_product_name der Products-Tabelle in ixProductName um:

```
ALTER INDEX IX_product_name ON GROUP0.Products  
RENAME TO ixProductName;
```

ALTER LDAP SERVER-Anweisung

Ändert ein LDAP-Serverkonfigurationsobjekt.

Syntax

```
ALTER LDAP SERVER ldapua-server-name  
[ ldapua-server-attrs ... ]  
[ WITH { SUSPEND | ACTIVATE | REFRESH } ]
```

ldapua-server-attribs :

```
SEARCH DN search-dn-attributes ...
| AUTHENTICATION URL { 'url-string' | NULL }
| CONNECTION TIMEOUT timeout-value
| CONNECTION RETRIES retry-value
| TLS { ON | OFF }
```

search-dn-attributes :

```
URL { 'url-string' | NULL }
| ACCESS ACCOUNT { 'dn-string' | NULL }
| IDENTIFIED BY ( 'password' | NULL }
| IDENTIFIED BY ENCRYPTED { encrypted-password | NULL }
```

Parameter

SEARCH DN-Klausel Es gibt keine Standardwerte für Parameter in der SEARCH DN-Klausel.

- **URL** Verwenden Sie diese Klausel zum Angeben des Hosts (als Namen oder als IP-Adresse), der Portnummer und der auszuführenden Suche, um den **LDAP-Distinguished Name (DN)** für eine bestimmte Benutzer-ID nachzuschlagen. *url-string* wird vor dem Speichern in ISYSLDAPSERVER im Hinblick auf die Richtigkeit der LDAP-URL-Syntax validiert. Die maximale Größe für diese Zeichenfolge ist 1024 Byte.

Das Format der *url-string* muss dem LDAP-URL-Standard entsprechen. Siehe <http://www.isode.com/whitepapers/ldap-standards.html>.

- **ACCESS ACCOUNT** Verwenden Sie diese Klausel zum Angeben des LDAP-Distinguished Name (DN), der vom Datenbankserver verwendet wird, um eine Verbindung mit dem LDAP-Server herzustellen. Dies ist kein SQL Anywhere-Benutzer, sondern ein Benutzer, der im LDAP-Server speziell für das Anmelden beim LDAP-Server erstellt wurde. Dieser Benutzer muss Berechtigungen im LDAP-Server haben, um über Benutzer-IDs an den in der SEARCH DN URL-Klausel angegebenen Orten nach DN's suchen zu können. Die maximale Größe für diese Zeichenfolge ist 1024 Byte.
- **IDENTIFIED BY** Verwenden Sie diese Klausel, um das Kennwort anzugeben, das dem Benutzer durch ACCESS ACCOUNT identifizierten Benutzer zugeordnet ist. Die maximal zulässige Länge beträgt 255 Byte und kann nicht auf NULL gesetzt werden.
- **IDENTIFIED BY ENCRYPTED** Verwenden Sie diese Klausel, um das Kennwort anzugeben, das dem durch ACCESS ACCOUNT identifizierten Benutzer zugeordnet ist. Dieses wird in verschlüsselter Form bereitgestellt und ist als Binärwert auf der Festplatte gespeichert. Die maximale Größe des Binärwerts beträgt 289 Byte und kann nicht auf NULL gesetzt werden. Mithilfe von IDENTIFIED BY ENCRYPTED kann das Kennwort abgerufen und verwendet werden, ohne dass es bekannt wird.

AUTHENTICATION URL-Klausel Geben Sie mithilfe dieser Klausel den Host als Namen oder als IP-Adresse sowie die Portnummer des LDAP-Servers an, der zum Authentifizieren eines Benutzers verwendet werden soll. Der bei einer früheren DN-Suche abgerufene DN des Benutzers und das Benutzerkennwort werden verwendet, um eine neue Verbindung an die Authentifizierungs-URL zu binden. Eine erfolgreiche Verbindung mit dem LDAP-Server gilt als Nachweis der Identität des Benutzers, der die Verbindung herstellt. Es gibt keinen Standardwert für diesen Parameter.

Größenbeschränkungen für diese Zeichenfolge finden Sie unter „[SYSLDAPSERVER-Systemansicht](#)“ auf Seite 1461.

CONNECTION TIMEOUT-Klausel Verwenden Sie diese Klausel, um das Verbindungs-Timeout des LDAP-Servers in Millisekunden anzugeben, sowohl für DN-Suchvorgänge als auch für die Authentifizierung. Der Standardwert beträgt 10 Sekunden.

CONNECTION RETRIES-Klausel Verwenden Sie diese Klausel, um die Anzahl von Wiederholungen für Verbindungen mit dem LDAP-Server anzugeben, sowohl für DN-Suchvorgänge als auch für die Authentifizierung. Der gültige Wertebereich liegt zwischen 1 und 60. Der Standardwert ist 3.

TLS-Klausel Verwenden Sie diese Klausel, um die Verwendung des TLS-Protokolls für Verbindungen mit dem LDAP-Server anzugeben, sowohl für DN-Suchvorgänge als auch für die Authentifizierung. Die gültigen Werte sind ON und OFF. Der Standardwert ist OFF. Wenn Sie das Secure LDAP-Protokoll verwenden möchten, geben Sie am Anfang der URL `ldaps://` statt `ldap://` ein. Die TLS-Option muss auf OFF gesetzt sein, wenn Sie Secure LDAP verwenden.

WITH-Klausel

- **WITH SUSPEND** Setzt den Zustand der LDAP-Serverkommunikation auf SUSPENDED (Wartungsmodus). Die Verbindungen mit dem LDAP-Server werden geschlossen und die Authentifizierung beim LDAP-Server wird nicht mehr ausgeführt.
- **WITH ACTIVATE** Aktiviert den LDAP-Server für die sofortige Verwendung. Damit wird der Zustand der LDAP-Serverkommunikation in READY geändert.
- **WITH REFRESH** Initialisiert die LDAP-Benutzerauthentifizierung neu. Dieser Befehl ändert nicht den Zustand des LDAP-Servers, wenn dieser SUSPENDED lautet. Wenn WITH REFRESH für einen LDAP-Server im Zustand READY oder ACTIVE angegeben wird, werden Verbindungen mit dem LDAP-Server geschlossen. Anschließend werden die Werte der Serveroptionen erneut aus der ISYSLDAPSERVER-Systemtabelle gelesen und auf neue Verbindungen mit dem LDAP-Server sowie auf eingehende Authentifizierungsanforderungen an den Datenbankserver angewendet.

Bemerkungen

ALTER LDAP SERVER...WITH REFRESH wird häufig auf einem LDAP-Server im Zustand ACTIVE oder READY verwendet, um gehaltene Ressourcen freizugeben oder um Änderungen an Dateien erneut zu lesen, die außerhalb des Servers vorgenommen wurden, z.B. Änderungen am Inhalt der durch die `trusted_certificates_file`-Datenbankoption angegebenen Datei.

Bei anderen Zustandswerten hat ALTER LDAP SERVER...WITH REFRESH keine Auswirkungen.

Privilegien

Sie müssen das MANAGE ANY LDAP SERVER-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „LDAP-Benutzerauthentifizierung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „CREATE LDAP SERVER-Anweisung“ auf Seite 642
- „DROP LDAP SERVER-Anweisung“ auf Seite 811
- „VALIDATE LDAP SERVER-Anweisung“ auf Seite 1116

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird der fiktive LDAP-Server apps_primary gestoppt.

```
ALTER LDAP SERVER apps_primary WITH SUSPEND;
```

Im folgenden Beispiel wird der LDAP Server apps_primary so geändert, dass er eine andere URL für die Authentifizierung auf dem Host fairfax, Portnummer 1066, verwendet. Außerdem wird die Anzahl der Verbindungsversuche auf 10 gesetzt und der Server wird aktiviert.

```
ALTER LDAP SERVER apps_primary
AUTHENTICATION URL 'ldap://fairfax:1066/'
CONNECTION RETRIES 10
WITH ACTIVATE;
```

ALTER LOGIN POLICY-Anweisung

Ändert eine bestehende Login-Richtlinie.

Syntax

```
ALTER LOGIN POLICY policy-name policy-options
```

```
policy options :
policy-option [ policy-option ... ]
```

```
policy-option :
policy-option-name = policy-option-value
```

```
policy-option-value :
{ UNLIMITED
| DEFAULT
| legal-option-value }
```

Parameter

policy-name Der Name der Login-Richtlinie. Geben Sie "root" an, um die Root-Login-Richtlinie zu ändern.

policy-option-name Der Name der Richtlinienoption.

policy-option-value Der Wert, der der Login-Richtlinienoption zugewiesen wird. Wenn Sie UNLIMITED angeben, werden keine Limits verwendet. Wenn Sie DEFAULT angeben, werden die Standardlimits verwendet.

Richtlinienoptionsname	Beschreibung	Standardwert	Gilt für:
auto_unlock_time	Die Zeitspanne, nach der gesperrte Konten automatisch freigegeben werden.	Unlimited	Alle Benutzer außer denjenigen mit MANAGE ANY USER-Systemprivileg
change_password_dual_control	Wenn der Wert für diese Option ON ist, muss das Kennwort von zwei Administratoren festgelegt werden. Die Einstellung für die verify_password_function-Option wird ignoriert, wenn diese Option auf ON gesetzt ist, weil das Kennwort separat in zwei Teilen konfiguriert wird. Es wird keine Überprüfung durchgeführt.	OFF	Alle Benutzer
ldap_primary_server	Der Name des LDAP-Primärservers.	(keiner)	Alle Benutzer
ldap_secondary_server	Der Name des LDAP-Sekundärservers.	(keiner)	Alle Benutzer
ldap_auto_failback_period	Die Zeitspanne in Minuten, nach der ein automatischer Failback auf den Primärserver versucht wird.	15 Minuten	Alle Benutzer
ldap_failover_to_std	Angabe, ob die Authentifizierung mit Standardauthentifizierung zulässig sein soll, wenn die Authentifizierung über den LDAP-Server fehlschlägt, weil der Distinguished Name (DN) für einen Benutzer nicht gefunden wird, Systemressourcen fehlen bzw. Netzwerkausfälle, Verbindungs-Timeouts oder ähnliche Systemfehler auftreten. Diese Einstellung lässt nicht zu, dass bei Rückgabe eines tatsächlichen Authentifizierungsfehlers durch einen LDAP-Server ein Failover auf die Standardauthentifizierung erfolgt (wie es der Fall ist, wenn der Benutzer gefunden wird, aber das angegebene Kennwort nicht passt).	ON	Alle Benutzer

Richtlinienoptionsname	Beschreibung	Standardwert	Gilt für:
ldap_refresh_dn	<p>Zum Zeitpunkt der Festlegung dieser Richtlinienoption durch eine CREATE LOGIN POLICY- oder ALTER LOGIN POLICY-Anweisung wird der aktuelle Zeitwert mit der Login-Richtlinie gespeichert. Dieser Wert ist der Zeitstempel, der bei der Benutzerauthentifizierung mit dem in ISYSUSER für den betreffenden Benutzer gefundenen user_dn_cached_at-Wert verglichen wird. Wenn der Wert in der Richtlinie neuer ist als der user_dn_cached_at-Wert in ISYSUSER, wird nach dem Distinguished Name (DN) eines Benutzers gesucht, um den user_dn-Wert in ISYSUSER zu aktualisieren.</p> <p>Der Wert NOW ist der einzige gültige Wert, der dieser Richtlinienoption zugeordnet werden kann. Alle anderen führen zu einer Fehlermeldung. Der Wert wird in der Coordinated Universal Time (UTC) angegeben und als Zeichenfolge im Standardformat des Servers gespeichert.</p>	(keiner)	Alle Benutzer
locked	Wenn der Wert für diese Option ON ist, dürfen die Benutzer keine neuen Verbindungen mehr herstellen. Die reason_locked-Spalte der sa_get_user_status-Systemprozedur gibt eine vom Datenbankserver erstellte Zeichenfolge zurück, die anzeigt, warum ein Benutzer gesperrt ist.	OFF	Alle Benutzer außer denjenigen mit MANAGE ANY USER-Systemprivileg
max_connections	Die maximale Anzahl gleichzeitiger Verbindungen, die einem Benutzer gestattet sind.	Unlimited	Alle Benutzer außer denjenigen mit SERVER OPERATOR- oder DROP CONNECTION-Systemprivileg

Richtlinienoptionsname	Beschreibung	Standardwert	Gilt für:
max_failed_login_attempts	Die maximale Anzahl fehlgeschlagener Login-Versuche seit dem letzten erfolgreichen Versuch, bevor der Benutzer gesperrt wird. Benutzer mit SYS_AUTH_DBA_ROLE-Kompatibilitätsrolle werden freigegeben, wenn seit dem letzten gescheiterten Login-Versuch eine Minute vergangen ist.	Unlimited	
max_days_since_login	Die maximale Anzahl von Tagen zwischen erfolgreichen Logins durch denselben Benutzer.	Unlimited	Alle Benutzer außer denjenigen mit MANAGE ANY USER-Systemprivileg
max_non_dba_connections	Die maximale Anzahl gleichzeitiger Verbindungen, die Benutzer herstellen können. Diese Option wird nur bei der Root-Login-Richtlinie verwendet.	Unlimited	Alle Benutzer außer denjenigen mit SERVER OPERATOR- oder DROP CONNECTION-Systemprivileg
password_life_time	Die maximale Anzahl von Tagen, bis ein Kennwort geändert werden muss	Unlimited	Alle Benutzer
password_grace_time	Die Anzahl der Tage bis zum Kennwortablauf. Das Login ist möglich, aber die Standardprozedur post_login gibt Warnungen aus.	0	Alle Benutzer
password_expiry_on_next_login	Wenn der Wert für diese Option ON ist, läuft das Kennwort des Benutzers nach dem nächsten Login ab.	OFF	Alle Benutzer

Richtlinienoptionsname	Beschreibung	Standardwert	Gilt für:
root_auto_unlock_time	Die Zeitspanne, nach der gesperrte Konten automatisch freigegeben werden. Diese Option wird nur von der Root-Login-Richtlinie unterstützt.	1 Minute	Benutzer mit MANAGE ANY USER-Sys- temprivileg

Bemerkungen

Wenn eine Login-Richtlinie geändert wird, gelten die Änderungen sofort für alle Benutzer.

Wenn Sie keine Richtlinienoption angeben, werden Werte für diese Login-Richtlinie aus der Root-Login-Richtlinie übernommen. Neue Richtlinien erben nicht die Richtlinienoptionen MAX_NON_DBA_CONNECTIONS und ROOT_AUTO_UNLOCK_TIME.

Alle neuen Datenbanken enthalten eine Root-Login-Richtlinie. Sie können die Werte der Root-Login-Richtlinie ändern, aber Sie können die Richtlinie nicht löschen. Einen Überblick über die Standardwerte für die Root-Login-Richtlinie finden Sie in der Tabelle oben.

Privilegien

Sie müssen das MANAGE ANY LOGIN POLICY-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „Login-Richtlinien ändern“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „ALTER USER-Anweisung“ auf Seite 540
- „COMMENT-Anweisung“ auf Seite 573
- „CREATE LOGIN POLICY-Anweisung“ auf Seite 647
- „CREATE USER-Anweisung“ auf Seite 770
- „DROP LOGIN POLICY-Anweisung“ auf Seite 812
- „DROP USER-Anweisung“ auf Seite 838
- „Login-Richtlinien“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Im folgenden Beispiel wird die fiktive Login-Richtlinie "Test1" mithilfe der Richtlinienoptionen LOCKED und MAX_CONNECTIONS geändert. Der LOCKED-Wert gibt an, dass Benutzer mit der Richtlinie keine neuen Verbindungen herstellen können, und der MAX_CONNECTIONS-Wert beschränkt die zulässige Anzahl von gleichzeitigen Verbindungen.

```
ALTER LOGIN POLICY Test1  
LOCKED=ON  
MAX_CONNECTIONS=5;
```

In diesem Beispiel werden die Richtlinienoptionen LOCKED und MAX_CONNECTIONS aus der Root-Login-Richtlinie aufgehoben.

```
ALTER LOGIN POLICY root  
LOCKED=ON  
MAX_CONNECTIONS=5;
```

Im folgenden Beispiel werden ein primärer und ein sekundärer LDAP-Server für eine fiktive Login-Richtlinie "ldap_user_policy" festgelegt und die Möglichkeit, ein Failover auf die Standard-Authentifizierung durchzuführen, wird deaktiviert, auch wenn die login_mode-Datenbankoption den Wert "Standard" enthält. Dies ermöglicht eine strikte Kontrolle der Benutzer dieser Login-Richtlinie, damit nur die LDAP-Benutzerauthentifizierung für die Authentifizierung verwendet werden kann. Falls das Aufkommen an Login-Verbindungen so hoch wird, dass der LDAP-Server nicht mehr schnell reagieren und die Benutzer authentifizieren kann, werden für Benutzer, deren Wiederholungen und Timeouts erschöpft sind, Fehler bei der Verbindung mit dem Datenbankserver angezeigt, statt ein Failover auf die Standard-Authentifizierung durchzuführen.

```
ALTER LOGIN POLICY ldap_user_policy  
LDAP_PRIMARY_SERVER=ldapsrv1  
LDAP_SECONDARY_SERVER=ldapsrv2  
LDAP_FAILOVER_TO_STD=OFF;
```

Im folgenden Beispiel wird der Zeitstempelwert für eine fiktive Login-Richtlinie "application_user_policy" auf die aktuelle Zeit zurückgesetzt. Für alle Benutzer, denen diese Richtlinie zugeordnet ist, wird beim nächsten Login-Versuch nach dem Distinguished Name (DN) gesucht, statt den jeweiligen in ISYSUSER im Cache abgelegten Wert zu verwenden. Durch diese Strategie werden bei der nächsten Authentifizierung alte DN-Werte in ISYSUSER für Benutzer gelöscht, die dieser Richtlinie zugeordnet sind.

```
ALTER LOGIN POLICY application_user_policy  
LDAP_REFRESH_DN=NOW;
```

ALTER MATERIALIZED VIEW-Anweisung

Ändert eine materialisierte Ansicht.

Syntax

```
ALTER MATERIALIZED VIEW [ owner.]materialized-view-name {  
  SET HIDDEN  
  { ENABLE | DISABLE }  
  { ENABLE | DISABLE } USE IN OPTIMIZATION  
  { ADD PCTFREE percent-free-space | DROP PCTFREE }  
  [ NOT ] ENCRYPTED  
  [ { IMMEDIATE | MANUAL } REFRESH ]  
}
```

percent-free-space : integer

Parameter

SET HIDDEN-Klausel Verwenden Sie die SET HIDDEN-Klausel, um die Definition einer materialisierten Ansicht zu verschleiern. *Diese Einstellung ist irreversibel.*

ENABLE-Klausel Verwenden Sie die ENABLE-Klausel, um eine deaktivierte materialisierte Ansicht zu aktivieren und damit für die Verwendung durch den Datenbankserver verfügbar zu machen. Diese Klausel hat keine Auswirkung bei einer Ansicht, die bereits aktiviert ist. Nachdem Sie diese Klausel verwendet haben, müssen Sie die Ansicht aktualisieren, um sie zu initialisieren, und Textindizes neu erstellen, die beim Deaktivieren der Ansicht gelöscht wurden.

DISABLE-Klausel Verwenden Sie die DISABLE-Klausel, um die Verwendung der Ansicht durch den Datenbankserver zu deaktivieren. Wenn Sie eine materialisierte Ansicht deaktivieren, löscht der Datenbankserver die Daten und Indizes aus der Ansicht.

{ ENABLE | DISABLE } USE IN OPTIMIZATION-Klausel Verwenden Sie diese Klausel, um anzugeben, ob die materialisierte Ansicht dem Optimierer zur Verfügung stehen soll. Wenn Sie DISABLE USE IN OPTIMIZATION angeben, wird die materialisierte Ansicht nur verwendet, wenn Abfragen ausgeführt werden, die diese Ansicht explizit referenzieren. Der Standardwert ist ENABLE USE IN OPTIMIZATION.

ADD PCTFREE-Klausel Geben Sie den Prozentsatz des Speicherplatzes an, den Sie für jede Seite reservieren möchten. Der freie Speicherplatz wird verwendet, wenn die Zeilen durch die Datenaktualisierung anwachsen. Wenn in einer Seite kein freier Speicherplatz verfügbar ist, führt jede Vergrößerung einer Zeile dieser Seite zu einer Aufteilung der Zeile auf mehrere Seiten, was Zeilenfragmentierung und eventuell Performanceverlust nach sich zieht.

Der Wert von *percent-free-space* ist eine Ganzzahl zwischen 0 und 100. Der Wert 0 bedeutet, dass auf den einzelnen Seiten kein freier Platz zur Verfügung stehen darf - jede Seite wird vollgeschrieben. Ein hoher Wert führt dazu, dass jede Zeile auf eine eigene Seite geschrieben wird. Wenn PCTFREE nicht festgelegt oder gelöscht wurde, wird die PCTFREE-StandardEinstellung entsprechend der Datenbank-Seitengröße angewendet (200 Byte für eine 4-kB-Seitengröße und 100 Byte für eine 2-kB-Seitengröße).

DROP PCTFREE-Klausel Entfernt die derzeit für die materialisierte Ansicht gültige PCTFREE-Einstellung und wendet den PCTFREE-Standardwert entsprechend der Datenbank-Seitengröße an.

[NOT] ENCRYPTED-Klausel Gibt an, ob die Daten der materialisierten Ansicht verschlüsselt werden sollen. Standardmäßig werden während der Erstellung die Daten einer materialisierten Ansicht nicht verschlüsselt. Um eine materialisierte Ansicht zu verschlüsseln, geben Sie ENCRYPTED an. Um eine materialisierte Ansicht zu entschlüsseln, geben Sie NOT ENCRYPTED an.

REFRESH-Klausel Benutzen Sie die REFRESH-Klausel, um den Aktualisierungstyp für die materialisierte Ansicht zu ändern:

- **IMMEDIATE REFRESH** Verwenden Sie die IMMEDIATE REFRESH-Klausel, um eine manuelle Ansicht in eine Sofortansicht zu ändern. Die manuelle Ansicht muss gültig und nicht initialisiert sein, wenn der Aktualisierungstyp auf IMMEDIATE REFRESH geändert werden soll. Wenn die Ansicht im initialisierten Zustand ist, führen Sie eine TRUNCATE-Anweisung aus, um den Status auf "nicht initialisiert" zu ändern, bevor Sie die Anweisung ALTER MATERIALIZED VIEW...IMMEDIATE REFRESH ausführen.

- **MANUAL REFRESH** Verwenden Sie die MANUAL REFRESH-Klausel, um eine Sofortansicht auf manuelle Ansicht zu ändern.

Bemerkungen

Wenn Sie eine materialisierte Ansicht eines anderen Eigentümers ändern, müssen Sie den Namen qualifizieren, indem Sie den Eigentümer eingeben (z.B. GROUPO.EmployeeConfidential). Wenn Sie den Namen nicht qualifizieren, sucht der Datenbankserver nach einer Ihnen gehörenden materialisierten Ansicht mit diesem Namen und ändert sie. Wenn eine solche nicht vorhanden ist, wird ein Fehler zurückgegeben.

Wenn Sie eine materialisierte Ansicht deaktivieren (DISABLE-Klausel), steht sie dem Datenbankserver zur Beantwortung von Abfragen nicht mehr zur Verfügung. Außerdem werden die Daten und Indizes gelöscht und der Aktualisierungstyp ändert sich auf "Manuell". Eventuelle abhängige reguläre Ansichten werden ebenfalls deaktiviert.

Die DISABLE-Klausel erfordert einen exklusiven Zugriff nicht nur auf die Ansicht, die deaktiviert wird, sondern auch auf alle abhängigen Ansichten, da diese ja auch deaktiviert werden.

Damit eine materialisierte Ansicht verschlüsselt werden kann (ENCRYPTED-Klausel), muss die Tabellenverschlüsselung bereits in der Datenbank aktiviert sein. Die Verschlüsselung erfolgt unter Verwendung des Chiffrierschlüssels und des Algorithmus, die bei der Datenbankerstellung festgelegt wurden.

Die einzigen Vorgänge, die ein Benutzer in einer materialisierten Ansicht durchführen kann, um ihre Daten zu verändern, sind: Aktualisieren, Kürzen, Deaktivieren. Sofortansichten werden allerdings vom Datenbankserver automatisch aktualisiert. Das bedeutet: Sobald eine Sofortansicht aktiviert und initialisiert ist, verwaltet der Datenbankserver sie automatisch, ohne dass weitere Privilegüberprüfungen erfolgen.

Privilegien

Sie müssen entweder Eigentümer der materialisierten Ansicht sein oder das ALTER ANY MATERIALIZED VIEW-Systemprivileg oder das ALTER ANY OBJECT-Systemprivileg haben.

Wenn Sie ein erforderliches Privileg nicht haben, aber eine materialisierte Ansicht in eine Sofortansicht ändern möchten (ALTER MATERIALIZED VIEW...IMMEDIATE REFRESH), müssen Sie Eigentümer der Ansicht und aller von ihr referenzierten Tabellen sein.

Nebenwirkungen

- Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE MATERIALIZED VIEW-Anweisung“ auf Seite 652
- „REFRESH MATERIALIZED VIEW-Anweisung“ auf Seite 987
- „DROP MATERIALIZED VIEW-Anweisung“ auf Seite 813
- „TRUNCATE-Anweisung“ auf Seite 1089
- „sa_refresh_materialized_views-Systemprozedur“ auf Seite 1294
- „Materialisierte Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Ansichtenabhängigkeiten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- Einschränkungen beim Ändern des Typs einer materialisierten Ansicht von "Manuell" auf "Sofort" [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Aktivieren oder Deaktivieren einer materialisierten Ansicht“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Ver- und Entschlüsseln einer materialisierten Ansicht“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Definition einer materialisierten Ansicht verbergen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Einstellen des Aktualisierungstyps auf "Manuell" oder "Sofort"“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Fortgeschrittene Aufgaben: Status und Eigenschaften von materialisierten Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Aktivieren und Deaktivieren der Verwendung einer materialisierten Ansicht durch den Optimierer“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgenden Anweisungen erstellen die materialisierte Ansicht EmployeeConfid88 und deaktivieren dann ihre Verwendung für die Optimierung. Um dieses Beispiel ausführen zu können, benötigen Sie auch das CREATE ANY MATERIALIZED VIEW-Systemprivileg sowie das SELECT-Privileg für die Tabellen "Employees" und "Departments".

Vorsicht

Wenn Sie dieses Beispiel durchgeführt haben, löschen Sie die von Ihnen erstellte materialisierte Ansicht. Andernfalls können Sie keine Schemaänderungen an ihren Basistabellen Employees und Departments durchführen, wenn Sie andere Beispiele ausprobieren. Sie können das Schema einer Tabelle nicht ändern, die aktivierte, materialisierte Ansichten haben.

```
CREATE MATERIALIZED VIEW EmployeeConfid88 AS
  SELECT EmployeeID, Employees.DepartmentID, SocialSecurityNumber, Salary,
  ManagerID,
  Departments.DepartmentName, Departments.DepartmentHeadID
  FROM GROUPO.Employees, GROUPO.Departments
  WHERE Employees.DepartmentID=Departments.DepartmentID;
REFRESH MATERIALIZED VIEW EmployeeConfid88;
ALTER MATERIALIZED VIEW EmployeeConfid88 DISABLE USE IN OPTIMIZATION;
```

ALTER MIRROR SERVER-Anweisung

Hinweis

Scale-Out mit Schreibschutz und Datenbankspiegelung erfordern jeweils eine getrennte Lizenz. Siehe „Getrennt lizenzierbare Komponenten“ [[SQL Anywhere 16 - Einführung](#)].

Ändert die Attribute eines Spiegelservers.

Syntax 1

```
ALTER MIRROR SERVER mirror-server-name
[AS { PRIMARY | MIRROR | ARBITER | PARTNER}]
[ server-option = { string | NULL } [ ... ]]
```

Syntax 2

```
ALTER MIRROR SERVER mirror-server-name
[AS COPY]
[ { FROM SERVER parent-name [ OR SERVER server-name ] | USING AUTO PARENT } | ALTER
PARENT FROM mirror-server-name ]
[ server-option = { string | NULL } [ ... ]]
```

parent-name :
server-name | PRIMARY

server-option :
connection_string
logfile
preferred
state_file

Parameter

AS-Klausel Verwenden Sie die AS-Klausel, um den Servertyp des Spiegelservers von PARTNER in COPY bzw. von COPY in PARTNER zu ändern. Diese Klausel wird nur dann benötigt und ist auch nur dann empfehlenswert, wenn Sie den Typ des Spiegelservers ändern möchten.

- **PARTNER** Nur ein Datenbankserver mit dem Spiegelservertyp COPY kann diesen Wert verwenden, um seinen Typ in PARTNER zu ändern. Die übergeordneten Definitionen für den Kopieknoten werden gelöscht.

Der Name des Spiegelservers muss dem Namen des Datenbankservers entsprechen, der durch die Serveroption -n angegeben wurde, und muss mit dem Wert des SERVER-Verbindungszeichenfolgenparameters übereinstimmen, der in der Spiegelserveroption connection_string angegeben ist.

In einem Datenbankspiegelungssystem verwenden die Partner den Wert in der Verbindungszeichenfolge, um sich miteinander zu verbinden. In einem Scale-Out-System mit Schreibschutz wird die Verbindungszeichenfolge von einem Kopieknoten verwendet, für den dieser Server der übergeordnete Knoten ist.

- **COPY** Nur ein Datenbankserver, dessen Spiegelservertyp PARTNER ist, kann diesen Wert verwenden, um seinen Servertyp in einen Kopieknoten zu ändern. Dieser Partnerserver muss außerdem derzeit die MIRROR-Rolle haben.

In einem Scale-Out-System mit Schreibschutz gibt dieser Wert an, dass der Datenbankserver ein Kopieknoten ist. Alle Verbindungen zur Datenbank auf diesem Server sind schreibgeschützt. Der Name des Spiegelservers muss dem Namen des Datenbankservers entsprechen, der durch die Serveroption `-n` angegeben wurde, und muss mit dem Wert des `SERVER-Verbindungszeichenfolgenparameters` übereinstimmen, der in der Spiegelserveroption `connection_string` angegeben ist.

FROM SERVER-Klausel Diese Klausel kann nur verwendet werden, wenn `AS COPY` angegeben ist. Diese Klausel konstruiert eine Struktur von Servern für ein Scale-Out-System und gibt an, von welchen Servern die Kopieknoten Transaktionslogseiten abrufen.

Der übergeordnete Server kann mit dem Namen des Spiegel- oder Primärservers angegeben werden. Ein alternativer übergeordneter Server für die Kopieknoten kann mit der `OR SERVER`-Klausel angegeben werden.

In einem Datenbankspiegelungssystem mit nur zwei Ebenen (Stamm- und Kopieknoten) erhalten die Kopieknoten Transaktionslogseiten vom aktuellen Primär- oder Spiegelserver.

Ein Kopieknoten legt mithilfe der in der Datenbank gespeicherten Spiegelserverdefinition fest, zu welchem Server die Verbindung hergestellt wird. Aus seiner Definition kann er die Definition des übergeordneten Servers ermitteln und aus dessen Definition die Verbindungszeichenfolge für die Verbindung zu diesem übergeordneten Server.

Sie müssen nicht explizit Kopieknoten für das Scale-Out-System definieren, sondern können festlegen, dass der Stammknoten die Kopieknoten bei der Verbindungsaufnahme definiert.

USING AUTO PARENT-Klausel Diese Klausel kann nur verwendet werden, wenn `AS COPY` angegeben ist. Diese Klausel veranlasst den Primärserver zum Zuweisen eines übergeordneten Servers.

ALTER PARENT FROM-Klausel Diese Klausel kann nur verwendet werden, wenn `AS COPY` angegeben ist. Diese Klausel ändert den übergeordneten Server für diesen Spiegelserver und weist alle gleichwertigen Server als dessen untergeordnete Server zu. Der Name des durch die `ALTER PARENT FROM`-Klausel angegebenen Servers wird verwendet, um zu überprüfen, ob der aktuelle übergeordnete Server dem angegebenen Wert entspricht. Auf diese Weise wird sichergestellt, dass nur einer der gleichwertigen Server an die Stelle des übergeordneten Servers treten kann, wenn diese Änderung von allen gleichzeitig angefordert wird.

server-option-Klausel Die folgenden Optionen werden unterstützt:

- **connection_string-Serveroption** Gibt die Verbindungszeichenfolge an, die für die Verbindung mit dem Server verwendet werden soll. Die Verbindungszeichenfolge für einen Spiegelserver sollte weder Benutzer-ID noch Kennwort enthalten, weil diese nicht benutzt werden, wenn ein Spiegelserver eine Verbindung mit einem anderen Spiegelserver herstellt.

Eine Liste der Verbindungsparameter finden Sie unter „[Verbindungsparameter](#)“ [*SQL Anywhere Server - Datenbankadministration*].

- **logfile-Serveroption** Gibt den Speicherort der Datei an, die eine Zeile für jede zwischen Spiegelservern gesendete Anforderung enthält, wenn die Datenbankspiegelung verwendet wird. Diese Datei wird nur für die Fehlersuche verwendet.

- **preferred-Serveroption** Gibt an, ob der Server der bevorzugte Server im Spiegelungssystem ist. Sie können entweder YES oder NO angeben. Der bevorzugte Server übernimmt wenn möglich die Rolle des Primärservers. Geben Sie diese Option an, wenn Sie PARTNER-Server definieren.
- **state_file-Serveroption** Gibt den Speicherort der Datei an, die zur Verwaltung von Statusinformationen über das Spiegelungssystem verwendet wird. Diese Option ist bei der Datenbankspiegelung erforderlich. In einem Spiegelungssystem muss für Server mit dem Typ PARTNER eine Statusdatei angegeben werden. Bei Arbiterservern wird der Speicherort als Teil des Befehls zum Starten des Servers angegeben.

Bemerkungen

In einem Datenbankspiegelungssystem kann der Spiegelservertyp PRIMARY, MIRROR, ARBITER oder PARTNER sein.

In einem Scale-Out-System mit Schreibschutz kann der Spiegelservertyp PRIMARY, PARTNER oder COPY sein.

Sie können den Spiegelservertyp nur von COPY in PARTNER oder von PARTNER in COPY ändern. Wenn Sie den Servertyp von oder in PRIMARY, MIRROR oder ARBITER ändern möchten, müssen Sie die Spiegelserverdefinition löschen und neu erstellen.

Spiegelservernamen für Server vom Typ PARTNER oder COPY müssen mit den Namen der Datenbankserver übereinstimmen, die Teil des Spiegelungssystems sein sollen (den Namen, die mit der Serveroption -n verwendet werden). Diese Anforderung ermöglicht es jedem Datenbankserver, seine eigene Definition und die des übergeordneten Servers zu finden. Die Werte für *mirror-server-name*, *parent-name* und *server-name* dürfen nur 7-Bit-ASCII-Zeichen enthalten.

Wenn Sie einen Kopieknoten in einen Partner konvertieren, werden die übergeordnete Definitionen aus der Spiegelserverdefinition gelöscht.

Wenn Sie eine Spiegelserverdefinition ersetzen möchten, verwenden Sie die CREATE MIRROR SERVER-Anweisung mit der OR REPLACE-Klausel.

Privilegien

Sie müssen das MANAGE ANY MIRROR SERVER-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Hinzufügen von untergeordneten Kopieknotten“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Fehlerbehandlung: Statusinformationsdateien von Partner und Arbitr“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Partnerserver in Kopieknotten konvertieren“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Automatisches Zuweisen des übergeordneten Knotens eines Kopieknottens“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Bevorzugte Datenbankserver in Datenbankspiegelungssystemen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „SYSMIRRORSERVER-Systemansicht“ auf Seite 1464
- „CREATE MIRROR SERVER-Anweisung“ auf Seite 656
- „COMMENT-Anweisung“ auf Seite 573
- „DROP MIRROR SERVER-Anweisung“ auf Seite 815

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im nächsten Beispiel werden folgende Vorgänge ausgeführt:

1. Erstellt einen Primärserver namens scaleout_primary1.
2. Erstellt den Kopieknotten scaleout_child1 für scaleout_primary1.
3. Erstellt den Spiegelserver scaleout_mirror1.
4. Ordnet mit der ALTER MIRROR SERVER-Anweisung scaleout_child1 als Kopieknotten von scaleout_mirror1 zu.

```
CREATE MIRROR SERVER "scaleout_primary1"
  AS PRIMARY
  connection_string =
  'server=scaleout_primary1;host=winxp-2:6871,winxp-3:6872';

CREATE MIRROR SERVER "scaleout_child1"
  AS COPY FROM SERVER "scaleout_primary1"
  connection_string = 'server=scaleout_child1;host=winxp-2:6878';

CREATE MIRROR SERVER "scaleout_mirror1"
  AS MIRROR
  connection_string =
  'server=scaleout_mirror1;host=winxp-2:6871,winxp-3:6872';

ALTER MIRROR SERVER "scaleout_child1"
  FROM SERVER "scaleout_mirror1"
  connection_string = 'server=scaleout_child1;host=winxp-2:6878';
```

ALTER PROCEDURE-Anweisung

Ändert eine Prozedur.

Syntax 1

ALTER PROCEDURE [*owner.*]*procedure-name* *procedure-definition*

procedure-definition: Siehe „[CREATE PROCEDURE-Anweisung](#)“ auf Seite 681.

Syntax 2

ALTER PROCEDURE [*owner.*]*procedure-name*
SET HIDDEN

Syntax 3

ALTER PROCEDURE [*owner.*]*procedure-name*
RECOMPILE

Bemerkungen

Die ALTER PROCEDURE-Anweisung muss die vollständige neue Prozedur enthalten. PROC kann als Synonym für PROCEDURE verwendet werden.

- **Syntax 1** Die ALTER PROCEDURE-Anweisung unterscheidet sich in der Syntax nur durch das erste Wort von der CREATE PROCEDURE-Anweisung. Sowohl Watcom- als auch Transact-SQL-Dialektprozeduren können mithilfe von ALTER PROCEDURE geändert werden.

Mit ALTER PROCEDURE werden vorhandene Privilegien für die Prozedur nicht geändert. Beim Ausführen von DROP PROCEDURE, gefolgt von CREATE PROCEDURE, werden EXECUTE-Privilegien neu zugeordnet.

- **Syntax 2** Sie können SET HIDDEN verwenden, um die Definition der zugehörigen Prozedur zu verschleiern und somit unlesbar zu machen. Die Prozedur kann aus Ihrer Datenbank entladen und in andere Datenbanken geladen werden.

Wenn SET HIDDEN verwendet wird, ist die Definition der Prozedur bei der Fehlerbehandlung mit dem Debugger nicht zu sehen und sie wird auch nicht in den Prozedurprofilen angezeigt.

Es ist nicht möglich, Syntax 2 mit Syntax 1 zu kombinieren.

Hinweis

Diese Einstellung ist irreversibel. Es wird empfohlen, die ursprüngliche Prozedurdefinition außerhalb der Datenbank aufzubewahren.

- **Syntax 3** Verwenden Sie die RECOMPILE-Syntax, um eine gespeicherte Prozedur neu zu kompilieren. Wenn Sie eine Prozedur neu kompilieren, wird die im Katalog gespeicherte Definition neuerlich syntaktisch analysiert und die Syntax wird geprüft. Bei Prozeduren, die eine Ergebnismenge generieren, aber keine RESULT-Klausel einbeziehen, versucht der Datenbankserver, die Eigenschaften der Ergebnismenge für die Prozedur zu bestimmen, und speichert die Informationen im Katalog. Dies kann sinnvoll sein, wenn eine von der Prozedur referenzierte Tabelle durch Hinzufügen, Entfernen oder Umbenennen von Spalten geändert wurde, seit die Prozedur erstellt wurde.

Die Prozedurdefinition wird durch Neukompilierung nicht geändert. Sie können Prozeduren mit durch die SET HIDDEN-Klausel ausgeblendeten Definitionen neu kompilieren, die Definitionen bleiben aber verborgen.

Privilegien

Sie müssen Eigentümer der Prozedur sein oder eines der folgenden Privilegien haben:

- ALTER ANY PROCEDURE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE PROCEDURE-Anweisung“ auf Seite 681
- „CREATE PROCEDURE-Anweisung [Webdienste]“ auf Seite 670
- „ALTER FUNCTION-Anweisung“ auf Seite 465
- „DROP PROCEDURE-Anweisung“ auf Seite 816
- „Verbergen des Inhalts von Prozeduren, Funktionen, Triggern, Ereignissen oder Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. ALTER PROCEDURE ist die optionale SQL-Sprachenfunktion F381 des SQL/2008-Standards. Im SQL-Standard kann ALTER PROCEDURE jedoch nicht dazu verwendet werden, die Definition einer gespeicherten Prozedur neu festzulegen, und Transact-SQL-Dialektprozeduren werden nicht unterstützt. SQL/2008 umfasst keine Unterstützung für SET HIDDEN oder RECOMPILE.

ALTER PUBLICATION-Anweisung [MobiLink] [SQL Remote]

Ändert eine Publikation. In MobiLink kennzeichnet eine Publikation synchronisierte Daten in einer entfernten SQL Anywhere-Datenbank. In SQL Remote kennzeichnet eine Publikation Replikatdaten sowohl in konsolidierten als auch in entfernten Datenbanken.

Syntax

ALTER PUBLICATION [*owner.*]*publication-name alterpub-clause, ...*

alterpub-clause :

ADD *article-definition*

| **ALTER** *article-definition*

| { **DELETE** | **DROP** } **TABLE** [*owner.*]*table-name*

| **RENAME** *publication-name*

article-definition :

TABLE *table-name* [(*column-name, ...*)]

[**WHERE** *search-condition*]

[**SUBSCRIBE BY** *expression*]

[**USING** ([**PROCEDURE**] [*owner.*]*[procedure-name]*]

FOR UPLOAD { **INSERT** | **DELETE** | **UPDATE** }, ...)]

Bemerkungen

Diese Anweisung gilt nur für MobiLink und SQL Remote.

Der Beitrag einer Tabelle zu einer Publikation wird als Artikel bezeichnet. Änderungen an einer Publikation können durch Hinzufügen, Ändern oder Löschen von Artikeln sowie durch ein Umbenennen der Publikation vorgenommen werden. Wenn ein Artikel geändert wird, muss die gesamte Definition des geänderten Artikels eingegeben werden.

Es wird empfohlen, dass Sie eine Synchronisation der Publikation erfolgreich abschließen, bevor Sie sie ändern.

Sie können die WHERE-Klausel nicht bei Publikationen verwenden, die als FOR DOWNLOAD ONLY oder WITH SCRIPTED UPLOAD definiert sind.

Die SUBSCRIBE BY-Klausel gilt nur für SQL Remote.

Die USING-Klausel ist nur für skriptgesteuerte Uploads bestimmt.

Die Optionen für eine MobiLink-Publikation können Sie mit der ADD OPTION-Klausel in der ALTER SYNCHRONIZATION SUBSCRIPTION- oder CREATE SYNCHRONIZATION SUBSCRIPTION-Anweisung festlegen.

Wenn Sie eine MobiLink-Publikation ändern, kann ein Artikel erst nach der Ausführung einer START SYNCHRONIZATION SCHEMA CHANGE-Anweisung gelöscht werden.

Erfordert exklusiven Zugriff auf alle Tabellen, die in der Anweisung referenziert werden, sowie auf alle Tabellen in der zu ändernden Publikation.

Privilegien

Sie müssen Eigentümer der Publikation sein oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Publikation
- SYS_REPLICATION_ADMIN_ROLE-Systemrolle

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ auf Seite 690
- „DROP PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ auf Seite 817
- SQL Anywhere MobiLink-Clients: „Publikationen“ [*MobiLink - Clientadministration*]
- UltraLite MobiLink-Clients: „UltraLite-Clientsynchronisationsplanung“ [*UltraLite - Datenbankverwaltung*]
- „Publikationen und Artikel“ [*SQL Remote*]
- „ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink]“ auf Seite 512
- „CREATE SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink]“ auf Seite 733
- „SYSSYNC-Systemansicht“ auf Seite 1489
- „START SYNCHRONIZATION SCHEMA CHANGE-Anweisung [MobiLink]“ auf Seite 1072
- „In einem laufenden System zu vermeidende Änderungen“ [*SQL Remote*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung fügt die Tabelle 'Customers' zur pub_contact-Publikation hinzu.

```
ALTER PUBLICATION pub_contact  
ADD TABLE GROUPO.Customers;
```

ALTER REMOTE MESSAGE TYPE-Anweisung [SQL Remote]

Ändert das Nachrichtensystem des Publikationseigentümers oder die Adresse des Publikationseigentümers bei einem angegebenen Nachrichtensystem für einen erstellten Nachrichtentyp.

Syntax

```
ALTER REMOTE MESSAGE TYPE message-system  
ADDRESS address
```

message-system : **FILE** | **FTP** | **SMTP**

address : *string*

Parameter

message-system Eines der Nachrichtensysteme, die von SQL Remote unterstützt werden. Hierfür muss einer der folgenden Werte verwendet werden: **FILE**, **FTP** oder **SMTP**.

address Eine Zeichenfolge, die eine gültige Adresse für das angegebene Nachrichtensystem enthält.

Bemerkungen

Die Anweisung ändert die Adresse des Publikationseigentümers bei einem gegebenen Nachrichtentyp.

Der Nachrichtenagent versendet ausgehende Nachrichten von einer Datenbank über eine der unterstützten Nachrichtenverbindungen. Das Extraktionsdienstprogramm verwendet diese Adresse, wenn die GRANT CONSOLIDATE-Anweisung in der entfernten Datenbank ausgeführt wird.

Die Adresse ist die des Publikationseigentümers unter dem angegebenen Nachrichtensystem. Wenn es sich um ein E-Mail-System handelt, muss die Adressenzeichenfolge eine gültige E-Mail-Adresse sein. Wenn es sich um ein System mit gemeinsamer Dateinutzung handelt, ist die Adressenzeichenfolge ein Unterverzeichnis des Verzeichnisses, das durch die SQLREMOTE-Umgebungsvariable bestimmt ist, oder des aktuellen Verzeichnisses, wenn sie nicht festgelegt ist. Sie können diese Einstellung für die GRANT CONSOLIDATE-Anweisung in der entfernten Datenbank außer Kraft setzen.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE REMOTE [MESSAGE] TYPE-Anweisung [SQL Remote]“ auf Seite 694
- „GRANT CONSOLIDATE-Anweisung [SQL Remote]“ auf Seite 889
- „SQLREMOTE-Umgebungsvariable“ [*SQL Anywhere Server - Datenbankadministration*]
- „SQL Remote-Nachrichtensysteme“ [*SQL Remote*]
- „Einen Nachrichtentyp ändern“ [*SQL Remote*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung ändert die Adresse des Publikationseigentümers bei der FILE-Nachrichtenverbindung auf "new_addr".

```
ALTER REMOTE MESSAGE TYPE file  
ADDRESS 'new_addr';
```

ALTER ROLE-Anweisung

Migriert eine Kompatibilitätsrolle in eine benutzerdefinierte Rolle und löscht anschließend die Kompatibilitätsrolle.

Syntax

```
ALTER ROLE compatibility-role-name  
MIGRATE TO new-role-name [, new-sa-role-name, new-sso-role-name ]
```

Parameter

- ***compatibility-role-name*** Verwenden Sie diesen Parameter, um den Namen der zu migrierenden Kompatibilitätsrolle anzugeben.

- ***new-role-name*** Verwenden Sie diesen Parameter, um den Namen der neuen Rolle anzugeben, die Sie erstellen.
- ***new-sa-role-name*** Verwenden Sie diesen Parameter, um den Namen der neuen Rolle anzugeben, in die SYS_AUTH_SA_ROLE migriert werden soll. Dieser Parameter ist erforderlich, wenn Sie SYS_AUTH_DBA_ROLE migrieren möchten, wodurch SYS_AUTH_SA_ROLE automatisch migriert wird.
- ***new-ssso-role-name*** Verwenden Sie diesen Parameter, um den Namen der neuen Rolle anzugeben, in die SYS_AUTH_SSO_ROLE migriert werden soll. Dieser Parameter ist erforderlich, wenn Sie SYS_AUTH_DBA_ROLE migrieren möchten, wodurch SYS_AUTH_SSO_ROLE automatisch migriert wird.

Bemerkungen

Der Name der neuen Rolle darf nicht sowohl mit "SYS_" anfangen als auch mit "_ROLE" enden. SYS_MyBackup_ROLE ist beispielsweise kein zulässiger Name für eine benutzerdefinierte Rolle, während MyBackup_ROLE und SYS_MyBackup zulässig sind.

Wenn Sie die ALTER ROLE-Anweisung ausführen, wird Berechtigungsempfängern der Kompatibilitätsrolle die neue Rolle erteilt.

Sie können Kompatibilitätsrollen, die migriert und anschließend gelöscht wurden, wiederherstellen, indem Sie eine CREATE ROLE-Anweisung ausführen und den Namen der Kompatibilitätsrolle angeben. `CREATE ROLE SYS_AUTH_BACKUP_ROLE;` stellt beispielsweise die SYS_AUTH_BACKUP_ROLE-Kompatibilitätsrolle wieder her.

Anfänglich können nur Benutzer mit vollen Administrationsrechten (DBAs) die neue Rolle verwalten, aber Sie können die CREATE ROLE-Anweisung mit der OR REPLACE-Klausel verwenden, um zusätzliche Administratoren anzugeben.

Verwenden Sie die Anweisungen GRANT und REVOKE, um der Rolle Systemprivilegien zu erteilen bzw. zu entziehen.

Sie können die Kompatibilitätsrollen SYS_AUTH_SA_ROLE und SYS_AUTH_SSO_ROLE migrieren, indem Sie die SYS_AUTH_DBA_ROLE-Kompatibilitätsrolle migrieren, wodurch SYS_AUTH_SA_ROLE und SYS_AUTH_SSO_ROLE automatisch migriert werden. Beim Migrieren von SYS_AUTH_DBA_ROLE müssen Sie die Parameter *new-sa-role-name* und *new-ssso-role-name* einbeziehen, um neue Namen für die migrierten Rollen SYS_AUTH_SA_ROLE und SYS_AUTH_SSO_ROLE festzulegen.

Privilegien

Sie müssen das MANAGE ROLES-Systemprivileg und Administrationsrechte für die zu migrierende Kompatibilitätsrolle haben.

Nebenwirkungen

Keine

Siehe auch

- „Rollen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „CREATE ROLE-Anweisung“ auf Seite 696
- „min_role_admins-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „DROP ROLE-Anweisung“ auf Seite 820
- „REVOKE-Anweisung“ auf Seite 1009

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung migriert alle Benutzer und zugrunde liegenden Systemprivilegien, die der SYS_AUTH_BACKUP_ROLE-Rolle erteilt wurden, in eine neue Rolle namens custom_Backup_ROLE und löscht anschließend SYS_AUTH_BACKUP_ROLE aus der Datenbank.

```
ALTER ROLE SYS_AUTH_BACKUP_ROLE  
MIGRATE TO custom_Backup_ROLE;
```

Die folgende Anweisung migriert alle Benutzer, zugrunde liegenden Systemprivilegien und Rollen die der SYS_AUTH_DBA_ROLE-Kompatibilitätsrolle erteilt wurden, in eine neue Rolle namens custom_DBA. Danach migriert sie automatisch alle Benutzer, zugrunde liegenden Systemprivilegien und Rollen, die SYS_AUTH_SA_ROLE und SYS_AUTH_SSO_ROLE erteilt wurden, in neue Rollen namens custom_SA und custom_SSO. Abschließend löscht sie SYS_AUTH_DBA_ROLE, SYS_AUTH_SA_ROLE und SYS_AUTH_SSO_ROLE aus der Datenbank.

```
ALTER ROLE SYS_AUTH_DBA_ROLE  
MIGRATE TO custom_DBA, custom_SA, custom_SSO;
```

ALTER SEQUENCE-Anweisung

Ändert eine Sequenz.

Syntax

```
ALTER SEQUENCE [ owner.] sequence-name  
[ RESTART WITH signed-integer ]  
[ INCREMENT BY signed-integer ]  
[ MINVALUE signed-integer | NO MINVALUE ]  
[ MAXVALUE signed-integer | NO MAXVALUE ]  
[ CACHE integer | NO CACHE ]  
[ CYCLE | NO CYCLE ]
```

Parameter

RESTART WITH-Klausel Startet die benannte Sequenz mit dem angegebenen Wert.

INCREMENT BY-Klausel Definiert den Betrag, um den der nächste Sequenzwert gegenüber dem letzten zugeordneten Wert erhöht wird. Standardwert ist "1". Geben Sie einen negativen Wert an, um für eine absteigende Sequenz zu erstellen. Wenn der INCREMENT BY-Wert 0 ist, wird ein Fehler zurückgegeben.

MINVALUE-Klausel Definiert den kleinsten durch die Sequenz generierten Wert. Standardwert ist "1". Ein Fehler wird zurückgegeben, wenn MINVALUE größer ist als $(2^{63}-1)$ oder kleiner als $-(2^{63}-1)$. Außerdem wird ein Fehler zurückgegeben, wenn MINVALUE größer ist als MAXVALUE.

MAXVALUE-Klausel Definiert den größten durch die Sequenz generierten Wert. Standardwert ist $2^{63}-1$. Ein Fehler wird zurückgegeben, wenn MAXVALUE größer ist als $2^{63}-1$ oder kleiner als $-(2^{63}-1)$.

CACHE-Klausel Gibt die Anzahl der vorab zugewiesenen Sequenzwerte an, die für einen schnelleren Zugriff im Speicher aufbewahrt werden. Wenn der Cache belegt ist, wird der Sequenzcache neu gefüllt und ein entsprechender Eintrag in das Transaktionslog geschrieben. Zur Checkpoint-Zeit wird der aktuelle Wert des Cache an die ISYSEQUENCE-Systemtabelle weitergeleitet. Der Standardwert ist 100.

CYCLE-Klausel Legt fest, ob weitere Werte generiert werden sollen, nachdem der Höchst- oder Mindestwert erreicht wurde.

Bemerkungen

Wenn die benannte Sequenz nicht gefunden werden kann, wird eine Fehlermeldung zurückgegeben.

Privilegien

Sie müssen Eigentümer der Sequenz sein oder eines der folgenden Privilegien haben:

- ALTER ANY SEQUENCE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Nebenwirkungen

Keine

Siehe auch

- „Verwendung einer Sequenz zum Generieren von eindeutigen Werten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „CREATE SEQUENCE-Anweisung“ auf Seite 699
- „DROP SEQUENCE-Anweisung“ auf Seite 822

Standards und Kompatibilität

- **SQL/2008** Die ALTER SEQUENCE-Anweisung ist Teil der optionalen SQL-Sprachenfunktion T176 des SQL/2008-Standards. Die CACHE-Klausel ist eine Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird ein neuer Höchstwert für eine Sequenz mit dem Namen "Test" gesetzt:

```
ALTER SEQUENCE Test
MAXVALUE 1500;
```

ALTER SERVER-Anweisung

Ändert die Attribute eines Fremdservers.

Syntax 1

```
ALTER SERVER server-name
[ CLASS server-class ]
[ USING connection-string-info ]
[ CAPABILITY cap-name { ON | OFF } ]
[ CONNECTION CLOSE [ CURRENT | ALL | connection-id ] ]
```

```
server-class :
'ADSODBC'
| 'ASEODBC'
| 'DB2ODBC'
| 'HANAODBC'
| 'IQODBC'
| 'MIRROR'
| 'MSACCESSODBC'
| 'MSSODBC'
| 'MYSQLODBC'
| 'ODBC'
| 'ORAODBC'
| 'SAODBC'
| 'ULODBC'
```

```
connection-info-string :
{ 'data-source-name' | 'sqlanywhere-connection-string' }
```

Syntax 2

```
ALTER SERVER server-name
[ CLASS 'DIRECTORY' ]
[ USING using-string ]
[ CAPABILITY cap-name { ON | OFF } ]
[ CONNECTION CLOSE [ CURRENT | ALL | connection-id ] ]
```

```
using-string :
'ROOT = path [ ;SUBDIRS = n ] [ ;READONLY = { YES | NO } ] [ ;CREATEDIRS = { YES | NO } ]
[ ;DELIMITER = { / | \ } ]'
```

Parameter

CLASS-Klausel Die CLASS-Klausel wird angegeben, um die Klasse des Servers zu ändern.

USING-Klausel Die USING-Klausel wird angegeben, um die Verbindungsinformationen des Servers zu ändern.

Die Zeichenfolge in der USING-Klausel kann auch lokale oder globale Variablennamen in geschweiften Klammern enthalten (*{variable-name}*). Der SQL-Variablenname muss vom Typ CHAR, VARCHAR oder LONG VARCHAR sein. Eine USING-Klausel mit 'DSN={@mydsn}' zeigt beispielsweise an, dass @mydsn eine SQL-Variable ist und dass der aktuelle Inhalt der @mydsn-Variablen beim Herstellen der Verbindung mit dem Ferndatenzugriffsserver ersetzt werden muss.

CAPABILITY-Klausel Die CAPABILITY-Klausel schaltet die jeweilige Fähigkeit eines Servers ON oder OFF. Die Serverfähigkeiten werden in der Systemtabelle ISYSCAPABILITY gespeichert. Die Namen dieser Fähigkeiten sind über die SYSCAPABILITYNAME-Systemansicht zugänglich. Die ISYSCAPABILITY-Systemtabelle und die SYSCAPABILITYNAME-Systemansicht werden erst mit Daten gefüllt, wenn die erste Verbindung zu einem Fremdsystem aufgenommen wird. Für nachfolgende

Verbindungen werden die Fähigkeiten des Datenbankservers aus der Tabelle ISYSCAPABILITY bezogen.

Im Allgemeinen brauchen Sie die Fähigkeiten eines Servers nicht zu ändern. Es kann allerdings nötig sein, die Fähigkeiten eines generischen Servers der Klasse ODBC anzupassen.

CONNECTION CLOSE-Klausel (nicht mehr empfohlen) Diese Klausel wird nicht mehr empfohlen. Verwenden Sie die DROP REMOTE CONNECTION-Anweisung.

Bemerkungen

Die Anweisung ALTER SERVER ändert die Attribute eines Servers. Diese Änderungen werden erst wirksam, wenn die nächste Verbindung zum Fremdserver aufgenommen wird.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Die CONNECTION CLOSE-Klausel bewirkt kein automatisches Festschreiben.

Siehe auch

- „CREATE SERVER-Anweisung“ auf Seite 701
- „DROP REMOTE CONNECTION-Anweisung“ auf Seite 819
- „DROP SERVER-Anweisung“ auf Seite 823
- „Ferndatenzugriff“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Fremdserver“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Verzeichniszugriffsserver“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Fehlerbehandlung beim Ferndatenzugriff“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „SYSCAPABILITY-Systemansicht“ auf Seite 1439
- „SYSCAPABILITYNAME-Systemansicht“ auf Seite 1439

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Mit dem folgenden Beispiel wird die Serverklasse des Adaptive Server Enterprise-Servers namens "ase_prod" geändert, damit seine Verbindung zu SQL Anywhere über ODBC erfolgt. Der Datenquellename lautet ase_datasource.

```
ALTER SERVER ase_prod
CLASS 'ASEODBC'
USING 'ase_datasource';
```

Mit dem folgenden Beispiel wird die Serverklasse des Adaptive Server Enterprise-Servers namens "ase_prod" geändert, damit seine Verbindung zu SQL Anywhere über ODBC erfolgt. Der Datenquellename wird aus der ase_source-Variablen abgerufen.

```
ALTER SERVER ase_prod
CLASS 'ASEODBC'
```

```
USING '{ase_source}';

CREATE VARIABLE ase_source VARCHAR(128);
SET ase_source = 'ase_datasource';
```

Mit dem folgenden Beispiel wird eine Funktion des Servers ase_prod geändert.

```
ALTER SERVER ase_prod
CAPABILITY 'insert select' OFF;
```

Im folgenden Beispiel wird ein Verzeichniszugriffsserver so geändert, dass er 9 Ebenen von Unterverzeichnissen innerhalb des Verzeichnisses *c:\temp* abruft.

```
ALTER SERVER ase_prod
CLASS 'DIRECTORY'
USING 'ROOT=c:\\temp;SUBDIRS=9';
```

ALTER SERVICE-Anweisung [HTTP-Webdienst]

Ändert einen vorhandenen HTTP-Webdienst.

Syntax

```
ALTER SERVICE service-name
[ TYPE { 'RAW' | 'HTML' | 'JSON' | 'XML' } ]
[ URL [ PATH ] { ON | OFF | ELEMENTS } ]
[ common-attributes ]
[ AS { statement | NULL } ]
```

```
common-attributes :
[ AUTHORIZATION { ON | OFF } ]
[ ENABLE | DISABLE ]
[ METHODS 'method,...' ]
[ SECURE { ON | OFF } ]
[ USER { user-name | NULL } ]
```

```
method :
DEFAULT
| POST
| GET
| HEAD
| PUT
| DELETE
| NONE
| *
```

Parameter

service-name Webdienstnamen können aus einer beliebigen Sequenz von alphanumerischen Zeichen bestehen sowie die Zeichen Schrägstrich (/), Bindestrich (-), Unterstrich (_), Punkt (.), Ausrufezeichen (!), Tilde (~), Sternchen (*), Apostroph ('), öffnende Klammer (()) oder schließende Klammer ()) enthalten. Allerdings darf der Dienstname nicht mit einem Schrägstrich (/) beginnen oder enden oder zwei oder mehr aufeinander folgende Schrägstriche (z.B. //) enthalten.

Sie können Ihren Dienst **root** nennen, aber dieser Name hat eine spezielle Funktion.

TYPE-Klausel Gibt den Typ des Dienstes an, wobei für jeden Dienst ein bestimmtes Antwortformat definiert wird. Der Typ muss einer der aufgelisteten Typen sein. Es gibt keine Standardeinstellung.

- **'RAW'** Die Ergebnismenge wird ohne weitere Formatierung an den Client gesendet. Die Nutzung dieses Dienstes erfordert, dass alle Inhalts-Markups explizit zur Verfügung gestellt werden. Komplexe dynamische Inhalte, die aktuellen Inhalt mit Markup, JavaScript und Grafiken enthalten, können bei Bedarf generiert werden. Der Medientyp kann durch Einstellen des Antwort-Headers "Content-Type" unter Verwendung der Prozedur `sa_set_http_header` angegeben werden. Die Einstellung 'text/html' für den Content-Type-Header wird empfohlen, wenn Sie HTML-Markup generieren, damit alle Browser die Markup als HTML und nicht "text/plain" anzeigen.
- **'HTML'** Die Ergebnismenge wird als HTML-Darstellung einer Tabelle oder einer Ansicht zurückgegeben.
- **'JSON'** Die Ergebnismenge wird in JavaScript Object Notation (JSON) zurückgegeben. Ein JSON-Dienst verarbeitet nicht automatisch eine JSON-Eingabe. Er stellt lediglich Daten (in der Antwort) im JSON-Format bereit. JSON akzeptiert POST/PUT-Methoden, wobei **application/x-www-form-urlencoded** unterstützt wird. Wenn für eine POST/PUT METHODE **Content-Type: application/json** angegeben ist, kann die Anwendung `http_variable('body')` zum Abrufen des JSON-(Anforderung)-Inhalts verwenden. SQL Anywhere führt nicht automatisch eine syntaktische Analyse der JSON-Eingabe durch. Die Anwendung muss sie syntaktisch analysieren. Weitere Hinweise zu JSON finden Sie unter <http://www.json.org>
- **'XML'** Die Ergebnismenge wird als XML zurückgegeben. Wenn die Ergebnismenge bereits XML ist, wird keine weitere Formatierung angewendet. Andernfalls wird sie automatisch als XML formatiert. Als alternativer Ansatz könnte ein RAW-Dienst unter Verwendung der FOR XML RAW-Klausel eine Auswahl zurückgeben, nachdem Sie mit der Prozedur `sa_set_http_header` einen gültigen Content-Type wie zum Beispiel 'text/xml' festgelegt haben.

URL-Klausel Bestimmt, ob URL-Pfade akzeptiert werden und, falls ja, wie sie verarbeitet werden sollen. Wenn Sie URL PATH angeben, hat dies dieselbe Wirkung wie URL.

- **OFF** Gibt an, dass auf den Dienstnamen in einer URL-Anforderung kein Pfad folgen darf. OFF ist die Standardeinstellung. Zum Beispiel ist das folgende Format aufgrund der Pfadelemente `/aaa/bbb/ccc` nicht zulässig.

`http://host-name/service-name/aaa/bbb/ccc`

Nehmen wir an, bei der Erstellung des Webdiensts wurde `CREATE SERVICE echo URL PATH OFF` angegeben. Eine URL ähnlich wie `http://localhost/echo?id=1` erzeugt die folgenden Werte:

Function call	Result
<code>HTTP_VARIABLE('id')</code>	1
<code>HTTP_HEADER('@HttpQueryString')</code>	id=1

- **ON** Gibt an, dass auf den Dienstnamen in einer URL-Anforderung ein Pfad folgen darf. Der Pfadwert wird zurückgegeben durch Abfragen einer ausschließlich für ihn vorgesehenen HTTP-

Variable mit dem Namen **URL**. Ein Dienst kann so definiert werden, dass er explizit den URL-Parameter bereitstellt, oder er kann mit der Funktion `HTTP_VARIABLE` abgerufen werden. Die folgende Form ist beispielsweise zulässig:

```
http://host-name/service-name/aaa/bbb/ccc
```

Nehmen wir an, bei der Erstellung des Webdiensts wurde `CREATE SERVICE echo URL PATH ON` angegeben. Eine URL ähnlich wie `http://localhost/echo/one/two?id=1` erzeugt die folgenden Werte:

Function call	Result
<code>HTTP_VARIABLE('id')</code>	1
<code>HTTP_VARIABLE('URL')</code>	one/two
<code>HTTP_HEADER('@HttpQueryString')</code>	id=1

- **ELEMENTS** Gibt an, dass auf den Dienstnamen in einer URL-Anforderung ein Pfad folgen darf. Der Pfad wird in Segmenten abgerufen durch die Angabe des einzelnen Parameter-Schlüsselworts **URL1**, **URL2** usw. Jeder Parameter kann mit der Funktion `HTTP_VARIABLE` oder `NEXT_HTTP_VARIABLE` abgerufen werden. Diese Wiederholfunktionen können in Anwendungen verwendet werden, in denen eine variable Anzahl von Pfadelementen bereitgestellt werden kann. Die folgende Form ist beispielsweise zulässig:

```
http://host-name/service-name/aaa/bbb/ccc
```

Nehmen wir an, bei der Erstellung des Webdiensts wurde `CREATE SERVICE echo URL PATH ELEMENTS` angegeben. Eine URL ähnlich wie `http://localhost/echo/one/two?id=1` erzeugt die folgenden Werte:

Function call	Result
<code>HTTP_VARIABLE('id')</code>	1
<code>HTTP_VARIABLE('URL1')</code>	one
<code>HTTP_VARIABLE('URL2')</code>	two
<code>HTTP_VARIABLE('URL3')</code>	NULL
<code>HTTP_HEADER('@HttpQueryString')</code>	id=1

Bis zu 10 Elemente können abgerufen werden. NULL wird zurückgegeben, wenn das entsprechende Element nicht angegeben wurde. In diesem Beispiel gibt `HTTP_VARIABLE('URL3')` NULL zurück, weil kein entsprechendes Element angegeben wurde.

Weitere Hinweise zu URLs finden Sie unter „[Auf einem SQL Anywhere-HTTP-Webserver navigieren](#)“ [*SQL Anywhere Server - Programmierung*] und „[Zugriff auf vom Client bereitgestellte HTTP-Variablen und -Header](#)“ [*SQL Anywhere Server - Programmierung*].

AUTHORIZATION-Klausel Bestimmt, ob die Benutzer einen Benutzernamen und ein Kennwort über die HTTP-Grundautorisierung angeben müssen, wenn sie eine Verbindung mit dem Dienst herstellen. Der Standardwert ist ON. Wenn AUTHORIZATION auf OFF gesetzt ist, muss die AS-Klausel für alle Dienste verwendet werden und ein Benutzer muss mit der USER-Klausel angegeben werden. Alle Anforderungen werden mit dem Konto und den Privilegien dieses Benutzers ausgeführt. Wenn AUTHORIZATION auf ON gesetzt ist, müssen alle Benutzer einen Benutzernamen und ein Kennwort angeben. Der Zugriff auf diesen Dienst kann durch Angabe eines Benutzer- oder Gruppennamens mithilfe der USER-Klausel eingeschränkt werden. Wenn der Benutzername NULL ist, haben alle bekannten Benutzer Zugriff auf den Dienst. Die AUTHORIZATION-Klausel ermöglicht es Ihren Webdiensten, mithilfe von Datenbankautorisierung und Privilegien den Zugriff auf die Daten in Ihrer Datenbank zu steuern.

Wenn der Autorisierungswert ON ist, benutzt ein HTTP-Client, der sich mit einem Webdienst verbindet, die Basisauthentifizierung (RFC 2617), die die Benutzer- und Kennwortdaten mit base-64-Kodierung verschleiert. Es wird empfohlen, für erweiterte Sicherheit das HTTPS-Protokoll zu verwenden.

ENABLE- und DISABLE-Klausel Legt fest, ob der Dienst verfügbar ist. Standardmäßig ist ein Dienst aktiviert, wenn er erstellt wird. Beim Erstellen oder Ändern eines Diensts können Sie eine ENABLE- oder DISABLE-Klausel einbauen. Durch das Deaktivieren eines Diensts wird dieser offline gesetzt. Später kann er mit ALTER SERVICE und einer ENABLE-Klausel aktiviert werden. Eine HTTP-Anforderung an einen deaktivierten Dienst gibt in der Regel den HTTP-Status 404 `Not Found` zurück.

METHODS-Klausel Gibt die HTTP-Methoden an, die vom Dienst unterstützt werden. Gültige Werte sind DEFAULT, POST, GET, HEAD, PUT, DELETE und NONE. Ein Sternchen (*) kann als Platzhalter für die Methoden POST, GET und HEAD verwendet werden, welche die Standard-Anforderungstypen für die Diensttypen RAW, HTML und XML sind. Nicht alle HTTP-Methoden sind für alle Diensttypen gültig. Die folgende Tabelle enthält eine Übersicht über die gültigen HTTP-Methoden, die auf die einzelnen Diensttypen angewendet werden können:

Methodenwert	Gilt für Dienst	Beschreibung
DEFAULT	alle	Verwenden Sie DEFAULT, um die Gruppe der standardmäßigen HTTP-Methoden für den angegebenen Diensttyp zurückzusetzen. Kann nicht in eine Liste mit anderen Methodenwerten aufgenommen werden.
POST	RAW, HTML, JSON, XML	Standardmäßig aktiviert.
GET	RAW, HTML, JSON, XML	Standardmäßig aktiviert.
HEAD	RAW, HTML, JSON, XML	Standardmäßig aktiviert.
PUT	RAW, HTML, JSON, XML	Standardmäßig nicht aktiviert.

Methodenwert	Gilt für Dienst	Beschreibung
DELETE	RAW, HTML, JSON, XML	Standardmäßig nicht aktiviert.
NONE	alle	Verwenden Sie NONE, um den Zugriff auf einen Dienst zu deaktivieren.
*	RAW, HTML, JSON, XML	Wie 'POST,GET,HEAD' .

Zum Beispiel können Sie eine der folgenden Klauseln verwenden, um festzulegen, dass ein Dienst alle HTTP-Methodentypen unterstützt:

```
METHODS ' *, PUT, DELETE '
METHODS ' POST, GET, HEAD, PUT, DELETE '
```

Um die Liste der Anforderungstypen für einen beliebigen Dienstyp auf den Standardwert zu setzen, können Sie die folgende Klausel verwenden:

```
METHODS ' DEFAULT '
```

SECURE-Klausel Gibt an, ob der Dienst auf einem sicheren oder nicht sicheren Listener zugänglich sein soll. ON bedeutet, dass nur HTTPS-Verbindungen akzeptiert werden und dass über den HTTP-Port empfangene Verbindungen automatisch zum HTTPS-Port umgeleitet werden. OFF bedeutet, dass sowohl HTTP- als auch HTTPS-Verbindungen akzeptiert werden, vorausgesetzt, die erforderlichen Ports werden beim Starten des Webserver angegeben. Der Standardwert ist OFF.

USER-Klausel Gibt einen Datenbankbenutzer oder eine Gruppe von Benutzern mit den erforderlichen Privilegien an, um die Webdienstanforderung auszuführen. Eine USER-Klausel muss angegeben werden, wenn der Dienst mit AUTHORIZATION OFF konfiguriert wird, und sollte angegeben werden, wenn AUTHORIZATION ON (Standardwert) verwendet wird. Eine HTTP-Anforderung an einen Dienst, der Autorisierung erfordert, liefert den HTTP-Antwortstatus "401 Authorization Required". Basierend auf dieser Antwort fordert der Webbrowser den Benutzer auf, eine Benutzer-ID und ein Kennwort einzugeben.

Vorsicht

Es wird dringend empfohlen, dass Sie eine USER-Klausel angeben, wenn die Autorisierung aktiviert ist (Standardwert). Andernfalls wird die Autorisierung allen Benutzern erteilt.

Die USER-Klausel steuert, welche Datenbank-Benutzerkonten verwendet werden können, um Dienstanforderungen zu verarbeiten. Datenbankzugriffsberechtigungen werden auf die dem Benutzer des Dienstes zugewiesenen Berechtigungen beschränkt.

statement Gibt einen Befehl an, zum Beispiel einen Aufruf einer gespeicherten Prozedur, der beim Zugreifen auf den Dienst aufgerufen werden soll.

Eine HTTP-Anforderung an einen Nicht-DISH-Dienst ohne *statement* gibt den SQL-Ausdruck zurück, der innerhalb seiner URL ausgeführt werden soll. Obwohl eine Autorisierung erforderlich ist, darf diese

Fähigkeit nicht in Produktionssystemen benutzt werden, weil dies den Server für SQL-Injektionen anfällig macht. Wenn eine Anweisung in dem Dienst definiert ist, kann die angegebene SQL-Anweisung als einzige Anweisung über den Dienst ausgeführt werden.

In einer typischen Webdienstanwendung verwenden Sie *statement* zum Aufrufen einer Funktion oder Prozedur. Sie können Hostvariablen als Parameter für den Zugriff auf vom Client gelieferte HTTP-Variablen übergeben.

Die folgende *statement* zeigt einen Prozeduraufruf, der zwei Hostvariablen an eine Prozedur mit dem Namen **AuthenticateUser** übergibt. Dieser Aufruf setzt voraus, dass ein Webclient die Variablen **user_name** und **user_password** liefert:

```
CALL AuthenticateUser ( :user_name, :user_password );
```

Weitere Hinweise zum Übergeben von Hostvariablen an eine Funktion oder Prozedur finden Sie unter „Zugriff auf vom Client bereitgestellte HTTP-Variablen und -Header“ [[SQL Anywhere Server - Programmierung](#)].

Bemerkungen

Die ALTER SERVICE-Anweisung ändert die Attribute eines Webdiensts.

Privilegien

Sie müssen Eigentümer des Diensts sein oder das MANAGE ANY WEB SERVICE-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „CREATE SERVICE-Anweisung [HTTP-Webdienst]“ auf Seite 706
- „DROP SERVICE-Anweisung“ auf Seite 824
- „sp_parse_json-Systemprozedur“ auf Seite 1383
- „Praktische Einführung: Verwenden von SQL Anywhere für den Zugriff auf einen SOAP/DISH-Dienst“ [[SQL Anywhere Server - Programmierung](#)]
- „SYSWEBSERVICE-Systemansicht“ auf Seite 1513
- „Datenbankserveroption -xs“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sa_set_http_header-Systemprozedur“ auf Seite 1323
- „SQL Anywhere als HTTP-Webserver“ [[SQL Anywhere Server - Programmierung](#)]
- „root-Webdienste erstellen und anpassen“ [[SQL Anywhere Server - Programmierung](#)]
- „Webdienstanwendungen auf einem HTTP-Webserver entwickeln“ [[SQL Anywhere Server - Programmierung](#)]
- „Bezeichner“ auf Seite 4
- „ROW-Konstruktor [zusammengesetzt]“ auf Seite 371
- „ARRAY-Konstruktor [zusammengesetzt]“ auf Seite 168

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel zeigt, wie Sie einem vorhandenen Webdienst mit der ALTER SERVICE-Anweisung deaktivieren:

```
CREATE SERVICE WebServiceTable
  TYPE 'RAW'
  AUTHORIZATION OFF
  USER DBA
  AS SELECT *
  FROM SYS.SYSTAB;

ALTER SERVICE WebServiceTable DISABLE;
```

ALTER SERVICE-Anweisung [SOAP-Webdienst]

Ändert einen vorhandenen HTTP über SOAP- oder DISH-Dienst.

Syntax 1 - SOAP über HTTP-Dienste

```
ALTER SERVICE service-name
[ TYPE 'SOAP' ]
[ DATATYPE { ON | OFF | IN | OUT } ]
[ FORMAT { 'DNET' | 'CONCRETE' [ EXPLICIT { ON | OFF } ] | 'XML' | NULL } ]
[ common-attributes ]
[ AS statement ]
```

common-attributes :

```
[ AUTHORIZATION { ON | OFF } ]
[ ENABLE | DISABLE ]
[ METHODS 'method,...' ]
[ SECURE { ON | OFF } ]
[ USER { user-name | NULL } ]
```

method :

```
DEFAULT
| POST
| HEAD
| NONE
```

Syntax 2 - DISH-Dienste

```
ALTER SERVICE service-name
[ TYPE 'DISH' ]
[ GROUP { group-name | NULL } ]
[ FORMAT { 'DNET' | 'CONCRETE' [ EXPLICIT { ON | OFF } ] | 'XML' | NULL } ]
[ common-attributes ]
```

common-attributes :

```
[ AUTHORIZATION { ON | OFF } ]
[ ENABLE | DISABLE ]
[ METHODS 'method,...' ]
[ SECURE { ON | OFF } ]
[ USER { user-name | NULL } ]
```

method :

```
DEFAULT
```

```

| POST
| GET
| HEAD
| NONE
| *

```

Parameter

Die Beschreibung der Klauseln für die ALTER SERVICE-Anweisung ist mit der Beschreibung der Klauseln für die CREATE SERVICE-Anweisung identisch.

Parameter

service-name Webdienstnamen können aus einer beliebigen Sequenz von alphanumerischen Zeichen bestehen sowie die Zeichen Schrägstrich (/), Bindestrich (-), Unterstrich (_), Punkt (.), Ausrufezeichen (!), Tilde (~), Sternchen (*), Apostroph (') , öffnende Klammer (()) oder schließende Klammer (()) enthalten. Allerdings darf der Dienstname nicht mit einem Schrägstrich (/) beginnen oder enden oder zwei oder mehr aufeinander folgende Schrägstriche (z.B. //) enthalten.

Im Gegensatz zu anderen Diensten dürfen Sie in einem DISH-Dienstnamen keinen Schrägstrich (/) verwenden.

Sie können Ihren Dienst **root** nennen, aber dieser Name hat eine spezielle Funktion.

TYPE-Klausel Gibt den Typ des Dienstes an, wobei für jeden Dienst ein bestimmtes Antwortformat definiert wird. Der Typ muss einer der aufgelisteten Typen sein. Es gibt keine Standardeinstellung.

- **'SOAP'** Die Ergebnismenge wird im XML-Format zurückgegeben, das als SOAP-Envelope bezeichnet wird. Das Format der Daten wird durch die FORMAT-Klausel bestimmt. Eine Anforderung an einen SOAP-Dienst muss eine gültige SOAP-Anforderung und nicht nur eine einfache HTTP-Anforderung sein. Weitere Hinweise zu den SOAP-Standards finden Sie unter <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>.
- **'DISH'** Ein DISH-Dienst (Determine SOAP Handler) ist ein SOAP-Endpunkt, der alle SOAP-Dienste in seinem GROUP-Kontext referenziert. Außerdem exponiert er die Schnittstellen zu seinen SOAP-Diensten durch die Erstellung eines WSDL-Objekts (Web Services Description Language), das von SOAP-Client-Toolkits verwendet werden kann.

GROUP-Klausel Ein DISH Dienst ohne GROUP-Klausel exponiert alle in der Datenbank definierten SOAP-Dienste. Laut Konvention kann der SOAP-Dienstname aus einem GROUP- und einem NAME-Element bestehen. Der Name wird durch den letzten Schrägstrich von der Gruppe getrennt. Beispiel: Der Name eines als **'aaa/bbb/ccc'** definierten SOAP-Dienstes ist **'ccc'**, und die Gruppe **'aaa/bbb'**. Die Angabe eines DISH-Dienstes unter Verwendung dieser Konvention ist ungültig. Stattdessen wird eine GROUP-Klausel angewendet, um die Gruppe von SOAP-Diensten anzugeben, für die er als SOAP-Endpunkt fungieren soll.

Hinweis

Schrägstriche werden in WSDL in Unterstriche konvertiert, um gültigen XML-Code zu erzeugen. Lassen Sie bei der Verwendung von einem DISH-Dienst Vorsicht walten, dessen GROUP-Klausel nicht so angegeben ist, dass alle SOAP-Dienste exponiert werden, die möglicherweise Schrägstriche enthalten. Achten Sie darauf, Mehrdeutigkeiten zu vermeiden, wenn Sie Gruppen in Verbindung mit SOAP-Dienstnamen verwenden, die Unterstriche enthalten.

DATATYPE-Klausel Gilt nur für SOAP-Dienste. Wenn DATATYPE OFF angegeben ist, werden SOAP-Eingabeparameter und -Antwortdaten als XMLSchema-Zeichenfolgetypen definiert. In den meisten Fällen werden die tatsächlichen Datentypen bevorzugt, weil durch sie die Notwendigkeit entfällt, dass der SOAP-Client die Daten vor der Berechnung festlegt. Parameterdatentypen werden im Schema-Abschnitt der WDSL angezeigt, der vom DISH-Dienst erstellt wird. Ausgabedatentypen werden als XML-Schematypattribute für die jeweilige Datenspalte dargestellt.

Die folgenden Werte sind bei der DATATYPE-Klausel zulässig:

- **ON** Generiert die Datentypisierung für Eingabeparameter und zurückgegebene Ergebnismengen
- **OFF** Alle Eingabeparameter und Antwortdaten haben den Datentyp **XMLSchema**-Zeichenfolge (Standardwert).
- **IN** Generiert die tatsächlichen Datentypen nur für Eingabeparameter. Antwortdatentypen sind **XMLSchema**-Zeichenfolgen.
- **OUT** Generiert die tatsächlichen Datentypen nur für Antworten. Eingabeparameter werden als **XMLSchema**-Zeichenfolge eingegeben.

Weitere Hinweise zu SOAP-Diensten finden Sie unter „[Praktische Einführung: Verwenden von SQL Anywhere für den Zugriff auf einen SOAP/DISH-Dienst](#)“ [[SQL Anywhere Server - Programmierung](#)].

Weitere Hinweise zur Zuordnung von XML-Schematypen zu SQL-Datentypen finden Sie unter „[SOAP-Datentypen](#)“ [[SQL Anywhere Server - Programmierung](#)].

FORMAT-Klausel Diese Klausel gibt das Ausgabeformat an, das beim Senden von Antworten an SOAP-Clientanwendungen verwendet wird.

Das SOAP-Dienstformat wird durch die dazugehörige DISH-Dienstformatspezifikation bestimmt, wenn es nicht vom SOAP-Dienst angegeben wird. Das Standardformat ist DNET.

SOAP-Anforderungen sollten an den DISH-Dienst (den SQL Anywhere-SOAP-Endpunkt) gerichtet werden, um gemeinsame Formatierungsregeln für eine Gruppe von SOAP-Diensten (SOAP-Vorgängen) zu nutzen. Eine FORMAT-Spezifikation für einen SOAP-Dienst überschreibt diejenige eines DISH-Dienstes. Die Formatspezifikation des DISH-Dienstes wird verwendet, wenn für den SOAP-Dienst keine FORMAT-Klausel definiert ist. Wenn für keinen der beiden Dienste ein FORMAT-Objekt zur Verfügung gestellt wird, ist der Standardwert **'DNET'**.

Folgende Formate werden unterstützt:

- **'DNET'** Die Ausgabe erfolgt in einem System.Data.DataSet-kompatiblen Format zur Verwendung durch .NET-Clientanwendungen. (Standardwert)

- **'CONCRETE'** Dieses Ausgabeformat wird zur Unterstützung von Client-SOAP-Toolkits verwendet, die in der Lage sind, Schnittstellen zu generieren, die Arrays von Zeilen- und Spaltenobjekten darstellen, aber nicht in der Lage sind, das DNET-Format zu bearbeiten. Java- und .NET-Clients können dieses Ausgabeformat einfach bearbeiten.

Das spezifische Format wird innerhalb des WSDL-Codes als explizites DataSet-Objekt oder als **SimpleDataset** bereitgestellt. Beide DataSet-Darstellungen beschreiben eine Datenstruktur für einen Zeilen-Array, in dem jede Zeile ein Array von Spalten enthält. Ein explizites DataSet-Objekt hat den Vorteil, dass es die tatsächliche Form der Ergebnismenge darstellt, indem es Parameternamen und Datentypen für jede Spalte in der Zeile angibt. Im Gegensatz dazu stellt das **SimpleDataset** Zeilen mit einer unbegrenzten Anzahl der Spalten beliebigen Typs bereit.

FORMAT 'CONCRETE' EXPLICIT ON setzt voraus, dass die **Service**-Anweisung eine gespeicherte Prozedur aufruft, die wiederum eine **RESULT**-Klausel definiert. Wenn diese Bedingung erfüllt ist, stellt der SOAP-Dienst ein explizites DataSet bereit, dessen Name mit dem Dienstenamen und dem Zusatz **Dataset** beginnt.

Wenn die Bedingung nicht erfüllt ist, wird ein **SimpleDataset** verwendet.

- **'XML'** Die Ausgabe wird in einem XMLSchema-Zeichenfolgenformat generiert. Die Antwort ist ein XML-Dokument, das zum Extrahieren der Spaltendaten eine weitere Verarbeitung durch den SOAP-Client erfordert. Dieses Format ist für SOAP-Clients geeignet, die keine Zwischen-Schnittstellenobjekte generieren, welche Arrays von Zeilen und Spalten darstellen.
- **NULL** Im Falle eines NULL-Typs verwendet der SOAP- oder DISH-Dienst das Standardverhalten. Der Formattyp eines bestehenden Diensts wird überschrieben, wenn der NULL-Typ in einer ALTER SERVICE-Anweisung verwendet wird.

AUTHORIZATION-Klausel Bestimmt, ob die Benutzer einen Benutzernamen und ein Kennwort über die HTTP-Grundautorisierung angeben müssen, wenn sie eine Verbindung mit dem Dienst herstellen. Der Standardwert ist ON. Wenn AUTHORIZATION auf OFF gesetzt ist, muss die AS-Klausel für SOAP-Dienste verwendet werden und ein Benutzer muss mit der USER-Klausel angegeben werden. Alle Anforderungen werden mit dem Konto und den Privilegien dieses Benutzers ausgeführt. Wenn AUTHORIZATION auf ON gesetzt ist, müssen alle Benutzer einen Benutzernamen und ein Kennwort angeben. Der Zugriff auf diesen Dienst kann durch Angabe eines Benutzer- oder Gruppennamens mithilfe der USER-Klausel eingeschränkt werden. Wenn der Benutzername NULL ist, haben alle bekannten Benutzer Zugriff auf den Dienst. Die AUTHORIZATION-Klausel ermöglicht es Ihren Webdiensten, mithilfe von Datenbankautorisierung und Privilegien den Zugriff auf die Daten in Ihrer Datenbank zu steuern.

Wenn der Autorisierungswert ON ist, benutzt ein HTTP-Client, der sich mit einem Webdienst verbindet, die Basisauthentifizierung (RFC 2617), die die Benutzer- und Kennwortdaten mit base-64-Kodierung verschleiert. Es wird empfohlen, für erweiterte Sicherheit das HTTPS-Protokoll zu verwenden.

ENABLE- und DISABLE-Klausel Legt fest, ob der Dienst verfügbar ist. Standardmäßig ist ein Dienst aktiviert, wenn er erstellt wird. Beim Erstellen oder Ändern eines Diensts können Sie eine ENABLE- oder DISABLE-Klausel einbauen. Durch das Deaktivieren eines Diensts wird dieser offline gesetzt. Später kann er mit ALTER SERVICE und einer ENABLE-Klausel aktiviert werden. Eine HTTP-Anforderung an einen deaktivierten Dienst gibt in der Regel den HTTP Status "404 Not Found" zurück.

METHODS-Klausel Gibt die HTTP-Methoden an, die vom Dienst unterstützt werden. Gültige Werte sind DEFAULT, POST, GET, HEAD, und NONE. Ein Sternchen (*) kann als Kurzform für die Darstellung der POST-, GET- und HEAD-Methoden verwendet werden. Die Standard-Methodentypen für SOAP-Dienste sind POST und HEAD. Die Standard-Methodentypen für DISH-Dienste sind GET, POST und HEAD. Nicht alle HTTP-Methoden sind für alle Diensttypen gültig. Die folgende Tabelle enthält eine Übersicht über die gültigen HTTP-Methoden, die auf die einzelnen Diensttypen angewendet werden können:

Methodenwert	Gilt für Dienst	Beschreibung
DEFAULT	Beide	Verwenden Sie DEFAULT, um die Gruppe der standardmäßigen HTTP-Methoden für den angegebenen Diensttyp zurückzusetzen. Kann nicht in eine Liste mit anderen Methodenwerten aufgenommen werden.
POST	Beide	Standardmäßig aktiviert für SOAP.
GET	Nur DISH	Standardmäßig aktiviert für DISH.
HEAD	Beide	Standardmäßig aktiviert für SOAP und DISH.
NONE	Beide	Verwenden Sie NONE, um den Zugriff auf einen Dienst zu deaktivieren. Bei der Anwendung auf einen SOAP-Dienst kann auf den Dienst nicht direkt über eine SOAP-Anforderung zugegriffen werden. Dies erzwingt exklusiven Zugriff auf einen SOAP-Vorgang über einen als SOAP-Endpunkt fungierenden DISH-Dienst. Es wird empfohlen, für die einzelnen SOAP-Dienste METHOD NONE anzugeben.
*	Nur DISH	Wie ' POST,GET,HEAD '.

Zum Beispiel können Sie die folgende Klausel verwenden, um festzulegen, dass ein Dienst alle SOAP über HTTP-Methodentypen unterstützt:

```
METHODS 'POST,HEAD'
```

Um die Liste der Anforderungstypen für einen beliebigen Dienstyp auf den Standardwert zu setzen, können Sie die folgende Klausel verwenden:

```
METHODS 'DEFAULT'
```

SECURE-Klausel Gibt an, ob der Dienst auf einem sicheren oder nicht sicheren Listener zugänglich sein soll. ON bedeutet, dass nur HTTPS-Verbindungen akzeptiert werden und dass über den HTTP-Port empfangene Verbindungen automatisch zum HTTPS-Port umgeleitet werden. OFF bedeutet, dass sowohl HTTP- als auch HTTPS-Verbindungen akzeptiert werden, vorausgesetzt, die erforderlichen Ports werden beim Starten des Webserver angegeben. Der Standardwert ist OFF.

USER-Klausel Gibt einen Datenbankbenutzer oder eine Gruppe von Benutzern mit den erforderlichen Privilegien an, um die Webdienstanforderung auszuführen. Eine USER-Klausel muss angegeben werden, wenn der Dienst mit AUTHORIZATION OFF konfiguriert wird, und sollte angegeben werden, wenn AUTHORIZATION ON (Standardwert) verwendet wird. Eine HTTP-Anforderung an einen Dienst, der Autorisierung erfordert, liefert den HTTP-Antwortstatus "401 Authorization Required". Basierend auf dieser Antwort fordert der Webbrowser den Benutzer auf, eine Benutzer-ID und ein Kennwort einzugeben.

Vorsicht

Es wird dringend empfohlen, dass Sie eine USER-Klausel angeben, wenn die Autorisierung aktiviert ist (Standardwert). Andernfalls wird die Autorisierung allen Benutzern erteilt.

Die USER-Klausel steuert, welche Datenbank-Benutzerkonten verwendet werden können, um Dienstanforderungen zu verarbeiten. Datenbankzugriffsberechtigungen werden auf die dem Benutzer des Dienstes zugewiesenen Berechtigungen beschränkt.

statement Gibt einen Befehl an, zum Beispiel einen Aufruf einer gespeicherten Prozedur, der beim Zugreifen auf den Dienst aufgerufen werden soll.

Ein DISH-Dienst ist der einzige Dienst, der entweder eine Null-Anweisung definieren oder keine Anweisung definieren muss. Ein SOAP-Dienst muss eine Anweisung definieren. Jeder andere SERVICE-Wert kann eine NULL-Anweisung definieren, jedoch nur, wenn er mit AUTHORIZATION ON konfiguriert ist.

Eine HTTP-Anforderung an einen Nicht-DISH-Dienst ohne *statement* gibt den SQL-Ausdruck zurück, der innerhalb seiner URL ausgeführt werden soll. Obwohl eine Autorisierung erforderlich ist, darf diese Fähigkeit nicht in Produktionssystemen benutzt werden, weil dies den Server für SQL-Injektionen anfällig macht. Wenn eine Anweisung in dem Dienst definiert ist, kann die angegebene SQL-Anweisung als einzige Anweisung über den Dienst ausgeführt werden.

In einer typischen Webdienstanwendung verwenden Sie *statement* zum Aufrufen einer Funktion oder Prozedur. Sie können Hostvariablen als Parameter für den Zugriff auf vom Client gelieferte HTTP-Variablen übergeben.

Die folgende *statement* zeigt einen Prozeduraufruf, der zwei Hostvariablen an eine Prozedur mit dem Namen **AuthenticateUser** übergibt. Dieser Aufruf setzt voraus, dass ein Webclient die Variablen **user_name** und **user_password** liefert:

```
CALL AuthenticateUser ( :user_name, :user_password );
```

Weitere Hinweise zum Übergeben von Hostvariablen an eine Funktion oder Prozedur finden Sie unter „Zugriff auf vom Client bereitgestellte HTTP-Variablen und -Header“ [*SQL Anywhere Server - Programmierung*].

Bemerkungen

Die ALTER SERVICE-Anweisung ändert die Attribute eines Webdiensts.

Privilegien

Sie müssen Eigentümer des Diensts sein oder das MANAGE ANY WEB SERVICE-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „CREATE SERVICE-Anweisung [SOAP-Webdienst]“ auf Seite 713
- „DROP SERVICE-Anweisung“ auf Seite 824
- „Praktische Einführung: Verwenden von SQL Anywhere für den Zugriff auf einen SOAP/DISH-Dienst“ [*SQL Anywhere Server - Programmierung*]
- „SYSWEBSERVICE-Systemansicht“ auf Seite 1513
- „Datenbankserveroption -xs“ [*SQL Anywhere Server - Datenbankadministration*]
- „SQL Anywhere als HTTP-Webserver“ [*SQL Anywhere Server - Programmierung*]
- „root-Webdienste erstellen und anpassen“ [*SQL Anywhere Server - Programmierung*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel zeigt, wie Sie einem vorhandenen Webdienst mit der ALTER SERVICE-Anweisung deaktivieren:

```
CREATE SERVICE WebServiceTable
  TYPE 'SOAP'
  AUTHORIZATION OFF
  USER DBA
  AS SELECT *
    FROM SYS.SYSTAB;

ALTER SERVICE WebServiceTable DISABLE;
```

ALTER SPATIAL REFERENCE SYSTEM-Anweisung

Ändert die Einstellungen eines vorhandenen räumlichen Bezugssystems. Vor dem Ändern eines räumlichen Bezugssystems lesen Sie bitte die Hinweise im Abschnitt 'Bemerkungen'.

Syntax

ALTER SPATIAL REFERENCE SYSTEM

srs-name

[*srs-attribute* [*srs-attribute* ...]]

srs-name : *string*

srs-attribute :

SRID *srs-id*

DEFINITION { *definition-string* | **NULL** }
ORGANIZATION { *organization-name* **IDENTIFIED BY** *organization-srs-id* | **NULL** }
TRANSFORM DEFINITION { *transform-definition-string* | **NULL** }
LINEAR UNIT OF MEASURE *linear-unit-name*
ANGULAR UNIT OF MEASURE { *angular-unit-name* | **NULL** }
TYPE { **ROUND EARTH** | **PLANAR** }
COORDINATE *coordinate-name* { **UNBOUNDED** | **BETWEEN** *low-number* **AND** *high-number* }
ELLIPSOID SEMI MAJOR AXIS *semi-major-axis-length* { **SEMI MINOR AXIS** *semi-minor-axis-length* |
INVERSE FLATTENING *inverse-flattening-ratio* }
SNAP TO GRID { *grid-size* | **DEFAULT** }
TOLERANCE { *tolerance-distance* | **DEFAULT** }
POLYGON FORMAT *polygon-format*
STORAGE FORMAT *storage-format*

srs-id : *integer*

semi-major-axis-length : *number*

semi-minor-axis-length : *number*

inverse-flattening-ratio : *number*

grid-size : *DOUBLE* : normalerweise zwischen 0 und 1

tolerance-distance : *number*

axis-order : { '*x/y/z/m*' | '*long/lat/z/m*' | '*lat/long/z/m*' }

polygon-format : { '**CounterClockWise**' | '**Clockwise**' | '**EvenOdd**' }

storage-format : { '**Internal**' | '**Original**' | '**Mixed**' }

Parameter

Vollständige Definitionen für jede der Klauseln werden unter der Anweisung CREATE SPATIAL REFERENCE SYSTEM beschrieben.

IDENTIFIED BY-Klausel Verwenden Sie diese Klausel zum Ändern der SRID-Nummer für das räumliche Bezugssystem.

DEFINITION-Klausel Verwenden Sie diese Klausel, um Standardeinstellungen für das Koordinatensystem festzulegen oder aufzuheben.

ORGANIZATION-Klausel Mit dieser Klausel geben Sie Informationen über die Organisation an, die das räumliche Bezugssystem erstellt hat, auf dem das räumliche Bezugssystem basiert.

TRANSFORM DEFINITION-Klausel Mit dieser Klausel geben Sie eine Beschreibung der Transformationsdefinition an, die für das räumliche Bezugssystem verwendet werden soll. Derzeit wird nur die PROJ.4-Transformationsdefinition unterstützt.

Die Transformationsdefinition wird von der ST_Transform-Methode beim Transformieren von Daten zwischen räumlichen Bezugssystemen verwendet. Einige Transformationen können möglicherweise auch dann durchgeführt werden, wenn keine *transform-definition-string* definiert ist.

COORDINATE-Klausel Mit dieser Klausel geben Sie die Grenzen für die Dimensionen des räumlichen Bezugssystems an. *coordinate-name* ist der Name des von dem räumlichen Bezugssystem verwendeten Koordinatensystems. Bei nicht geografischen Typen kann *coordinate-name* x, y oder m sein. Bei geografischen Typen kann *coordinate-name* LATITUDE, LONGITUDE, z oder m sein.

LINEAR UNIT OF MEASURE-Klausel Verwenden Sie diese Klausel zum Angeben der linearen Maßeinheit für das räumliche Bezugssystem. Der angegebene Wert muss einer linearen Maßeinheit gemäß der Definition in der ST_UNITS_OF_MEASURE-Systemansicht entsprechen.

ANGULAR UNIT OF MEASURE-Klausel Verwenden Sie diese Klausel zum Angeben der Winkelmaßeinheit für das räumliche Bezugssystem. Der angegebene Wert muss einer Winkelmaßeinheit gemäß der Definition in der ST_UNITS_OF_MEASURE-Systemansicht entsprechen.

TYPE-Klausel Verwenden Sie die TYPE-Klausel, um zu steuern, wie das räumliche Bezugssystem Linien zwischen Punkten interpretiert. Bei geografischen räumlichen Bezugssystemen kann die TYPE-Klausel entweder ROUND EARTH (Standardwert) oder PLANAR lauten. Bei nicht geografischen räumlichen Bezugssystemen muss die TYPE-Klausel PLANAR lauten.

ELLIPSOID-Klausel Verwenden Sie die ELLIPSOID-Klausel, um die Werte anzugeben, die bei räumlichen Bezugssystemen vom Typ ROUND EARTH für die Erddarstellung als Ellipsoid verwendet werden sollen. Wenn die DEFINITION-Klausel vorliegt, kann sie eine Ellipsoiddefinition angeben. Wenn die ELLIPSOID-Klausel angegeben ist, hebt sie diesen Standard-Ellipsoid auf.

SNAP TO GRID-Klausel Verwenden Sie bei räumlichen Bezugssystemen mit dem Modell "plane Erde" (PLANAR) die SNAP TO GRID-Klausel, um die Größe des Rasters zu definieren, das von SQL Anywhere beim Ausführen von Berechnungen verwendet wird. Geben Sie SNAP TO GRID DEFAULT an, um die Rastergröße auf den Standardwert zu setzen, den der Datenbankserver verwenden würde.

Bei räumlichen Bezugssystemen mit dem Modell "gewölbte Erde" (ROUND EARTH) muss SNAP TO GRID auf 0 gesetzt sein.

TOLERANCE-Klausel Verwenden Sie bei räumlichen Bezugssystemen mit dem Modell "plane Erde" (PLANAR) die TOLERANCE-Klausel, um die Genauigkeit anzugeben, die beim Vergleichen von Punkten verwendet werden soll.

Bei räumlichen Bezugssystemen mit dem Modell "gewölbte Erde" muss TOLERANCE auf 0 gesetzt sein.

POLYGON FORMAT-Klausel Verwenden Sie die POLYGON FORMAT-Klausel zum Ändern der Polygoninterpretation. Folgende Werte werden unterstützt:

- 'CounterClockwise'
- 'Clockwise'
- 'EvenOdd'

Der Standardwert für das Polygonformat ist 'EvenOdd'.

STORAGE FORMAT-Klausel Verwenden Sie die STORAGE FORMAT-Klausel, um zu steuern, was beim Laden von räumlichen Daten in die Datenbank gespeichert wird. Die möglichen Werte sind:

- **'Internal'** SQL Anywhere speichert nur die normalisierte Darstellung. Geben Sie dies an, wenn die ursprünglichen Eingabemerkmale nicht reproduziert werden müssen. Dies ist die Standardeinstellung für planare räumliche Bezugssysteme (TYPE PLANAR).

Hinweis

Wenn Sie MobiLink zum Synchronisieren Ihrer räumlichen Daten verwenden, sollten Sie stattdessen **Mixed** angeben. MobiLink testet die Gleichheit während der Synchronisation, was die Daten in ihrem ursprünglichen Format erfordert.

- **'Original'** SQL Anywhere speichert nur die ursprüngliche Darstellung. Die ursprünglichen Eingabemerkmale können reproduziert werden, aber bei allen Vorgängen mit den gespeicherten Werten müssen die Normalisierungsschritte wiederholt werden, wodurch Vorgänge mit den Daten möglicherweise verlangsamt werden.
- **'Mixed'** SQL Anywhere speichert die interne Version, und wenn diese von der ursprünglichen Version abweicht, speichert SQL Anywhere auch die Originalversion. Durch das Speichern beider Versionen können die ursprünglichen Darstellungsmerkmale reproduziert werden und bei Vorgängen mit den gespeicherten Werten müssen nicht die Normalisierungsschritte wiederholt werden. Die Speicheranforderungen werden jedoch möglicherweise signifikant erhöht, da potenziell für jede Geometrie zwei Darstellungen gespeichert werden.

"Mixed" ist das Standardformat für räumliche Bezugssysteme mit dem Modell "gewölbte Erde" (TYPE ROUND EARTH).

Bemerkungen

Sie können ein räumliches Bezugssystem nicht ändern, wenn es durch bestehende Daten referenziert wird. Wenn Sie z.B. eine Spalte als ST_Point(SRID=8743) deklariert haben, können Sie das räumliche Bezugssystem mit SRID 8743 nicht ändern. Dies liegt daran, dass viele Attribute von räumlichen Bezugssystemen, wie etwa das Speicherformat, Auswirkungen auf das Speicherformat der Daten haben. Wenn Sie über Daten verfügen, die die SRID referenzieren, erstellen Sie ein neues räumliches Bezugssystem und transformieren die Daten in die neue SRID.

Privilegien

Sie müssen Eigentümer des räumlichen Bezugssystems sein oder eines der folgenden Privilegien haben:

- ALTER-Privileg für das räumliche Bezugssystem
- MANAGE ANY SPATIAL OBJECT-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Nebenwirkungen

Keine

Siehe auch

- „CREATE SPATIAL REFERENCE SYSTEM-Anweisung“ auf Seite 719
- „DROP SPATIAL REFERENCE SYSTEM-Anweisung“ auf Seite 825
- „Räumliche Daten“ [*SQL Anywhere Server - Unterstützung für räumliche Daten*]
- ST_Transform-Methode für den ST_Geometry-Datentyp [*SQL Anywhere Server - Unterstützung für räumliche Daten*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Mit dem folgenden Beispiel wird das Polygonformat eines fiktiven räumlichen Bezugssystems mit dem Namen mySpatialRef in "EvenOdd" geändert.

```
ALTER SPATIAL REFERENCE SYSTEM mySpatialRef  
POLYGON FORMAT 'EvenOdd';
```

ALTER STATISTICS-Anweisung

Steuert, ob Statistiken für eine oder mehrere Spalten in einer Tabelle automatisch aktualisiert werden.

Syntax

```
ALTER STATISTICS  
[ ON ] table [ ( column1 [ , column2 ... ] ) ]  
AUTO UPDATE { ENABLE | DISABLE }
```

Parameter

ON Das Wort ON ist optional. Die Angabe hat keine Auswirkung auf die Anweisung.

AUTO UPDATE-Klausel Geben Sie an, ob das automatische Aktualisieren von Statistiken für die Spalte(n) aktiviert oder deaktiviert werden soll.

Bemerkungen

Während der normalen Ausführung von Abfragen, DML-Anweisungen und LOAD TABLE-Anweisungen führt der Datenbankserver automatisch Statistiken für die Verwendung durch den Optimierer. Der Vorteil, den das Führen von Statistiken für manche Spalten bietet, wird möglicherweise durch den Overhead aufgehoben, der für ihre Generierung notwendig ist. Wenn z.B. eine Spalte nicht häufig abgefragt wird oder wenn sie periodischen Massenänderungen unterworfen ist, die eventuell zurückgesetzt werden, lohnt es sich kaum, ihre Statistik zu aktualisieren. Verwenden Sie die ALTER STATISTICS-Anweisung, um das automatische Aktualisieren von Statistiken für diese Typen von Spalten zu unterdrücken.

Wenn das automatische Aktualisieren deaktiviert ist, können Sie weiterhin die Statistik für die Spalte aktualisieren, indem Sie die CREATE STATISTICS- und die DROP STATISTICS-Anweisung verwenden. Sie sollten sie allerdings nur aktualisieren, wenn feststeht, dass dies eine positive Auswirkung auf die Performance hat. Normalerweise sollten Spaltenstatistiken nicht deaktiviert werden.

Privilegien

Sie müssen Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- **MANAGE ANY STATISTICS**-Systemprivileg
- **ALTER ANY OBJECT**-Systemprivileg

Nebenwirkungen

Wenn das automatische Aktualisieren deaktiviert wurde, können Statistiken veralten. Eine Neu-Aktivierung bringt sie nicht unmittelbar auf den letzten Stand. Führen Sie ggf. die **CREATE STATISTICS**-Anweisung aus, um sie neu zu erstellen.

Siehe auch

- [„CREATE STATISTICS-Anweisung“ auf Seite 729](#)
- [„DROP STATISTICS-Anweisung“ auf Seite 827](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel deaktiviert das automatische Aktualisieren der Statistik in der Street-Spalte der Customers-Tabelle:

```
ALTER STATISTICS GROUPO.Customers ( Street ) AUTO UPDATE DISABLE;
```

ALTER SYNCHRONIZATION PROFILE-Anweisung [MobiLink]

Ändert ein SQL Anywhere-Synchronisationsprofil. Synchronisationsprofile sind benannte Sammlungen von Synchronisationsoptionen, die zur Steuerung der Synchronisation verwendet werden können.

Syntax

```
ALTER SYNCHRONIZATION PROFILE name  
MERGE string
```

Parameter

name Der Name des zu ändernden Synchronisationsprofils.

MERGE-Klausel Verwenden Sie diese Klausel, um bestehende Optionen in einem Synchronisationsprofil zu ändern oder ihm neue hinzuzufügen.

string Eine Zeichenfolge mit mindestens einem Paar "Option=Wert", getrennt durch Semikola.
Beispiel: 'option1=value1;option2=value2'.

Bemerkungen

Synchronisationsprofile legen fest, wie eine SQL Anywhere-Datenbank mit dem MobiLink-Server synchronisiert wird.

Wenn MERGE in der ALTER SYNCHRONIZATION PROFILE-Anweisung verwendet wird, werden in der Zeichenfolge angegebene Optionen den bereits im Synchronisationsprofil befindlichen Optionen hinzugefügt. Wenn eine in der Zeichenfolge enthaltene Option im Profil bereits vorhanden ist, ersetzt der Wert aus der Zeichenfolge den bereits im Profil gespeicherten Wert.

Beispiel: Wenn Sie folgende Anweisungen ausführen, bleibt das Profil *myProfile* beim Wert *subscription=s2;verbosity=high;uploadonly=on*.

```
CREATE SYNCHRONIZATION PROFILE myProfile 'subscription=p1;verbosity=high';
ALTER SYNCHRONIZATION PROFILE myProfile MERGE
'subscription=p2;uploadonly=on';
```

Verwenden Sie beim Festlegen von erweiterten Optionen die folgende Syntax:

```
ALTER SYNCHRONIZATION PROFILE myprofile MERGE
's=mysub;e={ctp=tcip;adr='host=localhost;port=2439'}';
```

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE SYNCHRONIZATION PROFILE-Anweisung [MobiLink]“ auf Seite 732
- „DROP SYNCHRONIZATION PROFILE-Anweisung [MobiLink]“ auf Seite 829

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink]

Ändert die Eigenschaften einer Synchronisationssubskription in einer entfernten SQL Anywhere-Datenbank.

Syntax

```
ALTER SYNCHRONIZATION SUBSCRIPTION
{ subscription-name | TO publication-name [ FOR ml-username, ... ] } { alter-clause ... }
```

alter-clause :

```
RENAME new-subscription-name
| TYPE network-protocol
| ADDRESS protocol-options
| ADD OPTION option=value, ...
| ALTER OPTION option=value, ...
| DELETE { ALL OPTION | OPTION option, ... }
| SET SCRIPT VERSION=script-version
```

subscription-name : *identifier*

publication-name : *identifier*

ml-username : *identifier*

new-subscription-name : *identifier*

network-protocol : **http** | **https** | **tls** | **tcip** | **NULL**

protocol-options : *string* | **NULL**

value : *string* | *integer*

option : *identifier*

script-version : *string* | **NULL**

Parameter

TO-Klausel Diese Klausel gibt den Namen einer Publikation an.

Wenn die TO-Klausel ohne FOR-Klausel verwendet wird, können Sie nicht mit der RENAME- oder SET SCRIPT VERSION-Klausel arbeiten.

FOR-Klausel Diese Klausel gibt einen oder mehrere MobiLink-Benutzernamen an.

Lassen Sie die FOR-Klausel weg, wenn Sie den Protokolltyp, Protokolloptionen und erweiterte Optionen für eine Publikation festlegen möchten.

Wenn die TO-Klausel ohne FOR-Klausel verwendet wird, können Sie nicht mit der RENAME- oder SET SCRIPT VERSION-Klausel arbeiten.

RENAME-Klausel Diese Klausel gibt einen neuen Namen für die Subskription an.

Wenn die TO-Klausel ohne FOR-Klausel verwendet wird, können Sie nicht mit der RENAME-Klausel arbeiten.

TYPE-Klausel Mit dieser Klausel wird das Netzwerkprotokoll für die Synchronisation angegeben. Das Standardprotokoll ist tcip.

ADDRESS-Klausel Diese Klausel gibt Netzwerkprotokolloptionen inklusive des Standorts des MobiLink-Servers an.

Klauseln ADD OPTION, ALTER OPTION, DELETE OPTION und DELETE ALL OPTION Mit diesen Klauseln können Sie erweiterte Optionen hinzufügen, ändern oder löschen bzw. alle erweiterten Optionen löschen. Sie können nur eine Option in jeder Klausel angeben. Für "Alle löschen" wird keine Option angegeben.

Die Werte für die einzelnen Optionen dürfen nicht die Zeichen "=" oder "," oder ";" enthalten.

SET SCRIPT VERSION-Klausel Mit dieser Klausel wird die Skriptversion für die Synchronisation angegeben. Sie können die Skriptversion ändern, ohne eine Schemaänderung durchführen zu müssen.

Wenn die TO-Klausel ohne FOR-Klausel verwendet wird, können Sie nicht mit der SET SCRIPT VERSION-Klausel arbeiten.

☛ **Cloud-Hinweis:** Sie können **NULL** für den *script-version*-Wert für Cloud-Tenant-Datenbanken angeben.

Bemerkungen

Die Parameter *network-protocol*, *protocol-options* und *options* können an mehreren Stellen festgelegt werden.

Diese Anweisung bewirkt, dass Optionen und andere Informationen in der SQL Anywhere-Systemtabelle ISYSSYNC gespeichert werden. Je nachdem, welche Privilegien ein Benutzer hat, kann er möglicherweise die Informationen anzeigen, zu denen auch Kennwörter und Verschlüsselungszertifikate gehören können. Um dieses potenzielle Sicherheitsproblem zu umgehen, können Sie die Informationen in der dbmlsync-Befehlszeile angeben.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ auf Seite 690
- „DROP PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ auf Seite 817
- „CREATE SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink]“ auf Seite 733
- SQL Anywhere MobiLink-Clients: „Erstellen von Synchronisationssubskriptionen“ [*MobiLink - Clientadministration*]
- UltraLite MobiLink-Clients: „UltraLite-Clientsynchronisationsplanung“ [*UltraLite - Datenbankverwaltung*]
- „SYSSYNC-Systemansicht“ auf Seite 1489
- „Netzwerkprotokolloptionen des MobiLink-Clients“ [*MobiLink - Clientadministration*]
- „Erweiterte Optionen von MobiLink SQL Anywhere-Clients“ [*MobiLink - Clientadministration*]
- „Erweiterte Option CommunicationType (ctp)“ [*MobiLink - Clientadministration*]
- „dbmlsync-Syntax“ [*MobiLink - Clientadministration*]
- Prioritätenfolge [*MobiLink - Clientadministration*]
- „Skriptversionen“ [*MobiLink - Serveradministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird die Adresse des MobiLink-Servers für die Subskription "sales" geändert:

```
ALTER SYNCHRONIZATION SUBSCRIPTION sales
TYPE TCPIP
ADDRESS 'host=10.11.12.132;port=2439';
```

ALTER SYNCHRONIZATION USER-Anweisung [MobiLink]

Ändert die Eigenschaften eines MobiLink-Benutzers in einer entfernten SQL Anywhere-Datenbank.

Syntax

```
ALTER SYNCHRONIZATION USER ml-username
[ TYPE network-protocol ]
[ ADDRESS protocol-options ]
[ ADD OPTION option=value, ... ]
[ ALTER OPTION option=value, ... ]
[ DELETE { ALL OPTION | OPTION option } ]
```

ml-username : *identifier*

network-protocol : **http** | **https** | **tls** | **tcpip** | **NULL**

protocol-options : *string* | **NULL**

value : *string* | *integer*

Parameter

TYPE-Klausel Mit dieser Klausel wird das Netzwerkprotokoll für die Synchronisation angegeben. Das Standardprotokoll ist tcpip.

ADDRESS-Klausel Diese Klausel gibt *protocol-options* in der Form *Schlüsselwort=Wert* an, durch Semikola getrennt. Welche Einstellungen Sie angeben, hängt von dem verwendeten Kommunikationsprotokoll ab (TCP/IP, TLS, HTTP oder HTTPS).

Klauseln ADD OPTION, ALTER OPTION, DELETE OPTION und DELETE ALL OPTION Mit diesen Klauseln können Sie erweiterte Optionen hinzufügen, ändern oder löschen bzw. alle erweiterten Optionen löschen. Sie können nur eine Option in jeder Klausel angeben. Für "Alle löschen" wird keine Option angegeben.

Bemerkungen

Die Parameter *network-protocol*, *protocol-options* und *options* können an mehreren Stellen festgelegt werden.

Diese Anweisung bewirkt, dass Optionen und andere Informationen in der SQL Anywhere-Systemtabelle ISYSSYNC gespeichert werden. Je nachdem, welche Privilegien ein Benutzer hat, sind möglicherweise Kennwörter und Verschlüsselungszertifikate sichtbar. Um dieses potenzielle Sicherheitsproblem zu umgehen, können Sie die Informationen in der dbmsync-Befehlszeile angeben.

Erfordert exklusiven Zugriff auf alle in der Publikation referenzierten Tabellen.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „dbmlsync-Syntax“ [*MobiLink - Clientadministration*]
- „CREATE SYNCHRONIZATION USER-Anweisung [MobiLink]“ auf Seite 735
- „Erweiterte Option CommunicationType (ctp)“ [*MobiLink - Clientadministration*]
- „DROP SYNCHRONIZATION USER-Anweisung [MobiLink]“ auf Seite 831
- „MobiLink-Benutzer“ [*MobiLink - Clientadministration*]
- „SYSSYNC-Systemansicht“ auf Seite 1489
- „Erweiterte Optionen von MobiLink SQL Anywhere-Clients“ [*MobiLink - Clientadministration*]
- Prioritätenfolge [*MobiLink - Clientadministration*]
- „Netzwerkprotokolloptionen des MobiLink-Clients“ [*MobiLink - Clientadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

ALTER TABLE-Anweisung

Ändert eine Tabellendefinition oder deaktiviert abhängige Ansichten.

Syntax 1 - Ändern einer vorhandenen Tabelle

```
ALTER TABLE [owner.]table-name { alter-clause, ... }
```

alter-clause :

ADD *create-clause*

| **ALTER** *column-name* *column-alteration*

| **ALTER** [**CONSTRAINT** *constraint-name*] **CHECK** (*condition*)

| **DROP** *drop-object*

| **RENAME** *rename-object*

| *table-alteration*

create-clause :

column-name [**AS**] *column-data-type* [*new-column-attribute* ...]

| *table-constraint*

| **PCTFREE** *integer*

column-alteration :

{ *column-data-type* | *alterable-column-attribute* } [*alterable-column-attribute* ...]

| **SET COMPUTE** (*compute-expression*)

| **ADD** [*constraint-name*] **CHECK** (*condition*)

| **DROP** { **DEFAULT** | **COMPUTE** | **CHECK** | **CONSTRAINT** *constraint-name* }

drop-object :

column-name

| **CHECK**

| **CONSTRAINT** *constraint-name*

| **UNIQUE** [**CLUSTERED**] (*index-columns-list*)

| **FOREIGN KEY** *fkey-name*

| **PRIMARY KEY**

rename-object :

new-table-name

| *column-name* **TO** *new-column-name*
 | **CONSTRAINT** *constraint-name* **TO** *new-constraint-name*

table-alteration :

PCTFREE **DEFAULT**
 | **[NOT] ENCRYPTED**

new-column-attribute :

[NOT] NULL
DEFAULT *default-value*
COMPRESSED
INLINE { *inline-length* | **USE DEFAULT** }
PREFIX { *prefix-length* | **USE DEFAULT** }
[NO] INDEX
IDENTITY
COMPUTE (*expression*)
column-constraint

table-constraint :

[CONSTRAINT *constraint-name*] {
 | **CHECK** (*condition*)
 | **UNIQUE** [**CLUSTERED** | **NONCLUSTERED**] (*column-name* [**ASC** | **DESC**], ...)
 | **PRIMARY KEY** [**CLUSTERED** | **NONCLUSTERED**] (*column-name* [**ASC** | **DESC**], ...)
 | *foreign-key*
}

column-constraint :

[CONSTRAINT *constraint-name*] {
 | **CHECK** (*condition*)
 | **UNIQUE** [**CLUSTERED** | **NONCLUSTERED**] [**ASC** | **DESC**]
 | **PRIMARY KEY** [**CLUSTERED** | **NONCLUSTERED**] [**ASC** | **DESC**]
 | **REFERENCES** *table-name* [(*column-name*)]
 | [**MATCH** [**UNIQUE**] { **SIMPLE** | **FULL** }]
 | [*actions*] [**CLUSTERED** | **NONCLUSTERED**]
 | **NOT NULL**
}

alterable-column-attribute :

[NOT] NULL
DEFAULT *default-value*
[CONSTRAINT *constraint-name*] **CHECK** { **NULL** | (*condition*) }
[NOT] COMPRESSED
INLINE { *inline-length* | **USE DEFAULT** }
PREFIX { *prefix-length* | **USE DEFAULT** }
[NO] INDEX

default-value :

special-value
string
global variable
[-] number
 (*constant-expression*)
 (*sequence-expression*)
built-in-function(*constant-expression*)
AUTOINCREMENT
GLOBAL AUTOINCREMENT [(*partition-size*)]

special-value :

CURRENT DATABASE
CURRENT DATE
CURRENT TIME
[CURRENT] TIMESTAMP
CURRENT PUBLISHER
CURRENT REMOTE USER
[CURRENT] USER
[CURRENT] UTC TIMESTAMP
LAST USER
NULL

foreign-key :

[NOT NULL] FOREIGN KEY [role-name]
[(column-name [ASC | DESC], ...)
REFERENCES table-name
[(pkey-column-list)]
[MATCH [UNIQUE] { SIMPLE | FULL }]
[actions] [CHECK ON COMMIT] [CLUSTERED]
[FOR OLAP WORKLOAD]

actions :

[ON UPDATE action] [ON DELETE action]

action :

CASCADE | SET NULL | SET DEFAULT | RESTRICT

Syntax 2 - Ansichtenabhängigkeiten deaktivieren

```
ALTER TABLE [owner.]table-name {  
  DISABLE VIEW DEPENDENCIES  
}
```

Syntax 3 - Tabelleneigentümer ändern

```
ALTER TABLE [owner.]table-name ALTER OWNER TO owner  
[ { PRESERVE | DROP } PRIVILEGES ]  
[ { PRESERVE | DROP } FOREIGN KEYS ]
```

Parameter

Hinzufügeklauseln Im folgenden Abschnitt werden die Klauseln erläutert, die zum Hinzufügen von Spalten oder Integritätsregeln zu einer Tabelle verwendet werden:

ADD column-name [AS] column-data-type [new-column-attribute ...]-Klausel Verwenden Sie diese Klausel, um der Tabelle eine neue Spalte hinzuzufügen, indem Sie den Datentyp und Attribute für die Spalte angeben.

NULL- und NOT NULL-Klauseln Verwenden Sie diese Klausel, um anzugeben, ob NULL in der Spalte zulässig ist. Standardmäßig ist NULL bei neuen Spalten zulässig. Bei BIT-Typ-Spalten wird automatisch die NOT NULL-Integritätsregel angewendet, wenn sie erstellt werden. Sie können aber eine BIT-Typ-Spalte als nullwertfähig deklarieren.

DEFAULT-Klausel Wenn ein DEFAULT-Wert angegeben ist, wird er als Wert für die Spalte in jeder INSERT-Anweisung benutzt, die für diese Spalte keinen Wert angibt. Wenn kein DEFAULT-Wert angegeben wird, entspricht dies DEFAULT NULL.

Die folgende Liste enthält mögliche Werte für DEFAULT:

- **special-value** Sie verwenden eine von mehreren Spezialwerten in der DEFAULT-Klausel.
- **[CURRENT] TIMESTAMP** Damit kann angezeigt werden, wann die einzelnen Zeilen in der Tabelle zuletzt geändert wurden. Ist eine Spalte mit DEFAULT TIMESTAMP deklariert, dann wird bei Einfügungen ein Standardwert bereitgestellt, und der Wert wird bei jeder Aktualisierung der Zeile mit Datum und Tageszeit aktualisiert.

Um einen Standardwert beim Einfügen bereitzustellen, die Spalte aber nicht bei jeder Zeilenaktualisierung zu aktualisieren, verwenden Sie DEFAULT CURRENT TIMESTAMP anstatt DEFAULT TIMESTAMP.

Spalten, die mit DEFAULT TIMESTAMP deklariert wurden, enthalten eindeutige Werte. Damit können Anwendungen fast gleichzeitige Aktualisierungen derselben Zeile ermitteln. Wenn der aktuelle TIMESTAMP-Wert mit dem letzten Wert übereinstimmt, wird er durch den Wert der Option default_timestamp_increment hochgezählt.

In SQL Anywhere können Sie TIMESTAMP-Werte automatisch kürzen, indem Sie die Option default_timestamp_increment verwenden. Dies ist hilfreich, wenn Sie die Kompatibilität mit anderen Datenbankprogrammen sicherstellen möchten, die timestamp-Werte weniger genau erfassen.

Die globale Variable @@dbts gibt einen TIMESTAMP-Wert zurück, der den zuletzt für eine Spalte mit DEFAULT TIMESTAMP generierten Wert repräsentiert.

- **[CURRENT] UTC TIMESTAMP** Damit kann angezeigt werden, wann die einzelnen Zeilen in der Tabelle zuletzt geändert wurden. Ist eine Spalte mit DEFAULT UTC TIMESTAMP deklariert, dann wird bei Einfügungen ein Standardwert bereitgestellt, und der Wert wird bei jeder Aktualisierung der Zeile mit der aktuellen Coordinated Universal Time (UTC) aktualisiert, wenn die Zeile aktualisiert wird.

Um einen Standardwert beim Einfügen bereitzustellen, die Spalte aber nicht bei jeder Zeilenaktualisierung zu aktualisieren, verwenden Sie DEFAULT CURRENT UTC TIMESTAMP anstatt DEFAULT UTC TIMESTAMP.

Das Verhalten dieses Standardwerts ist dasselbe wie TIMESTAMP und CURRENT TIMESTAMP mit dem Unterschied, dass das Datum und die Uhrzeit in der Coordinated Universal Time (UTC) angezeigt werden.

- **string** Siehe „[Zeichenfolgen](#)“ auf Seite 6.
- **global-variable** Siehe „[Globale Variablen](#)“ auf Seite 88.
- **constant-expression** Konstante Ausdrücke, die keine Datenbankobjekte referenzieren, sind in einer DEFAULT-Klausel zulässig. Daher können Funktionen wie GETDATE oder DATEADD verwendet werden. Wenn der Ausdruck keine Funktion oder kein einfacher Wert ist, muss er in Klammern eingeschlossen werden.
- **sequence-expression** Sie können DEFAULT auf den aktuellen oder nächsten Wert aus einer Sequenz in der Datenbank setzen.

- **AUTOINCREMENT** Wenn AUTOINCREMENT verwendet wird, muss die Spalte einer der Ganzzahl-Datentypen oder ein numerisch exakter Typ sein.

Ist beim Einfügen in die Tabelle kein Wert für die AUTOINCREMENT-Spalte vorgegeben, wird ein eindeutiger Wert erstellt, der größer ist als alle Werte in der Spalte. Wenn INSERT einen Wert für die Spalte angibt, der größer ist als der derzeitige Höchstwert für die Spalte, wird dieser Wert eingefügt und als Startpunkt für nachfolgende Einfügungen verwendet.

Das Löschen von Zeilen setzt den AUTOINCREMENT-Zähler nicht herunter. Lücken durch gelöschte Zeilen können nur durch explizite Zuweisungen wieder gefüllt werden, wenn eine Einfügung verwendet wird. Nach der expliziten Einfügung eines Spaltenwerts unter dem Maximum werden nachfolgende Zeilen ohne explizite Zuordnung weiterhin automatisch mit einem um 1 höheren Wert als das vorherige Maximum erhöht.

Sie können den zuletzt eingefügten Wert der Spalte herausfinden, indem Sie die globale Variable @@identity heranziehen.

AUTOINCREMENT-Werte werden als 64-Bit-Ganzzahlwerte mit Vorzeichen aufrechterhalten, entsprechend dem Datentyp der max_identity-Spalte in der SYSTABCOL-Systemansicht. Wenn der nächste zu generierende Wert den Höchstwert überschreitet, der in der Spalte mit der AUTOINCREMENT-Zuordnung gespeichert werden kann, wird NULL zurückgegeben. Wenn in der Spalte NULL nicht zulässig ist, wie im Fall von Primärschlüsselspalten, wird ein SQL-Fehler generiert.

Der nächste Wert für eine Spalte kann unter Verwendung der Prozedur sa_reset_identity zurückgesetzt werden.

- **GLOBAL AUTOINCREMENT** Dieser Standardwert ist vorgesehen, wenn mehrere Datenbanken in einer MobiLink-Synchronisationsumgebung oder einer SQL Remote-Replikation verwendet werden.

Diese Option ähnelt AUTOINCREMENT, außer dass die Domäne partitioniert ist. Jede Teilmenge enthält dieselbe Anzahl von Werten. Sie ordnen jeder Kopie der Datenbank eine eindeutige Datenbank-Identifizierungsnummer zu. SQL Anywhere liefert Standardwerte in einer Datenbank nur von der Partition, die eindeutig durch diese Datenbanknummer gekennzeichnet ist.

Die Partitionsgröße kann unmittelbar nach dem Schlüsselwort AUTOINCREMENT in Klammern angegeben werden. Die Partitionsgröße kann jede positive Ganzzahl sein, obwohl dieser Wert normalerweise so eingeteilt wird, dass seine Größe kaum jemals überschritten werden kann.

Wenn die Spalte vom Typ BIGINT oder UNSIGNED BIGINT ist, beträgt die Standard-Partitionsgröße $2^{32} = 4294967296$. Bei Spalten aller anderen Typen ist der Standardwert $2^{16} = 65536$. Da diese Standardwerte nicht immer sinnvoll sind, vor allem wenn die Spalte nicht vom Typ INT oder BIGINT ist, empfiehlt es sich, die Partitionsgröße explizit festzulegen.

Wenn Sie diesen Standardwert verwenden, muss der Wert der öffentlichen Option global_database_id in jeder Datenbank auf eine eindeutige, nicht-negative Ganzzahl gesetzt werden. Dieser Wert kennzeichnet die Datenbank eindeutig und zeigt an, von welcher Partition Standardwerte zugeordnet werden sollen. Der Bereich der zulässigen Werte geht von $np + 1$ bis $p(n + 1)$, wobei n der Wert der öffentlichen Option global_database_id und p die Partitionsgröße ist. Wenn Sie z.B. die

Partitionsgröße 1000 festlegen und `global_database_id` auf 3 gesetzt ist, liegt der Bereich zwischen 3001 und 4000.

Wenn der vorherige Wert kleiner ist als $p(n+1)$, wird der nächste Standardwert um eins größer sein als der vorherige größte Wert in der Spalte. Wenn die Spalte keine Werte enthält, ist der erste Standardwert $np+1$. Standardspaltenwerte sind von Werten, die sich außerhalb der aktuellen Partition befinden, nicht betroffen, d.h. von Nummern kleiner als $np+1$ oder größer als $p(n+1)$. Solche Werte können auftreten, wenn sie von einer anderen Datenbank über MobiLink oder SQL Remote repliziert wurden.

Sie können den zuletzt eingefügten Wert der Spalte herausfinden, indem Sie die globale Variable `@@identity` heranziehen.

GLOBAL AUTOINCREMENT-Werte werden als 64-Bit-Ganzzahlwerte mit Vorzeichen aufrechterhalten, entsprechend dem Datentyp der `max_identity`-Spalte in der SYSTABCOL-Systemansicht. Wenn der Wertevorrat innerhalb der Partition aufgebraucht ist, wird NULL zurückgegeben. Wenn in der Spalte NULL nicht zulässig ist, wie im Fall von Primärschlüsselspalten, wird ein SQL-Fehler generiert. In diesem Fall sollte der Datenbank ein neuer `global_database_id`-Wert zugewiesen werden, damit Standardwerte aus einer anderen Partition gewählt werden können. Um festzustellen, ob der Vorrat von ungenutzten Werten zu Ende geht, und um diese Situation zu beheben, erstellen Sie ein Ereignis vom Typ GlobalAutoincrement.

Da die öffentliche Option `global_database_id` nicht auf einen negativen Wert gesetzt werden kann, sind die ausgewählten Werte immer positiv. Die maximale Identifizierungsnummer wird nur durch den Spaltendatentyp und die Partitionsgröße beschränkt.

Wenn die öffentliche Option `global_database_id` auf den Standardwert 2147483647 gesetzt ist, wird NULL in die Spalte eingefügt. Falls NULL nicht zulässig ist, wird beim Versuch, die Zeile einzufügen, ein Fehler erzeugt.

Der nächste Wert für eine Spalte kann unter Verwendung der Prozedur `sa_reset_identity` zurückgesetzt werden.

- **LAST USER** LAST USER ist die Benutzer-ID des Benutzers, der die Zeile zuletzt geändert hat.

LAST USER kann als Standardwert in Spalten mit Zeichendatentypen verwendet werden.

Dieser Standardwert hat bei INSERT dieselbe Wirkung wie CURRENT USER.

Wenn Sie eine Spalte mit einem Standardwert LAST USER bei einer UPDATE-Anweisung nicht ausdrücklich ändern, wird sie auf den Namen des aktuellen Benutzers abgeändert.

In Kombination mit DEFAULT TIMESTAMP oder DEFAULT UTC TIMESTAMP kann der Standardwert LAST USER dazu verwendet werden, sowohl den Benutzer als auch Datum und Uhrzeit der letzten Änderung an einer Zeile zu erfassen (in getrennten Spalten).

column-constraint-Klausel Verwenden Sie diese Klausel, um der Spalte eine Integritätsregel hinzuzufügen.

Hinweis

Wenn eine neue Integritätsregel hinzugefügt wird, validiert der Datenbankserver (außer bei CHECK-Integritätsregeln) bestehende Werte, um zu bestätigen, dass sie der Integritätsregel entsprechen. CHECK-Integritätsregeln werden nur bei Vorgängen erzwungen, die nach Abschluss der Tabellenänderung auftreten.

Mögliche Spalten-Integritätsregeln sind:

- **CHECK-Klausel** Diese Integritätsregel ermöglicht die Überprüfung beliebiger Bedingungen. Zum Beispiel kann mit einer CHECK-Integritätsregel gewährleistet werden, dass eine Spalte mit dem Namen Geschlecht nur die Werte M oder W enthält.

Wenn Sie eine Prüf-Integritätsregel erstellen müssen, die eine Beziehung zwischen zwei oder mehr Spalten in der Tabelle umfasst (beispielsweise muss Spalte A kleiner sein als Spalte B), definieren Sie anstelle dessen eine Tabellenintegritätsregel.

- **UNIQUE-Klausel** Verwenden Sie diese Unterklausel, um anzugeben, dass die Werte in der Spalte eindeutig sein müssen, und ob ein Clustered- oder ein Nonclustered-Index erstellt werden soll.
- **PRIMARY KEY-Klausel** Verwenden Sie diese Unterklausel, um die Spalte zu einem Primärschlüssel zu machen und um anzugeben, ob ein Clustered-Index verwendet werden soll.
- **REFERENCES-Klausel** Verwenden Sie diese Unterklausel, um eine Referenz zu einer anderen Tabelle hinzuzufügen oder zu ändern, und um anzugeben, wie Übereinstimmungen gehandhabt und ob ein Clustered-Index verwendet werden soll.
- **MATCH-Klausel** Mit dieser Unterklausel können Sie steuern, was als Übereinstimmung gilt, wenn ein Mehrspalten-Fremdschlüssel verwendet wird. Sie ermöglicht es Ihnen auch, Eindeutigkeit für den Schlüssel anzugeben, wodurch sich ein separates Deklarieren der Eindeutigkeit erübrigt.
- **NULL- und NOT NULL-Klauseln** Verwenden Sie diese Klausel, um anzugeben, ob NULL in der Spalte zulässig ist. Standardmäßig ist NULL zulässig.

COMPRESSED-Klausel Verwenden Sie diese Klausel, um die Spalte zu komprimieren.

INLINE- und PREFIX-Klauseln Die INLINE-Klausel gibt in Byte die maximale BLOB-Größe an, die in einer Zeile gespeichert werden kann. BLOBs, die kleiner oder gleich dem in der INLINE-Klausel festgelegten Wert sind, werden in der Zeile gespeichert. BLOBs, die den in der INLINE-Klausel definierten Wert überschreiten, werden außerhalb der Zeile in Tabellenerweiterungsseiten gespeichert. Außerdem kann eine Kopie einiger Byte vom Beginn des BLOBs in der Zeile gespeichert werden, wenn ein BLOB größer als der INLINE-Wert ist. Verwenden Sie die PREFIX-Klausel, um anzugeben, wie viele Byte in der Zeile gehalten werden können. Die PREFIX-Klausel kann die Performance von Anforderungen verbessern, die die Präfixbyte eines BLOBs benötigen, um zu ermitteln, ob eine Zeile akzeptiert oder zurückgewiesen wird.

Die Präfixdaten für eine komprimierte Spalte werden nicht komprimiert gespeichert, daher ist keine Dekomprimierung erforderlich, wenn alle Daten, die zur Erfüllung einer Anforderung erforderlich sind, im Präfix gespeichert sind.

Wenn weder `INLINE` noch `PREFIX` angegeben sind oder wenn `USE DEFAULT` angegeben ist, werden Standardwerte wie folgt angewendet:

- Bei Spalten vom Zeichendaten-Typ wie `CHAR`, `NCHAR` und `LONG VARCHAR` ist der Standardwert für `INLINE` 256 und der Standardwert für `PREFIX` ist 8.
- Bei Spalten vom Typ Binärdaten wie `BINARY`, `LONG BINARY`, `VARBINARY`, `BIT`, `VARBIT`, `LONG VARBIT`, `BIT VARYING` und `UUID` ist der Standardwert für `INLINE` 256 und der Standardwert für `PREFIX` 0.

Hinweis

Es wird dringend empfohlen, die Standardwerte zu verwenden, es sei denn, spezielle Umstände erfordern eine andere Einstellung. Die Standardwerte wurden ausgewählt, um die Performance und den Festplattenspeicherbedarf möglichst gut aneinander anzupassen. Wenn Sie z.B. `INLINE` auf einen großen Wert setzen und alle BLOBs in der Zeile gespeichert werden, kann sich die Performance bei der Zeilenverarbeitung verschlechtern. Wenn Sie `PREFIX` zu hoch einstellen, erhöhen Sie den erforderlichen Speicherplatz zum Speichern von BLOBs, da die `PREFIX`-Daten ein Duplikat eines BLOB-Abschnitts sind.

Wenn nur ein Wert angegeben wird, wird der andere Wert automatisch auf die maximale Größe gesetzt, die nicht mit dem angegebenen Wert in Konflikt steht. Weder der `INLINE`- noch der `PREFIX`-Wert dürfen die Seitengröße der Datenbank überschreiten. Auch gibt es in einer Tabellenseite eine geringe Menge an reserviertem Overhead, der nicht zum Speichern von Zeilendaten verwendet werden kann. Daher führt das Angeben eines `INLINE`-Werts, der fast die Seitengröße der Datenbank erreicht, möglicherweise zu einer etwas niedrigeren Anzahl der in der Zeile gespeicherten Bytes.

INDEX- und NO INDEX-Klauseln Wenn Sie BLOBs speichern (nur Zeichen- oder Binärdatentypen), geben Sie `INDEX` an, um BLOB-Indizes für eingefügte Werte zu erstellen, die den internen BLOB-Schwellenwert übersteigen (etwa acht Datenbankseiten). Dies ist die Standardeinstellung.

BLOB-Indizes können die Performance verbessern, wenn Suchvorgänge mit wahlfreiem Zugriff innerhalb der BLOBs erforderlich sind. Allerdings kann bei BLOB-Werten wie Bildern und Multimediadateien, bei denen ein zufälliger Zugriff nicht erforderlich ist, die Performance gesteigert werden, wenn die BLOB-Indizierung deaktiviert ist. Um die BLOB-Indizierung für eine Spalte zu deaktivieren, geben Sie `NO INDEX` ein.

Hinweis

Ein BLOB-Index unterscheidet sich von einem Tabellenindex. Ein Tabellenindex wird erstellt, um Werte in einer oder mehreren Spalten zu indizieren.

IDENTITY-Klausel `IDENTITY` ist eine Transact-SQL-kompatible Alternative zur Verwendung von `DEFAULT AUTOINCREMENT`. In SQL Anywhere wird eine mit `IDENTITY` definierte Spalte als `DEFAULT AUTOINCREMENT` implementiert.

COMPUTE-Klausel Wenn eine Spalte unter Verwendung einer `COMPUTE`-Klausel erstellt wurde, entspricht ihr Wert in jeder Zeile dem Wert des angegebenen Ausdrucks. Mit dieser Integritätsregel erstellte Spalten sind schreibgeschützte Spalten für Anwendungen: Der Wert wird vom Datenbankserver geändert, wenn die Zeile geändert wird. Der `COMPUTE`-Ausdruck darf keinen nicht deterministischen

Wert zurückgeben. Er darf z.B. nicht einen Spezialwert wie CURRENT TIMESTAMP oder eine nicht deterministische Funktion enthalten. Wenn ein COMPUTE-Ausdruck einen nicht deterministischen Wert zurückgibt, kann er nicht verwendet werden, um eine Übereinstimmung in einer Abfrage zu suchen.

Die COMPUTE-Klausel wird für entfernte Tabellen ignoriert.

Eine UPDATE-Anweisung, die versucht, den Wert einer berechneten Spalte zu ändern, löst alle Trigger aus, die mit der Spalte verbunden sind.

ADD table-constraint-Klausel Verwenden Sie diese Klausel, um eine Tabellen-Integritätsregel hinzuzufügen. Tabellen-Integritätsregeln beschränken den Inhalt von Spalten in der Tabelle. Wenn Sie Tabellen-Integritätsregeln hinzufügen oder ändern, ermöglicht Ihnen der optionale Integritätsregelname, einzelne Integritätsregeln zu ändern oder zu löschen. Es folgt eine Liste der möglichen Tabellen-Integritätsregeln.

- **UNIQUE** Verwenden Sie diese Unterklausel, um anzugeben, dass Werte in den unter *column-list* angegebenen Spalten eindeutig sein müssen, und ob ein Clustered-Index verwendet werden soll (optional).
- **PRIMARY KEY** Verwenden Sie diese Unterklausel, um den Primärschlüssel für die Tabelle hinzuzufügen oder zu ändern, und um anzugeben, ob ein Clustered-Index verwendet werden soll. Die Tabelle darf noch keinen Primärschlüssel besitzen, der mit der Anweisung CREATE TABLE oder einer weiteren ALTER TABLE-Anweisung erstellt wurde.
- **foreign-key** Verwenden Sie diese Unterklausel, um einen Fremdschlüssel als Integritätsregel hinzuzufügen. Wenn Sie eine andere Subklausel als ADD FOREIGN KEY mit der ALTER TABLE-Anweisung bei einer Tabelle mit abhängigen materialisierten Ansichten verwenden, schlägt die ALTER TABLE Anweisung fehl. Bei allen anderen Klauseln müssen Sie die abhängigen materialisierten Ansichten deaktivieren und sie anschließend reaktivieren, nachdem Ihre Änderungen abgeschlossen sind.

Bei Verwendung eines Fremdschlüssels mit mehreren Spalten können Sie mithilfe einer MATCH-Unterklausel steuern, was als Übereinstimmung betrachtet wird. Sie ermöglicht es Ihnen auch, Eindeutigkeit für den Schlüssel anzugeben, wodurch sich ein separates Deklarieren der Eindeutigkeit erübrigt.

ADD PCTFREE-Klausel Geben Sie den Prozentsatz des Speicherplatzes an, den Sie in jeder Tabellenseite reservieren möchten. Der freie Speicherplatz wird verwendet, wenn die Zeilen durch die Datenaktualisierung anwachsen. Wenn in einer Tabellenseite kein freier Speicherplatz verfügbar ist, führt jede Vergrößerung einer Zeile dieser Seite zu einer Aufteilung der Zeile auf mehrere Tabellenseiten, was Zeilenfragmentierung und eventuell Performanceverlust nach sich zieht. Ein Prozentsatz für freien Speicherplatz von 0 bedeutet, dass auf den einzelnen Seiten kein freier Platz zur Verfügung stehen darf — jede Seite wird vollgeschrieben. Ein hoher Prozentsatz hat zur Folge, dass jede einzelne Zeile auf eine eigene Seite eingefügt wird. Wenn PCTFREE nicht festgelegt ist oder gelöscht wurde, wird der PCTFREE-Standardwert entsprechend der Datenbank-Seitengröße angewendet (200 Byte für eine Seitengröße von 4 kB und mehr). Der Wert für PCTFREE wird in der Systemtabelle ISYSTAB gespeichert. Wenn PCTFREE festgelegt ist, verwenden alle nachfolgenden Einträge in die Tabellenseiten den neuen Wert, bereits eingefügte Zeilen werden dadurch jedoch nicht beeinflusst. Der Wert bleibt bestehen, bis er geändert wird. Die Angabe von PCTFREE kann für Basistabellen, globale temporäre oder lokale temporäre Tabellen verwendet werden.

Änderungsklauseln Im folgenden Abschnitt werden die Klauseln erklärt, die zum Ändern von Definitionen einer Spalte oder Tabelle verwendet werden:

ALTER column-name column-alteration-Klausel Verwenden Sie diese Klausel, um die Attribute für die angegebene Spalte zu ändern. Wenn eine Spalte in einer Eindeutigkeits-Integritätsregel, einem Fremdschlüssel oder einem Primärschlüssel enthalten ist, können Sie nur den Standardwert für die Spalte ändern. Für alle anderen Änderungen müssen Sie hingegen den Schlüssel oder die Integritätsregel löschen, bevor Sie die Spalte ändern.

column-data-type-Klausel Verwenden Sie diese Klausel, um die Länge oder den Datentyp der Spalte zu ändern. Wenn nötig werden die Daten in der geänderten Spalte in einen neuen Datentyp konvertiert. Wenn ein Konvertierungsfehler auftritt, schlägt der Vorgang fehl und die Tabelle bleibt unverändert. Sie können die Größe einer Spalte nicht vermindern. Beispiel: Sie können eine Spalte nicht von VARCHAR(100) auf VARCHAR(50) ändern.

[NOT] NULL-Klausel Verwenden Sie diese Klausel, um die Einstellung in Bezug auf die Zulässigkeit von NULL in der Spalte zu ändern. Wenn NOT NULL angegeben wird und der Spaltenwert in einer bestehenden Zeile NULL ist, schlägt der Vorgang fehl und die Tabelle bleibt unverändert.

CHECK NULL-Klausel Verwenden Sie diese Klausel, um alle Prüf-Integritätsregeln für die Spalte zu löschen.

DEFAULT-Klausel Verwenden Sie diese Klausel, um den Standardwert für die Spalte zu ändern.

DEFAULT NULL-Klausel Verwenden Sie diese Klausel, um den Standardwert für die Spalte zu entfernen.

[CONSTRAINT constraint-name] CHECK { NULL | (condition) }-Klausel Verwenden Sie diese Klausel, um der Spalte eine Prüf-Integritätsregel hinzuzufügen.

Wenn Sie eine Prüf-Integritätsregel erstellen müssen, die eine Beziehung zwischen zwei oder mehr Spalten in der Tabelle umfasst (beispielsweise muss Spalte A kleiner sein als Spalte B), definieren Sie anstelle dessen eine Tabellenintegritätsregel.

[NOT] COMPRESSED-Klausel Verwenden Sie diese Klausel, um die Einstellung in Bezug auf die Spaltenkomprimierung zu ändern.

INLINE- und PREFIX-Klauseln Die INLINE-Klausel gibt in Byte die maximale BLOB-Größe an, die in einer Zeile gespeichert werden kann. BLOBs, die kleiner oder gleich dem in der INLINE-Klausel festgelegten Wert sind, werden in der Zeile gespeichert. BLOBs, die den in der INLINE-Klausel definierten Wert überschreiten, werden außerhalb der Zeile in Tabellenerweiterungsseiten gespeichert. Außerdem kann eine Kopie einiger Byte vom Beginn des BLOBs in der Zeile gespeichert werden, wenn ein BLOB größer als der INLINE-Wert ist. Verwenden Sie die PREFIX-Klausel, um anzugeben, wie viele Byte in der Zeile gehalten werden können. Die PREFIX-Klausel kann die Performance von Anforderungen verbessern, die die Präfixbyte eines BLOBs benötigen, um zu ermitteln, ob eine Zeile akzeptiert oder zurückgewiesen wird.

Die Präfixdaten für eine komprimierte Spalte werden nicht komprimiert gespeichert, daher ist keine Dekomprimierung erforderlich, wenn alle Daten, die zur Erfüllung einer Anforderung erforderlich sind, im Präfix gespeichert sind.

Wenn weder `INLINE` noch `PREFIX` angegeben sind oder wenn `USE DEFAULT` angegeben ist, werden Standardwerte wie folgt angewendet:

- Bei Spalten vom Zeichendaten-Typ wie `CHAR`, `NCHAR` und `LONG VARCHAR` ist der Standardwert für `INLINE` 256 und der Standardwert für `PREFIX` ist 8.
- Bei Spalten vom Typ Binärdaten wie `BINARY`, `LONG BINARY`, `VARBINARY`, `BIT`, `VARBIT`, `LONG VARBIT`, `BIT VARYING` und `UUID` ist der Standardwert für `INLINE` 256 und der Standardwert für `PREFIX` 0.

Hinweis

Es wird dringend empfohlen, die Standardwerte zu verwenden, es sei denn, spezielle Umstände erfordern eine andere Einstellung. Die Standardwerte wurden ausgewählt, um die Performance und den Festplattenspeicherbedarf möglichst gut aneinander anzupassen. Wenn Sie z.B. `INLINE` auf einen großen Wert setzen und alle BLOBs in der Zeile gespeichert werden, kann sich die Performance bei der Zeilenverarbeitung verschlechtern. Wenn Sie `PREFIX` zu hoch einstellen, erhöhen Sie den erforderlichen Speicherplatz zum Speichern von BLOBs, da die `PREFIX`-Daten ein Duplikat eines BLOB-Abschnitts sind.

Wenn nur ein Wert angegeben wird, wird der andere Wert automatisch auf die maximale Größe gesetzt, die nicht mit dem angegebenen Wert in Konflikt steht. Weder der `INLINE`- noch der `PREFIX`-Wert dürfen die Seitengröße der Datenbank überschreiten. Auch gibt es in einer Tabellenseite eine geringe Menge an reserviertem Overhead, der nicht zum Speichern von Zeilendaten verwendet werden kann. Daher führt das Angeben eines `INLINE`-Werts, der fast die Seitengröße der Datenbank erreicht, möglicherweise zu einer etwas niedrigeren Anzahl der in der Zeile gespeicherten Bytes.

INDEX- und NO INDEX-Klauseln Wenn Sie BLOBs speichern (nur Zeichen- oder Binärdatentypen), geben Sie `INDEX` an, um BLOB-Indizes für eingefügte Werte zu erstellen, die den internen BLOB-Schwellenwert übersteigen (etwa acht Datenbankseiten). Dies ist die Standardeinstellung.

BLOB-Indizes können die Performance verbessern, wenn Suchvorgänge mit wahlfreiem Zugriff innerhalb der BLOBs erforderlich sind. Allerdings kann bei BLOB-Werten wie Bildern und Multimediadateien, bei denen ein zufälliger Zugriff nicht erforderlich ist, die Performance gesteigert werden, wenn die BLOB-Indizierung deaktiviert ist. Um die BLOB-Indizierung für eine Spalte zu deaktivieren, geben Sie `NO INDEX` ein.

Hinweis

Ein BLOB-Index unterscheidet sich von einem Tabellenindex. Ein Tabellenindex wird erstellt, um Werte in einer oder mehreren Spalten zu indizieren.

SET COMPUTE-Klausel Wenn eine Spalte unter Verwendung einer `COMPUTE`-Klausel erstellt wurde, entspricht ihr Wert in jeder Zeile dem Wert des angegebenen Ausdrucks. Mit dieser Integritätsregel erstellte Spalten sind schreibgeschützte Spalten für Anwendungen: Der Wert wird vom Datenbankserver geändert, wenn die Zeile geändert wird. Der `COMPUTE`-Ausdruck darf keinen nicht deterministischen Wert zurückgeben. Er darf z.B. nicht einen Spezialwert wie `CURRENT TIMESTAMP` oder eine nicht deterministische Funktion enthalten. Wenn ein `COMPUTE`-Ausdruck einen nicht deterministischen Wert zurückgibt, kann er nicht verwendet werden, um eine Übereinstimmung in einer Abfrage zu suchen.

Die COMPUTE-Klausel wird für entfernte Tabellen ignoriert.

Eine UPDATE-Anweisung, die versucht, den Wert einer berechneten Spalte zu ändern, löst alle Trigger aus, die mit der Spalte verbunden sind.

ALTER CONSTRAINT *constraint-name* CHECK-Klausel Verwenden Sie diese Klausel, um eine benannte Prüf-Integritätsregel für die Tabelle zu ändern.

Wenn Sie die Integritätsregel ändern möchten, um eine Beziehung zwischen zwei oder mehr Spalten in der Tabelle anzugeben (z.B., dass Spalte A kleiner sein muss als Spalte B), müssen Sie stattdessen eine Tabellenintegritätsregel definieren.

Löschklauseln Im folgenden Abschnitt werden die DROP-Klauseln erklärt:

DROP DEFAULT Löscht den Standardwert, der für die Tabelle oder die angegebene Spalte gesetzt ist. Bestehende Werte ändern sich nicht.

DROP COMPUTE Entfernt das COMPUTE-Attribut für die angegebene Spalte. Diese Anweisung ändert keine bestehenden Werte in der Tabelle.

DROP CHECK Löscht alle Prüf-Integritätsregeln für die Tabelle oder angegebene Spalte. DELETE CHECK wird ebenfalls akzeptiert.

DROP CONSTRAINT *constraint-name* Löscht die benannte Integritätsregel für die Tabelle oder angegebene Spalte. DELETE CONSTRAINT wird ebenfalls akzeptiert.

DROP *column-name* Löscht die angegebene Spalte aus der Tabelle. DELETE *column-name* wird ebenfalls akzeptiert. Wenn die Spalte in einem Index, einer Eindeutigkeits-Integritätsregel, einem Fremdschlüssel oder Primärschlüssel enthalten ist, muss der Index, die Integritätsregel oder der Schlüssel gelöscht werden, bevor die Spalte gelöscht werden kann. Dadurch werden die Prüf-Integritätsregeln, welche diese Spalte referenzieren, nicht gelöscht.

DROP UNIQUE (*column-name ...*) Löscht die Eindeutigkeits-Integritätsregeln für die angegebene(n) Spalte(n). Alle Fremdschlüssel, die sich auf diese Eindeutigkeits-Integritätsregel beziehen, werden auch gelöscht. DELETE UNIQUE (*column-name ...*) wird ebenfalls akzeptiert.

DROP FOREIGN KEY *fkey-name* Löscht den angegebenen Fremdschlüssel. DELETE FOREIGN KEY *fkey-name* wird ebenfalls akzeptiert.

DROP PRIMARY KEY Löscht den Primärschlüssel. Alle Fremdschlüssel, die sich auf den Primärschlüssel für diese Tabelle beziehen, werden auch gelöscht. DELETE PRIMARY KEY wird ebenfalls akzeptiert.

Umbenennungsklauseln Im folgenden Abschnitt werden die Klauseln erklärt, die zum Umbenennen von Teilen einer Spalten- oder Definition verwendet werden:

RENAME *new-table-name* Ändert den Namen einer Tabelle in *new-table-name*. Alle Anwendungen, die den alten Tabellennamen benutzen, müssen ebenfalls geändert werden.

RENAME *column-name* TO *new-column-name* Ändert den Namen der Spalte in *new-column-name*. Alle Anwendungen, die den alten Spaltennamen benutzen, müssen ebenfalls geändert werden.

RENAME CONSTRAINT *constraint-name* TO *new-constraint-name* Ändert den Namen der Integritätsregel in *new-constraint-name*.

ALTER TABLE...RENAME CONSTRAINT *constraint-name* TO *new-constraint-name* benennt bei Anwendung auf eine RI-Integritätsregel nur die Integritätsregel um. Der darunterliegende Index oder, gegebenenfalls, der Fremdschlüssel-Rollename bleibt davon unberührt. Wenn Sie den zugrunde liegenden Index umbenennen oder den Rollennamen ändern möchten, verwenden Sie die ALTER INDEX-Anweisung.

table-alteration-Klauseln Verwenden Sie diese Klausel, um die nachstehenden Tabellenattribute zu ändern.

PCTFREE DEFAULT Verwenden Sie diese Klausel, um die PCTFREE-Einstellung für die Tabelle auf den Standardwert zu setzen (200 Byte für eine 4-kB-Seitengröße und höher).

[NOT] ENCRYPTED Verwenden Sie diese Klausel, um die Einstellung in Bezug auf die Tabellenverschlüsselung zu ändern. Um eine Tabelle zu verschlüsseln, muss die Tabellenverschlüsselung in der Datenbank aktiviert sein. Die Verschlüsselung erfolgt unter Verwendung des Chiffrierschlüssels und des Algorithmus, die bei der Datenbankerstellung festgelegt wurden.

Nach dem Verschlüsseln einer Tabelle bleiben Daten für diese Tabelle, die sich vor dem Verschlüsseln in temporären Dateien oder im Transaktionslog befanden, weiterhin unverschlüsselt zugänglich. Um dies zu verhindern, starten Sie die Datenbank neu, damit die temporären Dateien entfernt werden. Führen Sie das Sicherungsdienstprogramm (dbbackup) mit der Option -o aus oder verwenden Sie die BACKUP-Anweisung, um das Transaktionslog zu sichern und ein neues zu starten.

Wenn die Tabellenverschlüsselung aktiviert ist, werden Tabellenseiten für die verschlüsselte Tabelle, zugeordnete Indexseiten, Seiten der temporären Tabelle und Transaktionslogseiten, die Transaktionen von verschlüsselten Tabellen enthalten, verschlüsselt.

DISABLE VIEW DEPENDENCIES-Klausel Verwenden Sie diese Klausel, um abhängige reguläre Ansichten zu deaktivieren. Abhängige materialisierte Ansichten werden nicht deaktiviert. Sie müssen jede abhängige materialisierte Ansicht deaktivieren, in dem Sie die Anweisung ALTER MATERIALIZED VIEW...DISABLE ausführen.

ALTER OWNER-Klausel Eigentümer von Tabellen ändern

- Sie müssen das ALTER ANY OBJECT OWNER-Systemprivileg haben.
- Sie müssen eines der folgenden Privilegien haben: ALTER-Privileg für die Tabelle, ALTER ANY TABLE-Privileg oder ALTER ANY OBJECT-Privileg.
- Der neue Eigentümer darf nicht bereits Eigentümer einer Tabelle mit demselben Namen sein.
- Aktivierte materialisierte Ansichten können nicht die Tabelle referenzieren.

PRESERVE oder DROP PRIVILEGES Wenn Sie nicht möchten, dass der neue Eigentümer dieselben Privilegien hat wie der alte Eigentümer, können Sie DROP PRIVILEGES angeben, um alle explizit erteilten Privilegien zu löschen, die einem Benutzer Zugriff auf die Tabelle gewähren. Implizit erteilte Privilegien für den Eigentümer der Tabelle werden dem neuen Eigentümer erteilt und aus dem alten Eigentümer gelöscht.

PRESERVE oder DROP FOREIGN KEYS Um zu verhindern, dass der neue Eigentümer auf Daten in referenzierten Tabellen zugreift, können Sie DROP FOREIGN KEYS angeben, um alle Fremdschlüssel in der Tabelle zu löschen sowie alle Fremdschlüssel, die die Tabelle referenzieren.

Bemerkungen

Die ALTER TABLE-Anweisung ändert Tabellenattribute (Spaltendefinitionen, Integritätsregeln usw.) in einer bestehenden Tabelle.

Der Datenbankserver protokolliert Objektabhängigkeiten in der Datenbank. Änderungen am Schema einer Tabelle können sich auf abhängige Ansichten auswirken. Wenn materialisierte Ansichten vorhanden sind, die von der zu ändernden Tabelle abhängen, müssen Sie diese zuerst deaktivieren. Hierfür verwenden Sie die ALTER MATERIALIZED VIEW...DISABLE-Anweisung.

ALTER TABLE kann nicht für eine lokale, temporäre Tabelle verwendet werden.

ALTER TABLE wird verhindert, wenn die Anweisung eine Tabelle betrifft, die zeitgleich von einer anderen Verbindung benutzt wird. ALTER TABLE kann zeitaufwändig sein. Der Datenbankserver verarbeitet während der Anweisungsverarbeitung keine Anforderungen, die sich auf die Tabelle beziehen.

Wenn Sie eine Spalte ändern, für die ein als IMMEDIATE REFRESH festgelegter Textindex besteht, wird der Textindex sofort neu aufgebaut. Wenn der Textindex als AUTO REFRESH oder MANUAL REFRESH definiert ist, wird der Textindex bei der nächsten Aktualisierung neu aufgebaut.

Wenn Sie eine ALTER TABLE-Anweisung ausführen, versucht der Datenbankserver, Spaltenprivilegien für abhängige Ansichten wiederherzustellen, die automatisch neu kompiliert werden. Privilegien für Spalten, die in den neu kompilierten Ansichten nicht mehr vorhanden sind, gehen verloren.

ALTER TABLE erfordert exklusiven Zugriff auf die Tabelle.

Globale temporäre Tabellen können erst geändert werden, wenn alle Benutzer, welche die Tabelle referenzieren, ihre Verbindungen getrennt haben.

Diese Anweisung kann nicht innerhalb einer Snapshot-Transaktion verwendet werden.

Privilegien

Sie müssen Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Tabelle
- ALTER ANY TABLE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg
- MANAGE ANY DBSPACE-Systemprivileg

Wenn Sie den Tabelleneigentümer ändern möchten, müssen Sie außerdem das ALTER ANY OBJECT OWNER-Systemprivileg haben.

Wenn Sie Indizes für eine Tabelle erstellen, ändern oder löschen möchten, benötigen Sie das CREATE ANY INDEX-Systemprivileg, das ALTER ANY INDEX-Systemprivileg bzw. das DROP ANY INDEX-Systemprivileg.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Ein Checkpoint wird zu Beginn des ALTER TABLE-Vorgangs ausgeführt, und weitere Checkpoints werden vorübergehend unterbrochen, bis der ALTER TABLE-Vorgang abgeschlossen ist.

Sobald Sie eine Spalte oder Tabelle ändern, wird möglicherweise die Ausführung aller gespeicherten Prozeduren, Ansichten oder sonstigen Elemente, welche die geänderte Spalte referenzieren, beendet.

Wenn Sie die deklarierte Länge oder den Typ einer Spalte ändern oder eine Spalte löschen, wird die Statistik für diese Spalte gelöscht.

Siehe auch

- „Ereignisse“ [*SQL Anywhere Server - Datenbankadministration*]
- „sa_reset_identity-Systemprozedur“ auf Seite 1299
- Wählen zwischen Sequenzen und AUTOINCREMENT-Werten [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- Neuladen von Tabellen mit AUTOINCREMENT-Spalten [*SQL Anywhere 16 - Änderungen und Upgrades*]
- „ALTER MATERIALIZED VIEW-Anweisung“ auf Seite 476
- „BACKUP-Anweisung“ auf Seite 548
- „ALTER INDEX-Anweisung“ auf Seite 466
- „CREATE TABLE-Anweisung“ auf Seite 737
- „DROP TABLE-Anweisung“ auf Seite 832
- MATCH-Klausel, CREATE TABLE-Anweisung auf Seite 747
- „Sicherungsdienstprogramm (dbbackup)“ [*SQL Anywhere Server - Datenbankadministration*]
- „SQL-Datentypen“ auf Seite 95
- „Tabellenänderung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Die spezielle IDENTITY-Spalte“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Spezialwerte“ auf Seite 70
- „Tabellen- und Spalten-Integritätsregeln“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Verwendung einer Sequenz zum Generieren von eindeutigen Werten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „allow_nulls_by_default-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Tabellen verschlüsseln“ [*SQL Anywhere Server - Datenbankadministration*]
- „Clustered-Indizes“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Ansichtenabhängigkeiten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Tipp: Spaltenstatistiken aktualisieren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** ALTER TABLE ist eine Kernfunktion. Im SQL/2008-Standard werden ADD COLUMN und DROP COLUMN als Kernfunktionen unterstützt, wie auch ADD CONSTRAINT und DROP CONSTRAINT. ALTER [COLUMN] ist SQL-Funktion F381, wie auch die Möglichkeit, einen DEFAULT-Wert für eine Spalte hinzuzufügen, zu ändern oder zu löschen. In SQL/2008 erfolgt das Ändern des Datentyps einer Spalte mithilfe der SET DATA TYPE-Klausel, bei der es sich um SQL-Sprachenfunktion F382 handelt. Im Gegensatz dazu wird in SQL Anywhere das direkte Ändern des Datentyps einer Spalte durch die ALTER-Klausel unterstützt.

Andere von SQL Anywhere unterstützte Klauseln, einschließlich ALTER CONSTRAINT, RENAME, PCTFREE, ENCRYPTED und DISABLE MATERIALIZED VIEW, sind Erweiterungen des Herstellers. Die Unterstützung für Erweiterungen von Spaltendefinitionen sowie Integritätsregeldefinitionen für Spalten und Tabellen sind Herstellererweiterungen des SQL/2008-Standards oder spezifische optionale Funktionen von SQL/2008.

- **Transact-SQL** ALTER TABLE wird von Adaptive Server Enterprise unterstützt. Adaptive Server Enterprise unterstützt die Klauseln ADD COLUMN und DROP COLUMN zusätzlich zu ADD CONSTRAINT und DROP CONSTRAINT. Adaptive Server Enterprise verwendet für die ALTER-Klausel MODIFY und nicht das Schlüsselwort ALTER. Adaptive Server Enterprise verwendet die REPLACE-Klausel zum Ändern des DEFAULT Werts einer Spalte. In Adaptive Server Enterprise wird ALTER TABLE auch zum Aktivieren bzw. Deaktivieren von Triggern für eine bestimmte Tabelle verwendet. Diese Funktion wird in SQL Anywhere nicht unterstützt.

Beispiel

Im folgenden Beispiel wird eine neue Zeitstempelspalte namens TimeStamp der Tabelle Customers hinzugefügt. Um die nachfolgende Anweisung ausführen zu können, müssen Sie das ALTER-Privileg für die Tabelle "Customers" haben.

```
ALTER TABLE GROUPO.Customers
  ADD TimeStamp AS TIMESTAMP DEFAULT TIMESTAMP;
```

Im folgenden Beispiel wird die neue Zeitstempelspalte TimeStamp gelöscht, die Sie im vorherigen Beispiel erstellt haben.

```
ALTER TABLE GROUPO.Customers
  DROP TimeStamp;
```

Die Street-Spalte in der Tabelle Customers kann gegenwärtig bis zu 35 Zeichen enthalten. Damit sie bis zu 50 Zeichen umfassen kann, führen Sie Folgendes aus:

```
ALTER TABLE GROUPO.Customers
  ALTER Street CHAR(50);
```

Mit dem folgenden Beispiel wird eine Spalte zur Customers-Tabelle eingefügt, wobei jedem Kunden ein "sales contact" (Ansprechpartner) zugewiesen wird. Um die nachfolgende Anweisung ausführen zu können, müssen Sie außerdem das CREATE ANY INDEX-Systemprivileg haben, weil ein Fremdschlüssel erstellt wird.

```
ALTER TABLE GROUPO.Customers
  ADD SalesContact INTEGER
  REFERENCES GROUPO.Employees ( EmployeeID )
  ON UPDATE CASCADE
  ON DELETE SET NULL;
```

Dieser Fremdschlüssel wurde mit kaskadierenden Aktualisierungen konstruiert und wird bei Löschungen auf NULL gesetzt. Wenn sich die Mitarbeiterkennung eines Angestellten ändert, wird die Spalte entsprechend aktualisiert. Wenn ein Angestellter das Unternehmen verlässt und seine Kennung gelöscht wird, wird die Spalte auf den Wert NULL gesetzt.

Im folgenden Beispiel wird ein Fremdschlüssel FK_SalesRepresentative_EmployeeID2, für die Spalte SalesOrders.SalesRepresentative erstellt und mit Employees.EmployeeID verknüpft. Sie benötigen das ALTER-Privileg für die Tabelle "SalesOrders", um die folgende Anweisung ausführen zu können:

```
ALTER TABLE GROUPO.SalesOrders
  ADD CONSTRAINT FK_SalesRepresentative_EmployeeID2
  FOREIGN KEY ( SalesRepresentative )
  REFERENCES GROUPO.Employees (EmployeeID);
```

Im folgenden Beispiel wird eine Spalte hinzugefügt, in der der Standardwert AUTOINCREMENT ist. In diesem Beispiel werden alle bestehenden Kundenzeilen geändert, um eine nullwertfähige AUTOINCREMENT-Spalte mit zugeordnetem Spaltenwert zu erhalten, aber der Datenbankserver garantiert nicht, welcher Zeile welcher Wert zugeordnet ist:

```
ALTER TABLE GROUPO.Customers
  ADD Surrogate_key INTEGER DEFAULT AUTOINCREMENT;
```

Im folgenden Beispiel wird der Eigentümer der fiktiven Tabelle "mytable" in Bob geändert:

```
ALTER TABLE mytable ALTER OWNER TO bob;
```

ALTER TEXT CONFIGURATION-Anweisung

Ändert ein Textkonfigurationsobjekt

Syntax

```
ALTER TEXT CONFIGURATION [ owner.]config-name
STOPLIST stoplist-string
| DROP STOPLIST
| { MINIMUM | MAXIMUM } TERM LENGTH integer }
| TERM BREAKER { GENERIC [ EXTERNAL NAME external-call ] | NGRAM }
| PREFILTER EXTERNAL NAME external-call
| DROP PREFILTER
| SAVE OPTION VALUES [ FROM CONNECTION ]
```

external-call : '[operating-system:]library-function-name@library-name[;...]'

operating-system : **UNIX**

Parameter

STOPLIST-Klausel Verwenden Sie diese Klausel, um die Liste der zu ignorierenden Begriffe beim Aufbau des Textindexes zu erstellen oder zu ersetzen. Bei Verwendung dieses Textkonfigurationsobjekts werden in dieser Liste angegebene Begriffe auch in einer Abfrage ignoriert. Trennen Sie die Begriffe in der Stoppliste durch Leerstellen. Zum Beispiel: STOPLIST 'because about therefore only'. Begriffe in der Stoppliste können keine Leerstellen enthalten.

Beispiele von Stopplisten für verschiedene Sprachen finden Sie in *%SQLANYSAMPLE%\SQLAnywhere\SQL*.

Begriffe in einer Stoppliste dürfen keine nicht-alphanumerischen Zeichen enthalten. Die Stopplistenlänge muss kleiner als 8000 Byte sein.

Überlegen Sie gut, ob Sie Begriffe in Ihre Stoppliste aufnehmen wollen.

DROP STOPLIST-Klausel Mit dieser Klausel löschen Sie die Stoppliste für ein Textkonfigurationsobjekt.

MINIMUM TERM LENGTH-Klausel Der in der Klausel MINIMUM TERM LENGTH angegebene Wert wird ignoriert, wenn NGRAM-Textindizes verwendet werden.

Die in Zeichen angegebene Mindestlänge eines Begriffs, der in den Textindex aufzunehmen ist. Begriffe, die kürzer als diese Einstellung sind, werden beim Aufbau oder bei der Aktualisierung des Textindexes ignoriert. Der Wert dieser Option muss größer als 0 sein. Wenn Sie diese Option höher ansetzen als MAXIMUM TERM LENGTH, wird der Wert von MAXIMUM TERM LENGTH automatisch so berichtigt, dass er dem neuen Wert für MINIMUM TERM LENGTH entspricht.

MAXIMUM TERM LENGTH-Klausel Verwenden Sie bei NGRAM-Textindizes die MAXIMUM TERM LENGTH-Klausel zum Festlegen der Größe der N-Gramme, in die Zeichenfolgen aufgeteilt werden.

Bei GENERIC-Textindizes können Sie mit MAXIMUM TERM LENGTH die in Zeichen angegebene Maximallänge eines Begriffs festlegen, der in den Textindex aufzunehmen ist. Begriffe, die länger als diese Einstellung sind, werden beim Aufbau oder bei der Aktualisierung des Textindexes ignoriert. Der Wert von MAXIMUM TERM LENGTH muss kleiner oder gleich 60 sein. Wenn Sie diese Option niedriger ansetzen als MINIMUM TERM LENGTH, wird der Wert von MINIMUM TERM LENGTH automatisch so berichtigt, dass er dem neuen Wert für MAXIMUM TERM LENGTH entspricht.

TERM BREAKER-Klausel Der Name des Algorithmus, der für das Trennen von Spaltenwerten in Begriffe verwendet werden soll. Zur Auswahl stehen GENERIC (Standardwert) oder NGRAM.

- **GENERIC** Bei GENERIC können Sie TERM BREAKER GENERIC angeben, um den integrierten Begriffsegmentierer-Algorithmus GENERIC zu verwenden, oder mithilfe der TERM BREAKER GENERIC EXTERNAL NAME-Klausel einen externen Algorithmus angeben.

Der integrierte Algorithmus GENERIC verarbeitet Zeichenfolgen mit einem oder mehr alphanumerischen Zeichen, die durch nicht-alphanumerische Zeichen getrennt sind, als Begriff.

Geben Sie die TERM BREAKER GENERIC EXTERNAL NAME-Klausel an, um einen Eintrittspunkt für eine Begriffsegmentierer-Funktion in einer externen Bibliothek festzulegen. Dies ist nützlich, wenn Sie spezielle Anforderungen hinsichtlich der Frage haben, wie Begriffe vor dem Indizieren oder Abfragen aufgeteilt werden sollen (zum Beispiel, wenn ein Apostroph als Teil eines Begriffs und nicht als Begriffsegmentierer gelten soll).

external-call kann mehr als eine Funktion und/oder Bibliothek angeben sowie die Dateierweiterung der Bibliothek enthalten, normalerweise *.dll* unter Windows und *.so* unter Unix. Ohne Angabe der Dateierweiterung verwendet der Datenbankserver standardmäßig die plattformspezifische Dateierweiterung für Bibliotheken. Zum Beispiel ruft EXTERNAL NAME

'TermBreakFunct1@myTBlib;Unix:TermBreakFunct2@myTBlib' unter Windows die Funktion TermBreakFunct1 aus *myTBlib.dll* auf und unter Unix die Funktion TermBreakFunct2 aus *myTBlib.so*.

- **NGRAM** Der integrierte NGRAM-Algorithmus teilt Zeichenfolgen in N-Gramme auf. Ein N-Gramm ist eine *n*- Teilzeichenfolge einer größeren Zeichenfolge. Der NGRAM-Begriffsegmentierer ist für angenäherte Übereinstimmungen ("fuzzy matching") oder für Dokumente erforderlich, in denen keine Leerstellen oder nicht-alphanumerischen Zeichen für die Trennung von Begriffen verwendet werden, sofern kein externer Begriffsegmentierer angegeben ist.

PREFILTER EXTERNAL NAME-Klausel Geben Sie die PREFILTER EXTERNAL NAME-Klausel an, um einen Eintrittspunkt für eine Vorfilterfunktion in einer externen Bibliothek festzulegen. Dies ist hilfreich, wenn Text aus binären Daten (z.B. PDF) extrahiert werden muss. Darüber hinaus ist es hilfreich, wenn der zu indizierende Text Formatierungsinformationen und/oder Bilder enthält, die Sie vor der Indizierung der Daten entfernen möchten (z.B. HTML).

external-call kann mehr als eine Funktion und/oder Bibliothek angeben sowie die Dateierweiterung der Bibliothek enthalten, normalerweise *.dll* unter Windows und *.so* unter Unix. Ohne Angabe der Dateierweiterung verwendet der Datenbankserver standardmäßig die plattformspezifische Dateierweiterung für Bibliotheken. Zum Beispiel ruft `PREFILTER EXTERNAL NAME 'PrefilterFunct1@myPreFilterlib;Unix:PrefilterFunct2@myPreFilterlib'` unter Windows die Funktion `PrefilterFunct1` aus *myPreFilterlib.dll* auf und unter Unix die Funktion `PrefilterFunct2` aus *myPreFilterlib.so*.

DROP PREFILTER-Klausel Verwenden Sie die DROP PREFILTER-Klausel, um die Verwendung der angegebenen Vorfilter-Bibliothek für das Textkonfigurationsobjekt auszuschließen. Damit werden die Vorfilter nicht mehr ausgeführt, wenn der Datenbankserver Indizes aufbaut, für die dieses Textkonfigurationsobjekt verwendet wird.

SAVE OPTION VALUES-Klausel Wenn ein Textkonfigurationsobjekt erstellt wird, spiegeln die aktuellen Datenbankoptionen `date_format`, `time_format`, `timestamp_format` und `timestamp_with_time_zone_format` wider, wie die Spalten `DATE`, `TIME` und `TIMESTAMP` mit dem Textkonfigurationsobjekt gespeichert werden. Verwenden Sie die SAVE OPTION VALUES-Klausel zum Aktualisieren der für das Textkonfigurationsobjekt gespeicherten Optionswerte, damit diese die für die Verbindung aktiven Optionen widerspiegeln.

Bemerkungen

Bevor Sie die Einstellungen für die Begriffslänge ändern, lesen Sie die Hinweise zu den Auswirkungen der jeweiligen Einstellungen auf den Indizierungsumfang und die Interpretation der Suchbegriffe.

Textindizes sind von einem Textkonfigurationsobjekt abhängig. Bevor Sie diese Anweisung verwenden, müssen Sie abhängige `AUTO`- oder `MANUAL REFRESH`-Textindizes kürzen und `IMMEDIATE REFRESH`-Textindizes löschen.

Um die Einstellungen für Textkonfigurationsobjekte anzuzeigen, führen Sie eine Abfrage in der `SYSTEXTCONFIG`-Systemansicht durch.

Privilegien

Sie müssen Eigentümer des Textkonfigurationsobjekts sein oder eines der folgenden Privilegien haben:

- `ALTER`-Privileg für das Textkonfigurationsobjekt
- `ALTER ANY TEXT CONFIGURATION`-Systemprivileg
- `ALTER ANY OBJECT`-Systemprivileg

Wenn Sie einen externen Begriffsegmentierer ändern möchten, müssen Sie außerdem das `CREATE EXTERNAL REFERENCE`-Systemprivileg haben.

Wenn Sie einen Vorfilter angeben oder löschen bzw. einen externen Begriffsegmentierer angeben, benötigen Sie *darüber hinaus* das ALTER ANY TEXT-Systemprivileg CONFIGURATION oder das ALTER ANY OBJECT-Systemprivileg.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Ändern eines Textkonfigurationsobjekts“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Begriffe und Einstellungen für Textindizes anzeigen (Sybase Central)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Praktische Einführung: Volltextsuche in einem GENERIC-Textindex durchführen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Praktische Einführung: Eine Fuzzy-Volltextsuche durchführen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „CREATE TEXT CONFIGURATION-Anweisung“ auf Seite 757
- „DROP TEXT CONFIGURATION-Anweisung“ auf Seite 833
- „sa_char_terms-Systemprozedur“ auf Seite 1171
- „sa_nchar_terms-Systemprozedur“ auf Seite 1277
- „sa_refresh_text_indexes-Systemprozedur“ auf Seite 1295
- „sa_text_index_stats-Systemprozedur“ auf Seite 1342
- „SYSTEXTCONFIG-Systemansicht“ auf Seite 1499

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgenden Anweisungen erstellen das Textkonfigurationsobjekt "myTextConfig" und setzen anschließend die Begriff-Höchstlänge auf 16. Um diese Anweisung ausführen zu können, benötigen Sie das CREATE TEXT CONFIGURATION-Systemprivileg:

```
CREATE TEXT CONFIGURATION myTextConfig FROM default_char;
ALTER TEXT CONFIGURATION maxTerm16
    MAXIMUM TERM LENGTH 16;
```

Die folgende Anweisung fügt dem Konfigurationsobjekt "myTextConfig" eine Stoppliste hinzu:

```
ALTER TEXT CONFIGURATION myTextConfig
    STOPLIST 'because about therefore only';
```

Mit der folgenden Anweisung wird ein externer Begriffsegmentierer für das Textkonfigurationsobjekt myTextConfig konfiguriert. Die Schnittstelle wird sowohl für Windows als auch für Unix angegeben.

```
ALTER TEXT CONFIGURATION myTextConfig
    TERM BREAKER GENERIC
    EXTERNAL NAME
    'my_termbreaker@termbreaker.dll;Unix:my_termbreaker@libtermbreaker_r.so'
```

Im folgenden Beispiel wird ein externer Vorfilter für das Textkonfigurationsobjekt myTextConfig konfiguriert. Die Schnittstelle wird sowohl für Windows als auch für Unix angegeben.

```
ALTER TEXT CONFIGURATION myTextConfig
    PREFILTER EXTERNAL NAME
    'html_xml_filter@html_xml_filter.dll;UNIX:html_xml_filter@libhtml_xml_filter_
    r.so';
```

Im folgenden Beispiel wird der externe Vorfilter für das Textkonfigurationsobjekt myTextConfig gelöscht.

```
ALTER TEXT CONFIGURATION myTextConfig DROP PREFILTER;
```

ALTER TEXT INDEX-Anweisung

Ändert die Definition eines Textindexes.

Syntax

```
ALTER TEXT INDEX [ owner.]text-index-name
ON [ owner.]table-name
alter-clause

alter-clause :
rename-object
| refresh-alteration

rename-object :
RENAME { AS | TO } new-name

refresh-alteration :
{ MANUAL REFRESH
| AUTO REFRESH [ EVERY integer { MINUTES | HOURS } ] }
```

Parameter

RENAME-Klausel Verwenden Sie die RENAME-Klausel, um den Textindex umzubenennen.

REFRESH-Klausel Geben Sie die REFRESH-Klausel an, um den Aktualisierungstyp für den Textindex einzustellen.

Bemerkungen

Nachdem ein Textindex erstellt wurde, können Sie ihn nicht mehr von oder auf IMMEDIATE REFRESH ändern. Wenn eine dieser Änderungen erforderlich ist, müssen Sie den Textindex löschen und wiederherstellen.

Diese Anweisung kann nicht ausgeführt werden, wenn mit der WITH HOLD-Klausel geöffnete Cursor vorhanden sind, die entweder Anweisungs- oder Transaktions-Snapshots verwenden.

Sie können einen auf einer materialisierten Ansicht aufgebauten Textindex nur ändern, indem sie ihn umbenennen. Sie können nicht den Aktualisierungstyp für einen auf einer materialisierten Ansicht aufgebauten Textindex ändern.

Privilegien

Wenn Sie einen Textindex für eine Tabelle ändern möchten, müssen Sie der Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- REFERENCES-Privileg für die Tabelle
- ALTER ANY INDEX-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Wenn Sie einen Textindex für eine materialisierte Ansicht ändern möchten, müssen Sie der Eigentümer der materialisierten Ansicht sein oder eines der folgenden Privilegien haben:

- ALTER ANY INDEX-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE TEXT INDEX-Anweisung“ auf Seite 759
- „ALTER TEXT INDEX-Anweisung“ auf Seite 536
- „DROP TEXT INDEX-Anweisung“ auf Seite 834
- „REFRESH TEXT INDEX-Anweisung“ auf Seite 990
- „TRUNCATE TEXT INDEX-Anweisung“ auf Seite 1091
- „COMMENT-Anweisung“ auf Seite 573
- „Snapshot-Isolation“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Begriffe und Einstellungen für Textindizes anzeigen (Sybase Central)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Volltextsuche“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Praktische Einführung: Volltextsuche in einem GENERIC-Textindex durchführen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Praktische Einführung: Eine Fuzzy-Volltextsuche durchführen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die erste Anweisung erstellt einen Textindex namens txt_index_manual und definiert ihn als MANUAL REFRESH. Die zweite Anweisung ändert den Textindex, um ihn automatisch an jedem Tag zu aktualisieren. Die dritte Anweisung benennt den Textindex auf txt_index_daily um.

```
CREATE TEXT INDEX txt_index_manual ON GROUPO.MarketingInformation
( Description )
  MANUAL REFRESH;
ALTER TEXT INDEX txt_index_manual ON GROUPO.MarketingInformation
  AUTO REFRESH EVERY 24 HOURS;
ALTER TEXT INDEX txt_index_manual ON GROUPO.MarketingInformation
  RENAME AS txt_index_daily;
```

ALTER TRACE EVENT SESSION-Anweisung

Ermöglicht das Hinzufügen oder Entfernen von Trace-Ereignissen in einer Sitzung, das Hinzufügen oder Entfernen von Zielen in einer Sitzung sowie das Starten und Stoppen einer Trace-Sitzung.

Syntax

```
ALTER TRACE EVENT SESSION session-name  
{ add-drop-trace-event [...]  
  | add-drop-target [, ...] ]  
  | STATE = { START | STOP }  
}
```

add-drop-trace-event :
{ **ADD TRACE EVENT** *trace-event-name* ... | **DROP TRACE EVENT** *trace-event-name* }

add-drop-target :
{ **ADD TARGET** *target-name* ... | **DROP TARGET** *target-name* }

target-name : *filename*

Parameter

ADD TRACE EVENT Der Name des Trace-Ereignisses, das zur Sitzung hinzugefügt wird.

DROP TRACE EVENT Der Name des Trace-Ereignisses, das aus der Sitzung entfernt wird.

ADD TARGET Der Name des Ziels, das zur Sitzung hinzugefügt wird. Informationen zu den Trace-Ereignissen, die Teil der Sitzung sind, werden in diesem Ziel protokolliert.

DROP TARGET Der Name des Ziels, das aus der Sitzung entfernt wird.

Bemerkungen

Das Hinzufügen bzw. Löschen von Trace-Ereignissen oder Zielen in einer laufenden Trace-Sitzung führt dazu, dass die Sitzung kurz unterbrochen wird, um die Änderungen vorzunehmen, und anschließend wieder aufgenommen wird. Sie müssen eine Protokollierungssitzung nicht stoppen, um Trace-Ereignisse oder Ziele hinzuzufügen oder zu löschen. Das Hinzufügen bzw. Löschen eines Trace-Ereignisses aus einer Sitzung, die bereits gestartet wurde, hat die Nebenwirkung, dass einige Trace-Ereignisse verpasst werden, während die Sitzung vorübergehend unterbrochen wird.

Systemprivilegien

Sie müssen das **MANAGE ANY TRACE SESSION**-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Ereignisprotokollierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE TEMPORARY TRACE EVENT-Anweisung“ auf Seite 753
- „CREATE TEMPORARY TRACE EVENT SESSION-Anweisung“ auf Seite 755
- „DROP TRACE EVENT-Anweisung“ auf Seite 835
- „DROP TRACE EVENT SESSION-Anweisung“ auf Seite 836
- „NOTIFY TRACE EVENT-Anweisung“ auf Seite 963
- „sp_trace_events-Systemprozedur“ auf Seite 1407
- „sp_trace_event_fields-Systemprozedur“ auf Seite 1401
- „sp_trace_event_sessions-Systemprozedur“ auf Seite 1406
- „sp_trace_event_session_events-Systemprozedur“ auf Seite 1402
- „sp_trace_event_session_targets-Systemprozedur“ auf Seite 1405
- „sp_trace_event_session_target_options-Systemprozedur“ auf Seite 1403
- „Dienstprogramm für die Verwaltung von ETD-Dateien (dbmanageetd)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel werden ein Trace-Ereignis namens my_event und eine Trace-Ereignissitzung namens my_session erstellt und die Sitzung wird gestartet:

```
CREATE TEMPORARY TRACE EVENT my_event( id INTEGER, information LONG
VARCHAR );
CREATE TEMPORARY TRACE EVENT SESSION my_session
  ADD TRACE EVENT my_event, -- user event
  ADD TRACE EVENT SYS_ConsoleLog_Information -- system event
  ADD TARGET FILE ( SET filename_prefix='my_trace_file' ); -- add a target
ALTER TRACE EVENT SESSION my_session
  STATE = START;
```

ALTER TRIGGER-Anweisung

Ersetzt eine Triggerdefinition durch eine geänderte Version. Geben Sie die vollständige neue Triggerdefinition in der ALTER TRIGGER-Anweisung an.

Syntax 1 - Ändert die Definition eines Triggers

```
ALTER TRIGGER trigger-name trigger-definition
```

trigger-definition : CREATE TRIGGER-Syntax

Syntax 2 - Verschleiert eine Triggerdefinition

```
ALTER TRIGGER trigger-name ON [owner.] table-name SET HIDDEN
```

Bemerkungen

- **Syntax 1** Die ALTER TRIGGER-Anweisung unterscheidet sich in der Syntax nur durch das erste Wort von der CREATE TRIGGER-Anweisung.

Sie können sowohl die Transact-SQL- als auch die Watcom-SQL-Form der CREATE TRIGGER-Syntax verwenden.

- **Syntax 2** Sie können SET HIDDEN verwenden, um die Definition des zugehörigen Triggers zu verschleiern und somit unlesbar zu machen. Der Trigger kann aus Ihrer Datenbank entladen und in andere Datenbanken geladen werden. Wenn SET HIDDEN verwendet wird, ist die Definition des Triggers bei der Fehlerbehandlung mit dem Debugger nicht zu sehen und sie wird auch nicht in den Prozedurprofilen angezeigt.

Hinweis

Der SET HIDDEN-Vorgang ist irreversibel.

Privilegien

Sie müssen Eigentümer der Basistabelle sein oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Basistabelle mit CREATE ANY OBJECT-Systemprivileg
- ALTER ANY TRIGGER-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Wenn Sie einen Trigger in einer Ansicht eines anderen Eigentümers ändern möchten, benötigen Sie entweder die Systemprivilegien ALTER ANY TRIGGER und ALTER ANY VIEW oder das ALTER ANY OBJECT-Systemprivileg.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE TRIGGER-Anweisung“ auf Seite 762
- „CREATE TRIGGER-Anweisung [T-SQL]“ auf Seite 769
- „DROP TRIGGER-Anweisung“ auf Seite 837
- „Verbergen des Inhalts von Prozeduren, Funktionen, Triggern, Ereignissen oder Ansichten“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

ALTER USER-Anweisung

Ändert Benutzereinstellungen.

Syntax 1 - Ändert die Definition eines Datenbankbenutzers

```
ALTER USER user-name
[ IDENTIFIED BY password ]
[ LOGIN POLICY policy-name ]
[ FORCE PASSWORD CHANGE { ON | OFF } ]
```

Syntax 2 - Setzt die Login-Richtlinie eines Benutzers auf die ursprünglichen Werte zurück

```
ALTER USER user-name
[ RESET LOGIN POLICY ]
```

Syntax 3 - Distinguished Name (DN) für einen LDAP-Benutzer aktualisieren

```
ALTER USER user-name
REFRESH DN
```

Syntax 4 - Kennwortteil ändern

```
ALTER USER user-name
[ IDENTIFIED { FIRST | LAST } BY password-part ]
```

Parameter

user-name Der Name des Benutzers.

IDENTIFIED-Klausel Das Kennwort des Benutzers. Ein Benutzer ohne Kennwort kann keine Verbindung zur Datenbank herstellen.

- **IDENTIFIED BY-Klausel** Verwenden Sie diese Klausel, um das Kennwort für einen Benutzer zurückzusetzen.
- **IDENTIFIED {FIRST | LAST} BY-Klausel** Verwenden Sie diese Klausel, um einen Teil des Kennworts für einen Benutzer zurückzusetzen, der ein Kennwort für die Doppelkontrolle hat. In der Login-Richtlinie eines Benutzers mit Kennwort für die Doppelkontrolle ist die CHANGE_PASSWORD_DUAL_CONTROL-Option aktiviert.

Zwei Administratoren sind erforderlich, um ein Kennwort für die Doppelkontrolle zurückzusetzen. Ein Administrator führt die IDENTIFIED FIRST BY-Klausel aus, um den ersten Teil des Kennworts festzulegen, und ein anderer Administrator führt die IDENTIFIED LAST BY-Klausel aus, um den letzten Teil des Kennworts festzulegen. Der Benutzer kombiniert die beiden Kennwortteile und verwendet dieses kombinierte Kennwort für Verbindungen mit der Datenbank.

Siehe „[Kennwörter für die Doppelkontrolle](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

policy-name Der Name der Login-Richtlinie, die dem Benutzer zugewiesen werden soll. Es wird keine Änderung durchgeführt, wenn die LOGIN POLICY-Klausel nicht angegeben wird.

FORCE PASSWORD CHANGE-Klausel Steuert, ob der Benutzer beim Anmelden ein neues Kennwort eingeben muss. Diese Einstellung setzt die Einstellung der password_expiry_on_next_login-Option in der Richtlinie des Benutzers außer Kraft.

RESET LOGIN POLICY-Klausel Setzt die Einstellungen der Login-Richtlinie eines Benutzers auf die ursprüngliche Einstellung zurück. Wenn Sie eine Login-Richtlinie zurücksetzen, kann ein Benutzer auf

ein Konto zugreifen, das aufgrund des Überschreitens eines Limits einer Login-Richtlinienoption wie `max_failed_login_attempts` oder `max_days_since_login` gesperrt wurde.

REFRESH DN-Klausel REFRESH DN löscht den Distinguished Name (DN) und den Zeitstempel des Benutzers, damit bei der nächsten LDAP-Authentifizierung die Suche nach dem DN ausgeführt wird. Wenn die Authentifizierung bei der nächsten LDAP-Authentifizierung dieses Benutzers erfolgreich ist, werden DN und Zeitstempel mit dem neuen DN und der aktuellen Zeit aktualisiert.

Bemerkungen

In der folgenden Liste werden die Anforderungen für Benutzer-IDs und Kennwörter beschrieben. Die Anforderungen und Einschränkungen für einen Kennwortteil entsprechen der Beschreibung für ein Kennwort, bis auf die Tatsache, dass die maximale Länge für jeden Teil 127 Byte beträgt.

- Benutzer-IDs dürfen Folgendes nicht:
 - Sie dürfen nicht mit Leerstellen, Apostrophen oder Anführungszeichen beginnen.
 - Sie dürfen nicht mit Leerstellen enden.
 - Sie dürfen keine Semikola enthalten.
- Kennwörter berücksichtigen die Groß- und Kleinschreibung. Im Übrigen gilt Folgendes:
 - Sie dürfen nicht mit Leerstellen, Apostrophen oder Anführungszeichen beginnen.
 - Sie dürfen nicht mit Leerstellen enden.
 - Sie dürfen keine Semikola enthalten.
 - länger als 255 Byte sein

Mit der Login-Richtlinienoption `verify_password_function` können Sie eine Funktion zum Implementieren von Kennwortregeln angeben (z.B., dass Kennwörter mindestens eine Ziffer enthalten müssen). Wenn eine Funktion zur Kennwortüberprüfung verwendet wird, können Sie nicht mehr als eine Benutzer-ID mit Kennwort in der GRANT CONNECT-Anweisung angeben.

Wenn Sie `password_expiry_on_next_login` auf ON einstellen, läuft das Kennwort des Benutzers sofort ab, wenn er sich das nächste Mal anmeldet, auch wenn ihm dieselbe Richtlinie zugewiesen wurde. Sie können die Klauseln ALTER USER und LOGIN POLICY verwenden, um einen Benutzer zu zwingen, sein Kennwort beim nächsten Anmelden zu ändern.

Die ALTER USER...REFRESH DN-Syntax löscht den Distinguished Name (DN) und den Zeitstempel des Benutzers, damit bei der nächsten LDAP-Authentifizierung die Suche nach dem DN ausgeführt wird, statt den im Cache abgelegten DN zu verwenden, der möglicherweise veraltet ist. Wenn die Authentifizierung erfolgreich ist, werden DN und Zeitstempel mit dem neuen DN und der aktuellen Zeit aktualisiert.

Privilegien

Jeder Benutzer kann sein eigenes Kennwort ändern.

Wenn Sie Kennwörter für andere Benutzer ändern möchten, müssen Sie das CHANGE PASSWORD-Systemprivileg haben.

Für alle anderen Änderungen an anderen Benutzern, einschließlich der Festlegung, dass Benutzer ihr Kennwort ändern müssen, benötigen Sie das MANAGE ANY USER-Systemprivileg.

Nebenwirkungen

Keine.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung ändert einen Benutzer namens SQLTester, indem sie sein Kennwort auf "welcome123" und seine Login-Richtlinie auf "Test1" setzt und erlaubt ihm, eine erzwungene Kennwortänderung zu umgehen.

```
ALTER USER SQLTester IDENTIFIED BY welcome123
LOGIN POLICY Test1
FORCE PASSWORD CHANGE off;
```

Im folgenden Beispiel wird der LDAP-Distinguished Name für den Benutzer myusername aktualisiert.

```
ALTER USER myusername REFRESH DN;
```

Siehe auch

- „ALTER LOGIN POLICY-Anweisung“ auf Seite 471
- „COMMENT-Anweisung“ auf Seite 573
- „CREATE LOGIN POLICY-Anweisung“ auf Seite 647
- „CREATE USER-Anweisung“ auf Seite 770
- „DROP LOGIN POLICY-Anweisung“ auf Seite 812
- „DROP USER-Anweisung“ auf Seite 838
- „verify_password_function-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Login-Richtlinien“ [*SQL Anywhere Server - Datenbankadministration*]
- „Login-Richtlinien vorhandenen Benutzern zuordnen“ [*SQL Anywhere Server - Datenbankadministration*]
- „GRANT CONNECT-Anweisung“ auf Seite 888

ALTER VIEW-Anweisung

Ersetzt eine Ansichtsdefinition durch eine geänderte Version.

Syntax 1 - Ändert die Definition einer Ansicht

```
ALTER VIEW
[ owner.]view-name [ ( column-name, ... ) ] AS query-expression
[ WITH CHECK OPTION ]
```

Syntax 2 - Ändert die Attribute einer Ansicht

```
ALTER VIEW
[ owner.]view-name { SET HIDDEN | RECOMPILE | DISABLE | ENABLE }
```

Parameter

AS-Klausel Die SELECT-Anweisung, auf der die Ansicht basiert. Die SELECT-Anweisung darf sich nicht auf lokale temporäre Tabellen beziehen. Außerdem kann *query-expression* eine GROUP BY-,

HAVING-, WINDOW- oder ORDER BY-Klausel enthalten sowie UNION, EXCEPT, INTERSECT oder einen allgemeinen Tabellenausdruck.

Die Semantik von Abfragen bestimmt, dass die Reihenfolge der zurückgegebenen Zeilen nicht definiert ist, es sei denn, die Abfrage kombiniert eine ORDER BY-Klausel mit einer TOP- oder FIRST-Klausel in der SELECT-Anweisung.

WITH CHECK OPTION-Klausel Die WITH CHECK OPTION-Klausel weist alle Aktualisierungen und Einfügungen der Ansicht zurück, die nicht den Kriterien der von den für *query-expression* festgelegten Werten entsprechen.

SET HIDDEN-Klausel Verwenden Sie die SET HIDDEN-Klausel, um die Definition der Ansicht zu verschleiern und die Ansicht zu verbergen, z.B. in Sybase Central. Explizite Referenzen zur Ansicht funktionieren allerdings weiterhin.

Hinweis

Der SET HIDDEN-Vorgang ist irreversibel.

RECOMPILE-Klausel Verwenden Sie die RECOMPILE-Klausel, um Spaltendefinitionen für die Ansicht neu zu erstellen. Diese Klausel hat dieselbe Funktion wie die ENABLE-Klausel, nur dass Sie erstere auf eine Ansicht anwenden können, die nicht deaktiviert ist. Wenn eine Ansicht neu kompiliert wird, stellt der Datenbankserver die Spaltenprivilegien auf Basis der in der neuen Ansichtsdefinition angegebenen Spaltennamen wieder her. Die vorhandenen Privilegien gehen verloren, wenn eine Spalte nach der Neukompilierung nicht mehr vorhanden ist.

DISABLE-Klausel Verwenden Sie die DISABLE-Klausel, um die Verwendung der Ansicht durch den Datenbankserver zu deaktivieren.

ENABLE-Klausel Verwenden Sie die ENABLE-Klausel, um eine deaktivierte Ansicht zu aktivieren. Ein Aktivieren der Ansicht bewirkt, dass der Datenbankserver die Spaltendefinitionen für die Ansicht neu erstellt. Bevor Sie eine Ansicht aktivieren, müssen Sie alle Ansichten aktivieren, von denen diese abhängt.

Bemerkungen

Im *query-expression* kann eine TOP n-, FIRST- oder LIMIT-Klausel angegeben werden, auch wenn er keine ORDER BY-Klausel enthält. Es wird höchstens die angegebene Zeilenanzahl zurückgegeben, aber die Reihenfolge der zurückgegebenen Zeilen ist nicht definiert. Eine ORDER BY-Klausel kann also angegeben werden, ist aber nicht erforderlich. Wenn in einer Abfrage eine Ansicht verwendet wird, bestimmt ORDER BY in der Ansicht nicht die Reihenfolge der Zeilen in der Abfrage, auch wenn die FROM-Klausel keine anderen Tabellen enthält. Das bedeutet, dass ORDER BY nur dann für eine Ansicht verwendet werden sollte, wenn es benötigt wird, um die Zeilen auszuwählen, die in eine TOP n-, FIRST- oder LIMIT-Klausel einbezogen werden sollen. Andernfalls hat die ORDER BY-Klausel keine Auswirkungen und wird vom Datenbankserver ignoriert.

Wenn Sie eine ALTER VIEW-Anweisung für eine Ansicht ausführen, die einen oder mehr INSTEAD OF-Trigger hat, wird ein Fehler zurückgegeben. Sie müssen den Trigger löschen, bevor die Ansicht gelöscht oder geändert werden kann.

Wenn Sie eine Ansicht eines anderen Eigentümers ändern, müssen Sie den Namen qualifizieren, indem Sie den Eigentümer eingeben (z.B. GROUPO.ViewSalesOrders). Wenn Sie den Namen nicht

qualifizieren, sucht der Datenbankserver nach einer Ihnen gehörenden Ansicht mit diesem Namen und ändert sie. Wenn eine solche nicht vorhanden ist, wird ein Fehler zurückgegeben.

Wenn Sie eine Ansicht ändern, werden vorhandene Privilegien für die Ansicht beibehalten und müssen nicht neu zugeordnet werden. Anstatt die ALTER VIEW-Anweisung zu verwenden, können Sie die Ansicht auch löschen und neu erstellen, indem Sie die DROP VIEW- bzw. die CREATE VIEW-Anweisung verwenden. Wenn Sie dies jedoch tun, müssen die Privilegien für die Ansicht neu zugeordnet werden.

In einer Abfrage kann eine TOP n-, FIRST- oder LIMIT-Klausel angegeben werden, auch wenn er keine ORDER BY-Klausel enthält. Es wird höchstens die angegebene Zeilenanzahl zurückgegeben, aber die Reihenfolge der zurückgegebenen Zeilen ist nicht definiert. Eine ORDER BY-Klausel kann also angegeben werden, ist aber nicht erforderlich. Wenn in einer Abfrage eine Ansicht verwendet wird, bestimmt ORDER BY in der Ansicht nicht die Reihenfolge der Zeilen in der Abfrage, auch wenn die FROM-Klausel keine anderen Tabellen enthält. Das bedeutet, dass ORDER BY nur dann für eine Ansicht verwendet werden sollte, wenn es benötigt wird, um die Zeilen auszuwählen, die in eine TOP n-, FIRST- oder LIMIT-Klausel einbezogen werden sollen. Andernfalls hat die ORDER BY-Klausel keine Auswirkungen und wird vom Datenbankserver ignoriert.

Nachdem die Ansichtsänderung mit Syntax 1 abgeschlossen ist, rekompiliert der Datenbankserver die Ansicht. Wenn es abhängige Ansichten gibt, versucht der Datenbankserver diese je nach Typ der durchgeführten Änderung ebenfalls zu kompilieren. Wenn Sie eine Änderung durchgeführt haben, die sich auf eine abhängige Ansicht auswirkt, müssen Sie möglicherweise auch die Definition für die abhängige Ansicht ändern.

Vorsicht

Wenn die SELECT-Anweisung, die die Ansicht definiert, ein Sternchen (*) enthält, kann sich die Anzahl der Spalten in der Ansicht ändern, wenn Spalten in den Basistabellen hinzugefügt oder gelöscht wurden. Die Namen und Datentypen der Ansichtsspalten können sich ebenfalls ändern.

- **Syntax 1** Diese Syntax wird verwendet, um die Struktur der Ansicht zu ändern. Anders als bei Tabellenänderungen, wo sich die Änderungen auf einzelne Spalten beschränken können, erfordert das Ändern der Struktur einer Ansicht die Ersetzung der gesamten Ansichtsdefinition durch eine neue Definition, ähnlich wie beim Erstellen der Ansicht.
- **Syntax 2** Diese Syntax wird verwendet, um die Attribute für die Ansicht zu ändern, die z.B. festlegen, ob die Ansichtsdefinition verborgen ist.

Wenn Sie SET HIDDEN verwenden, kann die Ansicht entladen und in andere Datenbanken geladen werden. Wenn SET HIDDEN verwendet wird, ist die Definition der Ansicht bei der Fehlerbehandlung mit dem Debugger nicht zu sehen und sie wird auch nicht in den Prozedurprofilen angezeigt. Wenn Sie die Definition einer verborgenen Ansicht ändern wollen, müssen Sie die Ansicht löschen und mit der CREATE VIEW-Anweisung erneut erstellen.

Nachdem Sie die DISABLE-Klausel verwendet haben, steht die Ansicht dem Datenbankserver zur Beantwortung von Abfragen nicht mehr zur Verfügung. Das Deaktivieren einer Ansicht ähnelt ihrem Löschen, mit dem Unterschied, dass die Ansichtsdefinition in der Datenbank bleibt. Das Deaktivieren einer Ansicht deaktiviert auch etwaige abhängige Ansichten. Daher erfordert die DISABLE-Klausel

einen exklusiven Zugriff nicht nur auf die Ansicht, die deaktiviert wird, sondern auch auf alle abhängigen Ansichten, da diese ja auch deaktiviert werden.

Privilegien

Sie müssen Eigentümer der Ansicht sein oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Ansicht
- ALTER ANY VIEW-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Außerdem benötigen Sie die SELECT-Berechtigung für die zugrunde liegenden Objekte der Ansicht.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Alle Prozeduren und Trigger werden vom Speicher entladen, sodass jede Prozedur oder Trigger, der die Ansicht referenziert, der neuen Ansichtsdefinition entspricht. Das Entladen und Laden von Prozeduren und Triggern kann sich auf die Performance auswirken, wenn Sie Ansichten häufig ändern.

Siehe auch

- „CREATE VIEW-Anweisung“ auf Seite 773
- „DROP VIEW-Anweisung“ auf Seite 840
- „CREATE MATERIALIZED VIEW-Anweisung“ auf Seite 652
- „ALTER MATERIALIZED VIEW-Anweisung“ auf Seite 476
- „Verbergen des Inhalts von Prozeduren, Funktionen, Triggern, Ereignissen oder Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Ansichtenabhängigkeiten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Reguläre Ansichten erstellen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

ATTACH TRACING-Anweisung

Startet eine Diagnoseprotokollierungssitzung (und damit das Senden von Diagnosedaten an die Diagnosetabellen).

Syntax

```
ATTACH TRACING TO { LOCAL DATABASE | connect-string }  
[ LIMIT { size | history } ]
```

connect-string : Die Verbindungszeichenfolge für die Datenbank

size : **SIZE** *nnn* { **MB** | **GB** }

history : **HISTORY** *nnn* { **MINUTES** | **HOURS** | **DAYS** }

nnn : integer

Parameter

connect-string Die Verbindungszeichenfolge ist zur Herstellung einer Verbindung mit der Datenbank erforderlich, die die Protokollierungsinformationen empfängt. Dieser Parameter ist nur erforderlich, wenn die Datenbank, deren Profil erstellt wird, eine andere als die Datenbank ist, die die Daten empfängt.

Die folgenden Verbindungsparameter sind in *connect-string* zulässig: DBF, DBKEY, DBN, Host, Server, ENG, LINKS, PWD, UID.

Geben Sie DBF relativ zu dem Datenbankserver an, mit dem Sie eine Verbindung herstellen möchten. Wenn Sie keinen anderen Datenbankserver angeben, versucht der Datenbankserver, mit dem Sie gerade verbunden sind, die durch den Verbindungsparameter DBF identifizierte Protokollierungsdatenbank zu starten.

Ein Fehler wird zurückgegeben, wenn Sie den DBF-Parameter mit den Verbindungsparametern LINKS oder SERVER angeben.

LIMIT-Klausel Die Umfangsbeschränkung für Daten, die in der Protokollierungsdatenbank gespeichert werden, entweder nach Größe oder nach Dauer.

Bemerkungen

Die ATTACH TRACING-Anweisung wird verwendet, um eine Protokollierungssitzung für die Datenbank zu starten, deren Profil Sie erstellen möchten. Sie können sie erst verwenden, nachdem eine Protokollierungsstufe gesetzt wurde. Sie setzen die Protokollierungsstufe, indem Sie entweder Sybase Central oder die *sa_set_tracing_level*-Systemprozedur verwenden.

Wenn eine Sitzung gestartet ist, werden Protokollierungsinformationen entsprechend den Protokollierungsstufen generiert, die in der *sa_diagnostic_tracing_level*-Tabelle festgelegt sind. Sie können die Protokollierungsdaten an Protokollierungstabellen in der Datenbank senden, deren Profil erstellt wird, indem Sie LOCAL DATABASE angeben. Oder Sie können die Protokollierungsdaten an eine separate Protokollierungsdatenbank senden, indem Sie eine Verbindungszeichenfolge (*connect-string*) für diese Datenbank angeben. Die Protokollierungsdatenbank muss bereits bestehen, und Sie müssen Zugriff darauf haben.

Sie können die Menge an Protokollierungsdaten beschränken, indem Sie die LIMIT SIZE- oder LIMIT HISTORY-Klauseln verwenden. Verwenden Sie die LIMIT SIZE-Klausel, wenn Sie den Umfang der Protokollierungsdaten auf eine bestimmte Größe, in MB oder GB, beschränken wollen. Verwenden Sie die LIMIT HISTORY-Klausel, um den Umfang der Protokollierungsdaten auf eine Zeitperiode, in Minuten, Stunden oder Tagen gemessen, beschränken wollen. Beispiel: HISTORY 8 DAYS beschränkt die Menge der Protokollierungsdaten in der Protokollierungsdatenbank auf 8 Tage.

Um eine Protokollierungssitzung zu starten, muss TCP/IP auf dem/den Datenbankserver(n) laufen, auf dem die Protokollierungsdatenbank und die Produktionsdatenbank ausgeführt werden.

Pakete, die potenziell sensitive Daten enthalten, sind auf der Netzwerkschnittstelle sichtbar, selbst wenn die Protokollierung eine lokale Datenbank verwendet. Für Sicherheitszwecke können Sie eine Verschlüsselung in der Verbindungszeichenfolge festlegen.

Die aktuellen für eine Datenbank gesetzten Protokollierungsstufen finden Sie in der sa_diagnostic_tracing_level-Tabelle.

Um festzustellen, wohin Protokollierungsdaten gesendet werden, untersuchen Sie die Datenbankeigenschaft SendingTracingTo.

Sie müssen mit der Datenbank verbunden sein, deren Profil erstellt wird, um diese Anweisung ausführen zu können.

Privilegien

Sie müssen die DIAGNOSTICS-Systemrolle und das MANAGE PROFILING-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „DETACH TRACING-Anweisung“ auf Seite 801
- „REFRESH TRACING LEVEL-Anweisung“ auf Seite 993
- „Datenbankserveroption -x “ [*SQL Anywhere Server - Datenbankadministration*]
- „sa_diagnostic_tracing_level-Tabelle“ auf Seite 1156
- „sa_set_tracing_level-Systemprozedur“ auf Seite 1330
- SendingTracingTo-Datenbankeigenschaft [*SQL Anywhere Server - Datenbankadministration*]
- „Verbindungsparameter“ [*SQL Anywhere Server - Datenbankadministration*]
- „TCP/IP-Protokoll“ [*SQL Anywhere Server - Datenbankadministration*]
- „Diagnoseprotokollierung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Das folgende Beispiel setzt die Protokollierungsstufe auf 1, wobei die Systemprozedur sa_set_tracing_level verwendet wird. Dann wird eine Protokollierungssitzung gestartet. Für die lokale Datenbank generierte Protokollierungsdaten werden an die Protokollierungsdatenbank mytracingdb auf einem anderen Computer gesendet, wie durch die angegebene Verbindungszeichenfolge angezeigt. Ein Maximum von zwei Stunden an Protokollierungsdaten wird während der Protokollierungssitzung aufrecht erhalten. Das ATTACH TRACING-Anweisungsbeispiel wird in einer einzigen Zeile eingegeben.

```
CALL sa_set_tracing_level( 1 );  
ATTACH TRACING TO  
'uid=DBA;pwd=sql;server=remotedbsrv;dbn=mytracingdb;host=winxp-32'  
LIMIT HISTORY 2 HOURS;
```

BACKUP-Anweisung

Sichert die Datenbank und das Transaktionslog.

Syntax 1 - Sicherungskopie

```

BACKUP DATABASE
DIRECTORY backup-directory
[ backup-option [ backup-option ... ] ]

backup-directory : { string | variable }

backup-option :
WAIT BEFORE START
WAIT AFTER END
DBFILE ONLY
TRANSACTION LOG ONLY
TRANSACTION LOG RENAME [ MATCH ]
TRANSACTION LOG TRUNCATE
ON EXISTING ERROR
WITH COMMENT comment string
HISTORY { ON | OFF }
AUTO TUNE WRITERS { ON | OFF }
WITH CHECKPOINT LOG { AUTO | COPY | NO COPY | RECOVER }

```

Syntax 2 - Archivsicherung

```

BACKUP DATABASE TO archive-root
[ backup-option [ backup-option ... ] ]

archive-root : { string | variable }

backup-option :
WAIT BEFORE START
WAIT AFTER END
DBFILE ONLY
TRANSACTION LOG ONLY
TRANSACTION LOG RENAME [ MATCH ]
TRANSACTION LOG TRUNCATE
ATTENDED { ON | OFF }
WITH COMMENT comment string
HISTORY { ON | OFF }
WITH CHECKPOINT LOG [ NO ] COPY
MAX WRITE { number-of-writers | AUTO }
FREE PAGE ELIMINATION { ON | OFF }

```

comment-string : *string*

number-of-writers : *integer*

Parameter

DIRECTORY-Klausel Der Zielspeicherort auf der Festplatte für die Sicherungsdateien, relativ zum aktuellen Verzeichnis des Datenbankservers beim Start. Wenn das Verzeichnis nicht existiert, wird es erstellt. Wenn Sie eine leere Zeichenfolge als Verzeichnis eingeben, können Sie das Transaktionslog umbenennen oder verkürzen, ohne es vorher zu kopieren. Verwenden Sie diese Klausel nicht, wenn Sie mit einer Datenbankspiegelung arbeiten.

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „[Sandboxing](#)“ [*SQL Anywhere Server - Datenbankadministration*].

WAIT BEFORE START-Klausel Diese Klausel verzögert die Sicherung, bis keine aktiven Transaktionen mehr vorhanden sind. Alle anderen Aktivitäten in der Datenbank werden verhindert und ein Checkpoint wird ausgeführt.

Wenn Sie diese Klausel zusammen mit der WITH CHECKPOINT LOG NO COPY-Klausel verwenden, wird überprüft, ob die Sicherungskopie der Datenbank eine Wiederherstellung erfordert, und Sie können die Sicherungskopie der Datenbank im schreibgeschützten Modus öffnen und validieren. Zum Validieren der Sicherungsdatenbank müssen Sie keine weitere Kopie der Datenbank erstellen.

WAIT AFTER END-Klausel Diese Klausel gewährleistet, dass alle Transaktionen abgeschlossen werden, bevor das Transaktionslog umbenannt oder gekürzt wird. Der Datenbankserver wartet, bis alle offenen Transaktionen anderer Verbindungen festgeschrieben oder zurückgesetzt wurden, bevor er die Sicherung abschließt. Seien Sie mit dieser Klausel vorsichtig, da neue eingehende Transaktionen dazu führen können, dass die Sicherung unbegrenzt wartet.

DBFILE ONLY-Klausel Diese Klausel erstellt Sicherungskopien von der Hauptdatenbankdatei und allen ihr zugeordneten DBSpaces, aber nicht vom Transaktionslog. Sie können die DBFILE ONLY-Klausel nicht zusammen mit den TRANSACTION LOG RENAME- oder TRANSACTION LOG TRUNCATE-Klauseln verwenden.

TRANSACTION LOG ONLY-Klausel Mithilfe der TRANSACTION LOG ONLY-Klausel können Sie eine Sicherungskopie des Transaktionslogs erstellen, ohne die anderen Datenbankdateien zu kopieren.

TRANSACTION LOG RENAME [MATCH]-Klausel Diese Klausel veranlasst den Datenbankserver, das aktuelle Transaktionslog nach Erstellen der Sicherungskopie umzubenennen. Wenn das Schlüsselwort MATCH ausgelassen wird, erhält die Sicherungskopie des Transaktionslogs denselben Namen wie das aktuelle Transaktionslog für die Datenbank. Wenn Sie das Schlüsselwort MATCH angeben, erhält die Sicherungskopie des Transaktionslogs einen Namen mit dem Format *JJMMTTnn.log*, um der umbenannten Kopie des aktuellen Transaktionslogs zu entsprechen. Mit dem Schlüsselwort MATCH kann dieselbe Anweisung mehrmals ausgeführt werden, ohne dass alte Daten überschrieben werden.

Das Transaktionslog kann umbenannt werden, ohne eine Sicherung durchzuführen. Dazu müssen Sie mit der TRANSACTION LOG ONLY-Klausel einen leeren Verzeichnisnamen angeben. Beispiel:

```
BACKUP DATABASE DIRECTORY ''  
TRANSACTION LOG ONLY  
TRANSACTION LOG RENAME;
```

TRANSACTION LOG TRUNCATE-Klausel Mit dieser Klausel wird das aktuelle Transaktionslog nach Erstellung der Sicherungskopie verkürzt und neu gestartet. Verwenden Sie diese Klausel nicht, wenn Sie mit einer Datenbankspiegelung arbeiten.

Das Transaktionslog kann gekürzt werden, ohne eine Sicherung durchzuführen. Dazu müssen Sie mit der TRANSACTION LOG ONLY-Klausel einen leeren Verzeichnisnamen angeben. Beispiel:

```
BACKUP DATABASE DIRECTORY ''  
TRANSACTION LOG ONLY  
TRANSACTION LOG TRUNCATE;
```

archive-root-Klausel Der Dateiname oder Devicename des Bandlaufwerks für die Archivdatei

Um eine Sicherung auf Band vorzunehmen, müssen Sie den Devicenamen des Bandlaufwerks angeben. Die an das Ende des Archivdateinamens angehängte Nummer wird jedesmal erhöht, wenn Sie eine Archivsicherung durchführen.

Das Backslashzeichen (\) ist ein Escapezeichen in SQL-Zeichenfolgen, weshalb jeder Backslash verdoppelt werden muss.

ON EXISTING ERROR-Klausel Diese Klausel betrifft nur Sicherungskopien. Standardmäßig werden vorhandene Dateien überschrieben, wenn Sie eine BACKUP DATABASE-Anweisung ausführen. Bei Verwendung dieser Klausel kommt es zu einem Fehler, wenn eine der Dateien, die von der Sicherung erstellt werden sollen, bereits vorhanden ist.

ATTENDED-Klausel Diese Klausel gilt nur beim Sichern auf ein Bandgerät. ATTENDED ON (Standardwert) zeigt an, dass eine Person verfügbar ist, die den Status des Bandlaufwerks überwacht und bei Bedarf ein neues Band ins Laufwerk einlegt. Wenn das Bandlaufwerk bedient werden muss, wird eine Meldung an die Anwendung geschickt, welche die BACKUP DATABASE-Anweisung aufgerufen hat. Der Datenbankserver wartet dann, bis das Laufwerk bereit ist. Dies ist z.B. der Fall, wenn ein neues Band erforderlich ist.

Wenn ATTENDED OFF angegeben wird und ein neues Band erforderlich oder das Laufwerk nicht bereit ist, wird keine Meldung gesendet und ein Fehler wird ausgegeben.

WITH COMMENT-Klausel Diese Klausel zeichnet einen Kommentar in der Sicherungsverlaufsdatei (backup.syb) auf. Bei Archivsicherungen wird der Kommentar auch in der Archivdatei aufgezeichnet.

HISTORY-Klausel Diese Klausel aktiviert bzw. deaktiviert den Sicherungsverlauf. Standardmäßig ist diese Klausel auf ON gesetzt, was bedeutet, dass bei jedem Sicherungsvorgang in der Datei *backup.syb* eine Zeile angehängt wird. Das Angeben von HISTORY OFF verhindert Aktualisierungen der Datei *backup.syb* und wird in folgenden Fällen empfohlen:

- Die Datenbank wird häufig gesichert.
- Es ist keine Prozedur eingerichtet, um die Datei *backup.syb* regelmäßig zu archivieren oder zu löschen.
- Der Speicherplatz ist beschränkt.

AUTO TUNE WRITERS-Klausel Das Angeben dieser Klausel aktiviert bzw. deaktiviert die automatische Optimierung der Writer-Threads. Während des Sicherungsprozesses schreibt ein Writer-Thread die Sicherungsdateien in das Sicherungsverzeichnis. Wenn sich das Sicherungsverzeichnis auf einem Gerät befindet, das eine erhöhte Writer-Last verarbeiten kann (z.B. auf einem RAID-Array), verbessert der Standardwert AUTO TUNE WRITERS ON die Gesamtpformance der Sicherung, indem er die Anzahl der Writer-Threads erhöht. Der Datenbankserver überprüft regelmäßig die Lese- und Schreibperformances aller Geräte, die an der Sicherung teilnehmen. Durch Angeben von AUTO TUNE WRITERS OFF verhindern Sie, dass der Datenbankserver zusätzliche Writer-Threads erstellt.

WITH CHECKPOINT LOG-Klausel Diese Klausel gibt an, wie die Sicherung die Datenbankdateien verarbeitet, bevor sie in das Zielverzeichnis geschrieben werden. Die Entscheidung, entweder Pre-Images während einer Sicherung anzuwenden oder das Checkpoint-Log als Teil der Sicherung zu kopieren, wirkt

sich auf die Performance aus. Die Standardeinstellung ist AUTO für Image-Sicherungen und COPY für Archivsicherungen.

- **COPY-Klausel** Diese Option darf nicht mit der WAIT BEFORE START-Klausel in der BACKUP-Anweisung verwendet werden.

Wenn Sie COPY angeben, werden bei der Sicherung die Datenbankdateien gelesen, ohne dass geänderte Seiten angewendet werden. Das gesamte Checkpoint-Log und der System-DBSpace werden in das Sicherungsverzeichnis kopiert. Wenn der Datenbankserver das nächste Mal gestartet wird, stellt er automatisch den Datenbankstatus wieder her, der beim Checkpoint zum Zeitpunkt des Sicherungsstarts galt.

Da Seiten nicht in die temporäre Datei geschrieben werden müssen, kann das Verwenden dieser Option die Sicherungsperformance verbessern und interne Serverkonflikte bei anderen Verbindungen verringern, die während einer Sicherung aktiv sind. Da das Checkpoint-Log jedoch Original-Sicherungskopien von geänderten Seiten enthält, wird es bei Datenbankaktualisierungen größer. Wenn COPY angegeben ist, kann die gesicherte Kopie der Datenbankdateien größer als die Datenbankdateien zum Zeitpunkt des Sicherungsstarts sein. Die COPY-Option sollte nur verwendet werden, wenn im Zielverzeichnis der Plattenspeicherplatz kein Problem darstellt.

- **NO COPY-Klausel** Wenn Sie NO COPY angeben, wird das Checkpoint-Log nicht als Teil der Sicherung kopiert. Diese Option bewirkt, dass geänderte Seiten in der temporären Datei gespeichert werden, damit sie im Verlauf der Sicherung angewendet werden können. Die Sicherungskopien der Datenbankdateien haben dieselbe Größe wie die Datenbank zum Zeitpunkt des Sicherungsstarts.

Diese Option führt zu kleineren gesicherten Datenbankdateien, aber die Sicherung braucht möglicherweise länger und kann die Performance von anderen Vorgängen im Datenbankserver beeinträchtigen. Sie ist in Situationen nützlich, bei denen der Speicherplatz auf dem Ziellaufwerk beschränkt ist.

- **RECOVER-Klausel** Wenn Sie RECOVER angeben, kopiert der Datenbankserver das Checkpoint-Log (wie bei der COPY-Option), wendet aber das Checkpoint-Log für die Datenbank an, wenn die Sicherung abgeschlossen ist. Dies stellt die gesicherten Datenbankdateien im selben Status (und derselben Größe) wieder her, in der sie beim Start des Sicherungsvorgangs waren. Diese Option kann nützlich sein, wenn der Speicherplatz auf dem Sicherungslaufwerk beschränkt ist (sie benötigt dieselbe Menge an Speicherplatz wie die COPY-Option zum Sichern des Checkpoint-Logs, aber die resultierende Dateigröße ist kleiner).
- **AUTO-Klausel** Wenn Sie AUTO angeben, überprüft der Datenbankserver die Menge des verfügbaren Speicherplatzes auf dem Datenträger, der das Sicherungsverzeichnis enthält. Wenn mindestens doppelt so viel verfügbarer Plattenspeicherplatz vorhanden ist wie die Größe der Datenbank zum Zeitpunkt des Sicherungsstarts, verhält sich diese Option so, als ob COPY angegeben wäre. Ansonsten verhält sie sich als NO COPY. AUTO ist die Standardeinstellung.

MAX WRITE-Klausel Für Archivsicherungen wird standardmäßig ein Thread zum Schreiben der Sicherungsdateien verwendet. Wenn sich das Sicherungsverzeichnis auf einem Gerät befindet, das eine erhöhte Schreib-Arbeitslast verarbeiten kann (wie z.B. ein RAID-Array), kann die Sicherungsperformance gesteigert werden, indem Sie die Anzahl der Threads erhöhen, die als Schreibprogramme agieren.

Wenn AUTO angegeben ist, wird ein Ausgabedatenstrom für jeden Lese-Thread eingerichtet. Der Wert *n* gibt die maximale Anzahl von Ausgabedatenströmen an, die erstellt werden können, bis zur Anzahl von Lese-Threads. Der Standardwert für diese Klausel ist 1. Bei einer Sicherung auf Band kann nur ein Schreib-Thread benutzt werden.

Der erste Datenstrom, der Datenstrom 0, erzeugt Dateien mit der Bezeichnung *myarchive.X*, wobei *X* eine Zahl ist, die mit 1 beginnt und inkrementiert, bis die Anzahl der erforderlichen Dateien erreicht ist. Alle anderen Datenströme erzeugen Dateien mit der Bezeichnung *myarchive.Y.Z*, wobei *Y* die Datenstromnummer (beginnend bei 1) ist, und *Z* eine Zahl ist, die bei 1 beginnt und inkrementiert, bis die Anzahl der erforderlichen Dateien erreicht ist.

FREE PAGE ELIMINATION-Klausel Standardmäßig werden bei Archivsicherungen freie Seiten übersprungen, um das Speichervolumen zu reduzieren und den Vorgang zu beschleunigen. Da Transaktionslogdateien keine freien Seiten enthalten, wirkt sich die Eliminierung freier Seiten nicht auf die Sicherung von Transaktionslogdateien aus. Datenbanken mit großen Transaktionslogdateien profitieren möglicherweise weniger von der Eliminierung freier Seiten als Datenbanken mit kleinen Transaktionslogdateien.

Wenn Sie eine stark verschlüsselte Datenbank mit aktivierter Eliminierung freier Seiten sichern, müssen Sie bei der Wiederherstellung der Datenbank den Chiffrierschlüssel angeben. Wenn Sie eine stark verschlüsselte mit deaktivierter Eliminierung freier Seiten Datenbank sichern, müssen Sie bei der Wiederherstellung der Datenbank nicht den Chiffrierschlüssel angeben.

Ab Version 12 können Sie keine Archivsicherungen wiederherstellen, die mit Version 11 oder früheren Datenbankservern erstellt wurden.

Bemerkungen

Die BACKUP-Anweisung führt eine serverseitige Sicherung durch. Um eine clientseitige Sicherung durchzuführen, verwenden Sie das Dienstprogramm dbbackup.

Wenn die Sandboxing-Funktion für die Datenbank aktiviert ist, müssen Sie einen Schlüssel für gesicherte Funktionen angeben, der die Sandboxing-Funktion deaktiviert, damit der Datenbankserver die Sicherung in einem Verzeichnis außerhalb der Sandbox (des Verzeichnisses, in dem sich die Hauptdatenbankdatei befindet, und der dazugehörigen Unterverzeichnisse) erstellen kann. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Jeder Sicherungsvorgang, egal ob Sicherungskopie oder Archiv, aktualisiert eine Sicherungsverlaufsdatei namens *backup.syb*. Die Datei protokolliert die BACKUP- und RESTORE-Vorgänge, die auf einem Datenbankserver ausgeführt wurden. Hinweise darüber, wie der Standort der *backup.syb*-Datei bestimmt wird, finden Sie unter „[SALOGDIR-Umgebungsvariable](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Um eine Sicherung zu erstellen, die auf einem schreibgeschützten Server gestartet werden kann, ohne die Wiederherstellung durchführen zu müssen, müssen Sie die Klauseln WAIT BEFORE START und WITH CHECKPOINT LOG NO COPY verwenden. Die WAIT BEFORE START-Klausel stellt sicher, dass das Log für das Zurücksetzen leer ist und die Klausel WITH CHECKPOINT LOG NO COPY gewährleistet, dass das Checkpoint-Log leer ist. Wenn eine dieser Dateien fehlt, ist eine Wiederherstellung erforderlich. Sie können die Klausel WITH CHECKPOINT LOG RECOVER als Alternative zu WAIT BEFORE

START und WITH CHECKPOINT LOG NO COPY verwenden, wenn Sie die gesicherte Datenbank nicht wiederherstellen müssen.

- **Syntax 1 - Sicherungskopie** Eine Sicherungskopie erstellt Kopien von allen Datenbankdateien, auf die gleiche Weise wie das Sicherungsdienstprogramm (dbbackup). Standardmäßig erfolgt die Sicherung durch das Sicherungsdienstprogramm auf dem Clientcomputer. Sie können jedoch die Option -s festlegen, um mit dem Sicherungsdienstprogramm eine Sicherung auf dem Datenbankserver vorzunehmen. Bei einer BACKUP DATABASE-Anweisung kann eine Sicherung allerdings nur auf dem Datenbankserver erfolgen.

Optional können nur die Datenbankdatei(en) oder das Transaktionslog gesichert werden. Das Transaktionslog kann auch umbenannt oder gekürzt werden, wenn die Sicherung abgeschlossen ist.

Sie können auch eine leere Zeichenfolge als Verzeichnis eingeben, wenn Sie das Log umbenennen oder verkürzen wollen, ohne es vorher zu kopieren. Dies ist in einer Replikationsumgebung mit begrenztem Speicherplatz nützlich. Sie können diese Funktion mit einem Event-Handler für die Transaktionslog-Größe verwenden, um das Transaktionslog bei Erreichen einer bestimmten Größe umzubenenen, sowie mit der Option delete_old_logs, um das Transaktionslog zu löschen, wenn es nicht mehr benötigt wird.

Um anhand einer Sicherungskopie wiederherzustellen, kopieren Sie die gesicherten Dateien an ihre ursprünglichen Stellen zurück und übernehmen die Transaktionslogs wieder.

- **Syntax 2 - Archivsicherung** Eine Archivsicherung erstellt eine einzige Datei, die alle erforderlichen Sicherungsinformationen enthält. Das Ziel kann entweder ein Dateiname oder der Devicename eines Bandlaufwerks sein.

Auf einem Band kann es nur eine Sicherung geben. Das Band wird nach beendeter Sicherung ausgeworfen.

Nur ein Archiv pro Band ist zulässig, ein einzelnes Archiv kann dagegen mehrere Bänder umfassen. Um eine Datenbank aus einer Archivsicherung wiederherzustellen, verwenden Sie die RESTORE DATABASE-Anweisung.

Wenn eine RESTORE DATABASE-Anweisung eine Archivdatei referenziert, die nur ein Transaktionslog enthält, muss die Anweisung einen Dateinamen für den Speicherort der wiederherzustellenden Datenbankdatei angeben, selbst wenn die Datei nicht vorhanden ist. Die RESTORE DATABASE-Anweisung zur Wiederherstellung einer Datenbankdatei aus einem Archiv, das nur ein Transaktionslog enthält, in das Verzeichnis C:\MYNEWDB sieht beispielsweise wie folgt aus:

```
RESTORE DATABASE 'c:\\temp\\mynewdb\\my.db' FROM archive-root
```

Vorsicht

Sicherungskopien der Datenbank und des Transaktionslogs dürfen auf keinen Fall geändert werden. Wenn während der Sicherung keine Transaktionen aktiv waren oder Sie `BACKUP DATABASE WITH CHECKPOINT LOG RECOVER` oder `WITH CHECKPOINT LOG NO COPY` angegeben haben, können Sie die Validität der Sicherungsdatenbank im Schreibschutzmodus prüfen, oder indem Sie eine Kopie der Sicherungsdatenbank validieren.

Wenn jedoch Transaktionen aktiv waren oder Sie `BACKUP DATABASE WITH CHECKPOINT LOG COPY` angegeben haben, muss der Datenbankserver eine Wiederherstellung der Datenbank bei ihrem Start durchführen. Die Wiederherstellung ändert die Sicherungskopie, und das sollte vermieden werden.

Während des Ausführens dieser Anweisung können Sie Meldungen zum Verarbeitungsfortschritt anfordern.

Sie können auch mithilfe der Verbindungseigenschaft "Progress" feststellen, wie viel von der Anweisung ausgeführt wurde.

Privilegien

Sie müssen das `BACKUP DATABASE`-Systemprivileg haben.

Nebenwirkungen

Verursacht einen Checkpoint

Siehe auch

- „Serverseitige Sicherungen erstellen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankwiederherstellung mit mehreren Transaktionslogs“ [[SQL Anywhere Server - Datenbankadministration](#)]
- Progress-Verbindungseigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]
- „progress_messages-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Sicherungsdienstprogramm (dbbackup)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Sicherungskopien“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Zeichenfolgen“ auf Seite 6
- „RESTORE DATABASE-Anweisung“ auf Seite 1001
- KEY-Klausel, RESTORE DATABASE-Anweisung auf Seite 1002
- „Sicherung und Datenwiederherstellung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Verwaltung von Transaktionslogdateien in einem Datenbankspiegelungssystem“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „EXECUTE IMMEDIATE-Anweisung [SP]“ auf Seite 843
- „Parallele Datenbanksicherungen“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Windows Mobile** Nur die Syntax von `BACKUP DATABASE DIRECTORY` (Syntax 1) wird unter Windows Mobile unterstützt.

Beispiel

Die aktuelle Datenbank und das Transaktionslog werden in verschiedenen Dateien gesichert, das aktuelle Transaktionslog wird umbenannt. Eine Sicherungskopie wird erstellt.

```
BACKUP DATABASE
DIRECTORY 'c:\\temp\\backup'
TRANSACTION LOG RENAME;
```

Die Option zum Umbenennen des Transaktionslogs ist vor allem in Replikationsumgebungen von Bedeutung, wo das alte Transaktionslog noch benötigt wird.

Die aktuelle Datenbank und das Transaktionslog werden auf Band gesichert:

```
BACKUP DATABASE
TO '\\\\.\\tape0';
```

Das Transaktionslog wird umbenannt, ohne dass eine Kopie angelegt wird:

```
BACKUP DATABASE DIRECTORY ''
TRANSACTION LOG ONLY
TRANSACTION LOG RENAME;
```

Die BACKUP DATABASE-Anweisung wird mit einem dynamisch aufgebauten Verzeichnisnamen ausgeführt:

```
CREATE EVENT NightlyBackup
SCHEDULE
START TIME '23:00' EVERY 24 HOURS
HANDLER
BEGIN
    DECLARE dest LONG VARCHAR;
    DECLARE day_name CHAR(20);

    SET day_name = DATENAME( WEEKDAY, CURRENT DATE );
    SET dest = 'd:\\backups\\' || day_name;
    BACKUP DATABASE DIRECTORY dest
    TRANSACTION LOG RENAME;
END;
```

BEGIN SNAPSHOT-Anweisung

Verwenden Sie diese Anweisung, um zu einem festgelegten Zeitpunkt einen Snapshot zu starten, der mit Snapshot-Isolationstransaktionen verwendet werden soll.

Syntax

BEGIN SNAPSHOT

Bemerkungen

Standardmäßig gilt: Wenn eine Transaktion beginnt, verschiebt der Datenbankserver die Snapshot-Erstellung, bis die Anwendung einen Abruf der ersten Zeile einer Tabelle einleitet. Sie können die BEGIN SNAPSHOT-Anweisung verwenden, um den Snapshot früher in der Transaktion zu beginnen. Der Datenbankserver erstellt einen Snapshot, wenn die BEGIN SNAPSHOT-Anweisung durch eine Snapshot-Transaktion ausgeführt wird.

Die Anweisung schlägt fehl und gibt einen Fehler zurück, wenn eine der folgenden Bedingungen eintritt:

- Die Unterstützung für Snapshot-Transaktionen wurde für die Datenbank nicht aktiviert.
- Ein Snapshot wurde für die laufende Transaktion bereits gestartet.

Diese Anweisung ist auch für Nicht-Snapshot-Transaktionen nützlich, weil sie ermöglicht, einen Snapshot auszulösen, der später in der Transaktion für einen Snapshot-Vorgang auf Anweisungsebene benutzt wird.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „allow_snapshot_isolation-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Snapshot-Isolation“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

BEGIN-Anweisung

Gibt eine zusammengesetzte Anweisung an.

Syntax

```
[ statement-label : ]
BEGIN [ [ NOT ] ATOMIC ]
  [ local-declaration; ... ]
  statement-list
  [ EXCEPTION [ exception-case ... ] ]
END [ statement-label ]
```

local-declaration :

- variable-declaration*
- | *cursor-declaration*
- | *exception-declaration*
- | *temporary-table-declaration*

exception-case :

```
WHEN exception-name [ , ... ] THEN statement-list
| WHEN OTHERS THEN statement-list
```

variable-declaration und *exception-declaration* : Siehe „[DECLARE-Anweisung](#)“ auf Seite 786.

cursor-declaration : Siehe „[DECLARE CURSOR-Anweisung \[ESQL\] \[SP\]](#)“ auf Seite 778.

temporary-table-declaration : Siehe „[DECLARE LOCAL TEMPORARY TABLE-Anweisung](#)“ auf Seite 784.

Parameter

statement-label Wenn das *statement-label* am Ende angegeben ist, muss es mit dem *statement-label* am Anfang übereinstimmen. Die LEAVE-Anweisung kann verwendet werden, um die Ausführung bei der ersten Anweisung nach der zusammengesetzten Anweisung wieder aufzunehmen. Die zusammengesetzte Anweisung, die den Hauptteil einer Prozedur oder eines Triggers darstellt, hat ein implizites Label, das mit dem Namen der Prozedur oder des Triggers übereinstimmt.

ATOMIC-Klausel Eine atomare Anweisung wird entweder komplett oder gar nicht ausgeführt. Beispiel: Eine UPDATE-Anweisung, die Tausende von Zeilen aktualisiert, kann nach den ersten aktualisierten Zeilen auf einen Fehler stoßen. Wenn die Anweisung nicht abgeschlossen werden kann, werden alle bisher durchgeführten Aktualisierungen rückgängig gemacht, sodass der Ausgangsstatus wieder hergestellt wird. Ebenso gilt: Wenn Sie angeben, dass die BEGIN-Anweisung unteilbar ist, wird die Anweisung entweder vollständig oder gar nicht ausgeführt.

local-declaration Unmittelbar nach BEGIN können in einer zusammengesetzten Anweisung lokale Deklarationen für Objekte folgen, die nur innerhalb der zusammengesetzten Anweisung existieren. Eine zusammengesetzte Anweisung kann eine lokale Deklaration für eine Variable, einen Cursor, eine temporäre Tabelle oder eine Ausnahmebedingung enthalten. Lokale Deklarationen können von jeder beliebigen Anweisung in dieser zusammengesetzten Anweisung oder in jeder beliebigen darin verschachtelten zusammengesetzten Anweisung referenziert werden. Lokale Deklarationen der zusammengesetzten Anweisung sind für die Ausnahmeroutinen für die Anweisung sichtbar. Lokale Deklarationen sind für die anderen Prozeduren, die von innerhalb einer zusammengesetzten Anweisung aufgerufen werden, nicht sichtbar.

Bemerkungen

Der Hauptteil einer Prozedur oder eines Triggers besteht aus einer zusammengesetzten Anweisung. Zusammengesetzte Anweisungen können auch in Steueranweisungen innerhalb einer Prozedur oder eines Triggers verwendet werden.

Eine zusammengesetzte Anweisung ermöglicht die Gruppierung einer oder mehrerer SQL-Anweisungen, die dann als Einheit behandelt werden können. Eine zusammengesetzte Anweisung beginnt mit dem Schlüsselwort BEGIN und endet mit dem Schlüsselwort END.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Behandlung von Fehlern und Warnungen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- Routine bei Ausnahmefehlern und atomare zusammengesetzte Anweisungen [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Atomare zusammengesetzte Anweisungen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „CONTINUE-Anweisung“ auf Seite 581
- „SIGNAL-Anweisung [SP]“ auf Seite 1061
- „RESIGNAL-Anweisung [SP]“ auf Seite 1000
- „RAISERROR-Anweisung“ auf Seite 982
- „BEGIN-Anweisung [TSQL]“ auf Seite 560

Standards und Kompatibilität

- **SQL/2008** BEGIN kennzeichnet eine zusammengesetzte Anweisung und umfasst einen Teil der optionalen SQL-Sprachenfunktion P002 in SQL/2008.

Beispiel

Der Hauptteil einer Prozedur oder eines Triggers besteht aus einer zusammengesetzten Anweisung.

```
CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT TopValue INT)
BEGIN
    DECLARE err_notfound EXCEPTION FOR
        SQLSTATE '02000';
    DECLARE curThisCust CURSOR FOR
        SELECT CompanyName, CAST(
            sum( SalesOrderItems.Quantity *
              Products.UnitPrice ) AS INTEGER) VALUE
        FROM GROUPO.Customers
            LEFT OUTER JOIN SalesOrders
            LEFT OUTER JOIN SalesOrderItems
            LEFT OUTER JOIN Products
        GROUP BY CompanyName;
    DECLARE ThisValue INT;
    DECLARE ThisCompany CHAR( 35 );
    SET TopValue = 0;
    OPEN curThisCust;
CustomerLoop:
LOOP
    FETCH NEXT curThisCust
        INTO ThisCompany, ThisValue;
    IF SQLSTATE = err_notfound THEN
        LEAVE CustomerLoop;
    END IF;
    IF ThisValue > TopValue THEN
        SET TopValue = ThisValue;
        SET TopCompany = ThisCompany;
    END IF;
END LOOP CustomerLoop;
CLOSE curThisCust;
END;
```

Im nachstehenden Beispiel werden die folgenden Variablen deklariert:

- v1 als INT mit dem Anfangswert 5.
- v2 und v3 als CHAR(10), beide mit dem Anfangswert abc.

```
BEGIN
  DECLARE v1 INT = 5
  DECLARE v2, v3 CHAR(10) = 'abc'
  // ...
END
```

BEGIN-Anweisung [TSQL]

Gibt eine zusammengesetzte Anweisung an.

Syntax

BEGIN

statement-list

END

statement-list :

sql-statement

| *variable-declaration*

| *cursor-declaration*

| *temporary-table-declaration*

variable-declaration : Siehe „[DECLARE-Anweisung](#)“ auf Seite 786.

cursor-declaration : Siehe „[DECLARE CURSOR-Anweisung \[ESQL\] \[SP\]](#)“ auf Seite 778.

temporary-table-declaration : Siehe „[DECLARE LOCAL TEMPORARY TABLE-Anweisung](#)“ auf Seite 784.

Parameter

statement-list Eine Liste der Anweisungen und Deklarationen.

Bemerkungen

Eine BEGIN-Anweisung ermöglicht die Gruppierung einer oder mehrerer SQL-Anweisungen, die dann als Einheit behandelt werden können. Sie beginnt mit dem Schlüsselwort BEGIN und endet mit dem Schlüsselwort END.

Die Fehlerbehandlung unterscheidet sich für zusammengesetzte Anweisungen in Transact-SQL.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „Fehlerbehandlung in Transact-SQL-Prozeduren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „BEGIN-Anweisung“ auf Seite 557
- „CONTINUE-Anweisung“ auf Seite 581
- „GOTO-Anweisung [T-SQL]“ auf Seite 881
- „RAISERROR-Anweisung“ auf Seite 982
- „Transact-SQL-kompatible Datenbanken“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** BEGIN kennzeichnet eine zusammengesetzte Anweisung und umfasst einen Teil der optionalen SQL-Sprachenfunktion P002 in SQL/2008.

Beispiel

Im nachstehenden Beispiel werden die folgenden Variablen deklariert:

- v1 als INT mit dem Anfangswert 5.
- v2 und v3 als CHAR(10), beide mit dem Anfangswert abc.

```
BEGIN
  DECLARE v1 INT = 5
  DECLARE v2, v3 CHAR(10) = 'abc'
  // ...
END
```

BEGIN TRANSACTION-Anweisung [T-SQL]

Beginnt eine benutzerdefinierte Transaktion.

Syntax

BEGIN TRAN[SACTION] [*transaction-name*]

Bemerkungen

Der optionale Parameter *transaction-name* ist der Name, der dieser Transaktion zugeordnet wird. Er muss ein gültiger Bezeichner sein. Verwenden Sie Transaktionsnamen nur für das äußerste Paar von verschachtelten BEGIN/COMMIT- oder BEGIN/ROLLBACK-Anweisungen.

Wenn die BEGIN TRANSACTION-Anweisung innerhalb einer Transaktion ausgeführt wird, führt das zu einer Heraufsetzung der Verschachtelungsebene um eins. Die Verschachtelungsebene wird durch eine COMMIT-Anweisung heruntersetzt. Bei verschachtelten Transaktionen werden die Änderungen an der Datenbank nur durch die äußerste COMMIT-Anweisung permanent.

Sowohl Adaptive Server Enterprise als auch SQL Anywhere bieten zwei Transaktionsmodi.

Der Standard-Transaktionsmodus von Adaptive Server Enterprise wird unverketteter Modus genannt und schreibt jede Anweisung einzeln fest, es sei denn, es wird eine explizite BEGIN TRANSACTION-Anweisung ausgeführt, um eine Transaktion zu starten. Im Gegensatz dazu schreibt der ISO SQL/2008-

kompatible, verkettete Modus eine Transaktion nur fest, wenn eine explizite COMMIT-Anweisung ausgeführt wird oder eine Anweisung, die ein Autocommit durchführt (wie z.B. Datendefinitionsanweisungen).

Sie können den Modus steuern, indem Sie die Datenbankoption chained einstellen. Die Standardeinstellung für ODBC- und Embedded SQL-Verbindungen in SQL Anywhere ist ON, und SQL Anywhere läuft in diesem Fall im verketteten Modus. (ODBC-Benutzer sollten ebenfalls die ODBC-Einstellung AutoCommit überprüfen). Der Standardwert für TDS-Verbindungen ist OFF.

Im unverketteten Modus wird eine Transaktion implizit vor jeder Datenabfrage oder Änderungsanweisung gestartet. Diese Anweisungen sind: DELETE, INSERT, OPEN, FETCH, SELECT und UPDATE. Jede Transaktion muss jedoch explizit mit einer COMMIT- oder ROLLBACK-Anweisung abgeschlossen werden.

Die Einstellung der Option chained kann nicht innerhalb einer Transaktion geändert werden.

Vorsicht

Wenn Sie eine gespeicherte Prozedur aufrufen, müssen Sie sich vergewissern, dass sie unter dem gewünschten Transaktionsmodus auch funktioniert.

Die aktuelle Verschachtelungsebene ist in der globalen Variable @@trancount enthalten. Die Variable @@trancount hat vor der Ausführung der ersten BEGIN TRANSACTION-Anweisung einen Wert von 0. Datenbankänderungen werden erst durch eine COMMIT-Anweisung dauerhaft übernommen, die ausgeführt wird, wenn @@trancount eins beträgt.

Sie sollten die Werte von @@trancount nur heranziehen, um sich über die Anzahl der ausgegebenen, expliziten BEGIN TRANSACTION-Anweisungen auf dem Laufenden zu halten.

Wenn Adaptive Server Enterprise eine Transaktion implizit startet, wird die @@trancount-Variable auf 1 gesetzt. SQL Anywhere setzt den @@trancount-Wert nicht auf 1, wenn eine Transaktion implizit gestartet wird. Stattdessen hat die @@trancount-Variable in SQL Anywhere vor jeder BEGIN TRANSACTION-Anweisung einen Wert von 0 (obwohl es eine aktuelle Transaktion gibt), während sie in Adaptive Server Enterprise (im verketteten Modus) einen Wert von 1 hat.

Bei Transaktionen, die mit der Anweisung BEGIN TRANSACTION anfangen, hat @@trancount nach der ersten BEGIN TRANSACTION-Anweisung in SQL Anywhere und in Adaptive Server Enterprise einen Wert von 1. Wenn eine Transaktion implizit mit einer anderen Anweisung gestartet wird und anschließend eine BEGIN TRANSACTION-Anweisung ausgeführt wird, hat @@trancount nach der Anweisung BEGIN TRANSACTION sowohl in SQL Anywhere als auch in Adaptive Server Enterprise den Wert 2.

Eine ROLLBACK-Anweisung ohne den Namen einer Transaktion oder eines Savepoints setzt Anweisungen auf die äußerste (explizite oder implizite) BEGIN TRANSACTION-Anweisung zurück und bricht die gesamte Transaktion ab.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „COMMIT-Anweisung“ auf Seite 575
- „ROLLBACK-Anweisung“ auf Seite 1015
- „SAVEPOINT-Anweisung“ auf Seite 1019
- „isolation_level-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „chained-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Savepoints innerhalb von Transaktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** BEGIN TRANSACTION wird von Adaptive Server Enterprise unterstützt.

Beispiel

Der folgende Batch meldet aufeinanderfolgende Werte von @@trancount als "0", "1", "2", "1", "0". Die Werte werden im Meldungsfenster des Datenbankservers ausgegeben.

```
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
COMMIT
PRINT @@trancount
COMMIT
PRINT @@trancount
```

BREAK-Anweisung [T-SQL]

Beendet eine zusammengesetzte Anweisung oder eine Schleife.

Syntax

BREAK

Bemerkungen

Die BREAK-Anweisung ist eine Steueranweisung, die das Verlassen einer Schleife ermöglicht. Die Ausführung wird bei der ersten Anweisung nach der Schleife wieder aufgenommen.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „WHILE-Anweisung [T-SQL]“ auf Seite 1123
- „CONTINUE-Anweisung“ auf Seite 581
- „BEGIN-Anweisung“ auf Seite 557
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Transact-SQL-Erweiterung.

Beispiel

In diesem Beispiel unterbricht die BREAK-Anweisung die WHILE-Schleife, wenn der Preis für das teuerste Produkt über \$50 liegt. Sonst setzt sich die Schleife fort, bis der Durchschnittspreis größer oder gleich \$30 ist:

```
WHILE ( SELECT AVG( UnitPrice ) FROM Products ) < $30
BEGIN
    UPDATE GROUPO.Products
    SET UnitPrice = UnitPrice + 2
    IF ( SELECT MAX(UnitPrice) FROM Products ) > $50
        BREAK
END
```

CALL-Anweisung

Ruft eine Prozedur auf.

Syntax 1

[*variable* =] **CALL** *procedure-name* ([*expression*, ...])

Syntax 2

[*variable* =] **CALL** *procedure-name* ([*parameter-name* = *expression*, ...])

Bemerkungen

Die CALL-Anweisung ruft eine Prozedur auf, die zuvor mit einer CREATE PROCEDURE-Anweisung erstellt wurde. Wenn die Prozedur abgeschlossen ist, werden sämtliche INOUT- oder OUT-Parameterwerte zurück kopiert.

Die Argumentliste kann nach Position oder durch die Verwendung von Schlüsselwörtern angegeben werden. Die Argumente stimmen mit dem entsprechenden Parameter in der Parameterliste für die Prozedur hinsichtlich Position überein (DEFAULT kann für optionale Parameter verwendet werden). Bei Angabe nach Schlüsselwörtern werden den Argumenten die benannten Parameter zugeordnet.

Prozedurargumenten können in der CREATE PROCEDURE-Anweisung Standardwerte zugewiesen werden, und fehlenden Parametern wird der Standardwert zugeordnet. Falls kein Standardwert gesetzt und kein Argument angegeben sind, wird ein Fehler ausgegeben.

Innerhalb einer Prozedur kann eine CALL-Anweisung in einer DECLARE-Anweisung verwendet werden, wenn die Prozedur Ergebnismengen zurückgibt.

Unterabfragen und räumliche Methodenaufrufe sind nicht zulässig als Argumente für eine gespeicherte Prozedur in einer CALL-Anweisung.

Mit der RETURN-Anweisung können Prozeduren einen Ganzzahlwert zurückgeben (zum Beispiel als Zustandsindikator). Sie können diesen Rückgabewert in einer Variablen speichern, indem Sie das Gleichheitszeichen als Zuordnungsoperator verwenden:

Wenn die aufgerufene Prozedur INT zurückgibt und der Wert NULL ist, wird der Fehlerstatus, Wert 0, anstelle dessen zurückgegeben. Es gibt keine Möglichkeit, zwischen diesem Fall und dem Fall zu unterscheiden, bei dem tatsächlich ein Wert 0 zurückgegeben wird.

Hinweis

Diese Anweisung zum Aufrufen einer Funktion wird nicht mehr empfohlen. Wenn Sie eine Funktion aufrufen und das Ergebnis einer Variablen zuordnen möchten, verwenden Sie eine Zuordnungsanweisung. Beispiel:

```
DECLARE varname INT;
SET varname=test( );
```

Privilegien

Sie müssen Eigentümer der Prozedur sein oder eines der folgenden Privilegien haben:

- EXECUTE-Privileg für die Prozedur
- EXECUTE ANY PROCEDURE-Systemprivileg

Nebenwirkungen

Keine.

Siehe auch

- „CREATE FUNCTION-Anweisung“ auf Seite 633
- „CREATE FUNCTION-Anweisung [externer Aufruf]“ auf Seite 617
- „CREATE FUNCTION-Anweisung [Webdienst]“ auf Seite 624
- „CREATE PROCEDURE-Anweisung“ auf Seite 681
- „CREATE PROCEDURE-Anweisung [externer Aufruf]“ auf Seite 661
- „CREATE PROCEDURE-Anweisung [Webdienste]“ auf Seite 670
- „EXECUTE-Anweisung [T-SQL]“ auf Seite 848
- „Ergebnismengen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Kernfunktion. Die Verwendung der RETURN-Anweisung zum Zurückgeben eines Werts aus einer gespeicherten Prozedur ist eine Erweiterung des Herstellers. SQL/2008 unterstützt die Rückgabe von Werten nur für durch SQL aufgerufene Funktionen, nicht für Prozeduren. Die Standardwerte für Argumente gespeicherter Prozeduren werden in SQL/2008 nicht unterstützt.

Beispiel

In diesem Beispiel werden eine Prozedur, um die Anzahl der erteilten Aufträge des Kunden mit der angegebenen ID zurückzugeben, und eine Variable für das Ergebnis erstellt, die Prozedur wird aufgerufen und das Ergebnis angezeigt.

```
CREATE PROCEDURE OrderCount (IN customer_ID INT, OUT Orders INT)
BEGIN
    SELECT COUNT(GROUPO.SalesOrders.ID)
    INTO Orders
    FROM GROUPO.Customers
    KEY LEFT OUTER JOIN SalesOrders
    WHERE Customers.ID = customer_ID;
END
go
-- Create a variable to hold the result
CREATE VARIABLE Orders INT
go
-- Call the procedure, FOR customer 101
CALL OrderCount ( 101, Orders )
go
-- Display the result
SELECT Orders FROM DUMMY
go
```

CASE-Anweisung

Wählt einen Ausführungspfad auf der Basis mehrerer Fälle.

Syntax 1

```
CASE value-expression
WHEN [ constant | NULL ] THEN statement-list ...
[ WHEN [ constant | NULL ] THEN statement-list ] ...
[ ELSE statement-list ]
END [ CASE ]
```

Syntax 2

```
CASE
WHEN [ search-condition | NULL ] THEN statement-list ...
[ WHEN [ search-condition | NULL ] THEN statement-list ] ...
[ ELSE statement-list ]
END [ CASE ]
```

Bemerkungen

Syntax 1 Die CASE-Anweisung ist eine Steueranweisung, mit der Sie eine Liste von SQL-Anweisungen auswählen können, die auf der Grundlage des Wertes eines Ausdrucks ausgeführt werden sollen. Der *value-expression* ist ein Ausdruck, der nur einen Wert annimmt, wobei es sich um eine Zeichenfolge, eine Zahl, ein Datum oder um einen anderen SQL-Datentyp handeln kann. Wenn eine WHEN-Klausel für den Wert von *value-expression* existiert, dann wird die *statement-list* in der WHEN-Klausel ausgeführt. Wenn keine geeignete WHEN-Klausel existiert, es aber eine ELSE-Klausel gibt, dann wird die *statement-list* in der ELSE-Klausel ausgeführt. Die Ausführung wird bei der ersten Anweisung nach END CASE wieder aufgenommen.

Wenn *value-expression* Null sein kann, ersetzen Sie den *value-expression* NULL mithilfe der Funktion ISNULL durch einen anderen Ausdruck.

Syntax 2 Mit dieser Form werden die Anweisungen für die erste erfüllte *search-condition* in der CASE-Anweisung ausgeführt. Die ELSE-Klausel wird ausgeführt, wenn keine der *search-conditions* erfüllt wird.

Wenn der Ausdruck NULL sein kann, verwenden Sie die folgende Syntax für die erste *search-condition*:

```
WHEN search-condition IS NULL THEN statement-list
```

Hinweis

Verwechseln Sie die Syntax des CASE-Ausdrucks nicht mit der Syntax der CASE-Anweisung.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „ISNULL-Funktion [Verschiedene]“ auf Seite 295
- „Unbekannte Werte: NULL“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „BEGIN-Anweisung“ auf Seite 557
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „CASE-Ausdrücke“ auf Seite 25
- „CASE-Anweisung [T-SQL]“ auf Seite 568

Standards und Kompatibilität

- **SQL/2008** Die CASE-Anweisung ist Teil der Sprachenfunktion P002 (Verarbeitungsvollständigkeit) des SQL/2008-Standards. Die Verwendung von END alleine, statt END CASE, ist eine Erweiterung des Herstellers.
- **Transact-SQL** Die CASE-Anweisung wird von Adaptive Server Enterprise unterstützt.

Beispiel

Die folgende Prozedur verwendet eine CASE-Anweisung und klassifiziert die in der Tabelle Products der SQL Anywhere-Beispieldatenbank aufgeführten Produkte in "shirt", "hat", "shorts" oder "unknown".

```
CREATE PROCEDURE ProductType (IN product_ID INT, OUT type CHAR(10))
BEGIN
    DECLARE prod_name CHAR(20);
    SELECT Name INTO prod_name FROM GROUP1.Products
    WHERE ID = product_ID;
    CASE prod_name
    WHEN 'Tee Shirt' THEN
        SET type = 'Shirt'
    WHEN 'Sweatshirt' THEN
        SET type = 'Shirt'
```

```
WHEN 'Baseball Cap' THEN
    SET type = 'Hat'
WHEN 'Visor' THEN
    SET type = 'Hat'
WHEN 'Shorts' THEN
    SET type = 'Shorts'
ELSE
    SET type = 'UNKNOWN'
END CASE;
END;
```

Im folgenden Beispiel wird Syntax 2 benutzt, um eine Meldung über die Produktmenge in der SQL Anywhere-Beispieldatenbank zu erzeugen.

```
CREATE PROCEDURE StockLevel (IN product_ID INT)
BEGIN
    DECLARE qty INT;
    SELECT Quantity INTO qty FROM GROUPO.Products
    WHERE ID = product_ID;
    CASE
    WHEN qty < 30 THEN
        MESSAGE 'Order Stock' TO CLIENT;
    WHEN qty > 100 THEN
        MESSAGE 'Overstocked' TO CLIENT;
    ELSE
        MESSAGE 'Sufficient stock on hand' TO CLIENT;
    END CASE;
END;
```

CASE-Anweisung [T-SQL]

Wählt einen Ausführungspfad auf der Basis mehrerer Fälle.

Syntax 1

```
CASE value-expression
WHEN [ constant | NULL ] THEN statement-list ...
[ WHEN [ constant | NULL ] THEN statement-list ] ...
[ ELSE statement-list ]
END
```

Syntax 2

```
CASE
WHEN [ search-condition | NULL ] THEN statement-list ...
[ WHEN [ search-condition | NULL ] THEN statement-list ] ...
[ ELSE statement-list ]
END
```

Bemerkungen

Syntax 1 Die CASE-Anweisung ist eine Steueranweisung, mit der Sie eine Liste von SQL-Anweisungen auswählen können, die auf der Grundlage des Wertes eines Ausdrucks ausgeführt werden sollen. Der *value-expression* ist ein Ausdruck, der nur einen Wert annimmt, wobei es sich um eine Zeichenfolge, eine Zahl, ein Datum oder um einen anderen SQL-Datentyp handeln kann. Wenn eine WHEN-Klausel für den Wert von *value-expression* existiert, dann wird die *statement-list* in der WHEN-Klausel ausgeführt. Wenn keine geeignete WHEN-Klausel existiert, es aber eine ELSE-Klausel gibt, dann

wird die *statement-list* in der ELSE-Klausel ausgeführt. Die Ausführung wird bei der ersten Anweisung nach END CASE wieder aufgenommen.

Wenn *value-expression* Null sein kann, ersetzen Sie den *value-expression* NULL mithilfe der Funktion ISNULL durch einen anderen Ausdruck.

Syntax 2 Mit dieser Form werden die Anweisungen für die erste erfüllte *search-condition* in der CASE-Anweisung ausgeführt. Die ELSE-Klausel wird ausgeführt, wenn keine der *search-conditions* erfüllt wird.

Wenn der Ausdruck NULL sein kann, verwenden Sie die folgende Syntax für die erste *search-condition*:

```
WHEN search-condition IS NULL THEN statement-list
```

Hinweis

Verwechseln Sie die Syntax des CASE-Ausdrucks nicht mit der Syntax der CASE-Anweisung.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „ISNULL-Funktion [Verschiedene]“ auf Seite 295
- „BEGIN-Anweisung“ auf Seite 557
- „CASE-Ausdrücke“ auf Seite 25
- „Unbekannte Werte: NULL“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Die CASE-Anweisung ist Teil der Sprachenfunktion P002 (Verarbeitungsvollständigkeit) des SQL/2008-Standards. Der SQL-Standard erfordert END CASE zum Beenden der CASE-Anweisung, nicht END allein.
- **Transact-SQL** Kompatibel mit Adaptive Server Enterprise.

Beispiel

Die folgende Prozedur verwendet eine CASE-Anweisung und klassifiziert die in der Tabelle Products der SQL Anywhere-Beispieldatenbank aufgeführten Produkte in "shirt", "hat", "shorts" oder "unknown".

```
CREATE PROCEDURE DBA.ProductType( @product_ID INTEGER,@TYPE CHAR(10)
OUTPUT ) AS
BEGIN
    DECLARE @prod_name CHAR(20)
    SELECT Name INTO @prod_name FROM GROUPO.Products
    WHERE ID = @product_ID
    IF @prod_name
        = 'Tee Shirt'
```

```
        SET @TYPE = 'Shirt'
    ELSE IF @prod_name
        = 'Sweatshirt'
        SET @TYPE = 'Shirt'
    ELSE IF @prod_name
        = 'Baseball Cap'
        SET @TYPE = 'Hat'
    ELSE IF @prod_name
        = 'Visor'
        SET @TYPE = 'Hat'
    ELSE IF @prod_name
        = 'Shorts'
        SET @TYPE = 'Shorts'
    ELSE
        SET @TYPE = 'UNKNOWN'
END;
```

Im folgenden Beispiel wird Syntax 2 benutzt, um eine Meldung über die Produktmenge in der SQL Anywhere-Beispieldatenbank zu erzeugen.

```
CREATE PROCEDURE DBA.StockLevel( @product_ID INTEGER ) AS
BEGIN
    DECLARE @qty INTEGER
    SELECT Quantity INTO @qty FROM GROUPO.Products
    WHERE ID = @product_ID
    IF @qty < 30
        MESSAGE 'Order Stock' TO CLIENT
    ELSE IF @qty > 100
        MESSAGE 'Overstocked' TO CLIENT
    ELSE
        MESSAGE 'Sufficient stock on hand' TO CLIENT
END;
```

CHECKPOINT-Anweisung

Setzt Checkpoints in der Datenbank.

Syntax

CHECKPOINT

Bemerkungen

Die CHECKPOINT-Anweisung zwingt den Datenbankserver, einen Checkpoint auszuführen. Checkpoints werden vom Datenbankserver entsprechend einem internen Algorithmus auch automatisch ausgeführt. Normalerweise muss eine Anwendung keine CHECKPOINT-Anweisung ausgeben.

Privilegien

Sie müssen das CHECKPOINT-Systemprivileg haben, um an einen Checkpoint in einer Datenbank auszuführen, die auf einem Netzwerkservers (dbsrv) läuft.

Keine Privilegien sind erforderlich, um einen Checkpoint in einer Datenbank auszuführen, die auf einem Personal Datenbankserver (dbeng16) läuft.

Nebenwirkungen

Keine.

Siehe auch

- „Sicherung und Datenwiederherstellung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Checkpoint-Logs“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „checkpoint_time-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „recovery_time-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** Die CHECKPOINT-Anweisung wird von Adaptive Server Enterprise unterstützt.

CLEAR-Anweisung [Interactive SQL]

Schließt alle offenen Ergebnismengen in Interactive SQL.

Syntax

CLEAR

Bemerkungen

Schließt offene Ergebnismengen und lässt die Inhalte des Fensterausschnitts "SQL-Anweisungen" unverändert

Privilegien

Keine.

Nebenwirkungen

Schließt den Cursor, der den nicht mehr angezeigten Daten zugeordnet ist

Siehe auch

- „Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

CLOSE-Anweisung [ESQL] [SP]

Schließt einen Cursor.

Syntax

CLOSE *cursor-name*

cursor-name : *identifier* | *hostvar*

Bemerkungen

Diese Anweisung schließt den benannten Cursor.

Der Cursor muss zuvor geöffnet worden sein.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „OPEN-Anweisung [ESQL] [SP]“ auf Seite 964
- „DECLARE CURSOR-Anweisung [ESQL] [SP]“ auf Seite 778
- „PREPARE-Anweisung [ESQL]“ auf Seite 976

Standards und Kompatibilität

- **SQL/2008** Kernfunktion. In Embedded SQL ist die CLOSE-Anweisung Teil der optionalen Sprachenfunktion B031 (Basic Dynamic SQL).
- **Transact-SQL** Wird von Adaptive Server Enterprise unterstützt.

Beispiel

Die folgenden Beispiele schließen Cursor in Embedded SQL.

```
EXEC SQL CLOSE employee_cursor;  
EXEC SQL CLOSE :cursor_var;
```

Die folgende Prozedur benutzt einen Cursor.

```
CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT TopValue INT)  
BEGIN  
    DECLARE err_notfound EXCEPTION  
        FOR SQLSTATE '02000';  
    DECLARE curThisCust CURSOR FOR  
    SELECT CompanyName, CAST(      sum(SalesOrderItems.Quantity *  
    Products.UnitPrice) AS INTEGER) VALUE  
    FROM GROUPO.Customers  
    LEFT OUTER JOIN SalesOrders  
    LEFT OUTER JOIN SalesOrderItems  
    LEFT OUTER JOIN Products  
    GROUP BY CompanyName;  
    DECLARE ThisValue INT;  
    DECLARE ThisCompany CHAR(35);  
    SET TopValue = 0;  
    OPEN curThisCust;  
    CustomerLoop:  
    LOOP  
        FETCH NEXT curThisCust  
        INTO ThisCompany, ThisValue;  
        IF SQLSTATE = err_notfound THEN  
            LEAVE CustomerLoop;  
        END IF;  
        IF ThisValue > TopValue THEN
```

```

        SET TopValue = ThisValue;
        SET TopCompany = ThisCompany;
    END IF;
END LOOP CustomerLoop;
CLOSE curThisCust;
END

```

COMMENT-Anweisung

Speichert einen Kommentar für ein Datenbankobjekt in den Systemtabellen.

Syntax

```

COMMENT ON {
    COLUMN [ owner.]table-name.column-name
    | CERTIFICATE certificate-name
    | DBSPACE dbspace-name
    | EVENT [ owner.]event-name
    | EXTERNAL ENVIRONMENT environment-name
    | EXTERNAL [ ENVIRONMENT ] OBJECT object-name
    | FOREIGN KEY [ owner.]table-name.key-name
    | INDEX [ [ owner.] table.]index-name
    | INTEGRATED LOGIN integrated-login-id
    | JAVA CLASS java-class-name
    | JAVA JAR java-jar-name
    | KERBEROS LOGIN "client-Kerberos-principal"
    | LDAP SERVER ldapua-server-name
    | LOGIN POLICY policy-name
    | MATERIALIZED VIEW [ owner.]materialized-view-name
    | MIRROR SERVER mirror-server-name
    | PRIMARY KEY ON [ owner.]table-name
    | PROCEDURE [ owner.]procedure-name
    | PUBLICATION [ owner.] publication-name
    | REMOTE MESSAGE TYPE remote-message-type-name
    | ROLE role-name
    | SEQUENCE sequence-name
    | SERVICE web-service-name
    | SPATIAL REFERENCE SYSTEM srs-name
    | SPATIAL UNIT OF MEASURE uom-identifier
    | SYNCHRONIZATION PROFILE synchronization-profile-name
    | TABLE [ owner.]table-name
    | TEXT CONFIGURATION [ owner.]text-config-name
    | TEXT INDEX text-index-name ON [ owner.]table-name
    | TRIGGER [ [ owner.]tablename.]trigger-name
    | USER userid
    | VIEW [ owner.]view-name
}
IS comment

comment : string | NULL

environment-name :
    JAVA
    | PERL
    | PHP
    | CLR

```

C_ESQL32
C_ESQL64
C_ODBC32
C_ODBC64

Bemerkungen

Mit der COMMENT-Anweisung können Sie eine Bemerkung (Kommentar) für ein Objekt in der Datenbank setzen. Die COMMENT-Anweisung aktualisiert Bemerkungen, die in der ISYSREMARKS-Systemtabelle aufgelistet sind. Sie können einen Kommentar entfernen, indem Sie diesen auf NULL setzen. Um einen Index oder einen Trigger zu kommentieren, muss der Eigentümer des Kommentars Eigentümer der Tabelle sein, für die der Index oder der Trigger definiert ist.

Sie können keine Kommentare für lokale temporäre Tabellen hinzufügen.

Wenn Sie den **Assistenten zum Erstellen der Datenbankdokumentation** zur Dokumentation Ihrer SQL Anywhere-Datenbank verwenden, haben Sie die Option, die Kommentare für Prozeduren, Funktionen, Trigger, Ereignisse und Ansichten in die Dokumentation aufzunehmen.

Privilegien

Wenn Sie das COMMENT ANY OBJECT-Systemprivileg haben, können Sie jedes Objekt kommentieren, das Sie mit dem CREATE ANY OBJECT-Systemprivileg erstellen können. Wenn Sie nicht das COMMENT ANY OBJECT-Systemprivileg haben, benötigen Sie die äquivalenten Privilegien gemäß der folgenden Erläuterung:

- Bei Datenbankobjekten muss mindestens eine der folgenden Angaben zutreffen:
 - Sie sind der Eigentümer des Objekts.
 - Sie haben die Möglichkeit, Objekte desselben Typs zu erstellen oder zu ändern, deren Eigentümer andere Benutzer sind (z.B. CREATE ANY TABLE oder ALTER ANY OBJECT).
 - Sie haben die Möglichkeit, Objekte dieses Typs zu verwalten (z.B. MANAGE ANY USER).
- Bei Systemrollen müssen Sie Administrationsrechte für die Rolle haben.
- Bei benutzerdefinierten Rollen müssen Sie entweder das MANAGE ROLES-Systemprivileg oder Administrationsrechte für die jeweilige Rolle haben.
- Bei Kerberos- oder integrierten Logins müssen Sie das MANAGE ANY USER-Systemprivileg haben.
- Bei Java-Klassen oder JAR-Dateien müssen Sie das MANAGE ANY EXTERNAL OBJECT-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

- **Transact-SQL** Wird von Adaptive Server Enterprise nicht unterstützt.

Beispiel

Die folgenden Beispiele zeigen, wie Sie einen Kommentar hinzufügen und entfernen können:

1. Ein Kommentar wird zur Tabelle Employees hinzugefügt.

```
COMMENT ON TABLE GROUPO.Employees
IS 'Employee information';
```

2. Ein Kommentar wird aus der Tabelle Employees entfernt.

```
COMMENT
ON TABLE GROUPO.Employees
IS NULL;
```

Um den für ein Objekt festgelegten Kommentar anzuzeigen, verwenden Sie eine SELECT-Anweisung wie im folgenden Beispiel. Hier wird der Kommentar für die ViewSalesOrders-Ansicht in der SQL Anywhere-Beispieldatenbank abgerufen.

```
SELECT remarks
FROM SYSTAB t, SYSREMARK r
WHERE t.object_id = r.object_id
AND t.table_name = 'ViewSalesOrders';
```

Siehe auch

- „Datenbank dokumentieren“ [[SQL Anywhere Server - Datenbankadministration](#)]

COMMIT-Anweisung

Macht Änderungen in der Datenbank permanent oder beendet eine benutzerdefinierte Transaktion.

Syntax 1

```
COMMIT [ WORK ]
```

Syntax 2

```
COMMIT TRAN[SACTION] [ transaction-name ]
```

Parameter

transaction-name Ein optionaler Name, der dieser Transaktion zugewiesen wurde. Er muss ein gültiger Bezeichner sein. Verwenden Sie Transaktionsnamen nur für das äußerste Paar von verschachtelten BEGIN/COMMIT-oder BEGIN/ROLLBACK-Anweisungen.

Weitere Hinweise zum Verschachteln von Transaktionen in Adaptive Server Enterprise und SQL Anywhere finden Sie unter „[BEGIN TRANSACTION-Anweisung \[T-SQL\]](#)“ auf Seite 561.

Sie können eine Reihe von Optionen verwenden, um das Verhalten der COMMIT-Anweisung im Detail zu kontrollieren. Siehe:

- „cooperative_commit_timeout-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „cooperative_commits-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „delayed_commits-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „delayed_commit_timeout-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Permanente Datenänderungen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Mit der Verbindungseigenschaft "Commit" können Sie die Anzahl der Festschreibungen in der aktuellen Verbindung zurückgeben.

Bemerkungen

- **Syntax 1** Die COMMIT-Anweisung beendet eine Transaktion und macht alle Änderungen in der Datenbank dauerhaft, die während dieser Transaktion gemacht wurden.

Alle Datendefinitionsanweisungen schreiben automatisch fest. Weitere Hinweise finden Sie in der Rubrik "Nebenwirkungen" für jede SQL-Anweisung.

Die COMMIT-Anweisung schlägt fehl, wenn der Datenbankserver einen ungültigen Fremdschlüssel feststellt. Aufgrund dieses Verhaltens kann eine Transaktion mit einem ungültigen Fremdschlüssel nicht beendet werden. Im Allgemeinen wird Fremdschlüsselintegrität bei jeder Datenmanipulationsoperation überprüft. Wenn jedoch die Datenbankoption wait_for_commit auf "On" gesetzt ist oder ein bestimmter Fremdschlüssel durch eine CHECK ON COMMIT-Option definiert wurde, überprüft der Datenbankserver die Integrität erst, wenn die COMMIT-Anweisung ausgeführt wird.

Die Verwendung von COMMIT allein ist gleichwertig mit COMMIT WORK.

- **Syntax 2** Sie können die Anweisungen BEGIN TRANSACTION und COMMIT TRANSACTION paarweise verwenden, um verschachtelte Transaktionen zu erstellen. Verschachtelte Transaktionen haben Ähnlichkeit mit Savepoints. Wenn sie als äußerste von einer Reihe verschachtelter Transaktionen ausgeführt wird, macht die Anweisung Änderungen in der Datenbank dauerhaft. Wenn die COMMIT TRANSACTION-Anweisung innerhalb einer Transaktion ausgeführt wird, führt das zu einer Herabsetzung der Verschachtelungsebene um eins. Bei verschachtelten Transaktionen werden die Änderungen an der Datenbank nur durch die äußerste COMMIT-Anweisung permanent.

Syntax 2 ist eine Erweiterung von Transact-SQL.

In Interactive SQL können Sie eine COMMIT-Anweisung auch folgendermaßen ausführen:

- Drücken Sie Strg+Umschalttaste+C.
- Klicken Sie auf **SQL » Festschreiben**.

In Interactive SQL werden durch das Ausführen einer COMMIT-Anweisung über das **SQL**-Menü oder ein Tastenkürzel nicht die Inhalte des Fensterausschnitts **SQL-Anweisungen** geändert. Die Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** wird aber bereinigt.

Privilegien

Keine.

Nebenwirkungen

Schließt jeden Cursor, der nicht mit WITH HOLD geöffnet wurde

Löscht alle Zeilen deklarierter temporärer Tabellen in dieser Verbindung, es sei denn, sie wurden mit ON COMMIT PRESERVE ROWS deklariert

Wenn die Datenbank kein Transaktionslog verwendet, verursacht jeder COMMIT-Vorgang einen impliziten Checkpoint.

Siehe auch

- „wait_for_commit-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „auto_commit-Option [Interactive SQL]“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „commit_on_exit-Option [Interactive SQL]“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Tastenkürzel für Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „SAVEPOINT-Anweisung“ auf Seite 1019
- „Liste der Verbindungseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „BEGIN TRANSACTION-Anweisung [T-SQL]“ auf Seite 561
- „PREPARE TO COMMIT-Anweisung“ auf Seite 979
- „ROLLBACK-Anweisung“ auf Seite 1015

Standards und Kompatibilität

- **SQL/2008** Syntax 1 ist eine Kernfunktion. Syntax 2 ist eine Erweiterung von Transact-SQL.

Beispiel

Die folgende Anweisung schreibt die aktuelle Transaktion fest:

```
COMMIT;
```

Der folgende Transact-SQL-Batch meldet aufeinanderfolgende Werte von @@trancount als "0", "1", "2", "1", "0".

```
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
COMMIT TRANSACTION
PRINT @@trancount
COMMIT TRANSACTION
PRINT @@trancount
go
```

CONFIGURE-Anweisung [Interactive SQL]

Öffnet das Fenster **Optionen** von Interactive SQL.

Syntax

CONFIGURE

Bemerkungen

Mit der CONFIGURE-Anweisung wird das Interactive SQL-Fenster **Optionen** geöffnet. Dieses Fenster zeigt die aktuellen Einstellungen aller Interactive SQL-Optionen an. Datenbankoptionen können dort jedoch weder angezeigt noch geändert werden. Sie können dieses Fenster verwenden, um Interactive SQL zu konfigurieren.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „Interactive SQL anpassen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „SET OPTION-Anweisung“ auf Seite 1039
- „Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

CONNECT-Anweisung [ESQL] [Interactive SQL]

Stellt eine Verbindung zu einer Datenbank her.

Syntax 1 - Shared Memory-Verbindungen

CONNECT

```
[ TO database-server-name ]  
[ DATABASE database-name ]  
[ AS connection-name ]  
[ USER ] userid [ IDENTIFIED BY password ]
```

database-server-name, database-name, connection-name, userid, password :
{ *identifier* | *string* | *hostvar* }

Syntax 2 - TCP/IP-Verbindungen

CONNECT USING connect-string

connect-string : { *identifier* | *string* | *hostvar* }

Parameter

AS-Klausel Eine Verbindung kann wahlweise auch durch die Angabe einer AS-Klausel benannt werden. Dadurch können Sie mehrere Verbindungen zu einer Datenbank bzw. zu einem oder verschiedenen Datenbankservern gleichzeitig herstellen. Jede Verbindung besitzt ihre eigene zugeordnete

Transaktion. Es können sogar Sperrenkonflikte zwischen Ihren Transaktionen auftreten, wenn Sie beispielsweise versuchen, denselben Datensatz in einer Datenbank von zwei verschiedenen Verbindungen aus zu ändern.

Eine *connect-string* ist eine Liste von Parametereinstellungen in Form von Schlüsselwort=Wert, wobei die Werte durch Semikola getrennt sind. Sie müssen außerdem in Apostrophe eingeschlossen werden.

Bemerkungen

Die CONNECT-Anweisung stellt eine Verbindung zur Datenbank her, die durch *database-name* gekennzeichnet wird und auf dem mit *database-server-name* bezeichneten Datenbankserver läuft. Diese Anweisung wird in Prozeduren, Triggern, Ereignissen oder Batches nicht unterstützt.

Syntax 1 wird nur unterstützt für Shared Memory-Verbindungen mit Datenbankservern, die auf demselben Computer ausgeführt werden. Wenn Sie eine Verbindung mit einem lokalen Datenbankserver über TCP/IP oder mit einem Datenbankserver auf einem anderen Computer herstellen möchten, müssen Sie Syntax 2 verwenden.

- **Verhalten in Embedded SQL** Wenn der *database-server-name* nicht angegeben ist, wird in Embedded SQL der vorgegebene lokale Datenbankserver angenommen (der erste Datenbankserver, der gestartet wird). Wenn kein *database-name* angegeben ist, wird die erste Datenbank auf dem vorgegebenen Server angenommen.

Die Anweisungen WHENEVER, SET SQLCA und einige DECLARE-Anweisungen erzeugen keinen Code und können daher in der Quelldatei vor der CONNECT-Anweisung erscheinen. Andernfalls werden keine anderen Anweisungen zugelassen, bis eine CONNECT-Anweisung ausgeführt wurde.

Die Benutzer-ID und das Kennwort werden zur Überprüfung der Privilegien für alle Dynamic SQL-Anweisungen verwendet.

Hinweis

Bei SQL Anywhere ist nur Syntax 1 mit Embedded SQL gültig. Bei UltraLite können Syntax 1 und Syntax 2 mit Embedded SQL verwendet werden.

- **Verhalten in Interactive SQL** Wenn in der CONNECT-Anweisung weder Datenbank noch Server angegeben sind, hält Interactive SQL die Verbindung zur aktuellen Datenbank aufrecht und gibt weder Server noch Datenbank vor. Wenn der Name einer Datenbank ohne den Namen des Servers angegeben wird, versucht Interactive SQL, auf dem aktuellen Server eine Verbindung zur angegebenen Datenbank aufzubauen. Wenn der Name eines Servers ohne den Namen einer Datenbank angegeben wird, versucht Interactive SQL auf dem angegebenen Server eine Verbindung zur Standarddatenbank herzustellen.

Wenn z.B. der unten gezeigte Batch bei einer Datenbankverbindung ausgeführt wird, werden die beiden Tabellen in derselben Datenbank erstellt.

```
CREATE TABLE t1( c1 int );
CONNECT DBA IDENTIFIED BY sql;
CREATE TABLE t2 (c1 int );
```

Es werden keine anderen Datenbank-Anweisungen zugelassen, bis die CONNECT-Anweisung ausgeführt wurde.

Wenn Interactive SQL im Befehlsfenster-Modus ausgeführt wird, werden Sie zur Eingabe ggf. fehlender Verbindungsparameter aufgefordert.

Wenn Sie Interactive SQL im Befehlszeilenmodus (-nogui ist angegeben, wenn Sie Interactive SQL von einer Eingabeaufforderung aus starten) oder im Batchmodus ausführen, oder wenn Sie CONNECT ohne AS-Klausel ausführen, wird eine unbenannte Verbindung geöffnet. Wenn bereits eine unbenannte Verbindung geöffnet wurde, wird die alte automatisch geschlossen. Andernfalls werden vorhandene Verbindungen nicht geschlossen, wenn Sie eine CONNECT-Anweisung ausführen.

Mehrere Verbindungen können über das Konzept der aktuellen Verbindung verwaltet werden. Nach einer erfolgreichen Verbindungsanweisung wird aus der neuen Verbindung die aktuelle. Mit der SET CONNECTION-Anweisung können Sie zu einer anderen Verbindung wechseln. Die DISCONNECT-Anweisung wird zum Abbrechen von Verbindungen verwendet.

Bei der Erstellung einer Verbindung mit Interactive SQL entspricht die Angabe von CONNECT [USER] *userid* der Ausführung einer SETUSER WITH OPTION *userid*-Anweisung.

In Interactive SQL erscheinen die Verbindungsinformationen (der Datenbankname, Ihre Benutzer-ID und der Datenbankserver) in der Titelleiste über dem Fensterausschnitt mit den SQL-Anweisungen. Wenn keine Verbindung zu einer Datenbank besteht, erscheint "Nicht verbunden" in der Titelleiste.

Hinweis

Syntax 1 und Syntax 2 sind beide in Interactive SQL gültig, außer wenn Interactive SQL das *hostvar*-Argument nicht unterstützt.

Diese SQL-Anweisung wird nicht für SAP HANA-Datenbanken unterstützt.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „GRANT CONNECT-Anweisung“ auf Seite 888
- „DISCONNECT-Anweisung [ESQL] [Interactive SQL]“ auf Seite 802
- „SET CONNECTION-Anweisung [Interactive SQL] [ESQL]“ auf Seite 1032
- „SETUSER-Anweisung“ auf Seite 1059
- „Fehlerbehandlung: Verbindungen“ [SQL Anywhere Server - Datenbankadministration]
- „Verbindungsparameter“ [SQL Anywhere Server - Datenbankadministration]
- „Interactive SQL“ [SQL Anywhere Server - Datenbankadministration]

Standards und Kompatibilität

- **SQL/2008** Syntax 1 ist die optionale Sprachenfunktion F771 des SQL/2008-Standards. Syntax 2 ist eine Erweiterung des Herstellers.

- **Transact-SQL** Sowohl Syntax 1 als auch Syntax 2 werden von Adaptive Server Enterprise unterstützt.

Beispiele

Die folgenden Beispiele zeigen, wie Sie CONNECT in Embedded SQL verwenden können.

```
EXEC SQL CONNECT AS :conn_name
USER :userid IDENTIFIED BY :password;
EXEC SQL CONNECT USER "DBA" IDENTIFIED BY "sql";
```

In den folgenden Beispielen wird davon ausgegangen, dass die SQL Anywhere-Beispieldatenbank bereits gestartet wurde.

Stellen Sie eine Verbindung zu einer Datenbank in Interactive SQL her. Interactive SQL fordert zur Eingabe einer Benutzer-ID und eines Kennworts auf.

```
CONNECT;
```

Stellen Sie über Interactive SQL eine Verbindung mit der Standard-Datenbank als Benutzer DBA her. Interactive SQL fordert zur Eingabe eines Kennworts auf.

```
CONNECT USER "DBA";
```

Hier wird eine Verbindung zur Beispieldatenbank als Benutzer DBA in Interactive SQL hergestellt.

```
CONNECT
TO demo16
USER DBA
IDENTIFIED BY sql;
```

Hier wird eine Verbindung zur Beispieldatenbank unter Verwendung einer Verbindungszeichenfolge in Interactive SQL hergestellt.

```
CONNECT
USING 'UID=DBA;PWD=sql;DBN=demo';
```

CONTINUE-Anweisung

Startet eine Schleife neu.

Syntax

```
CONTINUE [ statement-label ]
```

Bemerkungen

Die CONTINUE-Anweisung ist eine Steueranweisung, die das Neustarten einer Schleife ermöglicht. Die Ausführung wird bei der ersten Anweisung in der Schleife fortgeführt. Wenn CONTINUE in einer Gruppe von Watcom-SQL-Anweisungen auftritt, muss *statement-label* angegeben werden.

Wenn CONTINUE in einer Gruppe von Anweisungen auftritt, welche Transact-SQL verwenden, muss *statement-label* nicht angegeben werden.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „LOOP-Anweisung“ auf Seite 950
- „WHILE-Anweisung [T-SQL]“ auf Seite 1123
- „FOR-Anweisung“ auf Seite 857
- „BEGIN-Anweisung“ auf Seite 557
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** CONTINUE ohne ein Anweisungslabel wird von Adaptive Server Enterprise unterstützt.

Beispiel

Das folgende Fragment zeigt, wie die CONTINUE-Anweisung verwendet wird, um eine Schleife neu zu starten. Das Beispiel zeigt die ungeraden Zahlen zwischen 1 und 10 an.

```
BEGIN
  DECLARE i INT;
  SET i = 0;
  lbl:
  WHILE i < 10 LOOP
    SET i = i + 1;
    IF mod( i, 2 ) = 0 THEN
      CONTINUE lbl
    END IF;
    MESSAGE 'The value ' || i || ' is odd.' TO CLIENT;
  END LOOP lbl;
END
```

CREATE CERTIFICATE-Anweisung

Fügt aus der angegebenen Datei oder Zeichenfolge ein Zertifikat in der Datenbank hinzu oder ersetzt es. Wenn Sie ein Zertifikat erstellen möchten, verwenden Sie das Dienstprogramm zum Erstellen von Zertifikaten (createcert).

Syntax

```
CREATE [ OR REPLACE ] CERTIFICATE certificate-name
FROM { certificate-string | variable-name | FILE file-name }
```

Parameter

FROM-Klausel Diese Klausel gibt eine Datei, Zeichenfolge oder Variable an, die ein Zertifikat enthält.

Bemerkungen

Die CREATE CERTIFICATE-Anweisung fügt aus der angegebenen Datei, Zeichenfolge oder Variablen ein Zertifikat in der Datenbank hinzu oder ersetzt es. Die Datei, Zeichenfolge oder Variable muss entweder ein Zertifikat im binären DER-Format oder ein Zertifikat im PEM-Textformat enthalten. Zertifikate im DER-Format werden konvertiert und als PEM-Zertifikate gespeichert.

In der Datenbank gespeicherte Zertifikate können von Webdienstprozeduren und -funktionen verwendet werden, die sichere HTTPS-Verbindungen mit einem Webserver herstellen. Außerdem können sie verwendet werden, um sichere Nachrichten mithilfe der xp_startsmtp-Systemprozedur zu senden.

Wenn Sie ein Zertifikat hinzufügen, wird es zur ISYSCERTIFICATE-Systemtabelle hinzugefügt. Verwenden Sie die entsprechende Systemansicht, SYSCERTIFICATE, um die Tabelle anzuzeigen.

Die CREATE CERTIFICATE-Anweisung dient nicht dazu, tatsächlich ein Zertifikat zu erstellen. Verwenden Sie dazu das Dienstprogramm zum Erstellen von Zertifikaten (createcert).

Privilegien

Sie müssen das MANAGE CERTIFICATES-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „DROP CERTIFICATE-Anweisung“ auf Seite 803
- „CREATE PROCEDURE-Anweisung [Webdienste]“ auf Seite 670
- „CREATE FUNCTION-Anweisung [Webdienst]“ auf Seite 624
- „xp_startsmtp-Systemprozedur“ auf Seite 1431
- „SYSCERTIFICATE-Systemansicht“ auf Seite 1440
- „sa_certificate_info-Systemprozedur“ auf Seite 1170
- „Dienstprogramm zum Erstellen von Zertifikaten [createcert]“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird ein Zertifikat namens mycert in der Datenbank erstellt, wobei der Inhalt der angegebenen Zertifikatdatei verwendet wird.

```
CREATE CERTIFICATE mycert
FROM FILE 'C:\\Users\\Public\\Documents\\SQL Anywhere 16\\Samples\\
\\Certificates\\rsaroot.crt';
```

CREATE DATABASE-Anweisung

Erstellt eine Datenbank.

Syntax

CREATE DATABASE *db-filename-string* [*create-option ...*]

create-option :

```
[ ACCENT { RESPECT | IGNORE | FRENCH } ]
[ ASE [ COMPATIBLE ] ]
[ BLANK PADDING { ON | OFF } ]
[ CASE { RESPECT | IGNORE } ]
[ CHECKSUM { ON | OFF } ]
[ COLLATION collation-label [ ( collation-tailoring-string ) ] ]
[ DATABASE SIZE size { KB | MB | GB | PAGES | BYTES } ]
[ DBA USER userid ]
[ DBA PASSWORD password ]
[ ENCODING encoding-label ]
[ ENCRYPTED [ TABLE ] { algorithm-key-spec | OFF } ]
[ JCONNECT { ON | OFF } ]
[ MIRROR mirror-filename-string ]
[ PAGE SIZE page-size ]
[ NCHAR COLLATION nchar-collation-label [ ( collation-tailoring-string ) ] ]
[ SYSTEM PROCEDURE AS DEFINER { ON | OFF } ]
[ [ TRANSACTION ] { LOG OFF | LOG ON [ log-filename-string ] ]
```

page-size :

2048 | 4096 | 8192 | 16384 | 32768

algorithm-key-spec :

```
ON
| [ ON ] KEY key [ ALGORITHM AES-algorithm ]
| [ ON ] ALGORITHM AES-algorithm KEY key
| [ ON ] ALGORITHM 'SIMPLE'
```

AES-algorithm :

'AES' | 'AES256' | 'AES_FIPS' | 'AES256_FIPS'

key: Zeichenfolge in Anführungszeichen

Parameter

CREATE DATABASE Die Dateinamen (*db-filename-string*, *log-filename-string* und *mirror-filename-string*) sind Zeichenfolgen, die Betriebssystem-Dateinamen enthalten. Als Literal-Zeichenfolgen müssen sie von Apostrophen umschlossen sein.

Hinweis

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

- Wenn Sie einen Suchpfad angeben, muss jedes Backslashzeichen (\), auf das ein n oder ein x folgt, verdoppelt werden. Dadurch wird vermieden, dass es als Zeilenendmarke (\n) oder als Hexadezimalzahl (\x) entsprechend den Regeln für Zeichenfolgen in SQL betrachtet wird.

In den folgenden Beispielen wird gezeigt, wo dies wichtig ist.

```
CREATE DATABASE 'c:\\temp\\\\x41\\x42\\x43xyz.db'
DBA USER 'DBA' DBA PASSWORD 'sql';
```

Die anfängliche Zeichenfolge `\\` steht für einen Backslash. Die Zeichenfolge `\x`, gefolgt von zwei Ziffern, steht für die Buchstaben "A", "B" bzw. "C". Der Dateiname lautet hier `ABCxyz.db`.

```
CREATE DATABASE 'c:\temp\\nest.db'
DBA USER 'DBA' DBA PASSWORD 'sql';
```

Bei der Zeichenfolge `\n` wird der Backslash doppelt angegeben, damit diese nicht als Zeilenumbruchzeichen interpretiert wird. Siehe [Escapesequenzen auf Seite 8](#).

Es ist immer sicherer, dem Backslashzeichen ein Escapezeichen voranzustellen. Beispiel:

```
CREATE DATABASE 'c:\\my_db.db'
DBA USER 'DBA' DBA PASSWORD 'sql'
LOG ON 'e:\\logdrive\\my_db.log';
```

- Wenn Sie keinen Suchpfad oder nur einen relativen Suchpfad angeben, wird die Datenbank relativ zum Arbeitsverzeichnis des Datenbankservers erstellt. Wenn Sie keinen Suchpfad für eine Transaktionslogdatei angeben, wird die Datei in demselben Verzeichnis erstellt wie die Datenbankdatei. Es wird empfohlen, dass Sie die Datenbankdatei und das Transaktionslog auf getrennten Datenträgern auf dem Computer speichern.
- Wenn Sie keine Dateierweiterung angeben, wird eine Datei mit der Erweiterung `.db` für Datenbanken, `.log` für das Transaktionslog und `.mlg` für den Transaktionslogspiegel erstellt.
- Der Verzeichnispfad ist relativ zum Datenbankserver.

Sie können nicht `utility_db` als *db-filename-string* eingeben. Dieser Name ist für die Dienstprogrammdateiabank reserviert.

ACCENT-Klausel Diese Klausel wird benutzt, um die Berücksichtigung von Akzenten in der Datenbank festzulegen. Die Unterstützung für diese Klausel ist veraltet. Verwenden Sie die Optionen der Kollationsanpassung in den Klauseln `COLLATION` und `NCHAR COLLATION`, um die Berücksichtigung von Akzenten festzulegen.

Die `ACCENT`-Klausel gilt nur, wenn der UCA-Algorithmus (Unicode Collation Algorithm) für die Kollation verwendet wird, die in der `COLLATION`- oder `NCHAR COLLATION`-Klausel angegeben ist. `ACCENT RESPECT` bewirkt, dass beim UCA-Zeichenfolgenvergleich Akzentunterschiede bei den Buchstaben beachtet werden. Beispiel: "e" ist weniger als "é". `ACCENT FRENCH` ist ähnlich wie `ACCENT RESPECT`, nur dass Akzente von rechts nach links verglichen werden, entsprechend den Regeln der französischen Sprache. `ACCENT IGNORE` bewirkt, dass Akzente bei Zeichenfolgenvergleichen ignoriert werden. Beispiel: "e" ist gleich "é".

Wenn die Berücksichtigung von Akzenten beim Erstellen der Datenbank nicht aktiviert wurde, ist der Standardwert für die Berücksichtigung von Akzenten für Vergleiche und Sortierungen auf *Insensitiv* eingestellt, wobei aber eine Ausnahme gilt: Bei japanischen Datenbanken, die mit der UCA-Kollation erstellt wurden, wird die Berücksichtigung von Akzenten standardmäßig auf *Sensitiv* eingestellt.

ASE COMPATIBLE-Klausel Erstellen Sie weder `SYS.SYSCOLUMNS`- noch `SYS.SYSEXES`-Ansichten. Standardmäßig werden diese Ansichten aus Kompatibilitätsgründen mit Systemtabellen erstellt, die in Watcom SQL (Softwareversion 4 und früher) verfügbar sind. Diese Anzeigen kollidieren mit den Adaptive Server Enterprise-Kompatibilitätsansichten `dbo.syscolumns` und `dbo.sysindexes`.

BLANK PADDING-Klausel SQL Anywhere vergleicht alle Zeichenfolgen, als ob sie eine variable Länge hätten und unter Verwendung der VARCHAR-Domäne gespeichert würden. Dies schließt Zeichenfolgenvergleiche ein, die CHAR- oder NCHAR-Spalten mit fester Länge betreffen. Überdies kürzt SQL Anywhere niemals Werte oder füllt sie mit nachgestellten Leerzeichen auf, wenn die Werte in der Datenbank gespeichert werden.

Standardmäßig behandelt SQL Anywhere Leerzeichen als signifikante Zeichen. Daher ist der Wert "a" (das Zeichen "a", gefolgt von einem Leerzeichen) nicht mit der Ein-Zeichen-Zeichenfolge "a" äquivalent. Ungleichheits-Vergleiche behandeln ebenfalls ein Leerzeichen wie jedes andere Zeichen in der Kollation.

Wenn das Auffüllen mit Leerzeichen aktiviert ist (durch Angabe von PADDING ON), befolgt die Semantik von Zeichenfolgenvergleichen eher den ANSI/ISO SQL-Standard. Bei aktiviertem Auffüllen mit Leerzeichen ignoriert SQL Anywhere in einem Vergleich nachgestellte Leerzeichen.

Im obenstehenden Beispiel gibt ein Gleichheitsvergleich von "a" mit "a" in einer mit Leerzeichen aufgefüllten Datenbank TRUE zurück. In einer mit Leerzeichen aufgefüllten Datenbank werden Zeichenfolgen fester Länge mit Leerzeichen aufgefüllt, wenn sie von einer Anwendung abgerufen werden. Ob die Anwendung bei so einer Zuordnung eine Zeichenfolge-Kürzungswarnung erhält, hängt von der Einstellung der Verbindungsoption `ansi_blanks` ab.

CASE-Klausel Diese Klausel wird benutzt, um die Berücksichtigung von Groß- und Kleinschreibung in der Datenbank festzulegen. Die Unterstützung für diese Klausel ist veraltet. Verwenden Sie die Optionen der Kollationsanpassung in den Klauseln COLLATION und NCHAR COLLATION, um die Berücksichtigung von Groß- und Kleinschreibung festzulegen.

CASE RESPECT bewirkt, dass die Groß-/Kleinschreibung bei Zeichenfolgenvergleichen in Bezug auf alle CHAR- und NCHAR-Datentypen berücksichtigt wird. Vergleiche mit UCA berücksichtigen die Groß-/Kleinschreibung eines Buchstabens nur, wenn die Basisbuchstaben und Akzente gleich sind. Bei allen anderen Kollationen wird zwischen Groß- und Kleinbuchstaben unterschieden. So ist beispielsweise a kleiner als A, dies wiederum kleiner als b, und so weiter. CASE IGNORE bewirkt Zeichenfolgenvergleiche ohne Berücksichtigung der Groß-/Kleinschreibung. Groß- und Kleinbuchstaben werden als exakt gleich angesehen.

Wenn die Berücksichtigung von Groß- und Kleinschreibung beim Erstellen der Datenbank nicht aktiviert wurde, ist der Standardwert für die Berücksichtigung von Groß- und Kleinschreibung für Vergleiche und Sortierungen auf *Insensitiv* eingestellt, wobei aber eine Ausnahme gilt: Bei japanischen Datenbanken, die mit der UCA-Kollation erstellt wurden, wird die Berücksichtigung von Groß- und Kleinschreibung standardmäßig auf *Sensitiv* eingestellt.

CASE RESPECT ist zur Gewährleistung der Kompatibilität mit dem ISO/ANSI SQL-Standard vorgesehen. Bezeichner in der Datenbank berücksichtigen die Groß-/Kleinschreibung niemals, selbst in Datenbanken nicht, in denen die Groß-/Kleinschreibung beachtet wird.

CHECKSUM-Klausel Mit Prüfsummen kann festgestellt werden, ob eine Datenbankseite auf der Festplatte geändert wurde. Wenn Sie eine Datenbank mit aktivierten globalen Prüfsummen erstellen, berechnet das System eine Prüfsumme, direkt bevor eine Datenbankseite auf die Festplatte geschrieben wird. Wenn die Seite von der Festplatte gelesen wird, ermittelt das System die Prüfsumme erneut und vergleicht sie mit dem auf der Seite gespeicherten Wert. Wenn die Prüfsummen unterschiedlich sind,

wurde die Seite auf der Festplatte geändert und ein Fehler tritt auf. Datenbanken, die mit aktivierten globalen Prüfsummen erstellt wurden, können auch mithilfe von Prüfsummen validiert werden. Sie können überprüfen, ob eine Datenbank mit aktivierten globalen Prüfsummen erstellt wurde, indem Sie die folgende Anweisung ausführen:

```
SELECT DB_PROPERTY ( 'Checksum' );
```

Diese Abfrage gibt ON zurück, wenn globale Prüfsummen aktiviert sind, sonst wird OFF zurückgegeben. Globale Prüfsummen sind standardmäßig aktiviert. Wird die CHECKSUM-Klausel weggelassen, wird also ON angewendet.

Ungeachtet der Einstellung dieser Klausel aktiviert der Datenbankserver das Schreiben von Prüfsummen immer für Datenbanken, die auf Speichermedien wie beispielsweise Wechseldatenträgern laufen, sowie für Datenbanken, die unter Windows Mobile ausgeführt werden, damit eine frühe Erkennung von Beschädigungen der Datenbankdatei möglich ist. Außerdem berechnet der Datenbankserver während der Validierung Prüfsummen für entscheidende Seiten.

Bei Datenbanken, für die keine globalen Prüfsummen aktiviert wurden, können Sie das Schreiben von Prüfsummen unter Verwendung der Optionen für -wc aktivieren.

COLLATION-Klausel Die durch die COLLATION-Klausel angegebene Kollation wird für die Sortierung und den Vergleich von Zeichendatentypen verwendet (CHAR, VARCHAR und LONG VARCHAR). Die Kollation liefert Zeichenvergleichs- und Sortierinformationen für die verwendete Kodierung (Zeichensatz). Wenn die COLLATION-Klausel nicht angegeben ist, wählt SQL Anywhere eine Kollation basierend auf der Betriebssystemsprache und Kodierung aus.

Die Kollation kann aus der Liste der Kollationen ausgewählt werden, die den SQL Anywhere Collation Algorithm (SACA) verwenden, oder eine UCA-Kollation (Unicode Collation Algorithm) sein. Wenn UCA angegeben wird, sollten Sie auch die ENCODING-Klausel angeben.

Es ist wichtig, dass Sie Ihre Kollation mit Umsicht auswählen. Sie kann nicht mehr geändert werden, nachdem die Datenbank erstellt wurde.

Optional können Sie Optionen der Kollationsanpassung (*collation-tailoring-string*) für eine zusätzliche Steuerung bei Zeichensortierungen und -vergleichen angeben. Diese Optionen werden in der Form von Schlüsselwort=Wert-Paaren, die in Klammern gesetzt werden, hinter dem Kollationsnamen angegeben. Beispiel: ... CHAR COLLATION 'UCA(locale=es;case=respect;accent=respect) '.

DATABASE SIZE-Klausel Verwenden Sie diese optionale Klausel, um die Anfangsgröße der Datenbankdatei festzulegen. Verwenden Sie KB, MB, GB oder PAGES, um die Einheit in kB, MB, GB bzw. Seiten anzugeben.

Das Angeben der Dateigröße zum Zeitpunkt der Erstellung ist eine Möglichkeit, Speicherplatz für die Datei vorab zuzuweisen. Dies dient dazu, das Risiko der Speicherplatzknappheit auf dem Laufwerk zu vermindern, auf dem die Datenbank läuft. Überdies kann die Performance verbessert werden, indem die Menge der Daten erhöht wird, die in der Datenbank gespeichert werden können, bevor der Datenbankserver die Datenbank vergrößern muss, was ein zeitaufwändiger Vorgang sein kann.

Klauseln DBA USER und DBA PASSWORD Verwenden Sie diese Klauseln, um eine DBA-Benutzer-ID und ein DBA-Kennwort für die Datenbank anzugeben. Wenn Sie diese Klauseln nicht

angeben, werden Sie zum Zeitpunkt der Erstellung aufgefordert, die DBA-Benutzer-ID und das DBA-Kennwort anzugeben.

- Benutzer-IDs dürfen Folgendes nicht:
 - Sie dürfen nicht mit Leerstellen, Apostrophen oder Anführungszeichen beginnen.
 - Sie dürfen nicht mit Leerstellen enden.
 - Sie dürfen keine Semikola enthalten.
- Kennwörter berücksichtigen die Groß- und Kleinschreibung. Im Übrigen gilt Folgendes:
 - Sie dürfen nicht mit Leerstellen, Apostrophen oder Anführungszeichen beginnen.
 - Sie dürfen nicht mit Leerstellen enden.
 - Sie dürfen keine Semikola enthalten.
 - länger als 255 Byte sein

ENCODING-Klausel Die meisten in der COLLATION-Klausel angegebenen Kollationen legen sowohl die Kodierung (Zeichensatz) als auch die Sortierreihenfolge fest. Bei diesen Kollationen sollte die ENCODING-Klausel nicht angegeben werden. Wenn jedoch der in der COLLATION-Klausel angegebene Wert UCA (Unicode Collation Algorithm) ist, verwenden Sie die ENCODING-Klausel, um eine standortspezifische Kodierung anzugeben und die Vorteile von UCA bei Vergleichen und Sortierungen zu nutzen. Die ENCODING-Klausel kann UTF-8 oder jede Einbyte-Kodierung für CHAR-Datentypen angeben. Sie darf keine andere Mehrbyte-Kodierung als UTF-8 angeben.

Wenn Sie die UCA-Kollation auswählen, können Sie Optionen der Kollationsanpassung festlegen.

Wenn COLLATION auf UCA eingestellt ist und ENCODING nicht angegeben wurde, verwendet SQL Anywhere UTF-8.

ENCRYPTED- oder ENCRYPTED TABLE-Klausel Eine Verschlüsselung macht gespeicherte Daten unlesbar. Verwenden Sie das ENCRYPTED-Schlüsselwort (ohne TABLE), um die gesamte Datenbank zu verschlüsseln. Verwenden Sie die ENCRYPTED TABLE-Klausel, wenn Sie nur die Tabellenverschlüsselung aktivieren wollen. Die Tabellenverschlüsselung zu aktivieren heißt, dass die Tabellen, die nachfolgend unter Verwendung der ENCRYPTED-Klausel erstellt oder geändert werden, unter Verwendung der Einstellungen verschlüsselt werden, die Sie bei der Datenbankerstellung angegeben haben.

Es gibt zwei Arten von Datenbank- und Tabellenverschlüsselung: einfache und starke. Einfache Verschlüsselung entspricht der Verschleierung. Die Daten sind unlesbar, aber eine Person mit kryptografischen Kenntnissen könnte die Daten entziffern. Bei starker Verschlüsselung sind die Daten unlesbar und es ist nahezu unmöglich, sie zu entziffern.

Für eine einfache Verschlüsselung geben Sie ENCRYPTED ON ALGORITHM SIMPLE bzw. ENCRYPTED ALGORITHM SIMPLE an, oder Sie geben die ENCRYPTED ON-Klausel ohne Festlegung eines Algorithmus bzw. Schlüssels an.

Um die starke Verschlüsselung zu aktivieren, geben Sie ENCRYPTED ON ALGORITHM mit einem 128- oder 256-Bit-Algorithmus und die KEY-Klausel zur Definition eines Chiffrierschlüssels an. Es wird empfohlen, dass Sie einen Wert für Ihren Schlüssel wählen, der mindestens 16 Zeichen umfasst und eine

Mischung aus Groß- und Kleinbuchstaben, Zahlen und Sonderzeichen enthält. Ein Schlüssel kann entweder als Zeichenfolge oder als Variablenname angegeben werden.

Unter Windows Mobile werden die AES_FIPS- und AES256_FIPS-Algorithmen nur mit ARM-Prozessoren unterstützt.

Vorsicht

Bei stark verschlüsselten Datenbanken achten Sie darauf, eine Kopie des Schlüssels an einem sicheren Ort zu verwahren. Wenn Sie den Chiffrierschlüssel verlieren, gibt es keine Möglichkeit, auf die Daten zuzugreifen, auch nicht mit Unterstützung durch den technischen Support. Die Datenbank muss verworfen und eine neue Datenbank muss erstellt werden.

Sie können auch eine verschlüsselte Kopie einer bestehenden Datenbank mit der Anweisung CREATE ENCRYPTED DATABASE erstellen.

JCONNECT-Klausel Um dem jConnect JDBC-Treiber den Zugriff auf Systemkataloginformationen zu ermöglichen, geben Sie JCONNECT ON an. Diese Klausel installiert die Systemobjekte, die jConnect-Unterstützung zur Verfügung stellen. Geben Sie JCONNECT OFF an, wenn Sie die jConnect-Systemobjekte ausschließen möchten. Sie können trotzdem JDBC verwenden, sofern Sie nicht auf Systemdaten zugreifen. JCONNECT ist standardmäßig ON.

PAGE SIZE-Klausel Die Seitengröße für eine Datenbank kann 2.048, 4.096, 8.192, 16.384 oder 32.768 Byte betragen. Die Standard-Seitengröße ist 4096 Byte. Große Datenbanken haben bei einer höheren Seitengröße eine bessere Performance, mit hohen Seitengrößen kann aber auch ein zusätzlicher Overhead verbunden sein.

Beispiel:

```
CREATE DATABASE 'c:\\temp\\my_db.db'
DBA USER 'DBA' DBA PASSWORD 'sql'
PAGE SIZE 4096;
```

Hinweis

Die Seitengröße kann nicht größer sein als die vom aktuellen Server verwendete Seitengröße. Die Seitengröße des Servers wird entweder aus der ersten Gruppe gestarteter Datenbanken übernommen oder in der Serverbefehlszeile mit der Option -gp festgelegt.

NCHAR COLLATION-Klausel Die durch die NCHAR COLLATION-Klausel angegebene Kollation wird für die Sortierung und den Vergleich von nationalen Zeichendatentypen verwendet (NCHAR, NVARCHAR und LONG NVARCHAR). Die Kollation liefert Informationen über die Zeichensortierfolge für die UTF-8-Kodierung (Zeichensatz), die für nationale Sonderzeichen verwendet wird. Wenn die NCHAR COLLATION-Klausel nicht angegeben ist, verwendet SQL Anywhere UCA (Unicode Collation Algorithm). Die einzige andere zulässige Kollation ist UTF8BIN, die eine binäre Sortierreihenfolge für alle Zeichen erstellt, deren Kodierung größer als 0x7E ist.

Optional können Sie Optionen der Kollationsanpassung (*collation-tailoring-string*) für eine zusätzliche Steuerung bei Zeichensortierungen und -vergleichen angeben. Diese Optionen werden in der Form von Schlüsselwort=Wert-Paaren, die in Anführungszeichen gesetzt werden, hinter dem Kollationsnamen angegeben. Zum Beispiel: . . . NCHAR COLLATION

'UCA(locale=es;case=respect;accent=respect)'. Wenn Sie die ACCENT- oder die CASE-Klausel und eine Zeichenfolge zur Kollationsanpassung angeben, die Angaben für die Berücksichtigung von Akzenten oder der Groß-/Kleinschreibung enthält, werden die Werte der ACCENT- und CASE-Klausel nur als Standardwerte verwendet.

Hinweis

Wenn Sie eine UCA-Kollation angeben, werden alle Optionen der Kollationsanpassung unterstützt. Bei allen anderen Kollationen wird nur die Kollationsanpassungsoption für Groß-/Kleinschreibung unterstützt.

Mit Optionen der Kollationsanpassung erstellte Datenbanken können nicht mit einem Datenbankserver vor Version 10.0.1 gestartet werden.

Klausel SYSTEM PROCEDURE AS DEFINER {ON | OFF} Die SYSTEM PROCEDURE AS DEFINER-Klausel gibt an, ob Systemprozeduren vor Version 16.0, die mit Privilegien verbundene Aufgaben ausführen, mit den Privilegien des Aufrufers oder des Definierers (Eigentümers) ausgeführt werden sollen. ON bedeutet, dass diese Systemprozeduren mit den Privilegien des Definierers (Eigentümers) ausgeführt werden. OFF bedeutet, dass diese Systemprozeduren mit den Privilegien des Aufrufers ausgeführt werden.

Wenn diese Klausel nicht angegeben wird, werden diese Prozeduren standardmäßig mit den Privilegien des Aufrufers ausgeführt.

Diese Einstellung betrifft nicht benutzerdefinierte Prozeduren oder Systemprozeduren, die in Version 16.0 oder später eingeführt wurden. Hinweise dazu, welche Systemprozeduren betroffen sind und welche Auswirkungen die Einstellung hat, finden Sie unter „Systemprozeduren vor Version 16.0 als Aufrufer oder Definierer ausführen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

TRANSACTION LOG-Klausel Das Transaktionslog ist eine Datei, in welcher der Datenbankserver alle Änderungen protokolliert, die in der Datenbank gemacht wurden. Das Transaktionslog spielt eine entscheidende Rolle beim Sichern und Wiederherstellen sowie bei der Datenreplikation.

Mithilfe der MIRROR-Klausel der TRANSACTION-Klausel können Sie einen Dateinamen angeben, wenn Sie einen Transaktionslog-Spiegel verwenden. Ein Transaktionslogspiegel ist eine identische Kopie eines Transaktionslogs, die normalerweise auf einem getrennten Medium gehalten wird, um die Daten besser zu schützen. Standardmäßig verwendet SQL Anywhere keinen Transaktionslogspiegel.

Bemerkungen

Erstellt eine Datenbankdatei mit den angegebenen Namen und Attributen. Sie wird als Betriebssystemdatei gespeichert. Diese Anweisung wird in Prozeduren, Triggern, Ereignissen oder Batches nicht unterstützt.

Sie müssen mit einer Datenbank verbunden sein, um eine andere Datenbank erstellen zu können, z.B. mit der Dienstprogrammdateiabank.

Das Konto, unter dem der Datenbankserver läuft, muss über eine Schreibberechtigung für die Verzeichnisse verfügen, in denen die Dateien erstellt werden.

Meldungen an den Client enthalten die Angabe, welcher Typ der Datenbankverschlüsselung für die Datenbank verwendet wird. Falls eine Verschlüsselung erfolgt, wird der verwendete Algorithmus ebenfalls angezeigt.

Privilegien

Ob Sie diese Anweisung ausführen können, hängt von der Einstellung der Datenbankoption `-gu` ab und davon, ob Sie das `SERVER OPERATOR`-Systemprivileg haben.

Nebenwirkungen

Eine Betriebssystemdatei wird erstellt.

Siehe auch

- „ALTER DATABASE-Anweisung“ auf Seite 451
- „DROP DATABASE-Anweisung“ auf Seite 804
- „Dienstprogramm Initialisierung (dbinit)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Verbindungsparameter DatabaseKey (DBKEY)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Das Transaktionslog“ [*SQL Anywhere Server - Datenbankadministration*]
- „Validierungs-Dienstprogramm (dbvalid)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Erkennung von Beschädigungen mithilfe von Prüfsummen“ [*SQL Anywhere Server - Datenbankadministration*]
- „VALIDATE-Anweisung“ auf Seite 1118
- „CREATE ENCRYPTED DATABASE-Anweisung“ auf Seite 601
- „Tabellenverschlüsselung“ [*SQL Anywhere Server - Datenbankadministration*]
- „Die Dienstprogrammdatei“ [*SQL Anywhere Server - Datenbankadministration*]
- „ansi_blanks-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -gp“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -gu“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -wc“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankoption -wc“ [*SQL Anywhere Server - Datenbankadministration*]
- „sa_validate-Systemprozedur“ auf Seite 1352
- „Optionen der Kollationsanpassung“ [*SQL Anywhere Server - Datenbankadministration*]
- „Hinweise zur Kollation“ [*SQL Anywhere Server - Datenbankadministration*]
- „Alternative Kollationen“ [*SQL Anywhere Server - Datenbankadministration*]
- „Einfache Verschlüsselung und starke Verschlüsselung“ [*SQL Anywhere Server - Datenbankadministration*]
- „jConnect unter Windows Mobile“ [*SQL Anywhere Server - Datenbankadministration*]
- „Internationale Sprachen und Zeichensätze“ [*SQL Anywhere Server - Datenbankadministration*]
- „Empfohlene Zeichensätze und Kollationen“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** Die `CREATE DATABASE`-Anweisung wird von Adaptive Server Enterprise unterstützt, wenn auch mit anderen Klauseln.

Beispiele

Die folgende Anweisung erstellt eine Datenbankdatei mit dem Namen `temp.db` im Verzeichnis `C:\temp`.

```
CREATE DATABASE 'c:\\temp\\temp.db'  
DBA USER 'DBA' DBA PASSWORD 'sql';
```

Die folgende Anweisung erstellt eine Datenbankdatei mit dem Namen *mydb.db* im Verzeichnis *C:\\temp*.

```
CREATE DATABASE 'C:\\temp\\mydb.db'  
DBA USER 'DBA' DBA PASSWORD 'sql'  
TRANSACTION LOG ON  
CASE IGNORE  
PAGE SIZE 4096  
ENCRYPTED OFF  
BLANK PADDING OFF;
```

Die folgende Anweisung erstellt eine Datenbank mithilfe von Codepage 1252 und verwendet UCA für CHAR- und NCHAR-Datentypen. Akzente sowie Groß-/Kleinschreibung werden bei Vergleich und Sortierung berücksichtigt.

```
CREATE DATABASE 'c:\\temp\\uca.db'  
DBA USER 'DBA' DBA PASSWORD 'sql'  
COLLATION 'UCA'  
ENCODING 'CP1252'  
NCHAR COLLATION 'UCA'  
ACCENT RESPECT  
CASE RESPECT;
```

Die folgende Anweisung erstellt die Datenbank *myencrypteddb.db*, die mit einfacher Verschlüsselung verschlüsselt ist:

```
CREATE DATABASE 'c:\\temp\\myencrypteddb.db'  
DBA USER 'DBA' DBA PASSWORD 'sql'  
ENCRYPTED ON;
```

Die folgende Anweisung erstellt die Datenbank *mystrongencryptdb.db*, die mithilfe des Schlüssels gh67AB2 (starke Verschlüsselung) verschlüsselt ist:

```
CREATE DATABASE 'c:\\temp\\mystrongencryptdb.db'  
DBA USER 'DBA' DBA PASSWORD 'sql'  
ENCRYPTED ON KEY 'gh67AB2';
```

Die folgende Anweisung erstellt die Datenbank *mytableencryptdb.db* mit aktivierter Tabellenverschlüsselung, die eine einfache Verschlüsselung verwendet. Beachten Sie das Schlüsselwort TABLE, das hinter ENCRYPTED eingefügt ist, um Tabellenverschlüsselung anstelle von Datenbankverschlüsselung anzugeben:

```
CREATE DATABASE 'c:\\temp\\mytableencryptdb.db'  
DBA USER 'DBA' DBA PASSWORD 'sql'  
ENCRYPTED TABLE ON;
```

Die folgende Anweisung erstellt die Datenbank *mystrongencrypttabledb.db* mit aktivierter Tabellenverschlüsselung, die eine einfache Verschlüsselung verwendet:

```
CREATE DATABASE 'c:\\temp\\mystrongencrypttabledb.db'  
DBA USER 'DBA' DBA PASSWORD 'sql'  
ENCRYPTED TABLE ON 'SIMPLE';
```

Die folgende Anweisung erstellt eine Datenbankdatei mit dem Namen *mydb.db*, die die Kollation 1252LATIN1 verwendet. Die NCHAR-Kollation ist auf UCA gesetzt, wobei die Sprachumgebung auf "es" gesetzt und die Berücksichtigung der Groß-/Kleinschreibung und der Akzente aktiviert ist:

```
CREATE DATABASE 'c:\\temp\\my2.db'
DBA USER 'DBA' DBA PASSWORD 'sql'
COLLATION '1252LATIN1(case=respect)'
NCHAR COLLATION 'UCA(locale=es;case=respect;accent=respect)';
```

Die folgende Anweisung erstellt eine Datenbank mit griechischer Kollation:

```
CREATE DATABASE 'c:\\temp\\mydb.db'
DBA USER 'DBA' DBA PASSWORD 'sql'
COLLATION '1253ELL';
```

Die folgende Anweisung erstellt eine Datenbank namens *mydb.db* mit einem Transaktionslog-Spiegel:

```
CREATE DATABASE 'c:\\mydb.db'
DBA USER 'DBA' DBA PASSWORD 'sql'
TRANSACTION LOG ON 'mydb.log'
MIRROR 'd:\\mydb.mlg';
```

CREATE DBSPACE-Anweisung

Definiert einen neuen Datenbankspeicher und erstellt die zugehörige Datenbankdatei.

Syntax

```
CREATE DBSPACE dbspace-name AS filename
```

Parameter

dbspace-name Geben Sie einen Namen für den DBSpace an. Dies ist nicht der eigentliche Datenbankdateiname, den Sie mit *filename* angeben. *dbspace-name* ist ein interner Name, den Sie beispielsweise in Anweisungen und Prozeduren verwenden können. Die folgenden Namen können nicht für DBSpaces verwendet werden, weil sie für vordefinierte DBSpaces reserviert sind: system, temporary, temp, translog und translogmirror.

Ein Fehler wird zurückgegeben, wenn Sie einen Wert eingeben, der einen Punkt (.) enthält.

filename Geben Sie einen Namen für die Datenbankdatei und optional den Pfad zur Datei ein. Wenn kein Pfad angegeben ist, wird die Datenbankdatei an der Stelle (Verzeichnis) erstellt, an der die Haupt-Datenbankdatei gespeichert ist. Wenn Sie einen anderen Speicherort angeben, ist der Pfad relativ zum Datenbankserver. Das Backslashzeichen (\) ist ein Escapezeichen in SQL-Zeichenfolgen, weshalb jeder Backslash verdoppelt werden muss.

Der Parameter *filename* muss ein Zeichenfolgenliteral oder eine Variable sein.

🔥 **Cloud-Hinweis:** Wenn Sie für Tenant-Datenbanken in einer Cloud den Standort eines DBSpace festlegen, können nur einen Dateinamen angeben. Sie können keinen Verzeichnispfad angeben.

Bemerkungen

Die CREATE DBSPACE-Anweisung erstellt eine neue Datenbankdatei. Wenn eine Datenbank erstellt wird, besteht sie aus einer Datei. Alle Tabellen und Indizes werden in dieser Datei abgelegt. CREATE DBSPACE fügt der Datenbank eine neue Datei hinzu. Diese Datei kann sich auf einem anderen Plattenlaufwerk befinden als die Stammdatei, was bedeutet, dass die Datenbank größer als ein physisches Medium sein kann.

Hinweis

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „[Sandboxing](#)“ [*SQL Anywhere Server - Datenbankadministration*].

Für jede Datenbank besteht ein Limit von zwölf DBSpaces zusätzlich zur Stammdatei.

Jedes Objekt, wie eine Tabelle oder ein Index, ist zur Gänze in einem einzigen DBSpace gespeichert. Die IN-Klausel der CREATE-Anweisung gibt den DBSpace an, in dem ein Objekt gespeichert wird. Objekte werden standardmäßig in der Datenbankdatei "system" gespeichert. Sie können auch angeben, in welchem DBSpace Tabellen erstellt werden, indem Sie die Option default_dbspace einstellen, bevor Sie die Tabellen erstellen.

Privilegien

Sie müssen das MANAGE ANY DBSPACE-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Automatischer Checkpoint

Siehe auch

- „DROP DBSPACE-Anweisung“ auf Seite 806
- „default_dbspace-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Zeichenfolgen“ auf Seite 6
- „Vordefinierte DBSpaces“ [*SQL Anywhere Server - Datenbankadministration*]
- „Zusätzliche Hinweise zu DBSpaces“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird ein DBSpace namens libbooks im Verzeichnis c:\ erstellt. Eine nachfolgende CREATE TABLE-Anweisung erstellt eine Tabelle namens LibraryBooks im DBSpace libbooks.

```
CREATE DBSPACE libbooks
AS 'c:\library.db';
CREATE TABLE LibraryBooks (
    title char(100),
    author char(50),
    isbn char(30),
) IN libbooks;
```

CREATE DECRYPTED DATABASE-Anweisung

Erstellt eine entschlüsselte Kopie einer bestehenden Datenbank inklusive aller Transaktionslogs und DBSpaces.

Syntax

```
CREATE DECRYPTED DATABASE newfile
FROM oldfile
[ KEY key ]
```

Parameter

FROM-Klausel Verwenden Sie diese Klausel, um den Namen der zu kopierenden Datenbank anzugeben (*oldfile*).

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „Sandboxing“ [[SQL Anywhere Server - Datenbankadministration](#)].

KEY-Klausel Benutzen Sie diese Klausel, um den Chiffrierschlüssel anzugeben, der für die Entschlüsselung der Datenbank erforderlich ist. Sie können für den Schlüssel entweder eine Zeichenfolge oder einen Variablennamen angeben. Geben Sie die KEY-Klausel nicht an, wenn die bestehende Datenbank mit einfacher Verschlüsselung chiffriert wurde, weil dafür kein Schlüssel erforderlich ist.

Bemerkungen

Die Anweisung CREATE DECRYPTED DATABASE erstellt eine neue Datenbankdatei (*newfile*) und ersetzt oder entfernt die alte Datenbankdatei (*oldfile*) nicht.

Alle verschlüsselten Tabellen in *oldfile* sind in *newfile* nicht verschlüsselt und die Tabellenverschlüsselung ist nicht aktiviert.

Hinweis

Für Datenbanken, die mit SQL Anywhere 16 oder später erstellt wurden, bleiben die Systemtabellen ISYSCOLSTAT, ISYSUSER und ISYSEXTNLOGIN grundsätzlich weiterhin verschlüsselt, um die Daten vor unberechtigtem Zugriff zu schützen.

Wenn *oldfile* Transaktionslog- oder Transaktionslogspiegeldateien verwendet, werden diese in *Neue_Datei.log* bzw. *Neue_Datei.mlg* umbenannt.

Wenn *oldfile* DBSpace-Dateien enthält, wird dem Dateinamen ein D (decrypted = entschlüsselt) hinzugefügt. Beispiel: Wenn Sie die Anweisung CREATE DECRYPTED DATABASE ausführen und *oldfile* gleich *mydbspace.dbs* ist, erhält *newfile* den Namen *mydbspace.dbsD*.

Hinweis

Wenn Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Hauptdatenbankdatei befindet. Siehe „Sandboxing“ [[SQL Anywhere Server - Datenbankadministration](#)].

Sie können diese Anweisung nicht in einer Datenbank ausführen, die eine Wiederherstellung benötigt. Diese Anweisung wird in Prozeduren, Triggern, Ereignissen oder Batches nicht unterstützt.

Sie können nicht mit der Datenbank verbunden sein, die Sie entschlüsseln möchten. Sie müssen mit einer anderen Datenbank verbunden sein, z.B. mit der Dienstprogrammdatenbank. Die Datenbank, die Sie verschlüsseln möchten, darf nicht laufen.

Privilegien

Ob Sie diese Anweisung ausführen können, hängt von der Einstellung der Datenbankoption -gu ab und davon, ob Sie das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „Datenbankverschlüsselung und -entschlüsselung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „CREATE ENCRYPTED DATABASE-Anweisung“ auf Seite 601
- „CREATE ENCRYPTED FILE-Anweisung“ auf Seite 604
- „CREATE DECRYPTED FILE-Anweisung“ auf Seite 596

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die erste Anweisung unten erstellt eine mit AES256 verschlüsselte Kopie der *demo.db* mit dem Namen *demoEncrypted.db*. Die zweite Anweisung erstellt eine entschlüsselte Kopie von *demoEncrypted.db* namens *demoDecrypted.db*.

```
CREATE ENCRYPTED DATABASE 'demoEncrypted.db'  
FROM 'demo.db'  
KEY 'Sd8f6654*Mnn'  
ALGORITHM 'AES256';  
CREATE DECRYPTED DATABASE 'demoDecrypted.db'  
FROM 'demoEncrypted.db'  
KEY 'Sd8f6654*Mnn';
```

CREATE DECRYPTED FILE-Anweisung

Erstellt eine entschlüsselte Kopie einer stark verschlüsselten Datenbank und kann verwendet werden, um entschlüsselte Kopien von Transaktionslogs, Transaktionslogspiegeln und DBSpaces zu erstellen.

Syntax

```
CREATE DECRYPTED FILE newfile  
FROM oldfile KEY key
```

Parameter

FROM-Klausel Gibt den Dateinamen der verschlüsselten Datei an

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

KEY-Klausel Führt den Schlüssel auf, der für den Zugriff auf die verschlüsselte Datei erforderlich ist. Der Schlüssel kann entweder eine Zeichenfolge oder ein Variablenname sein.

Bemerkungen

Verwenden Sie diese Anweisung, wenn Ihre Datenbank eine Wiederherstellung benötigt und Sie eine entschlüsselte Kopie der Datenbank für den Support benötigen. Sie müssen diese Anweisung auch verwenden, um Nebendateien einer Datenbank wie Transaktionslog, Transaktionslogspiegel und DBSpace zu entschlüsseln.

Die ursprüngliche Datenbankdatei muss mit einem Chiffrierschlüssel stark verschlüsselt worden sein. Die daraus resultierende Datei ist eine genaue Kopie der verschlüsselten Datei, jedoch ohne Verschlüsselung. Es ist also kein Schlüssel erforderlich, um auf sie zuzugreifen.

Hinweis

Wenn Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Hauptdatenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Wenn eine Datenbank mit dieser Anweisung entschlüsselt wird, muss auch die zugehörige Logdatei (und ggf. DBSpaces) entschlüsselt werden, damit die Datenbank benutzt werden kann.

Wenn eine wiederherzustellende Datenbank entschlüsselt wird, muss auch die Transaktionslogdatei entschlüsselt werden. Die Wiederherstellung wird dann mit der neuen Datenbank ausgeführt. Der Name der Transaktionslogdatei bleibt bei diesem Verfahren unverändert. Wenn also die Datenbank und die Transaktionslogdatei umbenannt werden, müssen Sie `dblog -t` mit der daraus resultierenden Datenbank ausführen.

Sie können diese Anweisung nicht bei einer Datenbank verwenden, bei der die Tabellenverschlüsselung aktiviert ist. Wenn Sie Tabellen entschlüsseln wollen, verwenden Sie die `NOT ENCRYPTED`-Klausel der `ALTER TABLE`-Anweisung.

Hinweis

Für Datenbanken, die mit SQL Anywhere 16 oder später erstellt wurden, bleiben die Systemtabellen `ISYSCOLSTAT`, `ISYSUSER` und `ISYSEXTERNLOGIN` grundsätzlich weiterhin verschlüsselt, um die Daten vor unberechtigtem Zugriff auf die Datenbankdatei zu schützen.

Diese Anweisung wird in Prozeduren, Triggern, Ereignissen oder Batches nicht unterstützt.

Sie können nicht mit der Datenbank verbunden sein, die Sie entschlüsseln möchten. Sie müssen mit einer anderen Datenbank verbunden sein, z.B. mit der Dienstprogrammdateiabank. Die Datenbank, die Sie verschlüsseln möchten, darf nicht laufen.

Privilegien

Ob Sie diese Anweisung ausführen können, hängt von der Einstellung der Datenbankoption `-gu` ab und davon, ob Sie das `SERVER OPERATOR`-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „ALTER TABLE-Anweisung“ auf Seite 516
- „CREATE ENCRYPTED FILE-Anweisung“ auf Seite 604
- „CREATE DECRYPTED DATABASE-Anweisung“ auf Seite 594
- „CREATE ENCRYPTED DATABASE-Anweisung“ auf Seite 601
- „Datenbankserveroption -gu “ [SQL Anywhere Server - Datenbankadministration]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird die fiktive verschlüsselte Datenbank "encContacts" entschlüsselt und die neue unverschlüsselte Datenbank "contacts" wird erstellt.

```
CREATE DECRYPTED FILE 'contacts.db'  
FROM 'encContacts.db'  
KEY 'Sd8f6654*Mnn';
```

CREATE DOMAIN-Anweisung

Erstellt eine Domäne in einer Datenbank.

Syntax

```
CREATE { DOMAIN | DATATYPE } [ AS ] domain-name data-type  
[ [ NOT ] NULL ]  
[ DEFAULT default-value ]  
[ CHECK ( condition ) ]  
[ AS USER user-name ]
```

domain-name : identifier

data-type : built-in data type, with precision and scale

Parameter

DOMAIN | DATATYPE-Klausel Es wird empfohlen, CREATE DOMAIN zu verwenden anstatt CREATE DATATYPE, da CREATE DOMAIN im SQL/2008-Standard definiert ist.

NULL-Klausel Mit dieser Klausel können Sie die Nullwertfähigkeit einer Domäne angeben. Wenn eine Domäne zur Definition einer Tabelle verwendet wird, wird die Nullwertfähigkeit folgendermaßen bestimmt:

- In der Spaltendefinition.
- In der Domänenendefinition.
- Wenn die Nullwertfähigkeit weder in der Spalten- noch in der Domänenendefinition explizit angegeben ist, wird die Einstellung der Option allow_nulls_by_default verwendet.

CHECK-Klausel Für die Erstellung einer Domäne mit CHECK-Integritätsregel können Sie in der Suchbedingung den Namen einer Variablen verwenden, dem das Zeichen "@" vorangestellt wird. Wenn

der Datentyp in der Definition einer Spalte verwendet wird, wird eine solche Variable durch den Spaltennamen ersetzt. Dadurch kann die CHECK-Integritätsregel einer Domäne auf jede mit dieser Domäne definierte Tabellenspalte angewendet werden.

AS USER-Klausel Gibt den Eigentümer des Objekts an.

Bemerkungen

Domänen sind Aliasnamen für integrierte Datentypen, und beinhalten ggf. auch Gesamtstellen- und Dezimalstellenwerte. Sie verbessern den Bedienungskomfort und unterstützen die Konsistenz in der Datenbank.

Domänen sind Objekte innerhalb einer Datenbank. Ihre Namen müssen den Regeln für Bezeichner entsprechen. Bei Domänennamen wird die Groß-/Kleinschreibung genauso wie bei integrierten Datentypnamen nicht berücksichtigt.

Der Benutzer, der einen Datentyp erstellt, wird automatisch zum Eigentümer dieses Datentyps. In der CREATE DATATYPE-Anweisung kann kein Eigentümer angegeben werden. Der Domänenname muss eindeutig sein, und alle Benutzer können auf den Datentyp zugreifen, ohne den Eigentümer als Präfix zu verwenden.

Domänen können CHECK-Bedingungen und DEFAULT-Werte haben. Außerdem können Sie angeben, ob der Datentyp NULL zulässt oder nicht. Diese Bedingungen und Werte werden an alle Spalten vererbt, die mit dieser Domäne definiert wurden. Alle explizit in der Spalte angegebenen Bedingungen oder Werte heben die für die Domäne angegebenen auf.

Privilegien

Sie müssen das CREATE DATATYPE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben, um Domänen erstellen zu können, deren Eigentümer Sie sind. Sie können keine Domänen erstellen, deren Eigentümer andere Benutzer sind.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „allow_nulls_by_default-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „DROP DOMAIN-Anweisung“ auf Seite 807
- „SQL-Datentypen“ auf Seite 95

Standards und Kompatibilität

- **SQL/2008** Die Domänenunterstützung ist die optionale SQL-Sprachenfunktion F251 des SQL/2008-Standards.

Beispiele

Die folgende Anweisung erstellt eine Domäne mit dem Namen address, die eine Zeichenfolge aus 35 Zeichen enthält und NULL zulässt.

```
CREATE DOMAIN address CHAR( 35 ) NULL;
```

Die folgende Anweisung erstellt eine Domäne namens ID, die NULL nicht zulässt und standardmäßig auf "autoincrement" gesetzt ist.

```
CREATE DOMAIN ID INT
NOT NULL
DEFAULT AUTOINCREMENT;
```

Die folgende Anweisung erstellt eine Domäne mit dem Namen PhoneNumber, die mithilfe eines regulären Ausdrucks innerhalb einer CHECK-Integritätsregel sicherstellt, dass die Zeichenfolge eine korrekt formatierte nordamerikanische Telefonnummer mit 12 Zeichen enthält, bestehend aus dreistelliger Vorwahl, dreistelliger Vermittlungsnummer und vierstelliger Durchwahl, getrennt entweder durch Bindestriche oder durch Leerzeichen.

```
CREATE DOMAIN PhoneNumber CHAR(12) NULL
CHECK( @PhoneNumber REGEXP '([2-9][0-9]{2}-[2-9][0-9]{2}-[0-9]{4})|([2-9][0-9]{2}\s[2-9][0-9]{2}\s[0-9]{4})' );
```

Einige Spalten in einer Datenbank werden für Namen von Mitarbeitern verwendet, andere für Adressen. Sie können dazu folgende Domänen definieren:

```
CREATE DOMAIN persons_name CHAR(30)
CREATE DOMAIN street_address CHAR(35);
```

Wenn Sie diese Domänen definiert haben, können Sie sie genauso verwenden wie die integrierten Datentypen. Sie können diese Definitionen folgendermaßen verwenden, um eine Tabelle zu definieren. Sie müssen das CREATE TABLE-Privileg haben, um die folgende Anweisung ausführen zu können.

```
CREATE TABLE myCustomers (
    ID INT DEFAULT AUTOINCREMENT PRIMARY KEY,
    Name persons_name,
    Street street_address);
```

Im obigen Beispiel wird der Primärschlüssel der Tabelle als Ganzzahl definiert. Viele Ihrer Tabellen benötigen ähnliche Bezeichner. Statt nun jedesmal anzugeben, dass es sich dabei um Ganzzahlen handelt, ist es bequemer, eine Bezeichnerdomäne für diese Anwendungen anzugeben.

Wenn Sie eine Domäne erstellen, können Sie einen Standardwert festlegen und eine Prüf-Integritätsregel bereitstellen, um sicherzustellen, dass keine falschen Werte in eine Spalte dieses Typs eingegeben werden.

Ganzzahlwerte werden im Allgemeinen als Tabellenbezeichner verwendet. Eine gute Methode für eindeutige Bezeichner besteht darin, positive Ganzzahlen zu verwenden. Da solche Bezeichner wahrscheinlich in vielen Tabellen verwendet werden, könnten Sie folgende Domäne definieren:

```
CREATE DOMAIN identifier UNSIGNED INT
DEFAULT AUTOINCREMENT;
```

Unter Verwendung dieser Definition können Sie die Definition der Tabelle "Customers" neu schreiben, wie es oben gezeigt wird.

```
CREATE TABLE Customers2 (
    ID identifier PRIMARY KEY,
    Name persons_name,
    Street street_address
);
```

CREATE ENCRYPTED DATABASE-Anweisung

Erstellt eine verschlüsselte Kopie einer bestehenden Datenbank einschließlich aller Transaktionslogs und DBSpaces.

Syntax 1 - Verschlüsselte Kopie einer Datenbank erstellen

```
CREATE ENCRYPTED DATABASE newfile
FROM oldfile
[ KEY newkey ]
[ ALGORITHM algorithm ]
[ OLD KEY oldkey ]
```

```
algorithm :
'SIMPLE'
| 'AES'
| 'AES256'
| 'AES_FIPS'
| 'AES256_FIPS'
```

Syntax 2 - Kopie einer Datenbank mit aktivierter Tabellenverschlüsselung erstellen

```
CREATE ENCRYPTED TABLE DATABASE newfile
FROM oldfile
[ KEY newkey ]
[ ALGORITHM algorithm ]
[ OLD KEY oldkey ]
```

Parameter

CREATE ENCRYPTED DATABASE-Klausel Verwenden Sie diese Klausel, um einen Namen für die neue verschlüsselte Datenbank einzugeben.

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

CREATE ENCRYPTED TABLE DATABASE-Klausel Verwenden Sie diese Klausel, um einen Namen für die neue Datenbank einzugeben. Die neue Datenbank ist nicht verschlüsselt, aber die Tabellenverschlüsselung ist aktiviert.

FROM-Klausel Verwenden Sie diese Klausel, um den Namen der Original-Datenbank anzugeben (*oldfile*).

Wenn Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Hauptdatenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

KEY-Klausel Wenn *algorithm-key* nicht SIMPLE ist, verwenden Sie diese Klausel, um den Verschlüsselungsschlüssel für *newfile* anzugeben. Der Schlüssel kann entweder eine Zeichenfolge oder ein Variablenname sein.

OLD KEY-Klausel Verwenden Sie diese Klausel, um den Chiffrierschlüssel für *oldfile* anzugeben. Der Schlüssel kann entweder eine Zeichenfolge oder ein Variablenname sein. Diese Klausel ist nur erforderlich, wenn *oldfile* nicht mit der SIMPLE-Verschlüsselung verschlüsselt ist.

ALGORITHM-Klausel Verwenden Sie diese Klausel, um den Verschlüsselungsalgorithmus für *newfile* anzugeben. Wenn Sie eine KEY-Klausel angeben, aber nicht die ALGORITHM-Klausel, wird standardmäßig die AES (128-Bit-Verschlüsselung) verwendet. Wenn Sie SIMPLE für *algorithm* angeben, geben Sie keine KEY-Klausel an.

Bemerkungen

Sie können diese Anweisung auch verwenden, um eine Kopie einer Datenbank zu erstellen und die Tabellenverschlüsselung in der Kopie zu aktivieren.

oldfile kann eine unverschlüsselte Datenbank, eine verschlüsselte Datenbank oder eine Datenbank mit aktivierter Tabellenverschlüsselung sein.

Syntax 1 nimmt eine bestehende Datenbank *oldfile* und erstellt eine verschlüsselte Kopie *newfile* davon.

Syntax 2 nimmt eine bestehende Datenbank *oldfile* und erstellt eine Kopie mit dem Namen *newfile*, wobei die Tabellenverschlüsselung aktiviert wird. Wenn Sie diese Syntax verwenden, werden alle in *oldfile* verschlüsselten Tabellen in *newfile* ebenfalls verschlüsselt. Wenn in *oldfile* keine Tabellen verschlüsselt waren, Sie aber eine Verschlüsselung wünschen, können Sie die Anweisung ALTER TABLE...ENCRYPTED für jede Tabelle ausführen, die Sie verschlüsseln möchten.

Hinweis

Wenn Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Hauptdatenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Weder die eine noch die andere Syntax ersetzt oder entfernt *oldfile*.

Wenn *oldfile* Transaktionslog- oder Transaktionslogspiegel-Dateien verwendet, werden Sie *Neue_Datei.log* und *Neue_Datei.mlg* genannt.

Wenn *oldfile* DBSpace-Dateien enthält, wird ein E (encrypted - verschlüsselt) an den Dateinamen angehängt. Beispiel: Wenn Sie die Anweisung CREATE ENCRYPTED DATABASE ausführen, wird die Datei *mydbspace.dbs* in *mydbspace.dbsE* umbenannt.

Sie können diese Anweisung verwenden, um den Verschlüsselungsalgorithmus und den Schlüssel für eine Datenbank zu ändern. Die CREATE ENCRYPTED DATABASE-Anweisung erstellt allerdings eine neue Datei (*newfile*), ohne die vorherige Version der Datei (*oldfile*) zu ersetzen oder zu entfernen.

CREATE ENCRYPTED DATABASE und CREATE ENCRYPTED TABLE DATABASE können nicht mit einer Datenbank verwendet werden, für die eine Wiederherstellung erforderlich ist. Diese Anweisungen werden nicht in Prozeduren, Triggern, Ereignissen oder Batches unterstützt.

Sie können nicht mit der Datenbank verbunden sein, die Sie verschlüsseln möchten. Sie müssen mit einer anderen Datenbank verbunden sein, z.B. mit der Dienstprogrammdateiabank. Die Datenbank, die Sie verschlüsseln möchten, darf nicht laufen.

Sie können auch mit der dbunload-Option -an und -ek oder -ep durch Entladen und Neuladen der Datenbank eine bestehende Datenbank verschlüsseln oder einen bereits existierenden Chiffrierschlüssel ändern.

Sie können eine verschlüsselte Datenbank oder eine Datenbank mit verschlüsselten Tabellen auch mit der Anweisung CREATE DATABASE erstellen.

Hinweis

Nicht alle Plattformen unterstützen FIPS-zertifizierte Verschlüsselung. Eine Liste der unterstützten Plattformen finden Sie unter <http://www.sybase.com/detail?id=1002288>.

Privilegien

Ob Sie diese Anweisung ausführen können, hängt von der Einstellung der Datenbankoption -gu ab und davon, ob Sie das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „Tipps zum Neuaufbau von Datenbanken mit dem Dienstprogramm Entladen (dbunload)“ [SQL Anywhere Server - SQL-Benutzerhandbuch]
- „Datenbankverschlüsselung und -entschlüsselung“ [SQL Anywhere Server - Datenbankadministration]
- „Tabellenverschlüsselung“ [SQL Anywhere Server - Datenbankadministration]
- „Einfache Verschlüsselung und starke Verschlüsselung“ [SQL Anywhere Server - Datenbankadministration]
- „CREATE DECRYPTED DATABASE-Anweisung“ auf Seite 594
- „CREATE ENCRYPTED FILE-Anweisung“ auf Seite 604
- „CREATE DECRYPTED FILE-Anweisung“ auf Seite 596
- „CREATE DATABASE-Anweisung“ auf Seite 583
- „ALTER TABLE-Anweisung“ auf Seite 516
- „Dienstprogramm Initialisierung (dbinit)“ [SQL Anywhere Server - Datenbankadministration]
- „Datenbankserveroption -gu“ [SQL Anywhere Server - Datenbankadministration]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird eine verschlüsselte Kopie der Beispieldatenbank namens *demoEnc.db* erstellt. Die neue Datenbank wird mit AES256 verschlüsselt.

```
CREATE ENCRYPTED DATABASE 'demoEnc.db'
FROM 'C:\\Users\\Public\\Documents\\SQL Anywhere 16\\Samples\\demo.db'
KEY 'Sd8f6654*Mnn'
ALGORITHM 'AES256';
```

Im folgenden Beispiel wird eine Kopie der Beispieldatenbank namens *demoTableEnc.db* erstellt. In der neuen Datenbank ist die Tabellenverschlüsselung aktiviert. Da ein Schlüssel ohne Algorithmus angegeben wurde, wird die AES-Verschlüsselung verwendet.

```
CREATE ENCRYPTED TABLE DATABASE 'demoTableEnc.db'
FROM 'C:\\Users\\Public\\Documents\\SQL Anywhere 16\\Samples\\demo.db'
KEY 'Sd8f6654';
```

CREATE ENCRYPTED FILE-Anweisung

Erstellt eine stark verschlüsselte Kopie einer Datenbankdatei, des Transaktionslogs, des Transaktionslogspiegels oder des DBSpaces.

Syntax

```
CREATE ENCRYPTED FILE newfile
FROM oldfile
{ KEY key | KEY key OLD KEY oldkey }
[ ALGORITHM {
  'AES'
  | 'AES256'
  | 'AES_FIPS'
  | 'AES256_FIPS' } ]
```

Parameter

FROM-Klausel Gibt den Namen der bestehenden Datei an (*oldfile*), für die die CREATE ENCRYPTED FILE-Anweisung ausgeführt werden soll

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

KEY-Klausel Gibt den zu verwendenden Chiffrierschlüssel an. Der Schlüssel kann entweder eine Zeichenfolge oder ein Variablenname sein.

OLD KEY-Klausel Gibt den aktuellen Schlüssel an, mit dem die Datei verschlüsselt ist. Der Schlüssel kann entweder eine Zeichenfolge oder ein Variablenname sein.

ALGORITHM-Klausel Gibt den Algorithmus an, der zum Verschlüsseln der Datei verwendet wird. Wenn Sie keinen Algorithmus angeben, wird AES (128-Bit-Verschlüsselung) standardmäßig verwendet.

Bemerkungen

Verwenden Sie diese Anweisung, wenn für Ihre Datenbank eine Wiederherstellung erforderlich ist und Sie eine verschlüsselte Kopie der Datenbank für den Support benötigen. Sie müssen diese Anweisung auch verwenden, um Nebendateien einer Datenbank wie Transaktionslog, Transaktionslogspiegel und DBSpace zu verschlüsseln.

Sie können nicht mit der Datenbank verbunden sein, für die Sie die verschlüsselte Datei erstellen möchten. Sie müssen mit einer anderen Datenbank verbunden sein, z.B. mit der Dienstprogrammdateiabank. Die Datenbank, die Sie verschlüsseln möchten, darf nicht laufen.

Wenn Nebendateien der Datenbank verschlüsselt werden, müssen sie denselben Algorithmus und denselben Schlüssel für alle Nebendateien dieser Datenbank angeben.

Wenn mit *oldfile* DBSpaces oder Transaktionslogs verbunden sind und Sie diese auch verschlüsseln, müssen Sie darauf achten, dass der neue Name und Speicherort dieser Dateien in der neuen Datenbank gespeichert werden. Dabei gehen Sie wie folgt vor:

- Führen Sie `dblog -t` in der neuen Datenbank aus, um den Namen und den Speicherort des Transaktionslogs zu ändern.

- Führen Sie dblog -m in der neuen Datenbank aus, um den Namen und den Speicherort des Transaktionslogspeiegels zu ändern.
- Führen Sie die Anweisung ALTER DBSPACE in der neuen Datenbank aus, um den Speicherort und den Namen der DBSpace-Dateien zu ändern.

Hinweis

Wenn Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Hauptdatenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Sie können diese Anweisung verwenden, um den Verschlüsselungsalgorithmus und den Schlüssel für eine Datenbank zu ändern. Die CREATE ENCRYPTED FILE-Anweisung erstellt allerdings eine neue Datei (*newfile*), ohne die vorherige Version der Datei (*oldfile*) zu ersetzen oder zu entfernen.

Der Name der Transaktionslogdatei bleibt bei diesem Verfahren unverändert. Wenn die Datenbank und Transaktionslogdatei also umbenannt werden, müssen Sie dblog -t mit der daraus resultierenden Datenbank ausführen.

Sie können auch mit der dbunload-Option -an und -ek oder -ep durch Entladen und Neuladen der Datenbank eine bestehende Datenbank verschlüsseln oder einen bereits existierenden Chiffrierschlüssel ändern.

Wenn Sie über eine Datenbank mit aktivierter Tabellenverschlüsselung verfügen, können Sie die Datenbank mit dieser Anweisung nicht verschlüsseln. Sie können allerdings diese Anweisung verwenden, um den bei der Tabellenverschlüsselung verwendeten Schlüssel zu ändern. Um eine Datenbank zu verschlüsseln, in der die Tabellenverschlüsselung aktiviert ist, verwenden Sie die Anweisung CREATE ENCRYPTED DATABASE.

Unter Windows Mobile werden die AES_FIPS- und AES256_FIPS-Algorithmen nur mit ARM-Prozessoren unterstützt.

Diese Anweisung wird in Prozeduren, Triggern, Ereignissen oder Batches nicht unterstützt.

Hinweis

Nicht alle Plattformen unterstützen FIPS-zertifizierte Verschlüsselung. Eine Liste der unterstützten Plattformen finden Sie unter <http://www.sybase.com/detail?id=1002288>.

Privilegien

Ob Sie diese Anweisung ausführen können, hängt von der Einstellung der Datenbankoption -gu ab und davon, ob Sie das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „CREATE ENCRYPTED DATABASE-Anweisung“ auf Seite 601
- „Datenbankverschlüsselung und -entschlüsselung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE ENCRYPTED DATABASE-Anweisung“ auf Seite 601
- „CREATE DECRYPTED FILE-Anweisung“ auf Seite 596
- „CREATE DECRYPTED DATABASE-Anweisung“ auf Seite 594
- „Dienstprogramm zum Entladen (dbunload)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Transaktionslog-Dienstprogramm (dblog)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -gu“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird die Beispieldatenbank *demo.db* verschlüsselt und eine neue Datenbank namens *demo2.db* mit AES_FIPS-Verschlüsselung erstellt. Die neue Datenbankdatei wird im aktuellen Arbeitsverzeichnis des Servers platziert.

```
CREATE ENCRYPTED FILE 'demo2.db'  
FROM 'C:\\Users\\Public\\Documents\\SQL Anywhere 16\\Samples\\demo.db'  
KEY 'Sd8f6654*Mnn'  
ALGORITHM 'AES_FIPS';
```

Im folgenden Beispiel werden die Beispieldatenbank *demo.db* und die dazugehörige Transaktionslogdatei *demo.log* verschlüsselt. Nach dem Erstellen werden Datenbank- und Logdatei im aktuellen Arbeitsverzeichnis des Servers platziert. Sie müssen `dblog -ek Sd8f6654*Mnn -t demo3.log demo3.db` ausführen, weil die neue Datenbankdatei *demo3.db* immer noch auf das alte Log zeigt.

```
CREATE ENCRYPTED FILE 'demo3.db'  
FROM 'C:\\Users\\Public\\Documents\\SQL Anywhere 16\\Samples\\demo.db'  
KEY 'Sd8f6654*Mnn';  
CREATE ENCRYPTED FILE 'demo3.log'  
FROM 'C:\\Users\\Public\\Documents\\SQL Anywhere 16\\Samples\\demo.log'  
KEY 'Sd8f6654*Mnn';
```

Um einen Chiffrierschlüssel für eine Datenbank zu ändern, erstellen Sie erst eine Kopie der Datenbankdatei mit dem neuen Schlüssel, wie in dieser Anweisung gezeigt:

```
CREATE ENCRYPTED FILE 'newdemo.db'  
FROM 'C:\\Users\\Public\\Documents\\SQL Anywhere 16\\Samples\\demo.db'  
KEY 'newkey' OLD KEY 'oldkey';
```

Die neue Datenbankdatei wird im aktuellen Arbeitsverzeichnis des Servers platziert. Nachdem Sie die verschlüsselte Datei erstellt haben, können Sie *demo.db* löschen sowie *newdemo.db* in dasselbe Verzeichnis wie die alte Datei verschieben und in *demo.db* umbenennen.

CREATE EVENT-Anweisung

Definiert ein Ereignis und den zugehörigen Handler zum Automatisieren vordefinierter Aktionen und definiert geplante Aktionen.

Syntax

```

CREATE EVENT [ owner.]event-name
[ TYPE event-type
  [ WHERE trigger-condition [ AND trigger-condition ] ... ]
  | SCHEDULE schedule-spec, ... ]
[ ENABLE | DISABLE ]
[ AT { CONSOLIDATED | REMOTE | ALL } ]
[ FOR { PRIMARY | ALL } ]
[ HANDLER
  BEGIN
...
  END ]

```

event-type :

```

BackupEnd
Connect
ConnectFailed
DatabaseStart
DBDiskSpace
Deadlock
"Disconnect"
GlobalAutoincrement
GrowDB
GrowLog
GrowTemp
LogDiskSpace
MirrorFailover
MirrorServerDisconnect
RAISERROR
ServerIdle
TempDiskSpace

```

trigger-condition :

```

event_condition( condition-name ) {
=
| <
| >
| !=
| <=
| >=
} value

```

schedule-spec :

```

[ schedule-name ]
{ START TIME start-time | BETWEEN start-time AND end-time }
[ EVERY period { HOURS | MINUTES | SECONDS } ]
[ ON { ( day-of-week, ... ) | ( day-of-month, ... ) } ]
[ START DATE start-date ]

```

event-name : *identifier*

schedule-name : *identifier*

day-of-week : *string*

day-of-month : *integer*

value : integer

period : integer

start-time : time

end-time : time

start-date : date

Parameter

CREATE EVENT-Klausel Der Ereignisname ist ein Bezeichner. Ein Ereignis hat einen Ersteller: den Benutzer, der das Ereignis erstellt hat. Der Event-Handler wird mit den Privilegien dieses Erstellers ausgeführt. Die Ausführung ist dieselbe wie bei gespeicherten Prozeduren. Sie können keine Ereignisse erstellen, die anderen Benutzern gehören.

TYPE-Klausel Sie können die TYPE-Klausel mit einer optionalen WHERE-Klausel angeben oder SCHEDULE festlegen.

Der *event-type* ist einer der aufgelisteten systemdefinierten Ereignistypen. Bei Ereignistypen wird die Groß-/Kleinschreibung nicht berücksichtigt. Die Bedingungen, unter welchen dieser *event-type* ein Ereignis auslöst, legen Sie mit der WHERE-Klausel fest.

- **DiskSpace-Ereignistypen** Wenn die Datenbank einen Event-Handler für einen der DiskSpace-Typen enthält, prüft der Datenbankserver alle 30 Sekunden den verfügbaren Speicherplatz auf jedem Medium, das der entsprechenden Datei zugeordnet ist.

Falls die Datenbank über mehr als einen DBSpace auf verschiedenen Laufwerken verfügt, prüft DBDiskSpace jedes Laufwerk und handelt abhängig vom kleinsten verfügbaren Speicherplatz.

Der Ereignistyp LogDiskSpace prüft den Speicherort des Transaktionslogs und aller Transaktionslogspiegel und gibt eine Meldung basierend auf dem kleinsten verfügbaren Speicherplatz aus.

DiskSpace-Ereignistypen werden unter Windows Mobile nicht unterstützt.

Der Ereignistyp TempDiskSpace prüft den Umfang des temporären Festplattenspeichers.

Wenn die geeigneten Ereignis-Handler definiert wurden (DBDiskSpace, LogDiskSpace oder TempDiskSpace), prüft der Datenbankserver alle 30 Sekunden den verfügbaren Speicher auf jedem Medium, das mit einer Datenbankdatei verbunden ist. Wenn ein Ereignis definiert wurde, um den Systemereignistyp ServerIdle zu verarbeiten, benachrichtigt der Datenbankserver den Handler, wenn während der abgelaufenen 30 Sekunden keine Anforderungen verarbeitet wurden.

Sie können die Option -fc beim Starten des Datenbankservers angeben, um eine Callback-Funktion zu implementieren, wenn der Datenbankserver Speichermangel im Dateisystem feststellt.

- **GlobalAutoIncrement-Ereignistyp** Das Ereignis wird bei *jeder* Einfügung ausgelöst, wenn die Anzahl der verbleibenden Werte für GLOBAL AUTOINCREMENT geringer als 1% des Endbereichs ist. Eine typische Aktion für den Handler könnte darin bestehen, einen neuen Wert für die Option

global_database_id basierend auf der Tabelle und der Anzahl der restlichen Werte anzufordern, die als Parameter für dieses Ereignis bereitgestellt wurden.

Sie können die Funktion event_condition mit RemainingValues als Argument für diesen Ereignistyp verwenden.

- **ServerIdle-Ereignistyp** Wenn die Datenbank einen Event-Handler für den Typ ServerIdle enthält, prüft der Datenbankserver die Serveraktivität alle 30 Sekunden.
- **Datenbankspiegelungs-Ereignistypen** Das MirrorServerDisconnect-Ereignis wird ausgelöst, wenn eine Verbindung vom Primärdatenbankserver zum Spiegel- bzw. Arbiterserver verloren geht, und das MirrorFailover-Ereignis wird ausgelöst, wenn ein Server Eigentümer der Datenbank wird.

WHERE-Klausel Die Triggerbedingung bestimmt die Bedingung, unter der ein Ereignis ausgelöst wird. Um z.B. eine Aktion auszuführen, wenn der Plattenspeicher mit dem Transaktionslog mehr als 80% voll ist, verwenden Sie die folgende Trigger-auslösende Bedingung:

```
...
WHERE event_condition( 'LogFreePercent' ) < 20
...
```

Das Argument der Funktion event_condition muss für den Ereignistyp gültig sein.

Sie können mehrere AND-Bedingungen für die WHERE-Klausel verwenden, OR-Bedingungen oder andere Bedingungen sind nicht zulässig.

SCHEDULE-Klausel Diese Klausel legt fest, wann geplante Aktionen stattfinden sollen. Die zeitliche Abfolge wird als eine Reihe von Trigger-auslösenden Bedingungen für die zugeordneten Aktionen behandelt, die im Event-Handler definiert sind.

Sie können mehr als einen Abfolgeplan für ein bestimmtes Ereignis und seine Verarbeitungsroutine erstellen. Damit können komplexe Abfolgen implementiert werden. Während Sie bei mehreren Abfolgeplänen einen *schedule-name* angeben müssen, ist der *schedule-name* bei einem einzigen Plan optional.

Ein geplantes Ereignis wiederholt sich, sobald seine Definition EVERY oder ON enthält. Wenn keines dieser Wörter verwendet wird, wird das Ereignis höchstens einmal ausgeführt. Beim Versuch, ein sich nicht-wiederholendes geplantes Ereignis zu erstellen, für das der Startzeitpunkt verstrichen ist, wird ein Fehler generiert. Wenn ein sich nicht-wiederholendes geplantes Ereignis abgelaufen ist, wird sein Abfolgeplan gelöscht, der Event-Handler jedoch nicht.

Die Zeiten für geplante Ereignisse werden berechnet, wenn die Abfolgepläne erstellt werden, und nochmals, wenn die Ausführung des Event-Handlers beendet wird. Die nächste Ereigniszeit wird berechnet, indem der Abfolgeplan oder die Pläne für das Ereignis untersucht werden und der nächste geplante Zeitpunkt ermittelt wird. Wenn ein Event-Handler angewiesen wird, jede Stunde zwischen 9:00 und 17:00 abzulaufen, und die Ausführung 65 Minuten dauert, findet diese um 9:00, 11:00, 13:00, 15:00 und 17:00 statt. Wenn sich die Ausführung überschneiden soll, müssen Sie mehr als ein Ereignis erstellen.

Die Unterklauseln einer Planungsdefinition lauten folgendermaßen:

- **START TIME-Klausel** Der erste geplante Zeitpunkt für jeden Tag, an dem das Ereignis geplant ist. Der Parameter *start-time* ist eine Zeichenfolge und kann keine Variable und kein Ausdruck wie

NOW () sein. Wenn START DATE angegeben wird, bezieht sich START TIME auf dieses Datum und jeden nachfolgenden Tag (sofern der Zeitplan EVERY oder ON enthält). Wenn START DATE nicht angegeben wurde, umfasst START TIME den aktuellen Tag (falls der Zeitpunkt noch nicht überschritten wurde) und alle darauf folgenden Tage (falls die Planung EVERY oder ON enthält). Die Klausel START TIME *start-time* ist gleichwertig mit BETWEEN *start-time* AND '23:59:59'.

- **BETWEEN...AND-Klausel** Ein Zeitabschnitt im Tagesverlauf, außerhalb dessen keine geplanten Zeitpunkte vorliegen. Die Parameter *start-time* und *end-time* sind Zeichenfolgen und können keine Variablen oder Ausdrücke wie NOW () sein. Wenn START DATE angegeben wurde, beginnen die geplanten Zeitpunkte erst ab diesem Datum.
- **EVERY-Klausel** Ein Intervall zwischen aufeinanderfolgenden geplanten Ereignissen. Geplante Ereignisse treten erst nach der unter START TIME angegebenen Startzeit für den Tag auf oder liegen in dem Bereich, der mit BETWEEN...AND festgelegt wurde.
- **ON-Klausel** Eine Liste von Tagen, an denen die geplanten Ereignisse eintreten. Der Standardwert ist täglich, sofern EVERY angegeben ist. Die Tage können als Tage der Woche oder des Monats angegeben werden.

Wochentage sind Montag, Dienstag etc. Sie können die Tage abgekürzt oder in Langform eingeben. Die Langform müssen Sie jedoch verwenden, wenn die verwendete Sprache nicht Englisch ist, nicht die vom Client in der Verbindungszeichenfolge verlangte Sprache ist, und nicht die Sprache ist, die im Meldungsfenster des Datenbankservers erscheint.

Monatstage sind Ganzzahlen zwischen 0 und 31. Ein Wert von "0" steht für den letzten Tag eines Monats.

- **START DATE-Klausel** Das Datum, an dem geplante Ereignisse zum ersten Mal eintreten sollen. Dieser Wert ist eine Zeichenfolge und kann keine Variable und kein Ausdruck wie TODAY () sein. Der Standardwert ist das aktuelle Datum.

Jedes Mal, wenn ein geplanter Event-Handler abgeschlossen ist, werden die folgenden Aktionen ausgeführt, um die nächste geplante Zeit und das Datum für das Ereignis zu berechnen:

1. Bei Verwendung der EVERY-Klausel muss festgestellt werden, ob der nächste geplante Zeitpunkt auf den aktuellen Tag fällt und (sofern dieser angegeben wurde) vor dem Ende des Bereichs BETWEEN...AND liegt. Wenn ja, ist dies der nächste geplante Zeitpunkt.
 2. Wenn der nächste geplante Zeitpunkt nicht auf den aktuellen Tag fällt, wird das nächste Datum, an dem das Ereignis ausgeführt werden soll, ermittelt und START TIME oder der Beginn des BETWEEN...AND-Bereichs für dieses Datum verwendet.
- **ENABLE | DISABLE-Klausel** Standardmäßig sind Event-Handler aktiviert. Wenn DISABLE angegeben ist, wird der Event-Handler nicht ausgeführt, selbst wenn der geplante Zeitpunkt erreicht wird oder die Trigger-auslösende Bedingung eintritt. Eine TRIGGER EVENT-Anweisung kann einen deaktivierten Event-Handler *nicht* zur Ausführung zwingen.
 - **AT-Klausel** Diese Klausel sollte nur unter folgenden Umständen verwendet werden: In einer SQL Remote-Umgebung verwenden Sie die AT-Klausel für die entfernten oder konsolidierten Datenbanken, um die Datenbanken zu begrenzen, in denen das Ereignis verarbeitet werden soll.

Wenn Sie die AT-Klausel beim Erstellen von Ereignissen für SQL Remote nicht verwenden, führen alle Datenbanken das Ereignis aus. Wenn diese Anweisung in einer konsolidierten Datenbank ausgeführt wird, hat sie keine Auswirkungen auf entfernte Datenbanken, die bereits extrahiert wurden.

- **FOR-Klausel** Diese Klausel sollte nur unter folgenden Umständen verwendet werden: Verwenden Sie in einem Datenbankspiegelungssystem oder einem Scale-Out-System mit Schreibschutz die FOR-Klausel, um die Datenbanken einzuschränken, in denen der Event-Handler ausgeführt wird.

Wenn Sie ein Ereignis für eine Datenbank in einem Spiegelungssystem oder einem Scale-Out-System mit Schreibschutz ohne die FOR-Klausel erstellen, wird das Ereignis nur von der Datenbank ausgeführt, die auf dem Primärserver läuft. Folgende Unterklauseln werden unterstützt:

- **FOR PRIMARY** Das Ereignis wird nur auf dem Server ausgeführt, der derzeit als Primärserver fungiert. Der Standardwert ist die PRIMARY-Unterklausel.

Wenn die FOR PRIMARY-Klausel zusammen mit dem DatabaseStart-Ereignistyp verwendet wird (oder die FOR-Klausel nicht angegeben wurde), wird das Ereignis ausgeführt, wenn ein Server die Rolle des Primärservers für die Datenbank übernimmt.

- **FOR ALL** Das Ereignis wird auf allen Servern im System ausgeführt.

Wenn die FOR ALL-Klausel zusammen mit dem DatabaseStart-Ereignistyp verwendet wird, wird das Ereignis beim Starten einer beliebigen Datenbank ausgeführt. Wenn in einem Spiegelungssystem das Ereignis nicht beim Starten der Datenbank ausgeführt wurde (z.B. weil die Datenbank ausgeführt wurde, bevor das Ereignis erstellt wurde), kann das Ereignis während eines Failovers ausgeführt werden. Beispiel: Sie starten ein Datenbankspiegelungssystem, erstellen ein DatabaseStart-Ereignis mit der FOR ALL-Klausel und stoppen anschließend den Primärserver, was einen Failover verursacht. In diesem Beispiel wird das Ereignis auf dem neuen Primärserver ausgeführt. Das DatabaseStart-Ereignis wird bei einem nachfolgenden Failover nicht ausgeführt.

- **HANDLER-Klausel** Jedes Ereignis umfasst eine Verarbeitungsroutine.

Bemerkungen

Ereignisse können für folgende Zwecke verwendet werden:

- **Geplante Aktionen** Der Datenbankserver führt Aktionen nach einem Zeitplan durch. Sie können diese Fähigkeit nutzen, um geplante Aufgaben wie Sicherungen, Gültigkeitsprüfungen und Abfragen durchzuführen, die verwendet werden, um Daten in Berichtstabellen einzufügen.
- **Aktionen zur Ereignisverarbeitung** Der Datenbankserver führt Aktionen aus, wenn ein vordefiniertes Ereignis eintritt. Sie können diese Fähigkeit verwenden, um geplante Aufgaben durchzuführen, etwa die Limitierung von Festplattenspeicher, wenn eine Datei über einen bestimmten Prozentsatz hinaus anwächst. Aktionen von Event-Handlern werden festgeschrieben, wenn während des Ausführens keine Fehler erkannt werden, und zurückgesetzt, wenn Fehler auftreten.

Eine Ereignisdefinition umfasst zwei unterschiedliche Bestandteile. Die Triggerbedingung kann ein auftretendes Ereignis sein, wie z.B. das Überschreiten eines bestimmten Grenzwerts für einen Plattenspeicher. Ein Abfolgeplan ist eine Reihe von Zeiten, die alle als Triggerbedingung gehandhabt werden. Wenn eine Triggerbedingung erfüllt ist, wird der Event-Handler ausgeführt. Der Event-Handler

umfasst eine oder mehrere festgelegte Aktionen innerhalb einer zusammengesetzten Anweisung (BEGIN... END).

Wenn keine Triggerbedingung oder Planungsspezifikation angegeben wird, kann nur eine explizite TRIGGER EVENT-Anweisung das Ereignis auslösen. Während der Entwicklung können Sie Event-Handler mithilfe von TRIGGER EVENT testen. Den Abfolgeplan oder die WHERE-Klausel fügen Sie hinzu, sobald der Test abgeschlossen ist.

Ereignisfehler werden im Datenbankserver-Meldungslog protokolliert.

Nach jeder Ausführung eines Event-Handlers wird ein COMMIT ausgeführt, falls kein Fehler aufgetreten ist. Wenn es einen Fehler gegeben hat, wird ein ROLLBACK ausgeführt.

Wenn Event-Handler ausgelöst werden, stellt der Datenbankserver dem Event-Handler mit der Funktion event_parameter Kontextinformationen bereit, wie z.B. die Verbindungs-ID, die das Ereignis ausgelöst hat.

Event-Handler werden auf einer separaten Verbindung ausgeführt, aber die separate Verbindung zählt nicht zu den zehn Verbindungen, auf die der Personal Datenbankserver beschränkt ist.

Privilegien

Sie müssen das MANAGE ANY EVENT-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „BEGIN-Anweisung“ auf Seite 557
- „ALTER EVENT-Anweisung“ auf Seite 461
- „COMMENT-Anweisung“ auf Seite 573
- „DROP EVENT-Anweisung“ auf Seite 808
- „TRIGGER EVENT-Anweisung“ auf Seite 1088
- „EVENT_PARAMETER-Funktion [System]“ auf Seite 256
- „EVENT_CONDITION-Funktion [System]“ auf Seite 254
- „Datenbankserveroption -fc “ [*SQL Anywhere Server - Datenbankadministration*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]
- „Triggerbedingungen für Ereignisse“ [*SQL Anywhere Server - Datenbankadministration*]
- „Protokollieren von Datenbankserver-Aktionen“ [*SQL Anywhere Server - Datenbankadministration*]
- „Verwendung von Datenbankspiegelungssystem-Ereignissen zum Senden von Benachrichtigungs-E-Mails bei Failover“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Weisen Sie den Datenbankserver an, jeden Tag um 1:00 Uhr eine automatische Sicherung auf Band unter Verwendung des ersten Bandlaufwerks auszuführen.

```

CREATE EVENT DailyBackup
SCHEDULE daily_backup
START TIME '1:00AM' EVERY 24 HOURS
HANDLER
BEGIN
    BACKUP DATABASE TO '\\\\.\tape0'
    ATTENDED OFF
END;

```

Weisen Sie den Datenbankserver an, täglich um 1:00 Uhr eine inkrementelle Sicherung ausführen.

```

CREATE EVENT IncrementalBackup
SCHEDULE
START TIME '1:00 AM' EVERY 24 HOURS
HANDLER
BEGIN
    BACKUP DATABASE DIRECTORY 'c:\\backup'
    TRANSACTION LOG ONLY
    TRANSACTION LOG RENAME MATCH
END;

```

Weisen Sie den Datenbankserver an, montags bis freitags zwischen 8 und 18 Uhr jede Stunde eine automatische Sicherung nur für das Transaktionslog auszuführen.

```

CREATE EVENT HourlyLogBackup
SCHEDULE hourly_log_backup
BETWEEN '8:00AM' AND '6:00PM'
EVERY 1 HOURS ON
('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday')
HANDLER
BEGIN
    BACKUP DATABASE DIRECTORY 'c:\\database\\backup'
    TRANSACTION LOG ONLY
    TRANSACTION LOG RENAME
END;

```

Ermitteln Sie, wann ein Ereignis das nächste Mal ausgeführt werden soll:

```

SELECT DB_EXTENDED_PROPERTY( 'NextScheduleTime', 'HourlyLogBackup' );

```

Erstellen Sie ein Ereignis, das auf jedem Server in einem Spiegelungssystem oder einem Scale-Out-System mit Schreibschutz ausgeführt werden kann.

```

CREATE EVENT evt1 FOR ALL
HANDLER
BEGIN
    MESSAGE CURRENT TIME || ' evt1 active' TO CONSOLE;
END

```

CREATE EXISTING TABLE-Anweisung

Erstellt eine neue Proxytabelle, die ein bestehendes Objekt auf einem Fremdserver repräsentiert.

Syntax

```

CREATE EXISTING TABLE [owner.]table-name
[ column-definition, ... ]
AT location-string

```

column-definition :
column-name data-type NOT NULL

location-string :
remote-server-name.[db-name].[owner].object-name
| *remote-server-name;[db-name];[owner];object-name*

Parameter

AT-Klausel Die AT-Klausel gibt den Standort des entfernten Objekts an. Die AT-Klausel unterstützt das Semikolon (;) als Begrenzer. Wenn in der Zeichenfolge *location-string* ein Semikolon vorhanden ist, dann ist dies als Feldbegrenzer zu betrachten. Wenn kein Semikolon vorhanden ist, gilt der Punkt als Feldbegrenzer. Damit können Dateinamen oder Erweiterungen in den Feldern für Datenbank und Eigentümer verwendet werden.

Die Zeichenfolge in der AT-Klausel kann auch lokale oder globale Variablennamen in geschweiften Klammern enthalten ({*variable-name*}). Der SQL-Variablenname muss vom Typ CHAR, VARCHAR oder LONG VARCHAR sein. Eine AT-Klausel mit 'access;{@myfile};;al' zeigt beispielsweise an, dass @myfile eine SQL-Variable ist und dass der aktuelle Inhalt der @myfile-Variablen beim Erstellen der Proxy-Tabelle ersetzt werden muss.

Bemerkungen

Die CREATE EXISTING TABLE-Anweisung erstellt eine neue lokale Proxytabelle, die ein Abbild einer externen Tabelle darstellt. Die CREATE EXISTING TABLE-Anweisung ist eine Variante der CREATE TABLE-Anweisung. Das EXISTING-Schlüsselwort wird mit CREATE TABLE verwendet, um anzugeben, dass bereits eine entfernte Tabelle vorhanden ist und deren Metadaten importiert werden sollen. Damit wird die entfernte Tabelle als Entität für Benutzer von SQL Anywhere sichtbar. Die Software überprüft vor dem Erstellen der Tabelle, ob die Tabelle am externen Speicherort vorhanden ist.

Wenn das Objekt nicht vorhanden ist (als Host-Datendatei oder Fremdserverobjekt), wird die Anweisung mit einer Fehlermeldung zurückgewiesen.

Indexinformationen aus den Hostdatendateien oder aus der Tabelle auf dem Fremdserver werden extrahiert und benutzt, um Zeilen für die Systemtabelle ISYSIDX zu erstellen. Mit diesen Informationen werden Indizes und Schlüssel in Serverbegriffen definiert, sodass der Abfrageoptimierer in die Lage versetzt wird, mit Indizes zu arbeiten, die für diese Tabelle eventuell existieren.

Integritätsregeln zur Erhaltung der referenziellen Integrität werden erforderlichenfalls an den Fremdserver weitergeleitet.

Wenn *column-definitions* nicht angegeben sind, leitet SQL Anywhere die Spaltenliste von den Metadaten ab, die aus der entfernten Tabelle bezogen werden. Wenn *column-definitions* angegeben sind, werden diese von SQL Anywhere überprüft. Spaltennamen, Datentypen, Längen, die Identitätseigenschaften und NULL-Eigenschaften werden nach folgenden Gesichtspunkten geprüft:

- Spaltennamen müssen genau übereinstimmen (Groß-/Kleinschreibung wird allerdings ignoriert).
- Datentypen in der Anweisung CREATE EXISTING TABLE müssen zu den Datentypen der Spalten in der entfernten Tabelle passen oder in sie konvertierbar sein. Beispiel: Der Datentyp einer lokalen Spalte ist als "money" definiert, während der Datentyp der entfernten Spalte "numeric" lautet.

- Die NULL-Eigenschaften aller Spalten werden geprüft. Wenn die NULL-Eigenschaft der lokalen Spalte nicht identisch mit der entfernten Spalte ist, wird eine Warnmeldung ausgegeben, die Anweisung allerdings nicht abgebrochen.
- Die Länge jeder einzelnen Spalte wird geprüft. Wenn die Länge von Spalten des Typs CHAR, VARCHAR, BINARY, VARBINARY, DECIMAL und/oder NUMERIC nicht übereinstimmt, wird eine Warnmeldung ausgegeben, die Anweisung aber nicht abgebrochen.

Sie können auch nur eine Teilmenge der entfernten Spaltenliste in die CREATE EXISTING-Anweisung aufnehmen.

Diese Anweisung wird unter Windows Mobile nicht unterstützt.

Privilegien

Sie müssen das CREATE PROXY TABLE-Systemprivileg haben, um Proxy-Tabellen erstellen zu können, deren Eigentümer Sie sind. Sie müssen das CREATE ANY TABLE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben, um Proxy-Tabellen erstellen zu können, deren Eigentümer andere Benutzer sind.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE TABLE-Anweisung“ auf Seite 737
- „Standorte von Proxy-Tabellen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** Wird von Adaptive Server Enterprise unterstützt. Das Format von *location-string* ist durch die Implementierung festgelegt.

Beispiele

Eine Proxytabelle namens "blurbs" wird für die Tabelle blurbs auf dem Fremdserverserver_a erstellt.

```
CREATE EXISTING TABLE blurbs
( author_id ID not null,
  copy text not null)
AT 'server_a.db1.joe.blurbs';
```

Erstellen Sie eine Proxy-Tabelle namens "blurbs" für die Tabelle "blurbs" auf dem Fremdserverserver_a. Der Datenbankserver leitet die Spaltenliste aus den Metadaten ab, die er aus der entfernten Tabelle abrufen.

```
CREATE EXISTING TABLE blurbs
AT 'server_a.db1.joe.blurbs';
```

Erstellen Sie eine Proxy-Tabelle namens "rda_employees" für die Tabelle "Employees" auf dem Fremdserverserver_rda.

```
CREATE EXISTING TABLE rda_employees
AT 'rda...Employees';
```

Erstellen Sie eine Proxy-Tabelle namens "rda_employees" für eine Tabelle, die durch die SQL-Variable `table_name` auf dem Fremdserver rda angegeben wird.

```
CREATE EXISTING TABLE rda_employees  
AT 'rda...{table_name}';
```

CREATE EXTERNLOGIN-Anweisung

Weist einen alternativen Loginnamen und ein Kennwort zu, die bei der Kommunikation mit dem Fremdserver verwendet werden.

Syntax

```
CREATE EXTERNLOGIN login-name  
TO remote-server  
[ REMOTE LOGIN remote-user [ IDENTIFIED BY remote-password ] ]
```

Parameter

login-name Gibt den lokalen Loginnamen des Benutzers an. Wenn Sie integrierte Logins verwenden, ist der *login-name* der Datenbankbenutzer, dem der Windows-Benutzer oder die Gruppe zugeordnet wird.

TO-Klausel Die TO-Klausel gibt den Namen des Fremdservers an.

REMOTE LOGIN-Klausel Die Klausel REMOTE LOGIN gibt das entsprechende Benutzerkonto auf dem *remote-server* für den lokalen Benutzer *login-name* an. Werte für die REMOTE LOGIN-Klausel sind auf 128 Byte beschränkt.

IDENTIFIED BY-Klausel Die IDENTIFIED BY-Klausel gibt das *remote-password* für *remote-user* an. Die Kombination von *remote-user* und *remote-password* muss auf dem Fremdserver gültig sein.

Wenn Sie die IDENTIFIED BY-Klausel weglassen, wird das Kennwort als NULL an den Fremdserver gesendet. Wenn Sie hingegen IDENTIFIED BY "" (eine leere Zeichenfolge) angeben, wird das Kennwort als leere Zeichenfolge gesendet.

Bemerkungen

Standardmäßig benutzt SQL Anywhere die Namen und Kennwörter des Clients, wenn für diese Clients eine Verbindung mit einem Fremdserver hergestellt wird. CREATE EXTERNLOGIN weist einen alternativen Loginnamen und ein Kennwort zu, die bei der Kommunikation mit dem Fremdserver verwendet werden.

Die Klausel REMOTE LOGIN ist nur erforderlich, wenn der Fremdserver eine Benutzer-ID und ein Kennwort benötigt, um die Verbindung herzustellen. Wenn ein DBA ein externes Login ohne entferntes Login hat, kann er kontrollieren, wer Zugriffsrechte für den Fremdserver hat. Er kann die Zugriffsschicht auf dem Fremdserver so einstellen, dass für das Anmelden beim Fremdserver keine Benutzer-ID und kein Kennwort benötigt wird. Beispiel: Die Verzeichniszugriffsserverklasse erfordert ein externes Login für die Beschränkung des Zugriffs auf den Verzeichnisserver, aber ein entferntes Login wird nicht benötigt, weil der Verzeichnisserver keine Validierung von Benutzer-ID und Kennwort durchführt.

Das Kennwort wird intern in verschlüsselter Form gespeichert. Der *remote-server* muss dem lokalen Server durch einen Eintrag in der Tabelle ISYSSERVER bekannt sein.

Computer mit automatischem Kennwortablauf müssen periodische Aktualisierungen von Kennwörtern für externe Logins planen.

CREATE EXTERNLOGIN kann nicht aus einer Transaktion heraus verwendet werden.

Diese Anweisung wird unter Windows Mobile nicht unterstützt.

Privilegien

Sie müssen das MANAGE ANY USER-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „DROP EXTERNLOGIN-Anweisung“ auf Seite 808
- „CREATE SERVER-Anweisung“ auf Seite 701
- „Externe Logins erstellen (Sybase Central)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem fiktiven Beispiel wird beim Verbinden mit dem Server sybase1 ein lokaler Benutzer, DBA, dem Benutzer "sa" mit dem Kennwort "Plankton" zugeordnet.

```
CREATE EXTERNLOGIN DBA
TO sybase1
REMOTE LOGIN sa
IDENTIFIED BY Plankton;
```

CREATE FUNCTION-Anweisung [externer Aufruf]

Erstellt eine Schnittstelle zu einer nativen oder externen Funktion.

Syntax

```
CREATE [ OR REPLACE ] FUNCTION [ owner. ] function-name
( [ parameter, ... ] )
[ SQL SECURITY { INVOKER | DEFINER } ]
RETURNS data-type
[ [ NOT ] DETERMINISTIC ]
{ EXTERNAL NAME 'native-call'
| EXTERNAL NAME 'c-call' LANGUAGE { C_ESQL32 | C_ESQL64 | C_ODBC32 | C_ODBC64 }
| EXTERNAL NAME 'clr-call' LANGUAGE CLR
| EXTERNAL NAME 'perl-call' LANGUAGE PERL
| EXTERNAL NAME 'php-call' LANGUAGE PHP
| EXTERNAL NAME 'java-call' LANGUAGE JAVA }
```

parameter :
[IN] parameter-name data-type [DEFAULT expression]

```

native-call :
[ operating-system:]function-name@library

result-column :
column-name data-type

c-call :
[ operating-system:]function-name@library; ...

clr-call :
dll-name::function-name( param-type-1[, ... ] )

perl-call :
<file=perl-file> $sa_perl_return = perl-subroutine( $sa_perl_arg0[, ... ] )

php-call :
<file=php-file> print php-func( $argv[1][, ... ] )

java-call :
[package-name.]class-name.method-name method-signature

operating-system :
Unix

method-signature :
( [ field-descriptor, ... ] ) return-descriptor

field-descriptor und return-descriptor :
{ Z
  | B
  | S
  | I
  | J
  | F
  | D
  | C
  | V
  | [descriptor
  | Lclass-name;
}
```

Parameter

Parameternamen müssen den Regeln für Datenbankbezeichner folgen. Sie müssen einen gültigen SQL-Datentyp umfassen und das Schlüsselwort IN kann vorangestellt werden. Dieses Präfix weist darauf hin, dass das Argument ein Ausdruck ist, welcher der Funktion einen Wert zur Verfügung stellt. Allerdings sind Funktionsparameter standardmäßig IN.

Wenn Funktionen ausgeführt werden, müssen nicht alle Parameter angegeben werden. Wenn in der CREATE FUNCTION-Anweisung ein Standardwert bereitgestellt wird, werden den fehlenden Parametern die Standardwerte zugeordnet. Falls kein Argument vom Aufrufer angegeben wurde und kein Standardwert gesetzt ist, wird ein Fehler ausgegeben.

OR REPLACE-Klausel Wenn Sie OR REPLACE angeben, wird eine neue Funktion erstellt oder eine bestehende Funktion mit demselben Namen ersetzt. Diese Klausel ändert die Definition der Funktion, lässt aber vorhandene Privilegien unberührt. Sie können die OR REPLACE-Klausel nicht mit temporären Funktionen verwenden.

RETURNS-Klausel Verwenden Sie die Klausel RETURNS, um den Datentyp für das Ergebnis der Funktion anzugeben.

SQL SECURITY-Klausel Die SQL SECURITY-Klausel legt fest, ob die Funktion als INVOKER (der Benutzer, der die Funktion aufruft) oder als DEFINER (der Benutzer, dem die Funktion gehört) aufgerufen wird. Standardwert ist DEFINER. Bei externen Aufrufen richtet diese Klausel den Eigentümerkontext für nicht qualifizierte Objektreferenzen in der externen Umgebung ein.

Wenn SQL SECURITY INVOKER angegeben ist, wird mehr Speicher verwendet, weil für jeden Benutzer, der die Funktion aufruft, ein Vermerk erfolgen muss. Außerdem erfolgt bei SQL SECURITY INVOKER auch eine Namensauflösung als Aufrufer. Sie sollten daher mit Umsicht vorgehen und alle Objektnamen (Tabellen, Prozeduren etc.) mit ihrem richtigen Eigentümer qualifizieren. Beispiel: Der Benutzer user1 erstellt die folgende Funktion:

```
CREATE FUNCTION user1.myFunc()
  RETURNS INT
  SQL SECURITY INVOKER
BEGIN
  DECLARE res INT;
  SELECT COUNT(*) INTO res FROM table1;
  RETURN res;
END;
```

Wenn der Benutzer user2 versucht, diese Funktion auszuführen und die Tabelle user2.table1 *nicht existiert*, kommt es zu einem Tabellensuchfehler. Wenn eine Tabelle user2.table1 *existiert*, wird diese Tabelle anstelle der beabsichtigten user1.table1 verwendet. Um dies zu verhindern, qualifizieren Sie die Tabellenreferenz in der Anweisung (user1.table1 anstelle von table1).

[NOT] DETERMINISTIC-Klausel Verwenden Sie diese Klausel, um anzugeben, ob Funktionen deterministisch oder nicht deterministisch sind. Wenn diese Klausel weggelassen wird, gilt das deterministische Verhalten der Funktion als nicht angegeben (Standard).

Wenn eine Funktion als DETERMINISTIC deklariert wird, muss Sie jedes Mal, wenn sie mit denselben Parametern aufgerufen wird, denselben Wert zurückgeben.

Wenn eine Funktion als NOT DETERMINISTIC deklariert wird, garantiert sie nicht, dass bei identischen Parametern ein identischer Wert zurückgegeben wird. Eine Funktion, die als NOT DETERMINISTIC deklariert ist, wird bei jedem Aufruf in einer Abfrage neu berechnet. Diese Klausel muss verwendet werden, wenn bekannt ist, dass das Ergebnis der Funktion mit gegebenen Parametern unterschiedlich ausfallen kann.

Außerdem sollten Funktionen mit Nebenwirkungen wie der Änderung von Basisdaten als NOT DETERMINISTIC deklariert werden. Beispiel: Eine Funktion, die Primärschlüsselwerte generiert und in einer INSERT...SELECT-Anweisung verwendet wird, muss als NOT DETERMINISTIC deklariert werden:

```
CREATE FUNCTION keygen( increment INTEGER )
  RETURNS INTEGER
```

```
NOT DETERMINISTIC
BEGIN
  DECLARE keyval INTEGER;
  UPDATE counter SET x = x + increment;
  SELECT counter.x INTO keyval FROM counter;
  RETURN keyval
END
INSERT INTO new_table
SELECT keygen(1), ...
FROM old_table;
```

Funktionen können als DETERMINISTIC deklariert werden, wenn sie stets denselben Wert für gegebene Eingabeparameter zurückgeben.

EXTERNAL NAME *native-call*-Klausel

native-call :
[*operating-system*:]*function-name*@*library*; ...

operating-system : **Unix**

Hinweis

Die Klausel EXTERNAL NAME wird für TEMPORARY-Funktionen nicht unterstützt.

Eine Funktion, die die EXTERNAL NAME-Klausel ohne LANGUAGE-Attribut verwendet, definiert eine Schnittstelle zu einer nativen Funktion, die in einer Programmiersprache wie C geschrieben ist. Die native Funktion wird vom Datenbankserver in seinen Adressraum geladen.

Der *library*-Name kann eine Dateierweiterung enthalten. In der Regel handelt es sich um *.dll* unter Windows und *.so* unter Unix. Wenn keine Dateierweiterung angegeben ist, hängt die Software die plattformspezifische Dateierweiterung für Bibliotheken an. Beispiel:

```
CREATE FUNCTION mystring( IN instr LONG VARCHAR )
RETURNS LONG VARCHAR
EXTERNAL NAME 'mystring@mylib.dll;Unix:mystring@mylib.so';
```

Eine einfachere Methode, die oben genannte EXTERNAL NAME-Klausel mit plattformspezifischen Standardwerten zu schreiben, lautet wie folgt:

```
CREATE FUNCTION mystring( IN instr LONG VARCHAR )
RETURNS LONG VARCHAR
EXTERNAL NAME 'mystring@mylib';
```

Nach ihrem Aufruf wird die Bibliothek, die die Funktion enthält, in den Adressraum des Datenbankservers geladen. Die native Funktion wird als Teil des Servers ausgeführt. In diesem Fall wird, falls die Funktion einen Fehler verursacht, der Datenbankserver beendet. Aus diesem Grund wird empfohlen, Funktionen in einer externen Umgebung mit dem LANGUAGE-Attribut zu laden und auszuführen. Wenn eine Funktion einen Fehler in einer externen Umgebung verursacht, läuft der Datenbankserver weiter.

EXTERNAL NAME *c-call* LANGUAGE {C_ESQL32 | C_ESQL64 | C_ODBC32 | C_ODBC64 }-

Klausel Um eine kompilierte native C-Funktion in einer externen Umgebung statt im Datenbankserver aufzurufen, wird die gespeicherte Prozedur oder Funktion mit der Klausel EXTERNAL NAME definiert, gefolgt vom Attribut LANGUAGE, das C_ESQL32, C_ESQL64, C_ODBC32 oder C_ODBC64 angibt.

Wenn das LANGUAGE-Attribut angegeben ist, wird die Bibliothek, die die Funktion enthält, von einem externen Prozess geladen und die externe Funktion wird als Teil dieses externen Prozesses ausgeführt. In diesem Fall läuft der Datenbankserver weiter, falls die Funktion einen Fehler verursacht.

```
CREATE FUNCTION ODBCinsert(
    IN ProductName CHAR(30),
    IN ProductDescription CHAR(50)
)
RETURNS INT
EXTERNAL NAME 'ODBCexternalInsert@extodbc.dll'
LANGUAGE C_ODBC32;
```

EXTERNAL NAME *clr-call* LANGUAGE CLR-Klausel Um eine .NET-Funktion in einer externen Umgebung aufzurufen, wird die Funktionsschnittstelle mit einer EXTERNAL NAME-Klausel definiert, auf die das LANGUAGE CLR-Attribut folgt.

Eine gespeicherte CLR-Prozedur oder -Funktion verhält sich wie eine gespeicherte SQL-Prozedur oder -Funktion, abgesehen davon, dass der Code für die Prozedur oder Funktion in einer .NET-Sprache wie C# oder Visual Basic geschrieben ist und die Ausführung der Prozedur oder Funktion außerhalb des Datenbankservers stattfindet (d.h. innerhalb eines separaten .NET-Programms).

```
CREATE FUNCTION clr_interface(
    IN p1 INT,
    IN p2 UNSIGNED SMALLINT,
    IN p3 LONG VARCHAR)
RETURNS INT
EXTERNAL NAME 'CLRlib.dll::CLRproc.Run( int, ushort, string )'
LANGUAGE CLR;
```

EXTERNAL NAME *perl-call* LANGUAGE PERL-Klausel Um eine Perl-Funktion in einer externen Umgebung aufzurufen, wird die Funktionsschnittstelle mit einer EXTERNAL NAME-Klausel definiert, auf die das LANGUAGE PERL-Attribut folgt.

Eine gespeicherte Perl-Prozedur oder Funktion verhält sich wie eine gespeicherte SQL-Prozedur oder Funktion, abgesehen davon, dass der Code für die Prozedur oder Funktion in Perl geschrieben ist und die Ausführung der Prozedur oder Funktion außerhalb des Datenbankservers stattfindet (d.h. innerhalb einer Perl-Programminstanz).

```
CREATE FUNCTION PerlWriteToConsole( IN str LONG VARCHAR)
RETURNS INT
EXTERNAL NAME '<file=PerlConsoleExample>
    WriteToServerConsole( $sa_perl_arg0 )'
LANGUAGE PERL;
```

EXTERNAL NAME *php-call* LANGUAGE PHP-Klausel Um eine PHP-Funktion in einer externen Umgebung aufzurufen, wird die Funktionsschnittstelle mit einer EXTERNAL NAME-Klausel definiert, auf die das LANGUAGE PHP-Attribut folgt.

Eine gespeicherte PHP-Prozedur oder -Funktion verhält sich wie eine gespeicherte SQL-Prozedur oder Funktion, abgesehen davon, dass der Code für die Prozedur oder Funktion in PHP geschrieben ist und die Ausführung der Prozedur oder Funktion außerhalb des Datenbankservers stattfindet (d.h. innerhalb einer PHP-Programminstanz).

```
CREATE FUNCTION PHPPopulateTable()
RETURNS INT
```

```
EXTERNAL NAME '<file=ServerSidePHPExample> ServerSidePHPSub()'
LANGUAGE PHP;
```

EXTERNAL NAME 'java-call' LANGUAGE JAVA-Klausel Um eine Java-Methode in einer externen Umgebung aufzurufen, wird die Funktionsschnittstelle mit einer EXTERNAL NAME-Klausel definiert, auf die das LANGUAGE JAVA-Attribut folgt.

Eine gespeicherte Prozedur oder Funktion über die Java-Schnittstelle verhält sich wie eine gespeicherte SQL-Prozedur oder Funktion, abgesehen davon, dass der Code für die Prozedur oder Funktion in Java geschrieben ist und die Ausführung der Prozedur oder Funktion außerhalb des Datenbankservers stattfindet (d.h. in einer Java VM).

```
CREATE FUNCTION HelloDemo( IN name LONG VARCHAR )
RETURNS INT
EXTERNAL NAME 'Hello.main([Ljava/lang/String;)V'
LANGUAGE JAVA;
```

Die Deskriptoren für Argumente und Rückgabewerte von Java-Methoden haben die folgende Bedeutung:

Feldtyp	Java-Datentyp
B	byte
C	char
D	double
F	float
I	int
J	long
L <i>class-name</i> ;	Eine Instanz der Klasse <i>class-name</i> . Die Klasse muss voll qualifiziert sein und alle Punkte im Namen müssen durch das Zeichen / ersetzt werden. Zum Beispiel, java/lang/String .
S	short
V	void
Z	Boolescher Wert
[Verwenden Sie jeweils eine für jede Array-Dimension.

Bemerkungen

Die CREATE FUNCTION-Anweisung erstellt eine Funktion in der Datenbank. Sie können Funktionen für andere Benutzer erstellen, indem Sie einen Eigentümer angeben. Eine Funktion wird als Teil eines SQL-Ausdrucks aufgerufen.

Bei der Referenzierung einer temporären Tabelle durch mehrere Funktionen kann ein Problem entstehen, wenn die Definitionen der temporären Tabelle nicht konsistent sind und Anweisungen, die die Tabelle referenzieren, im Cache abgelegt sind.

Privilegien

Wenn Sie externe Funktionen erstellen möchten, deren Eigentümer Sie sind, müssen Sie die Systemprivilegien CREATE PROCEDURE und CREATE EXTERNAL REFERENCE haben.

Wenn Sie externe Funktionen erstellen möchten, deren Eigentümer andere Benutzer sind, benötigen Sie das CREATE ANY PROCEDURE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg sowie das CREATE EXTERNAL REFERENCE-Systemprivileg.

Zum Erstellen von temporären Funktionen ist kein Privileg erforderlich.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „ALTER FUNCTION-Anweisung“ auf Seite 465
- „CALL-Anweisung“ auf Seite 564
- „CREATE FUNCTION-Anweisung“ auf Seite 633
- „CREATE FUNCTION-Anweisung [Webdienst]“ auf Seite 624
- „CREATE PROCEDURE-Anweisung [externer Aufruf]“ auf Seite 661
- „DROP FUNCTION-Anweisung“ auf Seite 809
- „GRANT-Anweisung“ auf Seite 881
- „Unterstützung für externe Umgebungen in SQL Anywhere“ [*SQL Anywhere Server - Programmierung*]
- „SQL Anywhere-Schnittstelle für externe Aufrufe“ [*SQL Anywhere Server - Programmierung*]
- „Die externen ESQL- und ODBC-Umgebungen“ [*SQL Anywhere Server - Programmierung*]
- „Die externe CLR-Umgebung“ [*SQL Anywhere Server - Programmierung*]
- „Die externe Java-Umgebung“ [*SQL Anywhere Server - Programmierung*]
- „Die externe PHP-Umgebung“ [*SQL Anywhere Server - Programmierung*]
- „Die externe PERL-Umgebung“ [*SQL Anywhere Server - Programmierung*]
- „SQL-Datentypen“ auf Seite 95
- „Referenzen auf temporäre Tabellen innerhalb von Prozeduren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** CREATE FUNCTION für eine externe Sprachenumgebung ist eine Kernfunktion des SQL/2008-Standards, obwohl einige der in SQL Anywhere unterstützten Komponenten optionale SQL/2008-Sprachenfunktionen sind. Zu diesen Funktionen gehören folgende:
 - Die SQL SECURITY-Klausel ist die optionale Sprachenfunktion T324.
 - Die Möglichkeit zum Übergeben eines LONG VARCHAR-, LONG NVARCHAR- oder LONG BINARY-Werts an eine SQL-Funktion ist Sprachenfunktion T041.

- Die Unterstützung für LANGUAGE JAVA ist die optionale SQL/2008-Sprachenfunktion J621.
- Die Möglichkeit, ein Schema-Objekt innerhalb einer externen Funktion mit Anweisungen wie CREATE TABLE und DROP TRIGGER zu erstellen oder zu ändern, ist Sprachenfunktion T653.
- Die Möglichkeit zum Verwenden einer Dynamic-SQL-Anweisung innerhalb einer externen Funktion, einschließlich der Anweisungen CONNECT, EXECUTE IMMEDIATE, PREPARE und DESCRIBE, ist Sprachenfunktion T654.

Einige Klauseln der CREATE FUNCTION-Anweisung sind Erweiterungen des Herstellers. Es handelt sich dabei um die Folgenden:

- Die Unterstützung für C_ESQL32, C_ESQL64, C_ODBC32, C_ODBC64, CLR, PERL und PHP in der LANGUAGES-Klausel ist eine Erweiterung des Herstellers.
- Das Format von *external-call* wird durch die Implementierung festgelegt.
- Die optionale DEFAULT-Klausel für einen bestimmten Routinenparameter ist eine Erweiterung des Herstellers.
- Die optionale OR REPLACE-Klausel ist eine Erweiterung des Herstellers.
- **Transact-SQL** CREATE FUNCTION für eine externe Routine wird von Adaptive Server Enterprise unterstützt. Adaptive Server Enterprise unterstützt nur LANGUAGE JAVA als externe Umgebung (SQL/2008-Sprachenfunktion J621) für eine externe Funktion.

CREATE FUNCTION-Anweisung [Webdienst]

Erstellt eine Webservice-Funktion, die eine HTTP- oder SOAP-über-HTTP-Anforderung durchführt.

Syntax

```
CREATE [ OR REPLACE ] FUNCTION [ owner.]function-name ( [ parameter, ... ] )
RETURNS data-type
URL url-string
[ HEADER header-string ]
[ SOAPHEADER soap-header-string ]
[ TYPE {
  'HTTP[:{ GET | POST[:MIME-type] | PUT[:MIME-type] | DELETE | HEAD } ]' |
  'SOAP[:{ RPC | DOC } ]' } ]
[ NAMESPACE namespace-string ]
[ CERTIFICATE certificate-string ]
[ CLIENTPORT clientport-string ]
[ PROXY proxy-string ]
[ SET protocol-option-string ]
```

parameter :
[IN] parameter-name data-type [DEFAULT expression]

url-string :
' { HTTP | HTTPS | HTTPS_FIPS } ://[user:password@]hostname[:port][/path]'

```

protocol-option-string
[ http-option-list
, soap-option-list ]

http-option-list :
HTTP(
[ CH[UNK]={ ON | OFF | AUTO } ]
[ ; VER[SION]={ 1.0 | 1.1 };kto=number-of-seconds ]
[ REDIR(COUNT=count, STATUS=status-list)
]
)

soap-option-list :
SOAP(OP[ERATION]=soap-operation-name)

```

Parameter

Parameternamen müssen den Regeln für Datenbankbezeichner folgen. Sie müssen einen gültigen SQL-Datentyp umfassen und das Schlüsselwort IN kann vorangestellt werden. Dieses Präfix weist darauf hin, dass das Argument ein Ausdruck ist, welcher der Funktion einen Wert zur Verfügung stellt. Allerdings sind Funktionsparameter standardmäßig IN.

Wenn Funktionen ausgeführt werden, müssen nicht alle Parameter angegeben werden. Wenn in der CREATE FUNCTION-Anweisung ein Standardwert bereitgestellt wird, werden den fehlenden Parametern die Standardwerte zugeordnet. Falls kein Argument vom Aufrufer angegeben wurde und kein Standardwert gesetzt ist, wird ein Fehler ausgegeben.

OR REPLACE-Klausel Wenn Sie OR REPLACE angeben, wird eine neue Funktion erstellt oder eine bestehende Funktion mit demselben Namen ersetzt. Diese Klausel ändert die Definition der Funktion, lässt aber vorhandene Privilegien unberührt. Sie können die OR REPLACE-Klausel nicht mit temporären Funktionen verwenden.

RETURNS-Klausel Geben Sie einen der folgenden Werte an, um den Rückgabetyt für die SOAP- oder HTTP-Funktion zu definieren:

- CHAR
- VARCHAR
- LONG VARCHAR
- TEXT
- NCHAR
- NVARCHAR
- LONG NVARCHAR
- NTEXT
- XML
- BINARY
- VARBINARY
- LONG BINARY

Der zurückgegebene Wert ist der Hauptteil der HTTP-Antwort. Es werden keine HTTP-Headerinformationen aufgenommen. Wenn weitere Informationen wie z.B. Statusinformationen benötigt werden, verwenden Sie eine Prozedur statt einer Funktion.

Der Datentyp hat keinen Einfluss auf die Verarbeitung der HTTP-Antwort.

URL-Klausel Verwenden Sie die URL-Klausel nur, wenn Sie eine Clientfunktion für HTTP- oder SOAP-Webdienste festlegen. Die URL-Klausel gibt den URI des Webdienstes an. Die optionalen Benutzernamen- und Kennwortparameter bieten eine Möglichkeit, die für die HTTP Basic Authentication erforderlichen Anmeldeinformationen zu liefern. Die HTTP Basic Authentication Base-64 verschlüsselt die Benutzer- und Kennwortinformationen und übergibt sie an den Header "Authentication" der HTTP-Anforderung.

Die Angabe von HTTPS_FIPS zwingt das System, die FIPS-zertifizierten Bibliotheken zu verwenden. Wenn HTTPS_FIPS angegeben wird, aber keine FIPS-zertifizierten Bibliotheken vorhanden sind, werden anstelle dessen Bibliotheken verwendet, die nicht FIPS-zertifiziert sind.

Für Funktionen des Typs HTTP:GET können Abfrageparameter innerhalb der URL-Klausel angegeben werden. Außerdem werden sie automatisch aus Parametern generiert, die an eine Prozedur übergeben werden.

```
URL 'http://localhost/service?parm=1'
```

HEADER-Klausel Wenn Sie HTTP-Webdienst-Clientfunktionen erstellen, verwenden Sie diese Klausel, um Headereinträge der HTTP-Anforderung hinzuzufügen oder zu ändern. Für HTTP-Header können nur druckbare ASCII-Zeichen angegeben werden, deren Groß- und Kleinschreibung nicht berücksichtigt wird.

SOAPHEADER-Klausel Wenn Sie einen SOAP-Webdienst als Funktion deklarieren, verwenden Sie diese Klausel, um einen oder mehrere Headereinträge der SOAP-Anforderung anzugeben. Ein SOAP-Header kann als statische Konstante deklariert oder dynamisch mithilfe des Parameterersetzungsmechanismus (IN-, OUT- oder INOUT-Parameter für hd1, hd2 etc. deklarieren) gesetzt werden. Eine Webdienstfunktion kann einen oder mehrere IN-Modus-Ersetzungsparameter definieren, aber keinen INOUT- oder OUT-Ersetzungsparameter.

TYPE-Klausel Die TYPE-Klausel gibt das für die Webdienstanforderung verwendete Format an. SOAP:RPC wird verwendet, wenn SOAP angegeben oder keine TYPE-Klausel enthalten ist. HTTP:POST wird verwendet, wenn HTTP angegeben ist.

Die TYPE-Klausel ermöglicht die Angabe eines MIME-Typs für HTTP:GET-, HTTP:POST- und HTTP:PUT-Typen. Mithilfe des Werts für den *MIME-type* wird der Anforderungs-Header "Content-Type" eingerichtet und der Betriebsmodus so eingestellt, dass nur ein einzelner Aufrufparameter den Hauptteil der Anforderung auffüllen kann. Wenn ein Webdienstauftrag einer gespeicherten Funktion erfolgt, nachdem Parameterersetzungen verarbeitet wurden, darf kein oder nur ein einziger Parameter verbleiben. Der Aufruf einer Webdienstfunktion mit NULL-Wert oder keinem Parameter (nach Ersetzungen) ergibt eine Anfrage ohne Hauptteil und mit einer Inhaltslänge von Null. Parameternamen und -werte (Mehrfachparameter sind erlaubt) werden innerhalb des Hauptteils der HTTP-Anforderungen URL-kodiert.

Zu den typischen MIME-Typen gehören folgende:

- text/plain
- text/html
- text/xml

Die Schlüsselwörter für die TYPE-Klausel haben die folgende Bedeutung:

- **HTTP:GET** Standardmäßig verwendet dieser Typ den MIME-Typ `application/x-www-form-urlencoded` zum Kodieren von in der URL angegebenen Parametern.

Die folgende Anforderung wird beispielsweise erzeugt, wenn ein Client eine Anforderung aus der URL `http://localhost/WebServiceName?arg1=param1&arg2=param2` sendet:

```
GET /WebServiceName?arg1=param1&arg2=param2 HTTP/1.1
// <End of Request - NO BODY>
```

- **HTTP:POST** Standardmäßig verwendet dieser Typ den MIME-Typ `application/x-www-form-urlencoded` zum Kodieren von im Hauptteil einer POST-Anforderung angegebenen Parametern. URL-Parameter werden im Hauptteil gespeichert.

Die folgende Anforderung wird beispielsweise erzeugt, wenn ein Client eine Anforderung aus der URL `http://localhost/WebServiceName?arg1=param1&arg2=param2` sendet:

```
POST /WebServiceName HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 19

arg1=param1&arg2=param2
// <End of Request>
```

- **HTTP:PUT** HTTP:PUT ist ähnlich wie HTTP:POST, aber der HTTP:PUT-Typ verfügt nicht über einen Standardmedientyp.

Das folgende Beispiel zeigt, wie sie eine allgemeine Clientprozedur für den Upload von Daten in einen SQL Anywhere-Server konfigurieren, der das Beispiel *put_data.sql* ausführt:

```
ALTER PROCEDURE CPUT("data" LONG VARCHAR, resnm LONG VARCHAR, mediatype
LONG VARCHAR)
  URL 'http://localhost/resource/!resnm'
  TYPE 'HTTP:PUT:!mediatype';

CALL CPUT('hello world', 'hello', 'text/plain');
```

- **HTTP:DELETE** Eine Webdienst-Clientprozedur kann so konfiguriert werden, dass eine auf einem Server befindliche Ressource gelöscht wird. Das Angeben des Medientyps ist optional.

Das folgende Beispiel zeigt, wie Sie eine allgemeine Clientprozedur für das Löschen von Daten auf einem SQL Anywhere-Server konfigurieren, auf dem das Beispiel *put_data.sql* ausgeführt wird:

```
ALTER PROCEDURE CDEL(resnm LONG VARCHAR, mediatype LONG VARCHAR)
  URL 'http://localhost/resource/!resnm'
  TYPE 'HTTP:DELETE:!mediatype';

CALL CDEL('hello', 'text/plain');
```

- **HTTP:HEAD** Die Head-Methode ist identisch mit einer GET-Methode, aber der Server gibt keinen Hauptteil zurück. Ein Medientyp kann angegeben werden.

```
ALTER PROCEDURE CHEAD(resnm LONG VARCHAR)
  URL 'http://localhost/resource/!resnm'
  TYPE 'HTTP:HEAD';

CALL CHEAD('hello');
```

- **SOAP:RPC** Dieser Typ setzt den Content-Type-Header auf 'text/xml'. SOAP-Vorgänge und-Parameter werden in SOAP Envelope-XML-Dokumenten gekapselt.
- **SOAP:DOC** Dieser Typ setzt den Content-Type-Header auf 'text/xml'. Er ähnelt dem SOAP:RPC-Typ, ermöglicht jedoch das Senden umfangreicherer Datentypen. SOAP-Vorgänge und-Parameter werden in SOAP Envelope-XML-Dokumenten gekapselt.

Das Angeben eines MIME-Typs für die TYPE-Klausel setzt automatisch den Content-Type-Header auf diesen MIME-Typ.

NAMESPACE-Klausel Gilt nur für SOAP-Clientfunktionen. Diese Klausel kennzeichnet den Methoden-Namespace, der üblicherweise sowohl bei SOAP:RPC- als auch bei SOAP:DOC-Anforderungen erforderlich ist. Der die Anforderung verarbeitende SOAP-Server verwendet diesen Namespace, um die Namen von Entitäten im Hauptteil der SOAP-Anforderung zu interpretieren. Der Namespace kann anhand der WSDL-Beschreibung (WSDL = Web Services Description Language) des SOAP-Diensts ermittelt werden, die der Webdienstserver zur Verfügung stellt. Der Standardwert ist die URL der Funktion bis zur optionalen Pfadkomponente (die aber nicht eingeschlossen ist).

CERTIFICATE-Klausel Um eine sichere (HTTPS-)Anforderung durchführen zu können, muss der Client Zugriff auf das vom HTTPS-Server verwendete Zertifikat haben. Die benötigten Informationen werden in einer Zeichenfolge von durch Semikola getrennten Schlüssel/Werte-Paaren angegeben. Sie können mithilfe des Dateischlüssels den Dateinamen für das Zertifikat angeben oder mithilfe des Zertifikatschlüssels das Serverzertifikat in einer Zeichenfolge angeben. Sie können nicht sowohl einen Dateischlüssel als auch einen Zertifikatschlüssel angeben. Folgende Schlüssel sind verfügbar:

Schlüssel	Abkürzung	Beschreibung
file		Der Dateiname des Zertifikats
certificate	cert	Das Zertifikat selbst
certificate_name	cert_name	Der Name eines in der Datenbank gespeicherten Zertifikats
company	co	Die im Zertifikat angegebene Organisation
unit		Die im Zertifikat angegebene Organisationseinheit
name		Der im Zertifikat angegebene allgemeine Name

Zertifikate sind nur bei Anforderungen erforderlich, die entweder an den HTTPS-Server gerichtet sind oder von einem nicht-sicheren zu einem sicheren Server umgeleitet werden können. Nur PEM-formatierte Zertifikate werden unterstützt.

CLIENTPORT-Klausel Kennzeichnet die Portnummer, auf der die HTTP-Clientfunktion mithilfe von TCP/IP kommuniziert. Sie steht für Verbindungen zur Verfügung, die durch ein Firewall-Schutzsystem

geleitet werden, da Firewalls entsprechend dem TCP/UDP-Port Filter anwenden. Die Klausel ist nur in diesem Fall empfehlenswert. Sie können eine einzelne Portnummer, Bereiche von Portnummern oder eine Kombination von beiden angeben, Beispiel: `CLIENTPORT '85,90-97'`

PROXY-Klausel Gibt den URI eines Proxyservers an. Wird verwendet, wenn der Client über einen Proxyserver auf das Netzwerk zugreifen muss. Diese Klausel zeigt an, dass die Funktion eine Verbindung zum Proxyserver herstellen soll, um die Anforderung durch ihn zum Webdienst zu senden.

SET-Klausel Gibt Optionen für protokollspezifisches Verhalten für HTTP und SOAP an. Die folgende Liste beschreibt die unterstützten SET-Optionen. CHUNK und VERSION gelten für das HTTP-Protokoll und OPERATION gilt für das SOAP-Protokoll. Die Ersetzung von Parametern wird bei dieser Klausel unterstützt.

- **'HTTP(CH[UNK]=option)'** (HTTP oder SOAP) Mit dieser Option können Sie angeben, ob das Aufteilen in Abschnitte verwendet werden soll. Dies ermöglicht es, HTTP-Nachrichten in mehrere Abschnitte aufzuteilen. Mögliche Werte sind: ON (immer Abschnitte verwenden), OFF (keine Abschnitte) und AUTO (Abschnitte nur verwenden, wenn der Inhalt, ausgenommen automatisch generiertes Markup, 8196 Byte überschreitet). Die folgende SET-Klausel aktiviert z.B. das Aufteilen in Abschnitte:

```
SET 'HTTP(CHUNK=ON)'
```

Wenn die CHUNK-Option nicht angegeben ist, ist das Standardverhalten AUTO. Wenn eine in Abschnitte aufgeteilte Anforderung im AUTO-Modus mit dem Status 505 (HTTP Version Not Supported - HTTP-Version nicht unterstützt), mit 501 (Not Implemented - Nicht implementiert) oder mit 411 (Length Required - Länge erforderlich) fehlschlägt, wiederholt der Client die Anforderung ohne eine in Abschnitte aufgeteilte Übertragungskodierung.

Definieren Sie die CHUNK-Option mit OFF (keine Aufteilung), wenn der HTTP-Server in Abschnitte aufgeteilte übertragungskodierte Anforderungen nicht unterstützt.

Da der CHUNK-Modus die Übertragungskodierung ab HTTP-Version 1.1 unterstützt, erfordert die Definition von CHUNK mit ON, dass die Version (VER) auf 1.1 oder gar nicht gesetzt wird. Im letzten Fall wird 1.1 als die Standardversion verwendet.

- **'HTTP(VER[SION]=ver;kto=number-of-seconds)'** (HTTP oder SOAP) Mit dieser Option können Sie die Version des HTTP-Protokolls angeben, das für das Format der HTTP-Nachricht verwendet wird. Beispiel: Die folgende SET-Klausel setzt die HTTP-Version auf 1.1:

```
SET 'HTTP(VERSION=1.1)'
```

Mögliche Werte sind 1.0 und 1.1. Wenn VERSION nicht angegeben ist:

- Wenn CHUNK auf ON gesetzt ist, wird die HTTP-Version 1.1 verwendet.
- Wenn CHUNK auf OFF gesetzt ist, wird die HTTP-Version 1.0 verwendet.
- Wenn CHUNK auf AUTO gesetzt ist, wird entweder 1.0 oder 1.1 verwendet, abhängig davon, ob der Client im CHUNK-Modus sendet.

- **kto=number-of-seconds** Gibt die Kriterien für das Keepalive-Timeout an (**kto**) und ermöglicht es so einer Webclient-Prozedur, eine HTTP/HTTPS-Keepalive-Verbindung für einen bestimmten Zeitraum zu instanziiieren und im Cache abzulegen. Damit eine HTTP-Keepalive-Verbindung im Cache abgelegt werden kann, muss die HTTP-Version auf 1.1 gesetzt werden und kto auf einen anderen Wert als Null. kto kann besonders bei HTTPS-Verbindungen nützlich sein, wenn Sie einen erheblichen Performanceunterschied zwischen HTTP- und HTTPS-Verbindungen feststellen. Eine Datenbankverbindung kann nur eine einzelne HTTP-Keepalive-Verbindung im Cache ablegen. Bei nachfolgenden Aufrufen einer Webclient-Prozedur mit demselben URI wird die Keepalive-Verbindung wiederverwendet. Der ausführende Webclient-Aufruf muss deshalb einen URI enthalten, dessen Schema, Zielhost und Port mit denjenigen des im Cache abgelegten URI übereinstimmen, und in der HEADER-Klausel darf nicht "Connection: close" angegeben werden. Wenn kto nicht angegeben oder auf Null gesetzt wird, werden HTTP/HTTPS-Verbindungen werden nicht im Cache abgelegt.
- **'REDIR(COUNT=count, STATUS=status-list)'** (HTTP oder SOAP) Die HTTP-Antwortstatuscodes wie **302 Found** und **303 See Other** werden verwendet, um Webanwendungen zu einem neuen URI umzuleiten, vor allem, wenn HTTP POST ausgeführt wurde. Eine Clientanforderung kann zum Beispiel folgendermaßen aussehen:

```
GET /people/alice HTTP/1.1
Host: www.example.com
Accept: text/html, application/xhtml+xml
Accept-Language: en, de
```

Die Webserverantwort kann folgendermaßen aussehen:

```
HTTP/1.1 302 Found
Location: http://www.example.com/people/alice.en.html
```

Als Antwort würde der Client eine andere HTTP-Anforderung an den neuen URI senden. Mit der REDIR-Option können Sie die maximale Anzahl der zulässigen Umleitungen steuern und festlegen, welche HTTP-Antwortstatuscodes automatisch umgeleitet werden.

SET 'REDIR(count=3, status=301,307)' ermöglicht beispielsweise eine maximale Anzahl von 3 Umleitungen und eine Umleitung für die Status 301 und 307. Wenn einer der anderen Umleitungsstatuscodes wie z.B. 302 oder 303 empfangen wird, wird ein Fehler ausgegeben (SQLCODE-983).

Die Anzahl (*count*) der Umleitungen ist standardmäßig auf 5 begrenzt. Standardmäßig wird eine HTTP-Clientprozedur automatisch als Antwort zu allen HTTP-Umleitungsstatuscodes (301, 302, 303, 307) umgeleitet. Um keine Umleitungsstatuscodes zuzulassen, verwenden Sie SET REDIR (COUNT=0). In diesem Modus führt die Antwort auf eine Umleitung nicht zu einem Fehler (SQLCODE-983). Stattdessen wird eine Ergebnismenge mit dem HTTP-Status und Antwort-Headern zurückgegeben. Dies ermöglicht es dem Aufrufer, die Anforderung basierend auf dem URI im **Location**-Header bedingt neu auszugeben.

Eine Webdienstprozedur, die eine POST HTTP-Methode festlegt und den Status 303 See Other empfängt, gibt mit der GET HTTP-Methode eine Umleitungsanforderung aus.

Der **Location**-Header kann entweder einen absoluten Pfad oder einen relativen Pfad enthalten. Die HTTP-Clientprozedur verarbeitet beide. Der Header kann auch Abfrageparameter enthalten. Diese

werden an den umgeleiteten Speicherort weitergeleitet. Beispiel: Wenn der Header Parameter wie z.B. die folgenden enthält, umfasst der nachfolgende GET- oder ein POST-Wert diese Parameter.

```
Location: alternate_service?a=1&b=2
```

Im oben gezeigten Beispiel lauten die Abfrageparameter a=1&b=2.

- **'SOAP(OP[ERATION]=soap-operation-name)'** (Nur SOAP) Mit dieser Option können Sie den Namen des SOAP-Vorgangs angeben, falls er sich vom Namen der Prozedur unterscheidet, die Sie erstellen. Der Wert für OPERATION ist mit dem Namen eines entfernten Prozeduraufrufs vergleichbar. Beispiel: Wenn Sie eine Prozedur mit dem Namen accounts_login erstellen wollen, die einen SOAP-Vorgang mit dem Namen login aufruft, geben Sie in etwa Folgendes ein:

```
CREATE FUNCTION accounts_login(
    name LONG VARCHAR,
    pwd LONG VARCHAR );
SET 'SOAP(OPERATION=login)';
```

Wenn die OPERATION-Option nicht angegeben ist, muss der Name des SOAP-Vorgangs mit dem Namen der Prozedur übereinstimmen, die Sie erstellen.

Die folgende Anweisung zeigt, wie mehrere Einstellungen der *protocol-option-string* in derselben SET-Klausel kombiniert werden:

```
CREATE FUNCTION accounts_login(
    name LONG VARCHAR,
    pwd LONG VARCHAR );
SET 'HTTP ( CHUNK=ON; VERSION=1.1 ), SOAP( OPERATION=login )'
...

```

Bemerkungen

Die CREATE FUNCTION-Anweisung erstellt eine Webdienstfunktion in der Datenbank. Eine Funktion kann für einen anderen Benutzer erstellt werden, indem der Name eines Eigentümers angegeben wird.

Parameterwerte werden als Teil der Anforderung weitergegeben. Die verwendete Syntax hängt vom Typ der Anforderung ab. Bei HTTP:GET werden die Parameter als Teil der URL weitergegeben. Bei HTTP:POST-Anforderungen werden die Werte in den Hauptteil der Anforderung platziert. Parameter von SOAP-Anforderungen werden immer im Anforderungshauptteil gebündelt.

Privilegien

Sie müssen das CREATE PROCEDURE-Systemprivileg haben, um Funktionen erstellen zu können, deren Eigentümer Sie sind.

Sie müssen das CREATE ANY PROCEDURE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben, um Funktionen erstellen zu können, deren Eigentümer andere Benutzer sind.

Sie müssen das CREATE EXTERNAL REFERENCE-Systemprivileg haben, um eine externe Funktion erstellen zu können.

Zum Erstellen von temporären Funktionen ist kein Privileg erforderlich.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „SQL-Datentypen“ auf Seite 95
- „Praktische Einführung: Verwenden von SQL Anywhere für den Zugriff auf einen SOAP/DISH-Dienst“ [*SQL Anywhere Server - Programmierung*]
- „HTTP- und SOAP-Anforderungsstrukturen“ [*SQL Anywhere Server - Programmierung*]
- „HTTP-Anforderungsheader verwalten“ [*SQL Anywhere Server - Programmierung*]
- „ALTER FUNCTION-Anweisung“ auf Seite 465
- „CREATE FUNCTION-Anweisung“ auf Seite 633
- „CREATE FUNCTION-Anweisung [externer Aufruf]“ auf Seite 617
- „CREATE PROCEDURE-Anweisung“ auf Seite 681
- „CREATE PROCEDURE-Anweisung [Webdienste]“ auf Seite 670
- „DROP FUNCTION-Anweisung“ auf Seite 809
- „RETURN-Anweisung“ auf Seite 1004
- „An Webdienste übergebene Variable“ [*SQL Anywhere Server - Programmierung*]
- „Praktische Einführung: Webserver erstellen und über einen Webclient darauf zugreifen“ [*SQL Anywhere Server - Programmierung*]
- „SQL Anywhere als HTTP-Webserver“ [*SQL Anywhere Server - Programmierung*]
- „Webclient-Anwendungsentwicklung“ [*SQL Anywhere Server - Programmierung*]
- „remote_idle_timeout-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** Wird von Adaptive Server Enterprise nicht unterstützt.

Beispiele

Die folgende Anweisung erstellt eine Funktion namens `cli_test1`, die Bilder aus dem auf localhost laufenden `get_picture`-Dienst zurückgibt:

```
CREATE FUNCTION cli_test1( image LONG VARCHAR )
RETURNS LONG BINARY
URL 'http://localhost/get_picture'
TYPE 'HTTP:GET';
```

Die folgende Anweisung gibt eine HTTP-Anforderung aus mit der URL `http://localhost/get_picture?image=widget`:

```
SELECT cli_test1( 'widget' );
```

Die folgende Anweisung verwendet einen Ersetzungsparameter, damit die Anforderungs-URL als Eingabeparameter übergeben werden kann. Die sichere HTTPS-Anforderung verwendet ein in der Datenbank gespeichertes Zertifikat. Die SET-Klausel wird verwendet, um den CHUNK-Modus für die Übertragungskodierung zu deaktivieren.

```
CREATE CERTIFICATE client_cert
FROM FILE 'C:\\Users\\Public\\Documents\\SQL Anywhere 16\\Samples\\
\\Certificates\\rsaroot.crt';
```

```
CREATE FUNCTION cli_test2( image LONG VARCHAR, myurl LONG VARCHAR )
RETURNS LONG BINARY
URL '!myurl'
CERTIFICATE 'certificate_name=client_cert'
TYPE 'HTTPS:GET'
SET 'HTTP(CH=OFF)'
HEADER 'ASA-ID';
```

Die folgende Anweisung gibt eine HTTP-Anforderung aus mit der URL *http://localhost/get_picture?image=widget*:

```
CREATE VARIABLE a_binary LONG BINARY;
SET a_binary = cli_test2( 'widget', 'https://localhost/get_picture' );
SELECT a_binary;
```

CREATE FUNCTION-Anweisung

Erstellt eine benutzerdefinierte SQL-Funktion in der Datenbank.

Syntax

```
CREATE [ OR REPLACE | TEMPORARY ] FUNCTION [ owner.]function-name
( [ parameter, ... ] )
[ SQL SECURITY { INVOKER | DEFINER } ]
RETURNS data-type
[ ON EXCEPTION RESUME ]
[ [ NOT ] DETERMINISTIC ]
compound-statement | AS tsq-compound-statement | AT location-string
```

parameter :

```
[ IN ] parameter-name data-type [ DEFAULT expression ]
```

tsq-compound-statement :

```
sql-statement
sql-statement
...
```

Parameter

Parameternamen müssen den Regeln für Datenbankbezeichner folgen. Sie müssen einen gültigen SQL-Datentyp umfassen und das Schlüsselwort IN kann vorangestellt werden. Dieses Präfix weist darauf hin, dass das Argument ein Ausdruck ist, welcher der Funktion einen Wert zur Verfügung stellt. Allerdings sind Funktionsparameter standardmäßig IN.

Wenn Funktionen ausgeführt werden, müssen nicht alle Parameter angegeben werden. Wenn in der CREATE FUNCTION-Anweisung ein Standardwert bereitgestellt wird, werden den fehlenden Parametern die Standardwerte zugeordnet. Falls kein Argument vom Aufrufer angegeben wurde und kein Standardwert gesetzt ist, wird ein Fehler ausgegeben.

OR REPLACE-Klausel Wenn Sie CREATE OR REPLACE FUNCTION angeben, wird eine neue Funktion erstellt oder eine bestehende Funktion mit demselben Namen ersetzt. Beim Ersetzen einer Funktion wird die Definition der Funktion geändert, aber die vorhandenen Privilegien bleiben unberührt.

Sie können die OR REPLACE-Klausel nicht mit temporären Funktionen verwenden.

TEMPORARY-Schlüsselwort Die Angabe von CREATE TEMPORARY FUNCTION bedeutet, dass die Funktion lediglich für die Verbindung sichtbar ist, die sie erstellt hat, und dass sie automatisch gelöscht wird, wenn die Verbindung beendet wird. Temporäre Funktionen können auch explizit gelöscht werden. Sie können kein ALTER, GRANT oder REVOKE für sie ausführen, und im Gegensatz zu anderen Funktionen werden temporäre Funktionen nicht im Katalog oder Transaktionslog aufgezeichnet.

Temporäre Funktionen werden mit den Privilegien ihres Erstellers (des aktuellen Benutzers) oder des angegebenen Eigentümers ausgeführt. Sie können unter folgenden Voraussetzungen einen Eigentümer für eine temporäre Funktion festlegen:

- Wenn die temporäre Funktion in einer permanenten gespeicherten Prozedur erstellt wird.
- Wenn der Eigentümer der temporären Funktion mit dem der permanenten gespeicherten Prozedur identisch ist.

Um den Eigentümer einer temporären Funktion zu löschen, müssen Sie erst die temporäre Funktion löschen.

Temporäre Funktionen können erstellt und gelöscht werden, wenn eine Verbindung mit einer schreibgeschützten Datenbank besteht.

Sie können die OR REPLACE-Klausel nicht mit temporären Funktionen verwenden.

RETURNS-Klausel Verwenden Sie die Klausel RETURNS, um den Datentyp für das Ergebnis der Funktion anzugeben.

SQL SECURITY-Klausel Die SQL SECURITY-Klausel legt fest, ob die Funktion als INVOKER (der Benutzer, der die Funktion aufruft) oder als DEFINER (der Benutzer, dem die Funktion gehört) aufgerufen wird. Standardwert ist DEFINER.

ON EXCEPTION RESUME-Klausel Fehlerbehandlung wie in Transact-SQL verwenden.

[NOT] DETERMINISTIC-Klausel Verwenden Sie diese Klausel, um anzugeben, ob Funktionen deterministisch oder nicht deterministisch sind. Wenn diese Klausel weggelassen wird, gilt das deterministische Verhalten der Funktion als nicht angegeben (Standard).

Wenn eine Funktion als DETERMINISTIC deklariert wird, muss Sie jedes Mal, wenn sie mit denselben Parametern aufgerufen wird, denselben Wert zurückgeben.

Wenn eine Funktion als NOT DETERMINISTIC deklariert wird, garantiert sie nicht, dass bei identischen Parametern ein identischer Wert zurückgegeben wird. Eine Funktion, die als NOT DETERMINISTIC deklariert ist, wird bei jedem Aufruf in einer Abfrage neu berechnet. Diese Klausel muss verwendet werden, wenn bekannt ist, dass das Ergebnis der Funktion mit gegebenen Parametern unterschiedlich ausfallen kann.

Außerdem sollten Funktionen mit Nebenwirkungen wie der Änderung von Basisdaten als NOT DETERMINISTIC deklariert werden. In diesem Beispiel handelt es sich um eine Funktion, die Primärschlüsselwerte generiert und in einer INSERT...SELECT-Anweisung verwendet wird. Sie wird als NOT DETERMINISTIC deklariert. Die referenzierten Tabellen sind fiktiv.

```
CREATE FUNCTION keygen( increment INTEGER )  
  RETURNS INTEGER
```

```

NOT DETERMINISTIC
BEGIN
    DECLARE keyval INTEGER;
    UPDATE counter SET x = x + increment;
    SELECT counter.x INTO keyval FROM counter;
    RETURN keyval
END
INSERT INTO new_table
SELECT keygen(1), ...
FROM old_table;

```

Funktionen können als DETERMINISTIC deklariert werden, wenn sie stets denselben Wert für gegebene Eingabeparameter zurückgeben.

compound-statement Eine Gruppe von SQL-Anweisungen in BEGIN und END, getrennt durch Semikola.

AS-Klausel Die *tsql-compound-statement* ist ein Batch von Transact-SQL-Anweisungen.

AT-Klausel Erstellen Sie eine Proxy-Funktion in der aktuellen Datenbank für eine entfernte Funktion, die durch die *location-string* angegeben wird. Die AT-Klausel unterstützt das Semikolon (;) als Feldbegrenzer in *location-string*. Wenn kein Semikolon vorhanden ist, gilt der Punkt als Feldbegrenzer. Mit Semikolon-Trennzeichen können Dateinamen und Erweiterungen in den Feldern für Datenbank und Eigentümer verwendet werden.

Die Zeichenfolge in der AT-Klausel kann auch lokale oder globale Variablennamen in geschweiften Klammern enthalten ({*variable-name*}). Der SQL-Variablenname muss vom Typ CHAR, VARCHAR oder LONG VARCHAR sein. Eine AT-Klausel mit 'bostonase.master.dbo.{@myfunction}' zeigt beispielsweise an, dass @myfunction eine SQL-Variable ist und dass der aktuelle Inhalt der @myfunction-Variablen beim Anwenden der entfernten Prozedur ersetzt werden muss.

Eine Proxy-Funktion kann jeden Datentyp zurückgeben außer DECIMAL, NUMERIC, LONG VARCHAR, LONG NVARCHAR, LONG BINARY, XML sowie räumliche Datentypen.

Bemerkungen

Die CREATE FUNCTION-Anweisung erstellt eine Funktion in der Datenbank. Eine Funktion kann für einen anderen Benutzer erstellt werden, indem der Name eines Eigentümers angegeben wird. Eine Funktion kann, entsprechend den Privilegien, genauso verwendet werden wie andere Nicht-Aggregatfunktionen.

Wenn SQL SECURITY INVOKER angegeben ist, wird mehr Speicher verwendet, weil für jeden Benutzer, der die Prozedur aufruft, ein Vermerk erfolgen muss. Außerdem wird, wenn SQL SECURITY INVOKER angegeben ist, die Namensauflösung als Aufrufer ausgeführt. Achten Sie daher darauf, alle Objektnamen (Tabellen, Prozeduren, usw.) mit dem jeweiligen Eigentümer zu qualifizieren.

Alle Funktionen werden als deterministisch behandelt, sofern sie nicht als NOT DETERMINISTIC deklariert sind. Deterministische Funktionen geben für die gleichen Parameter ein konsistentes Ergebnis zurück und weisen keine Nebenwirkungen auf. Der Datenbankserver nimmt also an, dass zwei aufeinanderfolgende Aufrufe derselben Funktion mit denselben Parametern dasselbe Ergebnis zurückgeben und keine unerwünschten Nebeneffekte auf die Semantik der Abfrage haben.

Wenn eine Funktion eine Ergebnismenge zurückgibt, kann sie nicht zusätzlich Ausgabeparameter setzen oder einen Rückgabewert zurückgeben.

Privilegien

Sie müssen das CREATE PROCEDURE-Systemprivileg haben, um Funktionen erstellen zu können, deren Eigentümer Sie sind.

Sie müssen das CREATE ANY PROCEDURE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben, um Funktionen erstellen zu können, deren Eigentümer andere Benutzer sind.

Sie müssen außerdem das CREATE EXTERNAL REFERENCE-Systemprivileg haben, um eine externe Funktion erstellen zu können.

Zum Erstellen von temporären Funktionen ist kein Privileg erforderlich.

Nebenwirkungen

Automatisches Festschreiben, auch für temporäre Funktionen.

Siehe auch

- „ALTER FUNCTION-Anweisung“ auf Seite 465
- „CREATE FUNCTION-Anweisung [externer Aufruf]“ auf Seite 617
- „CREATE FUNCTION-Anweisung [Webdienst]“ auf Seite 624
- „BEGIN-Anweisung“ auf Seite 557
- „CREATE PROCEDURE-Anweisung“ auf Seite 681
- „DROP FUNCTION-Anweisung“ auf Seite 809
- „RETURN-Anweisung“ auf Seite 1004
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Benutzerdefinierte Funktionen erstellen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „SQL-Datentypen“ auf Seite 95
- „Transact-SQLBatches“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** CREATE FUNCTION ist eine Kernfunktion des SQL/2008-Standards, obwohl einige der in SQL Anywhere unterstützten Komponenten optionale SQL-Sprachenfunktionen sind. Zu diesen Funktionen gehören:
 - Die SQL SECURITY-Klausel ist die optionale Sprachenfunktion T324.
 - Die Möglichkeit zum Übergeben eines LONG VARCHAR-, LONG NVARCHAR- oder LONG BINARY-Werts an eine SQL-Funktion ist Sprachenfunktion T041.
 - Die Möglichkeit, ein Schema-Objekt innerhalb einer SQL-Funktion mit Anweisungen wie CREATE TABLE und DROP TRIGGER zu erstellen oder zu ändern, ist Sprachenfunktion T651.
 - Die Möglichkeit zum Verwenden einer Dynamic-SQL-Anweisung innerhalb einer SQL-Funktion, einschließlich der Anweisungen CONNECT, EXECUTE IMMEDIATE, PREPARE und DESCRIBE, ist Sprachenfunktion T652.

Einige Klauseln der CREATE FUNCTION-Anweisung sind Erweiterungen des Herstellers. Es handelt sich dabei um die folgenden:

- Die TEMPORARY-Klausel.
- Die ON EXCEPTION RESUME-Klausel.
- Die optionale DEFAULT-Klausel für einen bestimmten Routinenparameter.
- Die Angabe einer Transact-SQL-Funktion mit der AS-Klausel.
- Die optionale OR REPLACE-Klausel.
- **Transact-SQL** CREATE FUNCTION wird von Adaptive Server Enterprise unterstützt. Adaptive Server Enterprise unterstützt nicht das optionale Schlüsselwort IN für Funktionsparameter.

Beispiele

Die folgende Funktion verkettet die Zeichenfolge 'Vorname' mit der Zeichenfolge 'Nachname'.

```
CREATE FUNCTION fullname(
    firstname CHAR(30),
    lastname CHAR(30) )
RETURNS CHAR(61)
BEGIN
    DECLARE name CHAR(61);
    SET name = firstname || ' ' || lastname;
    RETURN (name);
END;
```

Im folgenden Beispiel wird die Funktion fullname ersetzt, die im ersten Beispiel erstellt wurde. Nachdem die Funktion ersetzt wurde, wird die lokale Variable name entfernt:

```
CREATE OR REPLACE FUNCTION fullname(
    firstname CHAR(30),
    lastname CHAR(30) )
RETURNS CHAR(61)
BEGIN
    RETURN ( firstname || ' ' || lastname );
END;
```

Die folgenden Beispiele veranschaulichen die Verwendung der fullname-Funktion.

Rückgabe eines vollständigen Namens aus zwei angegebenen Zeichenfolgen:

```
SELECT fullname ( 'joe', 'smith' );
```

fullname('joe', 'smith')
joe smith

Namensliste aller Mitarbeiter:

```
SELECT fullname ( GivenName, Surname )
FROM GROUPO.Employees;
```

fullname (GivenName, Surname)
Fran Whitney
Matthew Cobb
Philip Chin
Julie Jordan
...

Die folgende Funktion benutzt Transact-SQL-Syntax:

```
CREATE FUNCTION DoubleIt( @Input INT )
RETURNS INT
AS
BEGIN
    DECLARE @Result INT
    SELECT @Result = @Input * 2
    RETURN @Result
END;
```

Die Anweisung `SELECT DoubleIt(5)` gibt den Wert 10 zurück.

CREATE INDEX-Anweisung

Erstellt einen Index für eine angegebene Tabelle oder materialisierte Ansicht.

Syntax 1 - Einen Index für eine Tabelle erstellen

```
CREATE [ VIRTUAL ] [ UNIQUE ] [ CLUSTERED ] INDEX [ IF NOT EXISTS ] index-name
ON [ owner. ] table-name
    ( column-name [ ASC | DESC ], ...
      | function-name ( argument, [ ... ] ) AS column-name )
[ [ WITH NULLS [ NOT ] DISTINCT ]
  [ { IN | ON } dbspace-name ]
  [ FOR OLAP WORKLOAD ]
```

Syntax 2 - Einen Index für eine materialisierte Ansicht erstellen

```
CREATE [ VIRTUAL ] [ UNIQUE ] [ CLUSTERED ] INDEX [ IF NOT EXISTS ] index-name
ON [ owner. ] materialized-view-name
    ( column-name [ ASC | DESC ], ... )
[ [ WITH NULLS NOT DISTINCT ]
  [ { IN | ON } dbspace-name ]
  [ FOR OLAP WORKLOAD ]
```

Parameter

VIRTUAL-Klausel Das Schlüsselwort VIRTUAL wird hauptsächlich für den Indexberater verwendet. Ein virtueller Index ahmt die Eigenschaften des richtigen physischen Indexes nach, während die Ausführungspläne durch den Indexberater ausgewertet werden und wenn die PLAN-Funktion verwendet wird. Sie können virtuelle Indizes mit der PLAN-Funktion verwenden, um die Auswirkungen eines

Indexes auf die Performance zu testen, ohne die zeit- und ressourcenraubende Erstellung eines tatsächlichen Indexes vornehmen zu müssen.

Virtuelle Indizes sind für andere Verbindungen nicht sichtbar und werden gelöscht, wenn die Verbindung getrennt wird. Virtuelle Indizes werden bei der Auswertung von Plänen für die tatsächliche Durchführung von Abfragen nicht verwendet und wirken sich daher nicht negativ auf die Performance aus.

Virtuelle Indizes sind auf vier Spalten begrenzt.

UNIQUE-Klausel Das Attribut UNIQUE stellt sicher, dass in der Tabelle oder materialisierten Ansicht keine zwei Zeilen mit identischen Werten in den Indexspalten vorhanden sind. Wenn Sie UNIQUE angeben, nicht aber WITH NULLS NOT DISTINCT, muss jeder einzelne Indexschlüssel eindeutig sein oder in mindestens einer Spalte NULL enthalten. Zum Beispiel gelten zwei Einträge ('a', NULL) und ('a', NULL) jeweils als eindeutig.

Wenn Sie UNIQUE...WITH NULLS NOT DISTINCT angeben, muss der Indexschlüssel unabhängig von den NULL-Werten eindeutig sein. Zum Beispiel gelten zwei Einträge ('a', NULL) und ('a', NULL) als gleich, nicht als eindeutig.

Es besteht ein Unterschied zwischen einer Eindeutigkeits-Integritätsregel und einem eindeutigen Index. Spalten mit einem eindeutigen Index lassen NULL zu, nicht aber Spalten in einer Eindeutigkeits-Integritätsregel. Ein Fremdschlüssel kann entweder einen Primärschlüssel oder eine Eindeutigkeits-Integritätsregel referenzieren, aber keinen eindeutigen Index, da dieser mehrere Instanzen von NULL enthalten kann.

Es wird empfohlen, keine angenäherten Datentypen wie FLOAT und DOUBLE für Primärschlüssel oder Spalten in Eindeutigkeits-Integritätsregeln zu verwenden. Bei angenäherten numerischen Datentypen können nach arithmetischen Vorgängen Rundungsfehler auftreten.

Spalten mit räumlichen Daten können nicht in einen eindeutigen Index einbezogen werden.

CLUSTERED-Klausel Das CLUSTERED-Attribut sorgt dafür, dass Zeilen in etwa in der Schlüsselreihenfolge des entsprechenden Indexes gespeichert werden. Der Datenbankserver versucht die Speicherreihenfolge der Zeilen in Schlüsselreihenfolge zu halten, kann dies aber nicht vollständig garantieren.

Wenn ein Clustered-Index existiert, fügt die LOAD TABLE-Anweisung Zeilen in der Reihenfolge des Indexschlüssels ein. Die INSERT-Anweisung versucht, neue Zeilen auf derselben Seite einzufügen, die benachbarte Zeilen bezüglich der Schlüsselreihenfolge enthält.

IF NOT EXISTS-Klausel Wenn das Attribut IF NOT EXISTS angegeben wurde und der benannte Index bereits vorhanden ist, werden keine Änderungen vorgenommen und es wird kein Fehler zurückgegeben.

ASC | DESC-Klausel Spalten werden in aufsteigender Reihenfolge sortiert, es sei denn, absteigend (DESC) wird explizit angegeben. Ein Index wird sowohl für eine aufsteigende als auch für eine absteigende ORDER BY-Klausel verwendet, je nachdem, ob der Index aufsteigend oder absteigend war. Wenn jedoch ORDER BY mit gemischten Attributen (aufsteigend und absteigend) ausgeführt wird, wird ein Index nur verwendet, wenn der Index mit denselben aufsteigenden und absteigenden Attributen erstellt wurde.

function-name Die Funktionsname-Klausel erstellt einen Index für eine Funktion. Diese Klausel kann nicht für deklarierte temporäre Tabellen oder materialisierte Ansichten verwendet werden.

Diese Form einer CREATE INDEX-Anweisung ist eine bequeme Methode zur Durchführung folgender Vorgänge:

1. Hinzufügen einer berechneten Spalte namens *column-name* zur Tabelle. Die Spalte ist mit einer COMPUTE-Klausel definiert, die die angegebene Funktion und alle angegebenen Argumente enthält. Unter der COMPUTE-Klausel der CREATE TABLE-Anweisung finden Sie Hinweise zu Einschränkungen beim Funktionstyp, der angegeben werden kann. Der Datentyp der Spalte basiert auf dem Ergebnistyp der Funktion.
2. Ausfüllen der berechneten Spalte für die bestehenden Tabellenzeilen
3. Erstellen eines Indexes für die Spalte

Das Löschen des Indexes bewirkt nicht das Löschen der zugehörigen berechneten Spalte.

IN | ON-Klausel Standardmäßig wird der Index in derselben Datenbankdatei abgelegt wie seine Tabelle oder materialisierte Ansicht. Sie können den Index in einer getrennten Datenbankdatei ablegen, indem Sie einen DBSpace-Namen angeben, unter dem der Index abgelegt werden soll. Diese Funktion ist vor allem bei großen Datenbanken hilfreich, da hiermit Beschränkungen der Dateigröße umgangen werden können, oder bei Performancesteigerungen, die möglicherweise durch die Verwendung von mehreren Festplattengeräten erreicht wird.

Wenn der neue Index den physischen Index mit einem bestehenden logischen Index gemeinsam nutzen kann, wird die IN-Klausel ignoriert.

WITH NULLS NOT DISTINCT-Klausel Diese Klausel kann nur angegeben werden, wenn Sie den Index als UNIQUE deklarieren, und mit ihr können Sie festlegen, dass NULL-Werte in Indexschlüsseln nicht eindeutig sind. Weitere Hinweise finden Sie unter der UNIQUE-Klausel.

FOR OLAP WORKLOAD-Klausel Wenn Sie FOR OLAP WORKLOAD angeben, führt der Datenbankserver bestimmte Optimierungen durch und sammelt Statistiken über den Schlüssel, um die Performance für OLAP-Verarbeitungslasten zu verbessern. Performanceverbesserungen sind besonders gut erkennbar, wenn optimization_workload auf OLAP eingestellt ist.

Bemerkungen

Syntax 1 gilt für Tabellen, Syntax 2 für materialisierte Ansichten.

Indizes können die Performance der Datenbank verbessern. SQL Anywhere verwendet physische und logische Indizes. Ein physischer Index ist die tatsächliche Indexstruktur, wie sie auf der Festplatte gespeichert ist. Ein logischer Index ist eine Referenz auf einen physischen Index. Wenn Sie einen Index erstellen, der in seinen physischen Attributen identisch mit einem bestehenden Index ist, erstellt der Datenbankserver einen logischen Index, der den bestehenden physischen Index mitbenutzt. Im Allgemeinen werden von Benutzern erstellte Indizes als logische Indizes angesehen. Der Datenbankserver erstellt physische Indizes, um logische Indizes zu implementieren, und kann denselben physischen Index gemeinsam für mehrere logische Indizes verwenden.

Die CREATE INDEX-Anweisung erstellt einen sortierten Index für die angegebenen Spalten der benannten Tabelle oder materialisierten Ansicht. Indizes werden automatisch verwendet, um die

Performance von Abfragen zu verbessern, die von der Datenbank ausgegeben werden, und um Abfragen mithilfe einer ORDER BY-Klausel zu sortieren. Sobald ein Index erstellt ist, wird er nicht mehr in einer SQL-Anweisung referenziert, außer wenn er validiert (VALIDATE INDEX), geändert (ALTER INDEX) oder gelöscht (DROP INDEX) bzw. in einem Hinweis an den Optimierer verwendet wird.

- **Indexeigentümer** In der CREATE INDEX-Anweisung gibt es keine Möglichkeit, den Indexeigentümer anzugeben. Indizes gehören automatisch dem Eigentümer der Tabelle oder materialisierten Ansicht.
- **Indizes für Ansichten** Sie können Indizes für materialisierte Ansichten, aber nicht für reguläre Ansichten erstellen.
- **Index-Namespace** Der Name jedes Indexes muss für eine gegebene Tabelle oder materialisierte Ansicht eindeutig sein.
- **Exklusive Verwendung** CREATE INDEX wird verhindert, wenn die Anweisung eine Tabelle oder materialisierte Ansicht betrifft, die gerade von einer anderen Verbindung verwendet wird. CREATE INDEX kann zeitaufwändig sein. Der Datenbankserver verarbeitet während der Anweisungsverarbeitung keine Anforderungen hinsichtlich der gleichen Tabelle.
- **Automatisch erstellte Indizes** SQL Anywhere erstellt automatisch Indizes für Primärschlüssel, Fremdschlüssel und Eindeutigkeits-Integritätsregeln. Diese automatisch erstellten Indizes werden in derselben Datenbankdatei aufbewahrt wie die Tabelle.

Diese Anweisung kann nicht ausgeführt werden, wenn mit der WITH HOLD-Klausel geöffnete Cursor vorhanden sind, die entweder Anweisungs- oder Transaktions-Snapshots verwenden.

Privilegien

Wenn Sie einen Index für eine Tabelle erstellen möchten, müssen Sie der Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- REFERENCES-Privileg für die Tabelle
- CREATE ANY INDEX-Systemprivileg
- CREATE ANY OBJECT-Systemprivileg

Wenn Sie einen Index für eine materialisierte Ansicht erstellen möchten, müssen Sie Eigentümer der materialisierten Ansicht sein oder eines der folgenden Privilegien haben:

- CREATE ANY INDEX-Systemprivileg
- CREATE ANY OBJECT-Systemprivileg

Nebenwirkungen

In den meisten Fällen erfolgt eine automatische Festschreibung. Wenn die auto_commit_on_create_local_temp_index-Option auf "Off" gesetzt ist, erfolgt keine Festschreibung, bevor ein Index für eine lokale temporäre Tabelle erstellt wird. Das Erstellen eines Indexes für eine Funktion (eine implizite berechnete Spalte) bewirkt einen Checkpoint.

Spaltenstatistiken werden aktualisiert (oder erstellt, falls sie nicht existieren).

Siehe auch

- „Indizes erstellen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „auto_commit_on_create_local_temp_index-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „DROP INDEX-Anweisung“ auf Seite 810
- „CREATE STATISTICS-Anweisung“ auf Seite 729
- „optimization_workload-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Fortgeschrittene Aufgaben: Logische und physische Indizes“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Indizes“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Abrufen von Empfehlungen des Indexberaters für eine Abfrage“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Indexberater“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Snapshot-Isolation“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Clustered-Indizes“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Berechnete Spalten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „OLAP-Unterstützung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „SQL Anywhere-Einschränkungen von Größe und Anzahl“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Erstellung eines zweispaltigen Indexes für die Tabelle Employees

```
CREATE INDEX employee_name_index
ON GROUPO.Employees
( Surname, GivenName );
```

Erstellung eines Indexes für die Tabelle SalesOrderItems für die ProductID-Spalte

```
CREATE INDEX item_prod
ON GROUPO.SalesOrderItems
( ProductID );
```

Verwenden Sie die SORTKEY-Funktion, um einen Index für die Description-Spalte der Tabelle "Products", nach russischer Kollation sortiert, zu erstellen. Die Nebenwirkung ist, dass die Anweisung eine berechnete Spalte namens "desc_ru" zur Tabelle hinzufügt. Damit dieses Beispiel erfolgreich ausgeführt werden kann, müssen Sie außerdem das SELECT-Privileg für die Tabelle "Products" haben.

```
CREATE INDEX ix_desc_ru
ON GROUPO.Products (
  SORTKEY( Description, 'rusdict' )
  AS desc_ru );
```

CREATE LDAP SERVER-Anweisung

Erstellt ein LDAP-Serverkonfigurationsobjekt.

Syntax

```
CREATE LDAP SERVER ldapua-server-name
[ ldapua-server-attrs ... ]
[ WITH ACTIVATE ]
```

ldapua-server-attrs :

```
SEARCH DN search-dn-attributes ...
| AUTHENTICATION URL { 'url-string' | NULL }
| CONNECTION TIMEOUT timeout-value
| CONNECTION RETRIES retry-value
| TLS { ON | OFF }
```

search-dn-attributes :

```
URL { 'url-string' | NULL }
| ACCESS ACCOUNT { 'dn-string' | NULL }
| IDENTIFIED BY ( 'password' | NULL }
| IDENTIFIED BY ENCRYPTED { encrypted-password | NULL }
```

Parameter

SEARCH DN-Klausel Es gibt keine Standardwerte für Parameter in der SEARCH DN-Klausel.

- **URL** Verwenden Sie diese Klausel zum Angeben des Hosts (als Namen oder als IP-Adresse), der Portnummer und der auszuführenden Suche, um den **LDAP-Distinguished Name (DN)** für eine bestimmte Benutzer-ID nachzuschlagen. *url-string* wird vor dem Speichern in ISYSLDAPSERVER im Hinblick auf die Richtigkeit der LDAP-URL-Syntax validiert. Die maximale Größe für diese Zeichenfolge ist 1024 Byte.

Das Format der *url-string* muss dem LDAP-URL-Standard entsprechen. Siehe <http://www.isode.com/whitepapers/ldap-standards.html>.

- **ACCESS ACCOUNT** Verwenden Sie diese Klausel zum Angeben des DN, der vom Datenbankserver verwendet wird, um eine Verbindung mit dem LDAP-Server herzustellen. Dies ist kein SQL Anywhere-Benutzer, sondern ein Benutzer, der im LDAP-Server speziell für das Anmelden beim LDAP-Server erstellt wurde. Dieser Benutzer muss Berechtigungen im LDAP-Server haben, um über Benutzer-IDs an den in der SEARCH DN URL-Klausel angegebenen Orten nach DN's suchen zu können. Die maximale Größe für diese Zeichenfolge ist 1024 Byte.
- **IDENTIFIED BY** Verwenden Sie diese Klausel, um das Kennwort anzugeben, das dem Benutzer durch ACCESS ACCOUNT identifizierten Benutzer zugeordnet ist. Die maximal zulässige Länge beträgt 255 Byte und kann nicht auf NULL gesetzt werden.
- **IDENTIFIED BY ENCRYPTED** Verwenden Sie diese Klausel, um das Kennwort anzugeben, das dem durch ACCESS ACCOUNT identifizierten Benutzer zugeordnet ist. Dieses wird in verschlüsselter Form bereitgestellt und ist als Binärwert auf der Festplatte gespeichert. Die maximale Größe des Binärwerts beträgt 289 Byte und kann nicht auf NULL gesetzt werden. Mithilfe von IDENTIFIED BY ENCRYPTED kann das Kennwort abgerufen und verwendet werden, ohne dass es bekannt wird.

AUTHENTICATION URL-Klausel Geben Sie mithilfe dieser Klausel die *url-string* an, die den Host nach Name oder IP-Adresse identifiziert, sowie die Portnummer des LDAP-Servers, der zum Authentifizieren eines Benutzers verwendet werden soll. Der bei einer früheren DN-Suche abgerufene

DN des Benutzers und das Benutzerkennwort werden verwendet, um eine neue Verbindung an die Authentifizierungs-URL zu binden. Eine erfolgreiche Verbindung mit dem LDAP-Server gilt als Nachweis der Identität des Benutzers, der die Verbindung herstellt. Es gibt keinen Standardwert für diesen Parameter. Weitere Hinweise zu Größenbeschränkungen für diese Zeichenfolge finden Sie in der Beschreibung der SYSLDAPSERVER-Systemansicht.

CONNECTION TIMEOUT-Klausel Verwenden Sie diese Klausel, um das Verbindungs-Timeout des LDAP-Servers in Millisekunden anzugeben, sowohl für DN-Suchvorgänge als auch für die Authentifizierung. Der Standardwert beträgt 10 Sekunden.

CONNECTION RETRIES-Klausel Verwenden Sie diese Klausel, um die Anzahl von Wiederholungen für Verbindungen mit dem LDAP-Server anzugeben, sowohl für DN-Suchvorgänge als auch für die Authentifizierung. Der gültige Wertebereich liegt zwischen 1 und 60. Der Standardwert ist 3.

TLS-Klausel Verwenden Sie diese Klausel, um die Verwendung des TLS-Protokolls für Verbindungen mit dem LDAP-Server anzugeben, sowohl für DN-Suchvorgänge als auch für die Authentifizierung. Die gültigen Werte sind ON und OFF. Der Standardwert ist OFF. Wenn Sie das Secure LDAP-Protokoll verwenden möchten, geben Sie am Anfang der URL `ldaps://` statt `ldap://` ein. Die TLS-Option muss auf OFF gesetzt sein, wenn Sie Secure LDAP verwenden.

WITH ACTIVATE-Klausel Verwenden Sie diese Klausel, um den LDAP-Server für die sofortige Verwendung zu aktivieren. Mithilfe dieser Klausel können Sie die LDAP-Benutzerauthentifizierung in einer Anweisung definieren und aktivieren, weil sie den Zustand des neuen LDAP-Servers in READY ändert.

Bemerkungen

Keine

Privilegien

Sie müssen das MANAGE ANY LDAP SERVER-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „LDAP-Benutzerauthentifizierung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „ALTER LDAP SERVER-Anweisung“ auf Seite 468
- „DROP LDAP SERVER-Anweisung“ auf Seite 811
- „VALIDATE LDAP SERVER-Anweisung“ auf Seite 1116

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel werden Suchparameter, eine Authentifizierungs-URL und ein Timeout von 3 Sekunden gesetzt und der LDAP-Server wird aktiviert, sodass er mit dem Authentifizieren von Benutzern beginnen kann. Eine Verbindung mit dem LDAP-Server wird ohne TLS- oder SECURE LDAP-Protokoll

hergestellt. Neben den zum Ausführen der CREATE LDAP SERVER-Anweisung erforderlichen Privilegien müssen Sie auch das SET ANY SECURITY-Systemprivileg haben, um die login_mode-Option im folgenden Beispiel setzen zu können.

```
SET OPTION PUBLIC.login_mode = 'Standard,LDAPUA';
CREATE LDAP SERVER apps_primary
  SEARCH DN
    URL 'ldap://voyager:389/dc=MyCompany,dc=com??sub?cn=*'
    ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'
    IDENTIFIED BY 'Secret99Password'
  AUTHENTICATION URL 'ldap://voyager:389/'
  CONNECTION TIMEOUT 3000
  WITH ACTIVATE;
```

In diesem Beispiel werden dieselben Suchparameter verwendet, jedoch wird ldaps:// angegeben und dadurch eine Secure LDAP-Verbindung mit dem LDAP-Server auf dem Host voyager, port 636, hergestellt. Nur LDAP-Clients, die das Secure LDAP-Protokoll verwenden, dürfen Verbindungen über diesen Port herstellen. Die Datenbanksicherheitsoption Trusted_certificate_file muss mit einem Dateinamen gesetzt werden, der das Zertifikat der Zertifizierungsstelle enthält, die das vom LDAP-Servers unter 'ldaps://voyager:636' verwendete Zertifikat signiert hat. Beim Handshake mit dem LDAP-Server wird das vom LDAP-Server vorgelegte Zertifikat vom Datenbankserver überprüft, um zu gewährleisten, dass es von einem der in der Datei aufgelisteten Zertifikate signiert wurde. Die an den LDAP-Server übergebenen Parameter ACCESS ACCOUNT und IDENTIFIED BY werden ebenfalls vom LDAP-Server überprüft.

```
SET OPTION PUBLIC.login_mode = 'Standard,LDAPUA';
SET OPTION PUBLIC.trusted_certificates_file = '/opt/sybase/shared/
trusted.txt';
CREATE LDAP SERVER secure_primary
  SEARCH DN
    URL 'ldaps://voyager:636/dc=MyCompany,dc=com??sub?cn=*'
    ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'
    IDENTIFIED BY 'Secret99Password'
  AUTHENTICATION URL 'ldaps://voyager:636/'
  CONNECTION TIMEOUT 3000
  WITH ACTIVATE;
```

CREATE LOCAL TEMPORARY TABLE-Anweisung

Erstellt eine lokale temporäre Tabelle innerhalb einer Prozedur, die nach Abschluss der Prozedur erhalten bleibt, bis entweder die Tabelle explizit gelöscht oder die Verbindung beendet wird.

Syntax

```
CREATE LOCAL TEMPORARY TABLE IF NOT EXISTS table-name
( { column-definition [ column-constraint ... ] | table-constraint | pctfree }, ... )
[ ON COMMIT { DELETE | PRESERVE } ROWS | NOT TRANSACTIONAL ]
```

pctfree : **PCTFREE** *percent-free-space*

percent-free-space : *integer*

Parameter

IF NOT EXISTS-Klausel Wenn die benannte Tabelle bereits vorhanden ist, werden keine Änderungen durchgeführt und es wird keine Fehlermeldung ausgegeben.

ON COMMIT-Klausel Standardmäßig werden die Zeilen einer temporären Tabelle bei COMMIT gelöscht. Sie können die ON COMMIT-Klausel benutzen, um die Zeilen bei COMMIT zu erhalten.

NOT TRANSACTIONAL-Klausel Die NOT TRANSACTIONAL-Klausel bietet Performanceverbesserungen unter bestimmten Umständen, da Vorgänge in nicht-transaktionalen temporären Tabellen keine Einträge im Rollback-Log bewirken. NOT TRANSACTIONAL kann z.B. sinnvoll sein, wenn Prozeduren, die die temporäre Tabelle verwenden, wiederholt ohne dazwischen liegende COMMITs oder ROLLBACKs aufgerufen werden.

Bemerkungen

In einer Prozedur verwenden Sie die CREATE LOCAL TEMPORARY TABLE-Anweisung anstelle der DECLARE LOCAL TEMPORARY TABLE-Anweisung, wenn Sie eine Tabelle erstellen wollen, die bestehen bleibt, nachdem die Prozedur abgeschlossen ist. Mit der CREATE LOCAL TEMPORARY TABLE-Anweisung erstellte lokale temporäre Tabellen bleiben erhalten, bis sie entweder explizit gelöscht werden oder die Verbindung beendet wird.

Tabellen, die mit CREATE LOCAL TEMPORARY TABLE erstellt wurden, erscheinen nicht in der SYSTABLE-Ansicht des Systemkatalogs.

Lokale temporäre Tabellen, die mit CREATE LOCAL TEMPORARY TABLE in IF-Anweisungen erstellt werden, bleiben auch erhalten, wenn die IF-Anweisung abgeschlossen ist.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „CREATE TABLE-Anweisung“ auf Seite 737
- „DECLARE LOCAL TEMPORARY TABLE-Anweisung“ auf Seite 784
- „Zusammengesetzte Anweisungen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** CREATE LOCAL TEMPORARY TABLE ist Teil der optionalen Sprachenfunktion F531 des SQL/2008-Standards. Die Klauseln PCTFREE und NOT TRANSACTIONAL sind Erweiterungen des Herstellers. Die in der Anweisung festgelegten Definitionen für Spalten und Integritätsregeln können ebenfalls Syntax enthalten, die eine Erweiterung des Herstellers darstellt. Im SQL/2008-Standard ist festgelegt, dass mit der CREATE LOCAL TEMPORARY TABLE-Anweisung erstellte Tabellen im Systemkatalog erscheinen. Dies ist bei SQL Anywhere nicht der Fall.
- **Transact-SQL** CREATE LOCAL TEMPORARY TABLE wird von Adaptive Server Enterprise nicht unterstützt. In Sybase Adaptive Server Enterprise wird eine temporäre Tabelle erstellt unter

Verwendung der CREATE TABLE-Anweisung mit einem Tabellennamen, der mit dem Sonderzeichen # beginnt.

Beispiel

Mit dem nachstehenden Beispiel wird eine lokale temporäre Tabelle namens TempTab erstellt:

```
CREATE LOCAL TEMPORARY TABLE TempTab ( number INT )
ON COMMIT PRESERVE ROWS;
```

CREATE LOGIN POLICY-Anweisung

Erstellt eine Login-Richtlinie.

Syntax

CREATE LOGIN POLICY *policy-name* *policy-options*

policy options :
policy-option [*policy-option* ...]

policy-option :
policy-option-name = *policy-option-value*

policy-option-value :
{ **UNLIMITED** | *legal-option-value* }

Parameter

policy-name Der Name der Login-Richtlinie

policy-option-name Der Name der Login-Richtlinienoption.

policy-option-value Der Wert, der der Login-Richtlinienoption zugewiesen wird. Wenn Sie UNLIMITED angeben, werden keine Limits auferlegt.

Richtlinienoptionsname	Beschreibung	Standardwert	Gilt für:
auto_unlock_time	Die Zeitspanne, nach der gesperrte Konten automatisch freigegeben werden.	Unlimited	Alle Benutzer außer denjenigen mit MANAGE ANY USER-Systemprivileg

Richtlinienoptionsname	Beschreibung	Standardwert	Gilt für:
change_password_dual_control	<p>Wenn der Wert für diese Option ON ist, muss das Kennwort von zwei Administratoren festgelegt werden.</p> <p>Die Einstellung für die verify_password_function-Option wird ignoriert, wenn diese Option auf ON gesetzt ist, weil das Kennwort separat in zwei Teilen konfiguriert wird. Es wird keine Überprüfung durchgeführt.</p>	OFF	Alle Benutzer
ldap_primary_server	Der Name des LDAP-Primärservers.	(keiner)	Alle Benutzer
ldap_secondary_server	Der Name des LDAP-Sekundärservers.	(keiner)	Alle Benutzer
ldap_auto_failback_period	Die Zeitspanne in Minuten, nach der ein automatischer Failback auf den Primärserver versucht wird.	15 Minuten	Alle Benutzer
ldap_failover_to_std	Angabe, ob die Authentifizierung mit Standardauthentifizierung zulässig sein soll, wenn die Authentifizierung über den LDAP-Server fehlschlägt, weil der Distinguished Name (DN) für einen Benutzer nicht gefunden wird, Systemressourcen fehlen bzw. Netzwerkausfälle, Verbindungs-Timeouts oder ähnliche Systemfehler auftreten. Diese Einstellung lässt nicht zu, dass bei Rückgabe eines tatsächlichen Authentifizierungsfehlers durch einen LDAP-Server ein Failover auf die Standardauthentifizierung erfolgt (wie es der Fall ist, wenn der Benutzer gefunden wird, aber das angegebene Kennwort nicht passt).	ON	Alle Benutzer

Richtlinienoptionsname	Beschreibung	Standardwert	Gilt für:
ldap_refresh_dn	<p>Zum Zeitpunkt der Festlegung dieser Richtlinienoption durch eine CREATE LOGIN POLICY- oder ALTER LOGIN POLICY-Anweisung wird der aktuelle Zeitwert mit der Login-Richtlinie gespeichert. Dieser Wert ist der Zeitstempel, der bei der Benutzerauthentifizierung mit dem in ISYSUSER für den betreffenden Benutzer gefundenen user_dn_cached_at-Wert verglichen wird. Wenn der Wert in der Richtlinie neuer ist als der user_dn_cached_at-Wert in ISYSUSER, wird nach dem Distinguished Name (DN) eines Benutzers gesucht, um den user_dn-Wert in ISYSUSER zu aktualisieren.</p> <p>Der Wert NOW ist der einzige gültige Wert, der dieser Richtlinienoption zugeordnet werden kann. Alle anderen führen zu einer Fehlermeldung. Der Wert wird in der Coordinated Universal Time (UTC) angegeben und als Zeichenfolge im Standardformat des Servers gespeichert.</p>	(keiner)	Alle Benutzer
locked	Wenn der Wert für diese Option ON ist, dürfen die Benutzer keine neuen Verbindungen mehr herstellen. Die reason_locked-Spalte der sa_get_user_status-Systemprozedur gibt eine vom Datenbankserver erstellte Zeichenfolge zurück, die anzeigt, warum ein Benutzer gesperrt ist.	OFF	Alle Benutzer außer denjenigen mit MANAGE ANY USER-Systemprivileg
max_connections	Die maximale Anzahl gleichzeitiger Verbindungen, die einem Benutzer gestattet sind.	Unlimited	Alle Benutzer außer denjenigen mit SERVER OPERATOR- oder DROP CONNECTION-Systemprivileg

Richtlinienoptionsname	Beschreibung	Standardwert	Gilt für:
max_failed_login_attempts	Die maximale Anzahl fehlgeschlagener Login-Versuche seit dem letzten erfolgreichen Versuch, bevor der Benutzer gesperrt wird. Benutzer mit SYS_AUTH_DBA_ROLE-Kompatibilitätsrolle werden freigegeben, wenn seit dem letzten gescheiterten Login-Versuch eine Minute vergangen ist.	Unlimited	
max_days_since_login	Die maximale Anzahl von Tagen zwischen erfolgreichen Logins durch denselben Benutzer.	Unlimited	Alle Benutzer außer denjenigen mit MANAGE ANY USER-Systemprivileg
max_non_dba_connections	Die maximale Anzahl gleichzeitiger Verbindungen, die Benutzer herstellen können. Diese Option wird nur bei der Root-Login-Richtlinie verwendet.	Unlimited	Alle Benutzer außer denjenigen mit SERVER OPERATOR- oder DROP CONNECTION-Systemprivileg
password_life_time	Die maximale Anzahl von Tagen, bis ein Kennwort geändert werden muss	Unlimited	Alle Benutzer
password_grace_time	Die Anzahl der Tage bis zum Kennwortablauf. Das Login ist möglich, aber die Standardprozedur post_login gibt Warnungen aus.	0	Alle Benutzer
password_expiry_on_next_login	Wenn der Wert für diese Option ON ist, läuft das Kennwort des Benutzers nach dem nächsten Login ab.	OFF	Alle Benutzer

Richtlinienoptionsname	Beschreibung	Standardwert	Gilt für:
root_auto_unlock_time	Die Zeitspanne, nach der gesperrte Konten automatisch freigegeben werden. Diese Option wird nur von der Root-Login-Richtlinie unterstützt.	1 Minute	Benutzer mit MANAGE ANY USER-Systemprivileg

Bemerkungen

Wenn Sie keine Richtlinienoption angeben, werden Werte für diese Login-Richtlinie aus der Root-Login-Richtlinie übernommen. Neue Richtlinien erben nicht die Richtlinienoptionen MAX_NON_DBA_CONNECTIONS und ROOT_AUTO_UNLOCK_TIME.

Alle neuen Datenbanken enthalten eine Root-Login-Richtlinie. Sie können die Werte der Root-Login-Richtlinie ändern, aber Sie können die Richtlinie nicht löschen. Einen Überblick über die Standardwerte für die Root-Login-Richtlinie finden Sie in der Tabelle oben.

Privilegien

Sie müssen das MANAGE ANY LOGIN POLICY-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „ALTER LOGIN POLICY-Anweisung“ auf Seite 471
- „ALTER USER-Anweisung“ auf Seite 540
- „COMMENT-Anweisung“ auf Seite 573
- „CREATE USER-Anweisung“ auf Seite 770
- „DROP LOGIN POLICY-Anweisung“ auf Seite 812
- „DROP USER-Anweisung“ auf Seite 838
- „Login-Richtlinien“ [*SQL Anywhere Server - Datenbankadministration*]
- „Neue Login-Richtlinien erstellen“ [*SQL Anywhere Server - Datenbankadministration*]
- „Login-Richtlinien vorhandenen Benutzern zuordnen“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Im folgenden Beispiel wird die Login-Richtlinie Test1 erstellt. Dieses Beispiel enthält ein unbegrenztes Kennwort und dem Benutzer werden maximal fünf Versuche zur Eingabe des richtigen Kennworts ermöglicht, bevor das Konto gesperrt wird.

```
CREATE LOGIN POLICY Test1
PASSWORD_LIFE_TIME=UNLIMITED
MAX_FAILED_LOGIN_ATTEMPTS=5;
```

Das folgende Beispiel zeigt typische Einstellungen für eine neue Login-Richtlinie (`ldap_user_policy`), die LDAP-Benutzerauthentifizierung verwendet. Sowohl ein primäres als auch ein sekundäres Serverkonfigurationsobjekt (die zuvor erstellt wurden) werden angegeben, damit ein Failover auf den sekundären LDAP-Server möglich ist, und ein Failover auf die Standard-Authentifizierung ist zulässig, wenn Systemressourcen, Netzwerkressourcen oder sowohl der primäre als auch der sekundäre LDAP-Server nicht mehr reagieren. In diesem Beispiel wird eine Kombination von Authentifizierungsoptionen angegeben, die Antworten mit im Cache abgelegten Werten zulässt, wenn ein LDAP-Server die eingehenden Anforderungen nicht bewältigen kann. In diesem Beispiel wird vorausgesetzt, dass die `login_mode`-Datenbankoption den Wert "Standard" enthält. Sie können dieses Beispiel nicht einfügen und ausführen, weil die in dem Beispiel als primär und sekundär genannten Server fiktiv sind.

```
CREATE LOGIN POLICY ldap_user_policy
LDAP_PRIMARY_SERVER=ldapsrv1
LDAP_SECONDARY_SERVER=ldapsrv2
LDAP_FAILOVER_TO_STD=ON;
```

CREATE MATERIALIZED VIEW-Anweisung

Erstellt eine materialisierte Ansicht.

Syntax

```
CREATE MATERIALIZED VIEW
[ owner. ] materialized-view-name [ ( alt-column-names, ... ) ]
[ IN dbspace-name ]
AS select-statement
[ CHECK { IMMEDIATE | MANUAL } REFRESH ]
```

alt-column-names :
(*column-name* [,...])

Parameter

alt-column-names-Klausel Verwenden Sie diese Klausel, um alternative Namen für die Spalten in der materialisierten Ansicht anzugeben. Wenn Sie alternative Spalten angeben, muss die Anzahl von Spalten in *alt-column-names* der Anzahl von Spalten in der *select-statement* entsprechen. Wenn Sie keine alternativen Spaltennamen angeben, werden als Namen die in der *select-statement* enthaltenen herangezogen.

IN-Klausel Verwenden Sie diese Klausel, um den DBSpace anzugeben, in dem die materialisierte Ansicht erstellt werden soll. Wenn diese Klausel nicht angegeben ist, wird die materialisierte Ansicht in dem DBSpace erstellt, der in der Option `default_dbspace` angegeben wurde. Sonst wird der SYSTEM-DBSpace benutzt.

AS-Klausel Verwenden Sie diese Klausel, um in Form einer SELECT-Anweisung die Daten anzugeben, die verwendet werden, um die materialisierte Ansicht zu füllen. Eine Definition einer materialisierten Ansicht kann nur Basistabellen referenzieren, keine Ansichten, anderen materialisierten Ansichten oder temporären Tabellen. *select-statement* muss Spaltennamen oder einen Aliasnamen enthalten. Wenn Sie *alt-column-name* angeben, werden diese Namen anstelle der Alias verwendet, die in der *select-statement* enthalten sind.

Spaltennamen in der SELECT-Anweisung müssen explizit angegeben werden, das Konstrukt `SELECT *` kann nicht verwendet werden. Beispiel: Sie können nicht `CREATE MATERIALIZED VIEW matview AS SELECT * FROM table-name` angeben. Objektnamen müssen darüber hinaus in der *select-statement* voll qualifiziert werden.

CHECK-Klausel Verwenden Sie diese Klausel, um die Anweisung ohne Erstellung der Ansicht zu validieren. Wenn Sie die CHECK-Klausel angeben:

- Der Datenbankserver führt die normalen Sprachprüfungen durch, die auch durchgeführt würden, wenn `CREATE MATERIALIZED VIEW` ohne Klausel ausgeführt worden wäre. Eventuell generierte Fehler werden wie üblich zurückgegeben.
- Der Datenbankserver führt nicht die eigentliche Erstellung der Ansicht aus, sodass bestimmte Fehler, die zum Erstellungszeitpunkt auftreten würden, nicht generiert werden. Beispiel: Ein Fehler, der angibt, dass der angegebene Ansichtsname bereits existiert, wird nicht generiert. Damit können Sie die CHECK-Klausel verwenden, um beabsichtigte Änderungen an der Definition der Ansicht zu prüfen, ohne einen Namenskonflikt mit der Ansicht befürchten zu müssen.
- Wenn `CHECK IMMEDIATE REFRESH` benutzt wird, prüft der Datenbankserver, ob die Syntax für eine Sofortansicht gültig ist, und gibt eventuelle Fehler zurück.
- Es werden keine Änderungen an der Datenbank durchgeführt und im Transaktionslog wird nichts aufgezeichnet.
- Ein implizites `COMMIT` erfolgt zu Beginn der Anweisungsausführung und ein `ROLLBACK` am Ende, um alle Sperren freizugeben, die während der Ausführung gesetzt wurden.

Siehe auch

- „Einschränkungen für materialisierte Ansichten“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Zusätzliche Hinweise zu DBSpaces“ [[SQL Anywhere Server - Datenbankadministration](#)]

Bemerkungen

Wenn Sie eine materialisierte Ansicht erstellen, ist sie eine manuelle Ansicht und nicht initialisiert. Das heißt: Sie hat den Aktualisierungstyp "manuell" und ist noch nicht aktualisiert (mit Daten gefüllt) worden. Um die Ansicht zu initialisieren, führen Sie die Anweisung `REFRESH MATERIALIZED VIEW` aus oder benutzen die Systemprozedur `sa_refresh_materialized_views`.

Sie können eine materialisierte Ansicht verschlüsseln, ihre `PCTFREE`-Einstellung und ihren Aktualisierungstyp ändern sowie ihre Verwendung durch den Optimierer aktivieren oder deaktivieren. Sie müssen allerdings zuerst die materialisierte Ansicht erstellen und dann `ALTER MATERIALIZED VIEW` verwenden, um diese Einstellungen zu ändern. Die Standardwerte für materialisierte Ansichten zum Erstellungszeitpunkt sind:

- `NOT ENCRYPTED`
- `ENABLE USE IN OPTIMIZATION`
- `PCTFREE` wird je nach Seitengröße der Datenbank eingestellt: 200 Byte für eine 4-kB-Seitengröße und 100 Byte für eine 2-kB-Seitengröße

- **MANUAL REFRESH**

Mehrere Datenbank- und Serveroptionen müssen aktiviert sein, um eine materialisierte Ansicht zu erstellen.

Die Systemprozedur `sa_recompile_views` wirkt sich nicht auf materialisierte Ansichten aus.

Siehe auch

- „Einschränkungen für materialisierte Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „REFRESH MATERIALIZED VIEW-Anweisung“ auf Seite 987

Privilegien

Sie müssen das `CREATE MATERIALIZED VIEW`-Systemprivileg haben, um materialisierte Ansichten erstellen zu können, deren Eigentümer Sie sind. Außerdem müssen Sie Eigentümer des zugrunde liegenden Objekts sein, das die materialisierte Ansicht referenziert, oder `SELECT`-Privilegien dafür haben.

Sie müssen das `CREATE ANY MATERIALIZED VIEW`-Systemprivileg oder das `CREATE ANY OBJECT`-Systemprivileg haben, um materialisierte Ansichten erstellen zu können, deren Eigentümer andere Benutzer sind.

Nebenwirkungen

Während der Ausführung setzt die `CREATE MATERIALIZED VIEW`-Anweisung Exklusivsperrern ohne Blockierung für alle von der materialisierten Ansicht referenzierten Tabellen. Wenn eine der referenzierten Tabellen nicht gesperrt werden kann, schlägt die Anweisung fehl und es wird ein Fehler zurückgegeben.

Siehe auch

- „Materialisierte Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Fortgeschrittene Aufgaben: Status und Eigenschaften von materialisierten Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „ALTER MATERIALIZED VIEW-Anweisung“ auf Seite 476
- „DROP MATERIALIZED VIEW-Anweisung“ auf Seite 813
- „REFRESH MATERIALIZED VIEW-Anweisung“ auf Seite 987
- „CREATE VIEW-Anweisung“ auf Seite 773
- „sa_refresh_materialized_views-Systemprozedur“ auf Seite 1294

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel erstellt in der SQL Anywhere-Beispieldatenbank eine materialisierte Ansicht, die vertrauliche Informationen über Mitarbeiter enthält. Sie müssen danach eine `REFRESH MATERIALIZED VIEW`-Anweisung ausführen, um die Ansicht für die Verwendung zu initialisieren, wie im Beispiel gezeigt wird.

```
CREATE MATERIALIZED VIEW EmployeeConfid2 AS
SELECT EmployeeID, Employees.DepartmentID,
```

```

        SocialSecurityNumber, Salary, ManagerID,
        Departments.DepartmentName, Departments.DepartmentHeadID
FROM GROUPO.Employees, GROUPO.Departments
WHERE Employees.DepartmentID=Departments.DepartmentID;
REFRESH MATERIALIZED VIEW EmployeeConfid2;

```

CREATE MESSAGE-Anweisung [T-SQL]

Erstellt ein Zeichenfolgenpaar aus Meldungsnummer und Meldung. Die Meldungsnummer kann in den Anweisungen PRINT und RAISERROR verwendet werden.

Syntax

CREATE MESSAGE *message-number* **AS** *message-text*

message-number : integer

message-text : string

Parameter

message-number Die Meldungsnummer der hinzuzufügenden Meldung. Die Meldungsnummer für eine benutzerdefinierte Meldung muss 20000 oder größer sein.

message-text Der Text der hinzuzufügenden Meldung. Die Höchstlänge beträgt 255 Byte. PRINT und RAISERROR erkennen Platzhalter im Meldungstext. Eine einzelne Meldung kann bis zu 20 eindeutige Platzhalter in jeder beliebigen Reihenfolge enthalten. Diese Platzhalter werden durch die formatierten Inhalte sämtlicher Argumente ersetzt, die auf diese Meldung folgen, wenn der Text der Meldung an den Client gesendet wird.

Die Platzhalter werden nummeriert, um eine Neusortierung der Argumente zu ermöglichen, wenn die Meldung in eine Sprache mit einer anderen grammatikalischen Struktur übersetzt wird. Ein Platzhalter für ein Argument hat das Format "%nn!": ein Prozentzeichen (%), gefolgt von einer Ganzzahl zwischen 1 und 20, gefolgt von einem Ausrufezeichen (!), wobei die Ganzzahl die Position des Arguments in der Argumentliste repräsentiert. "%1!" ist das erste Argument, "%2!" ist das zweite Argument, usw.

Es gibt keinen Parameter, der dem Argument *Sprache* für sp_addmessage entspricht.

Bemerkungen

Fügt der Systemtabelle ISYSUSERMESSAGE eine benutzerdefinierte Meldung hinzu, die von PRINT- und RAISERROR-Anweisungen verwendet wird.

Privilegien

Sie müssen das CREATE MESSAGE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „PRINT-Anweisung [T-SQL]“ auf Seite 980
- „RAISERROR-Anweisung“ auf Seite 982
- „DROP MESSAGE-Anweisung“ auf Seite 814
- „SYSUSERMESSAGE-Systemansicht“ auf Seite 1510

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** CREATE MESSAGE liefert die von der sp_addmessage-Systemprozedur in Adaptive Server Enterprise bereitgestellte Funktionalität.

Beispiel

Im folgenden Beispiel wird eine neue Meldung erstellt:

```
CREATE MESSAGE 20000 AS 'End of line reached';
```

CREATE MIRROR SERVER-Anweisung

Hinweis

Scale-Out mit Schreibschutz und Datenbankspiegelung erfordern jeweils eine getrennte Lizenz. Siehe „Getrennt lizenzierbare Komponenten“ [*SQL Anywhere 16 - Einführung*].

Erstellt oder ersetzt einen Spiegelserver, der bei der Datenbankspiegelung oder im Scale-Out mit Schreibschutz verwendet wird.

Syntax 1

```
CREATE [ OR REPLACE ] MIRROR SERVER mirror-server-name  
AS { PRIMARY | MIRROR | ARBITER | PARTNER }  
[ server-option = string [ ... ] ]
```

Syntax 2

```
CREATE [ OR REPLACE ] MIRROR SERVER mirror-server-name  
AS COPY  
{ FROM SERVER parent-name [ OR SERVER server-name ] | USING AUTO PARENT }  
[ server-option = string [ ... ] ]
```

server-option :
connection_string
logfile
preferred
state_file

parent-name :
server-name | **PRIMARY**

Parameter

OR REPLACE-Klausel CREATE MIRROR SERVER erstellt den Spiegelserver. Ein Fehler wird zurückgegeben, wenn bereits ein Spiegelserver mit dem angegebenen Namen in der Datenbank vorhanden ist.

Mit der Angabe von CREATE OR REPLACE wird ein Spiegelserver erstellt, wenn der Server nicht bereits in der Datenbank vorhanden ist, und der Server ersetzt, wenn er vorhanden ist.

AS-Klausel Sie können einen der folgenden Servertypen festlegen:

- **PRIMARY** Der Spiegelserver mit PRIMARY-Typ definiert einen virtuellen oder logischen Server, nicht aber einen tatsächlichen Datenbankserver. Der Name dieses Servers ist der alternative Servername für die Datenbank. Mithilfe des alternativen Servernamens können Anwendungen eine Verbindung mit dem als Primärserver verwendeten Server herstellen. Die Verbindungszeichenfolge für den als PRIMARY gekennzeichneten Server bewirkt Folgendes:
 - Definiert die Verbindungszeichenfolge, über die Kopieknoten Verbindungen mit dem Stammknoten oder dem übergeordneten PRIMARY-Knoten herstellen.
 - Definiert die Verbindungszeichenfolge, die vom PRIMARY-Wert des NodeType-Verbindungsparameters verwendet wird.

Es kann nur einen PRIMARY-Server für eine Datenbank geben.

- **MIRROR** Der Spiegelserver mit MIRROR-Typ definiert einen virtuellen oder logischen Server, nicht aber einen tatsächlichen Datenbankserver. Der Name dieses Servers ist der alternative Servername für die Datenbank. Mithilfe des alternativen Spiegelservernamens können Anwendungen eine Verbindung mit dem als schreibgeschützten Spiegelserver verwendeten Server herstellen. Außerdem definiert der als MIRROR gekennzeichnete Server die Verbindungszeichenfolge, die vom MIRROR-Wert des NodeType-Verbindungsparameters verwendet wird. Es kann nur einen MIRROR-Server für eine Datenbank geben.
- **ARBITER** In einem Datenbankspiegelungssystem hilft der Arbiterserver dabei zu ermitteln, welcher der PARTNER-Server Eigentümer der Datenbank wird. Der Arbiterserver muss mit einer Verbindungszeichenfolge definiert werden, die von den Partnerservern für die Verbindung mit dem Arbiterserver verwendet werden kann. Es kann nur einen ARBITER-Server für eine Datenbank geben.
- **PARTNER** Der Name des Spiegelservers muss dem Namen des Datenbankservers entsprechen, der durch die Serveroption -n angegeben wurde, und muss mit dem Wert des SERVER-Verbindungszeichenfolgenparameters übereinstimmen, der in der Spiegelserveroption connection_string angegeben ist.

In einem Datenbankspiegelungssystem verwenden die Partner den Wert in der Verbindungszeichenfolge, um sich miteinander zu verbinden. In einem Scale-Out-System mit Schreibschutz wird die Verbindungszeichenfolge von einem Kopieknoten verwendet, für den dieser Server der übergeordnete Knoten ist.

- **Spiegelung oder Spiegelung in Verbindung mit Scale-Out mit Schreibschutz** Sie müssen für eine Datenbankspiegelung zwei PARTNER-Server definieren und beide müssen eine Verbindungszeichenfolge und eine Statusdatei haben.

In einem Datenbankspiegelungssystem kommen als PARTNER definierte Server dafür in Frage, Primärserver und Eigentümer der Datenbank zu werden.

- **Scale-Out mit Schreibschutz ohne Spiegelung** Für Scale-Out mit Schreibschutz müssen Sie einen PARTNER-Server definieren und dieser muss eine Verbindungszeichenfolge und keine Statusdatei haben. Dieser Server ist der Stammserver und führt die einzige Kopie der Datenbank aus, in der sowohl Lese- als auch Schreibvorgänge zulässig sind
- **COPY** In einem Scale-Out-System mit Schreibschutz gibt dieser Wert an, dass der Datenbankserver ein Kopieknoten ist. Alle Verbindungen zur Datenbank auf diesem Server sind schreibgeschützt. Der Name des Spiegelservers muss dem Namen des Datenbankservers entsprechen, der durch die Serveroption -n angegeben wurde, und muss mit dem Wert des SERVER-Verbindungszeichenfolgenparameters übereinstimmen, der in der Spiegelserveroption connection_string angegeben ist.

Wenn AS COPY angegeben wird, müssen Sie außerdem die Klausel FROM SERVER oder USING AUTO PARENT angeben.

Die Verbindungszeichenfolge wird vom COPY-Wert des NodeType-Verbindungsparameters verwendet und außerdem von anderen Kopieknoten, für die dieser Server der übergeordnete Knoten ist.

Beim Hinzufügen von Kopieknoten zu einem Scale-Out-System mit Schreibschutz können Sie entweder eine CREATE MIRROR SERVER-Anweisung für den Kopieknoten ausführen oder den Spiegelserver vom Stammserver automatisch definieren lassen.

FROM SERVER-Klausel Diese Klausel kann nur verwendet werden, wenn AS COPY angegeben ist. Diese Klausel konstruiert eine Struktur von Servern für ein Scale-Out-System und gibt an, von welchen Servern die Kopieknoten Transaktionslogseiten abrufen.

Der übergeordnete Server kann mit dem Namen des Spiegel- oder Primärservers angegeben werden. Ein alternativer übergeordneter Server für die Kopieknoten kann mit der OR SERVER-Klausel angegeben werden.

In einem Datenbankspiegelungssystem mit nur zwei Ebenen (Stamm- und Kopieknoten) erhalten die Kopieknoten Transaktionslogseiten vom aktuellen Primär- oder Spiegelserver.

Ein Kopieknoten legt mithilfe der in der Datenbank gespeicherten Spiegelserverdefinition fest, zu welchem Server die Verbindung hergestellt wird. Aus seiner Definition kann er die Definition des übergeordneten Servers ermitteln und aus dessen Definition die Verbindungszeichenfolge für die Verbindung zu diesem übergeordneten Server.

Sie müssen nicht explizit Kopieknoten für das Scale-Out-System definieren, sondern können festlegen, dass der Stammknoten die Kopieknoten bei der Verbindungsaufnahme definiert.

OR SERVER-Klausel Verwenden Sie die OR SERVER-Klausel, um einen alternativen übergeordneten Server für den Kopieknoten anzugeben.

USING AUTO PARENT-Klausel Diese Klausel kann nur verwendet werden, wenn AS COPY angegeben ist. Diese Klausel veranlasst den Primärserver zum Zuweisen eines übergeordneten Servers.

Wenn Sie diese Klausel verwenden, um einen vorhandenen Kopieknottenserver zu ersetzen, ändern sich dadurch nicht die Definitionen des übergeordneten und alternativen übergeordneten Knotens für den Kopieknotten.

server-option-Klausel Die folgenden Optionen werden unterstützt:

- **connection_string-Serveroption** Gibt die Verbindungszeichenfolge an, die für die Verbindung mit dem Server verwendet werden soll. Die Verbindungszeichenfolge für einen Spiegelserver sollte weder Benutzer-ID noch Kennwort enthalten, weil diese nicht benutzt werden, wenn ein Spiegelserver eine Verbindung mit einem anderen Spiegelserver herstellt.

Eine Liste der Verbindungsparameter finden Sie unter „Verbindungsparameter“ [[SQL Anywhere Server - Datenbankadministration](#)].

- **logfile-Serveroption** Gibt den Speicherort der Datei an, die eine Zeile für jede zwischen Spiegelservern gesendete Anforderung enthält, wenn die Datenbankspiegelung verwendet wird. Diese Datei wird nur für die Fehlersuche verwendet.
- **preferred-Serveroption** Gibt an, ob der Server der bevorzugte Server im Spiegelungssystem ist. Sie können entweder YES oder NO angeben. Der bevorzugte Server übernimmt wenn möglich die Rolle des Primärservers. Geben Sie diese Option an, wenn Sie PARTNER-Server definieren.
- **state_file-Serveroption** Gibt den Speicherort der Datei an, die zur Verwaltung von Statusinformationen über das Spiegelungssystem verwendet wird. Diese Option ist bei der Datenbankspiegelung erforderlich. In einem Spiegelungssystem muss für Server mit dem Typ PARTNER eine Statusdatei angegeben werden. Bei Arbiterservern wird der Speicherort als Teil des Befehls zum Starten des Servers angegeben.

Bemerkungen

Diese Anweisung erstellt oder ersetzt eine Spiegelserverdefinition, ändert sie aber nicht. Wenn Sie eine Spiegelserverdefinition ändern möchten, verwenden Sie die ALTER MIRROR SERVER-Anweisung.

In einem Datenbankspiegelungssystem kann der Spiegelservertyp PRIMARY, MIRROR, ARBITER oder PARTNER sein.

In einem Scale-Out-System mit Schreibschutz kann der Spiegelservertyp PRIMARY, PARTNER oder COPY sein.

Spiegelservernamen für Server vom Typ PARTNER oder COPY müssen mit den Namen der Datenbankserver übereinstimmen, die Teil des Spiegelungssystems sind (den Namen, die mit der Serveroption -n verwendet werden). Diese Anforderung ermöglicht es jedem Datenbankserver, seine eigene Definition und die des übergeordneten Servers zu finden. Alle Kopieknottenserver müssen auch eindeutige Servernamen haben. Die Werte für *mirror-server-name*, *parent-name* und *server-name* müssen aus 7-Bit-ASCII-Zeichen bestehen.

Um einen Kopieknotten als Arbitrer für die Datenbank zu verwenden, die im Datenspiegelungssystem kopiert wird, erstellen Sie den Arbiterserver mit einem Namen, der nicht zum Servernamen einer der Datenbankserver im Hochverfügbarkeitssystem passt. In dieser Konfiguration wird der Name des Arbiters als Platzhalter in der Spiegelserverdefinition verwendet, um die Verbindungszeichenfolge für den Arbitrer aufzunehmen.

Privilegien

Sie müssen das **MANAGE ANY MIRROR SERVER**-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „**SYSMIRRORSERVER**-Systemansicht“ auf Seite 1464
- „Datenbankspiegelung“ [*SQL Anywhere Server - Datenbankadministration*]
- „**SQL Anywhere-Scale-Out mit Schreibschutz**“ [*SQL Anywhere Server - Datenbankadministration*]
- „**SET MIRROR OPTION**-Anweisung“ auf Seite 1034
- „**ALTER MIRROR SERVER**-Anweisung“ auf Seite 480
- „**COMMENT**-Anweisung“ auf Seite 573
- „**DROP MIRROR SERVER**-Anweisung“ auf Seite 815
- „Hinzufügen von untergeordneten Kopieknoten“ [*SQL Anywhere Server - Datenbankadministration*]
- „Fehlerbehandlung: Statusinformationsdateien von Partner und Arbitr“ [*SQL Anywhere Server - Datenbankadministration*]
- „Bevorzugte Datenbankserver in Datenbankspiegelungssystemen“ [*SQL Anywhere Server - Datenbankadministration*]
- „Automatisches Zuweisen des übergeordneten Knotens eines Kopieknotens“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers

Beispiel

Die folgende Anweisung erstellt einen Spiegelserver, der als Primärserver in einem Datenbankspiegelungssystem verwendet werden kann:

```
CREATE MIRROR SERVER "scaleout_primary"  
AS PRIMARY  
connection_string = 'server=scaleout_primary;host=winxp-2:6871,winxp-3:6872';
```

Die folgende Anweisung erstellt einen Spiegelserver, der als Spiegelserver in einem Datenbankspiegelungssystem verwendet werden kann:

```
CREATE MIRROR SERVER "scaleout_mirror"  
AS MIRROR  
connection_string = 'server=scaleout_mirror;host=winxp-2:6871,winxp-3:6872';
```

Die folgende Anweisung erstellt einen Spiegelserver, der als Arbiterserver in einem Datenbankspiegelungssystem verwendet werden kann:

```
CREATE MIRROR SERVER "scaleout_arbiter"  
AS ARBITER  
connection_string = 'server=scaleout_arbiter;host=winxp-4:6870';
```

Die folgende Anweisung erstellt zwei Spiegelserver, die als Partnerserver in einem Datenbankspiegelungssystem verwendet werden können:

```
CREATE MIRROR SERVER "scaleout_server1"  
AS PARTNER
```

```
connection_string = 'server=scaleout_server1;HOST=winxp-2:6871'
state_file = 'c:\\server1\\server1.state';
```

```
CREATE MIRROR SERVER "scaleout_server2"
AS PARTNER
connection_string = 'server=scaleout_server2;HOST=winxp-3:6872'
state_file = 'c:\\server2\\server2.state';
```

Die folgende Anweisung erstellt einen Kopieknotten, der in einem Datenbankspiegelungssystem als Arbitrer fungieren kann:

```
CREATE MIRROR SERVER "scaleout_child"
AS COPY FROM SERVER "scaleout_primary"
connection_string = 'server=scaleout_child;host=winxp-5:6878';
```

Die folgende Anweisung definiert einen Kopieknotten als Arbitrer für ein anderes Datenbankspiegelungssystem:

```
CREATE MIRROR SERVER "The Arbitrer"
AS ARBITER
connection_string = 'server=scaleout_child;host=winxp-5:6878';
```

Mit der folgenden Anweisung wird der aktuelle übergeordnete Knoten beibehalten, wenn *server-name* bereits vorhanden ist. Ein neuer übergeordneter Knoten wird jedoch nicht automatisch generiert.

```
CREATE OR REPLACE MIRROR SERVER "server-name" AS COPY USING AUTO PARENT;
```

CREATE PROCEDURE-Anweisung [externer Aufruf]

Erstellt eine Schnittstelle zu einer nativen oder externen Prozedur.

Syntax

```
CREATE [ OR REPLACE ] PROCEDURE [ owner. ] procedure-name
    ( [ parameter[, ... ] ] )
[ SQL SECURITY { INVOKER | DEFINER } ]
[ RESULT ( result-column [, ... ] ) | NO RESULT SET ]
[ DYNAMIC RESULT SETS integer-expression ]
{ EXTERNAL NAME 'native-call'
  | EXTERNAL NAME 'c-call' LANGUAGE { C_ESQL32 | C_ESQL64 | C_ODBC32 | C_ODBC64 }
  | EXTERNAL NAME 'clr-call' LANGUAGE CLR
  | EXTERNAL NAME 'perl-call' LANGUAGE PERL
  | EXTERNAL NAME 'php-call' LANGUAGE PHP
  | EXTERNAL NAME 'java-call' LANGUAGE JAVA }
```

```
parameter :
[ parameter-mode ] parameter-name data-type [ DEFAULT expression ]
| SQLCODE
| SQLSTATE
```

```
parameter-mode :
IN
| OUT
| INOUT
```

```

native-call :
[ operating-system:]function-name@library

result-column :
column-name data-type

c-call :
[ operating-system:]function-name@library; ...

clr-call :
dll-name::function-name( param-type-1[, ... ] )

perl-call :
<file=perl-file> $sa_perl_return = perl-subroutine( $sa_perl_arg0[, ... ] )

php-call :
<file=php-file> print php-func( $argv[1][, ... ] )

java-call :
[package-name.]class-name.method-name method-signature

operating-system :
Unix

method-signature :
( [ field-descriptor, ... ] ) return-descriptor

field-descriptor und return-descriptor :
{ Z
  | B
  | S
  | I
  | J
  | F
  | D
  | C
  | V
  | [descriptor
  | Lclass-name;
  }

```

Parameter

Sie können permanent gespeicherte Prozeduren erstellen, die externe oder native, mit unterschiedlichen Programmiersprachen erstellte Prozeduren aufrufen. PROC kann als Synonym für PROCEDURE verwendet werden.

OR REPLACE-Klausel Wenn Sie OR REPLACE angeben, wird eine neue Prozedur erstellt oder eine bestehende Prozedur mit demselben Namen ersetzt. Diese Klausel ändert die Definition der Prozedur, lässt aber vorhandene Privilegien unberührt. Ein Fehler wird zurückgegeben, wenn Sie eine Prozedur ersetzen, die gerade verwendet wird.

Parameter Parameternamen müssen den Regeln für andere Datenbankbezeichner, wie z.B. Spaltennamen, entsprechen. Es muss sich dabei um einen gültigen SQL-Datentyp handeln.

Parameter können eines der Schlüsselwörter IN, OUT oder INOUT vorangestellt haben. Wenn Sie keinen dieser Werte angeben, sind die Parameter standardmäßig INOUT. Die Schlüsselwörter haben die folgenden Bedeutungen:

- **IN** Dieser Parameter ist ein Ausdruck, der der Prozedur einen Wert zur Verfügung stellt.
- **OUT** Dieser Parameter ist eine Variable, die von der Prozedur einen Wert erhalten kann.
- **INOUT** Dieser Parameter ist eine Variable, die einen Wert für die Prozedur bereitstellt, und die von der Prozedur einen neuen Wert erhalten kann.

Wenn Prozeduren mit der CALL-Anweisung ausgeführt werden, müssen nicht alle Parameter angegeben werden. Wenn in der CREATE PROCEDURE-Anweisung ein Standardwert bereitgestellt wird, werden fehlenden Parametern die Standardwerte zugeordnet. Falls in der CALL-Anweisung kein Argument angegeben wurde und kein Standardwert gesetzt ist, wird ein Fehler ausgegeben.

SQLSTATE und SQLCODE sind spezielle OUT-Parameter, die den SQLSTATE- oder SQLCODE-Wert ausgeben, wenn die Prozedur beendet wird. Die Spezialwerte SQLSTATE und SQLCODE können sofort geprüft werden, nachdem ein Prozeduraufruf abgeschlossen wurde, um den Rückgabestatus der Prozedur zu testen.

Die Spezialwerte SQLSTATE und SQLCODE werden durch die nächste SQL-Anweisung geändert. Indem SQLSTATE oder SQLCODE als Prozedurargumente bereitgestellt werden, kann die Rückmeldung in einer Variablen gespeichert werden.

Wenn Sie OR REPLACE (CREATE OR REPLACE PROCEDURE) angeben, wird eine neue Prozedur erstellt oder eine bestehende Prozedur mit demselben Namen ersetzt. Diese Klausel ändert die Definition der Prozedur, lässt aber vorhandene Privilegien unberührt. Ein Fehler wird zurückgegeben, wenn Sie eine Prozedur ersetzen, die gerade verwendet wird.

Sie können keine extern aufgerufenen TEMPORARY-Prozeduren erstellen.

RESULT-Klausel Die RESULT-Klausel deklariert die Anzahl und den Typ der Spalten in der Ergebnismenge. Mit der Liste in Klammern nach dem Schlüsselwort RESULT werden die Namen und Typen der Ergebnisspalten festgelegt. Diese Angaben werden von Embedded SQL DESCRIBE oder von ODBC SQLDescribeCol zurückgegeben, wenn eine CALL-Anweisung beschrieben wird.

Externe Prozeduren in Embedded SQL (LANGUAGE C_ESQL32, LANGUAGE C_ESQL64) or ODBC (LANGUAGE C_ODBC32, LANGUAGE C_ODBC64) können 0 oder 1 Ergebnismenge zurückgeben.

Externe Prozeduren in Perl oder PHP (LANGUAGE PERL, LANGUAGE PHP) können keine Ergebnismengen zurückgeben. Prozeduren, die vom Datenbankserver geladene native Funktionen aufrufen, können ebenfalls keine Ergebnismengen zurückgeben.

Externe CLR- oder Java-Prozeduren (LANGUAGE CLR, LANGUAGE JAVA) können 0, 1 oder mehr Ergebnismengen zurückgeben.

Einige Prozeduren können mehr als eine Ergebnismenge mit unterschiedlicher Spaltenanzahl zurückgeben, je nachdem, wie sie ausgeführt werden. Die folgende Prozedur gibt beispielsweise unter bestimmten Bedingungen zwei Spalten und in anderen Fällen nur eine Spalte zurück.

```
CREATE PROCEDURE names( IN formal char(1))
BEGIN
  IF formal = 'n' THEN
    SELECT GivenName
    FROM GROUPO.Employees
  ELSE
    SELECT Surname, GivenName
    FROM Employees
  END IF
END;
```

Prozeduren mit variablen Ergebnismengen müssen ohne RESULT-Klausel oder in Transact-SQL geschrieben werden. Ihre Verwendung ist den folgenden Einschränkungen unterworfen:

- **Embedded SQL** Sie müssen den Prozeduraufruf mithilfe einer DESCRIBE-Anweisung beschreiben, nachdem der Cursor für die Ergebnismenge geöffnet wurde, aber bevor Zeilen zurückgegeben werden, damit die richtige Form der Ergebnismenge bezogen wird. Die Klausel *CURSOR cursor-name* in der DESCRIBE-Anweisung ist erforderlich.
- **ODBC, OLE DB, ADO.NET** Prozeduren mit variablen Ergebnismengen können von Anwendungen verwendet werden, die diese Schnittstellen benutzen. Die richtige Beschreibung der Ergebnismengen wird vom Treiber oder Provider vorgenommen.
- **Open Client-Anwendungen** Prozeduren mit variablen Ergebnismengen können von Open Client-Anwendungen verwendet werden.

Wenn Ihre Prozedur nur eine Ergebnismenge zurückgibt, sollten Sie eine RESULT-Klausel verwenden. Das Vorhandensein dieser Klausel verhindert, dass ODBC- und Open Client-Anwendungen die Ergebnismenge noch einmal beschreiben, nachdem der Cursor geöffnet wurde.

Um mehrere Ergebnismengen verarbeiten zu können, muss ODBC den aktuell ausgeführten Cursor beschreiben, nicht die definierte Ergebnismenge der Prozedur. Deshalb beschreibt ODBC die Spaltennamen nicht immer so, wie sie in der RESULT-Klausel der gespeicherten Prozedur definiert sind. Um dieses Problem zu vermeiden, verwenden Sie Spaltenaliasnamen in der SELECT-Anweisung, die die Ergebnismenge erzeugt.

NO RESULT SET-Klausel Deklariert, dass von dieser Prozedur keine Ergebnismenge zurückgegeben wird. Diese Deklaration kann zu einer Performanceverbesserung führen.

DYNAMIC RESULT SETS-Klausel Verwenden Sie diese Klausel mit LANGUAGE CLR- und LANGUAGE JAVA-Aufrufen. Die DYNAMIC RESULT SETS-Klausel wird verwendet, um die Anzahl der dynamischen Ergebnismengen anzugeben, die von der Prozedur zurückgegeben werden. Wenn eine RESULT-Klausel angegeben ist, nicht aber die Klausel DYNAMIC RESULT SETS, wird angenommen, dass die Anzahl der dynamischen Ergebnismengen 1 ist. Wenn weder die RESULT- noch die DYNAMIC RESULT SETS-Klausel angegeben sind, wird keine Ergebnismenge erwartet und ein Fehler gemeldet, wenn eine Ergebnismenge generiert wird.

Die externen Umgebungen C_ESQL32, C_ESQL64, C_ODBC32 und C_ODBC64 können auch Ergebnismengen zurückgeben, wie z.B. CLR und JAVA, sind aber auf nur eine dynamische Ergebnismenge beschränkt.

Prozeduren, die externe Funktionen in Perl oder PHP (LANGUAGE PERL, LANGUAGE PHP) aufrufen, können keine Ergebnismengen zurückgeben. Prozeduren, die native Funktionen aufrufen, die vom Datenbankserver geladen werden, können ebenfalls keine Ergebnismengen zurückgeben.

SQL SECURITY-Klausel Die SQL SECURITY-Klausel legt fest, ob die Prozedur als INVOKER (der Benutzer, der die Prozedur aufruft) oder als DEFINER (der Eigentümer der Prozedur) aufgerufen wird. Standardwert ist DEFINER. Bei externen Aufrufen richtet diese Klausel den Eigentümerkontext für nicht qualifizierte Objektreferenzen in der externen Umgebung ein.

Wenn SQL SECURITY INVOKER angegeben ist, wird mehr Speicher verwendet, weil für jeden Benutzer, der die Prozedur aufruft, ein Vermerk erfolgen muss. Außerdem erfolgt bei SQL SECURITY INVOKER auch eine Namensauflösung als Aufrufer. Sie sollten daher mit Umsicht vorgehen und alle Objektnamen (Tabellen, Prozeduren etc.) mit ihrem richtigen Eigentümer qualifizieren. Beispiel: Der Benutzer user1 erstellt die folgende Prozedur:

```
CREATE PROCEDURE user1.myProcedure()  
  RESULT( columnA INT )  
  SQL SECURITY INVOKER  
  BEGIN  
    SELECT columnA FROM table1;  
  END;
```

Wenn der Benutzer user2 versucht, diese Prozedur auszuführen und eine Tabelle user2.table1 *nicht existiert*, kommt es zu einem Tabellensuchfehler. Wenn eine Tabelle user2.table1 *existiert*, wird diese Tabelle anstelle der beabsichtigten user1.table1 verwendet. Um dies zu verhindern, qualifizieren Sie die Tabellenreferenz in der Anweisung (user1.table1 anstelle von table1).

EXTERNAL NAME-Klausel Eine Prozedur, die die EXTERNAL NAME-Klausel ohne LANGUAGE-Attribut verwendet, definiert eine Schnittstelle zu einer nativen Funktion, die in einer Programmiersprache wie C geschrieben ist. Die native Funktion wird vom Datenbankserver in seinen Adressenspeicher geladen.

Der *library*-Name kann eine Dateierweiterung enthalten. In der Regel handelt es sich um *.dll* unter Windows und *.so* unter Unix. Wenn keine Dateierweiterung angegeben ist, hängt die Software die plattformspezifische Dateierweiterung für Bibliotheken an. Es folgt ein formelles Beispiel.

```
CREATE PROCEDURE mystring( IN instr LONG VARCHAR )  
  EXTERNAL NAME 'mystring@mylib.dll;Unix:mystring@mylib.so';
```

Eine einfachere Methode, die oben genannte EXTERNAL NAME-Klausel mit plattformspezifischen Standardwerten zu schreiben, lautet wie folgt:

```
CREATE PROCEDURE mystring( IN instr LONG VARCHAR )  
  EXTERNAL NAME 'mystring@mylib';
```

Nach ihrem Aufruf wird die Bibliothek, die die Funktion enthält, in den Adressraum des Datenbankservers geladen. Die native Funktion wird als Teil des Servers ausgeführt. In diesem Fall wird, falls die Funktion einen Fehler verursacht, der Datenbankserver beendet. Aus diesem Grund wird empfohlen, Funktionen in einer externen Umgebung mit dem LANGUAGE-Attribut zu laden und auszuführen. Wenn eine Funktion einen Fehler in einer externen Umgebung verursacht, läuft der Datenbankserver weiter.

Hinweis zur Syntax bei Verwendung von *operating-system*: Wenn Sie *operating-system* nicht festlegen, dann wird angenommen, dass die Prozedur auf allen Plattformen läuft. Wenn Sie **Unix** für einen der Aufrufe festlegen, dann wird angenommen, dass der andere Aufruf für Windows vorgesehen ist.

- **EXTERNAL NAME 'c-call' LANGUAGE {C_ESQL32 | C_ESQL64 | C_ODBC32 | C_ODBC64 }-Klausel** Um eine kompilierte native C-Funktion in einer externen Umgebung statt im Datenbankserver aufzurufen, wird die gespeicherte Prozedur oder Funktion mit der Klausel EXTERNAL NAME definiert, gefolgt vom Attribut LANGUAGE, das C_ESQL32, C_ESQL64, C_ODBC32 oder C_ODBC64 angibt.

Wenn das LANGUAGE-Attribut angegeben ist, wird die Bibliothek, die die Funktion enthält, von einem externen Prozess geladen und die externe Funktion wird als Teil dieses externen Prozesses ausgeführt. In diesem Fall läuft der Datenbankserver weiter, falls die Funktion einen Fehler verursacht.

Nachstehend wird ein Beispiel für eine Prozedurdefinition angeführt.

```
CREATE PROCEDURE ODBCinsert(  
    IN ProductName CHAR(30),  
    IN ProductDescription CHAR(50)  
)  
NO RESULT SET  
EXTERNAL NAME 'ODBCexternalInsert@extodbc.dll'  
LANGUAGE C_ODBC32;
```

- **EXTERNAL NAME 'clr-call' LANGUAGE CLR-Klausel** Um eine .NET-Funktion in einer externen Umgebung aufzurufen, wird die Prozedurschnittstelle mit einer EXTERNAL NAME-Klausel definiert, auf die das LANGUAGE CLR-Attribut folgt.

Eine gespeicherte CLR-Prozedur oder -Funktion verhält sich wie eine gespeicherte SQL-Prozedur oder -Funktion, abgesehen davon, dass der Code für die Prozedur oder Funktion in einer .NET-Sprache wie C# oder Visual Basic geschrieben ist und die Ausführung der Prozedur oder Funktion außerhalb des Datenbankservers stattfindet (d.h. innerhalb eines separaten .NET-Programms).

```
CREATE PROCEDURE clr_interface(  
    IN p1 INT,  
    IN p2 UNSIGNED SMALLINT,  
    OUT p3 LONG VARCHAR)  
NO RESULT SET  
EXTERNAL NAME 'CLRlib.dll::CLRproc.Run( int, ushort, out string )'  
LANGUAGE CLR;
```

- **EXTERNAL NAME 'perl-call' LANGUAGE PERL-Klausel** Um eine Perl-Funktion in einer externen Umgebung aufzurufen, wird die Prozedurschnittstelle mit einer EXTERNAL NAME-Klausel definiert, auf die das LANGUAGE PERL-Attribut folgt.

Eine gespeicherte Perl-Prozedur oder Funktion verhält sich wie eine gespeicherte SQL-Prozedur oder Funktion, abgesehen davon, dass der Code für die Prozedur oder Funktion in Perl geschrieben ist und die Ausführung der Prozedur oder Funktion außerhalb des Datenbankservers stattfindet (d.h. innerhalb einer Perl-Programminstanz).

Nachstehend wird ein Beispiel für eine Prozedurdefinition angeführt.

```
CREATE PROCEDURE PerlWriteToConsole( IN str LONG VARCHAR)  
NO RESULT SET
```

```
EXTERNAL NAME '<file=PerlConsoleExample>
WriteToServerConsole( $sa_perl_arg0 )'
LANGUAGE PERL;
```

- **EXTERNAL NAME *php-call* LANGUAGE PHP-Klausel** Um eine PHP-Funktion in einer externen Umgebung aufzurufen, wird die Prozedurschnittstelle mit einer EXTERNAL NAME-Klausel definiert, auf die das LANGUAGE PHP-Attribut folgt.

Eine gespeicherte PHP-Prozedur oder -Funktion verhält sich wie eine gespeicherte SQL-Prozedur oder Funktion, abgesehen davon, dass der Code für die Prozedur oder Funktion in PHP geschrieben ist und die Ausführung der Prozedur oder Funktion außerhalb des Datenbankservers stattfindet (d.h. innerhalb einer PHP-Programminstanz).

Nachstehend wird ein Beispiel für eine Prozedurdefinition angeführt.

```
CREATE PROCEDURE PHPPopulateTable()
NO RESULT SET
EXTERNAL NAME '<file=ServerSidePHPEXample> ServerSidePHPSub()'
LANGUAGE PHP;
```

- **EXTERNAL NAME *java-call* LANGUAGE JAVA-Klausel** Um eine Java-Methode in einer externen Umgebung aufzurufen, wird die Prozedurschnittstelle mit einer EXTERNAL NAME-Klausel definiert, auf die das LANGUAGE JAVA-Attribut folgt.

Eine gespeicherte Prozedur oder Funktion über die Java-Schnittstelle verhält sich wie eine gespeicherte SQL-Prozedur oder Funktion, abgesehen davon, dass der Code für die Prozedur oder Funktion in Java geschrieben ist und die Ausführung der Prozedur oder Funktion außerhalb des Datenbankservers stattfindet (d.h. in einer Java VM).

Nachstehend wird ein Beispiel für eine Prozedurdefinition angeführt.

```
CREATE PROCEDURE HelloDemo( IN name LONG VARCHAR )
NO RESULT SET
EXTERNAL NAME 'Hello.main([Ljava/lang/String;)V'
LANGUAGE JAVA;
```

Die Deskriptoren für Argumente und Rückgabewerte von Java-Methoden haben die folgende Bedeutung:

Feldtyp	Java-Datentyp
B	byte
C	char
D	double
F	float
I	int
J	long

Feldtyp	Java-Datentyp
L <i>Klassenname</i> ;	Eine Instanz der Klasse <i>Klassenname</i> . Die Klasse muss voll qualifiziert sein und alle Punkte im Namen müssen durch das Zeichen / ersetzt werden. Zum Beispiel, java/lang/String .
S	short
V	void
Z	Boolescher Wert
[Verwenden Sie jeweils eine für jede Array-Dimension.

Bemerkungen

Die CREATE PROCEDURE-Anweisung erstellt eine Prozedur in der Datenbank. Sie können Prozeduren für andere Benutzer erstellen, indem Sie einen Eigentümer angeben. Eine Prozedur wird mit einer CALL-Anweisung aufgerufen.

Wenn eine gespeicherte Prozedur eine Ergebnismenge zurückgibt, kann sie nicht zusätzlich Ausgabeparameter setzen oder einen Rückgabewert zurückgeben.

Bei der Referenzierung einer temporären Tabelle durch mehrere Prozeduren kann ein Problem entstehen, wenn die Definitionen der temporären Tabelle nicht konsistent sind und Anweisungen, die die Tabelle referenzieren, im Cache abgelegt sind.

Privilegien

Sie müssen die CREATE PROCEDURE-Systemprivileg haben, um externe Prozeduren erstellen zu können, deren Eigentümer Sie sind.

Sie müssen das CREATE ANY PROCEDURE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben, um externe Prozeduren erstellen zu können, deren Eigentümer andere Benutzer sind.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Referenzen auf temporäre Tabellen innerhalb von Prozeduren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „ALTER PROCEDURE-Anweisung“ auf Seite 483
- „CALL-Anweisung“ auf Seite 564
- „CREATE FUNCTION-Anweisung“ auf Seite 633
- „CREATE FUNCTION-Anweisung [externer Aufruf]“ auf Seite 617
- „CREATE PROCEDURE-Anweisung“ auf Seite 681
- „CREATE PROCEDURE-Anweisung [Webdienste]“ auf Seite 670
- „CREATE PROCEDURE-Anweisung [T-SQL]“ auf Seite 679
- „DROP PROCEDURE-Anweisung“ auf Seite 816
- „GRANT-Anweisung“ auf Seite 881
- „SQL-Datentypen“ auf Seite 95
- „Die externen ESQL- und ODBC-Umgebungen“ [*SQL Anywhere Server - Programmierung*]
- „SQL Anywhere-Schnittstelle für externe Aufrufe“ [*SQL Anywhere Server - Programmierung*]
- „Unterstützung für externe Umgebungen in SQL Anywhere“ [*SQL Anywhere Server - Programmierung*]
- „Die externe PERL-Umgebung“ [*SQL Anywhere Server - Programmierung*]
- „Die externe CLR-Umgebung“ [*SQL Anywhere Server - Programmierung*]
- „Die externe PHP-Umgebung“ [*SQL Anywhere Server - Programmierung*]
- „Die externe Java-Umgebung“ [*SQL Anywhere Server - Programmierung*]

Standards und Kompatibilität

- **SQL/2008** CREATE PROCEDURE für eine externe Sprachenumgebung ist eine Kernfunktion des SQL/2008-Standards, obwohl einige der in SQL Anywhere unterstützten Komponenten optionale SQL/2008-Sprachenfunktionen sind. Zu diesen Funktionen gehören folgende:
 - Die SQL SECURITY-Klausel ist die optionale SQL/2008-Sprachenfunktion T324.
 - Die Möglichkeit zum Übergeben eines LONG VARCHAR-, LONG NVARCHAR- oder LONG BINARY-Werts an eine externe Prozedur ist SQL/2008-Sprachenfunktion T041.
 - Die Möglichkeit, ein Schema-Objekt innerhalb einer externen Prozedur mit Anweisungen wie CREATE TABLE und DROP TRIGGER zu erstellen oder zu ändern, ist SQL/2008-Sprachenfunktion T653.
 - Die Möglichkeit zum Verwenden einer Dynamic-SQL-Anweisung innerhalb einer externen Prozedur, einschließlich der Anweisungen CONNECT, EXECUTE IMMEDIATE, PREPARE und DESCRIBE, ist SQL/2008-Sprachenfunktion T654.
 - Externe JAVA-Prozeduren stellen SQL/2008-Sprachenfunktion J621 dar.

Einige Klauseln der CREATE PROCEDURE-Anweisung sind Erweiterungen des Herstellers. Es handelt sich dabei um die Folgenden:

- Die Unterstützung für C_ESQL32, C_ESQL64, C_ODBC32, C_ODBC64, CLR, PERL und PHP in der LANGUAGES-Klausel ist eine Erweiterung des Herstellers. Der SQL/2008-Standard unterstützt "C" als *environment-name* (optionale Sprachenfunktion B122).

- Das Format von *external_call* wird durch die Implementierung festgelegt.
- Die Klauseln **RESULT** und **NO RESULT SET** sind Erweiterungen des Herstellers. Der SQL/2008-Standard verwendet die **RETURNS**-Klausel.
- Die optionale **DEFAULT**-Klausel für einen bestimmten Routinenparameter ist eine Erweiterung des Herstellers.
- Die optionale **OR REPLACE**-Klausel ist eine Erweiterung des Herstellers.
- **Transact-SQL** **CREATE PROCEDURE** für eine externe Routine wird von Adaptive Server Enterprise unterstützt. Adaptive Server Enterprise unterstützt externe Routinen für die Programmiersprachen C und Java.

CREATE PROCEDURE-Anweisung [Webedienste]

Erstellt eine benutzerdefinierte Webclient-Prozedur, die HTTP- oder SOAP-Anforderungen an einen HTTP-Server sendet.

Syntax

```
CREATE [ OR REPLACE ] PROCEDURE [ owner.]procedure-name ( [ parameter, ... ] )
[ RESULT ( attribute-column-name datatype, value-column-name datatype ) ]
URL url-string
[ TYPE { http-type-spec-string | soap-type-spec-string } ]
[ HEADER header-string ]
[ CERTIFICATE certificate-string ]
[ CLIENTPORT clientport-string ]
[ PROXY proxy-string ]
[ SET protocol-option-string ]
[ SOAPHEADER soap-header-string ]
[ NAMESPACE namespace-string ]
```

http-type-spec-string :

```
HTTP[: { GET
| POST[:MIME-type ]
| PUT[:MIME-type ]
| DELETE
| HEAD } ]
```

soap-type-spec-string :

```
SOAP[:{ RPC | DOC }]
```

parameter :

```
parameter-mode parameter-name data-type [ DEFAULT expression ]
```

parameter-mode :

```
IN
| OUT
| INOUT
```

url-string :

```
{ HTTP | HTTPS | HTTPS_FIPS }://[user:password@]hostname[:port][/path]
```

```

protocol-option-string
[ http-option-list
, soap-option-list ]

http-option-list :
HTTP(
[ CH[UNK]={ ON | OFF | AUTO } ]
[ ; VER[SION]={ 1.0 | 1.1 };kto=number-of-seconds ]
[ REDIR(COUNT=count, STATUS=status-list) ]
)

soap-option-list :
SOAP(OP[ERATION]=soap-operation-name)

```

Parameter

Sie können eine Webdienst-Clientprozedur erstellen oder ersetzen. PROC kann als Synonym für PROCEDURE verwendet werden.

Für SOAP-Anforderungen wird der Prozedurname standardmäßig als SOAP-Vorgangsname verwendet. Weitere Hinweise finden Sie unter der SET-Klausel.

Hinweis

Sie können keine TEMPORARY-Webdienst-Prozeduren erstellen.

OR REPLACE-Klausel Wenn Sie OR REPLACE angeben, wird eine neue Prozedur erstellt oder eine bestehende Prozedur mit demselben Namen ersetzt. Diese Klausel ändert die Definition der Prozedur, lässt aber vorhandene Privilegien unberührt. Ein Fehler wird zurückgegeben, wenn Sie eine Prozedur ersetzen, die gerade verwendet wird.

Parameter Parameternamen müssen den Regeln für andere Datenbankbezeichner, wie z.B. Spaltennamen, entsprechen. Es muss sich dabei um einen gültigen SQL-Datentyp handeln.

Nur SOAP-Anforderungen unterstützen die Übertragung von eingetippten Daten des Typs FLOAT, INT etc. HTTP-Anforderungen unterstützen nur die Übertragung von Zeichenfolge, daher sind sie hier auf CHAR-Datentypen beschränkt.

Wenn Prozeduren mit der CALL-Anweisung ausgeführt werden, müssen nicht alle Parameter angegeben werden. Wenn in der CREATE PROCEDURE-Anweisung ein Standardwert bereitgestellt wird, werden fehlenden Parametern die Standardwerte zugeordnet. Falls kein Argument in der CALL-Anweisung angegeben wurde und kein Standardwert gesetzt ist, wird ein Fehler ausgegeben.

Parameter können eines der Schlüsselwörter IN, OUT oder INOUT vorangestellt haben. Wenn Sie keinen dieser Werte angeben, sind die Parameter standardmäßig INOUT. Die Schlüsselwörter haben die folgenden Bedeutungen:

- **IN** Dieser Parameter ist ein Ausdruck, der der Prozedur einen Wert zur Verfügung stellt.
- **OUT** Dieser Parameter ist eine Variable, die von der Prozedur einen Wert erhalten kann.
- **INOUT** Dieser Parameter ist eine Variable, die einen Wert für die Prozedur bereitstellt und von der Prozedur einen neuen Wert erhalten kann.

RESULT-Klausel Die RESULT-Klausel ist erforderlich, um die Prozedur in einer SELECT-Anweisung verwenden zu können. Die RESULT-Klausel muss zwei Spalten zurückgeben. Die erste Spalte enthält Attribute zu HTTP-Antwort-Header, Status und Antworthauptteil, die zweite Spalte enthält die Werte für diese Attribute. Die RESULT-Klausel muss zwei Zeichendatentypen angeben. Beispiel: VARCHAR oder LONG VARCHAR. Wenn die RESULT-Klausel nicht angegeben ist, lauten die Standardspaltennamen "Attribute" und "Value" und der dazugehörige Datentyp LONG VARCHAR.

URL-Klausel Gibt den URI des Webdiensts an. Die optionalen Benutzernamen- und Kennwortparameter bieten eine Möglichkeit, die für die HTTP Basic Authentication erforderlichen Anmeldeinformationen zu liefern. Die HTTP Basic Authentication Base-64 verschlüsselt die Benutzer- und Kennwortinformationen und übergibt sie an den Header "Authentication" der HTTP-Anforderung. Bei dieser Art der Angabe werden Benutzername und Kennwort unverschlüsselt, als Teil der URL, übergeben.

Für Prozeduren des Typs HTTP:GET können Abfrageparameter innerhalb der URL-Klausel angegeben werden. Außerdem werden sie automatisch aus Parametern generiert, die an eine Prozedur übergeben werden.

```
URL 'http://localhost/service?parm=1'
```

TYPE-Klausel Gibt das für die Webdienstanforderung verwendete Format an. SOAP:RPC wird verwendet, wenn SOAP angegeben oder keine TYPE-Klausel enthalten ist. HTTP:POST wird verwendet, wenn HTTP angegeben ist.

Die TYPE-Klausel ermöglicht die Angabe eines MIME-Typs für HTTP:GET-, HTTP:POST- und HTTP:PUT-Typen. Mithilfe des Werts für den *MIME-type* wird der Anforderungs-Header "Content-Type" eingerichtet und der Betriebsmodus so eingestellt, dass nur ein einzelner Aufrufparameter den Hauptteil der Anforderung auffüllen kann. Wenn ein Webdienstauftrag einer gespeicherten Prozedur erfolgt, nachdem Parameterersetzungen verarbeitet wurden, darf kein oder nur ein einziger Parameter verbleiben. Der Aufruf einer Webdienstfunktion mit NULL oder keinem Parameter (nach Ersetzungen) ergibt eine Anfrage ohne Hauptteil und mit einer Inhaltslänge von Null. Parameternamen und -werte (Mehrfachparameter sind erlaubt) werden innerhalb des Hauptteils der HTTP-Anforderungen URL-kodiert.

Zu den typischen MIME-Typen gehören folgende:

- text/plain
- text/html
- text/xml

Die Schlüsselwörter für die TYPE-Klausel haben die folgende Bedeutung:

- **'HTTP:GET'** Standardmäßig verwendet dieser Typ den MIME-Typ application/x-www-form-urlencoded zum Kodieren von in der URL angegebenen Parametern.

Die folgende Anforderung wird beispielsweise erzeugt, wenn ein Client eine Anforderung aus der URL `http://localhost/WebServiceName?arg1=param1&arg2=param2` sendet:

```
GET /WebServiceName?arg1=param1&arg2=param2 HTTP/1.1
// <End of Request - NO BODY>
```

- **'HTTP:POST'** Standardmäßig verwendet dieser Typ den MIME-Typ `application/x-www-form-urlencoded` zum Kodieren von im Hauptteil einer POST-Anforderung angegebenen Parametern. URL-Parameter werden im Hauptteil der Anforderung gespeichert.

Die folgende Anforderung wird beispielsweise erzeugt, wenn ein Client eine Anforderung aus der URL `http://localhost/WebServiceName?arg1=param1&arg2=param2` sendet:

```
POST /WebServiceName HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 19
arg1=param1&arg2=param2
// <End of Request>
```

- **HTTP:PUT** HTTP:PUT ist ähnlich wie HTTP:POST, aber der HTTP:PUT-Typ verfügt nicht über einen Standardmedientyp.

Das folgende Beispiel zeigt, wie Sie eine allgemeine Clientprozedur konfigurieren, die Daten auf einen SQL Anywhere-Server hochlädt, auf dem das Beispiel `%SQLANYAMP16%\SQLAnywhere\HTTP\put_data.sql` ausgeführt wird:

```
ALTER PROCEDURE CPUT("data" LONG VARCHAR, resnm LONG VARCHAR, mediatype
LONG VARCHAR)
URL 'http://localhost/resource/!resnm'
TYPE 'HTTP:PUT:!mediatype';

CALL CPUT('hello world', 'hello', 'text/plain');
```

- **HTTP:DELETE** Eine Webdienst-Clientprozedur kann so konfiguriert werden, dass eine auf einem Server befindliche Ressource gelöscht wird. Das Angeben des Medientyps ist optional.

Das folgende Beispiel zeigt, wie Sie eine allgemeine Clientprozedur für das Löschen von Daten aus einem SQL Anywhere-Server, der das Beispiel `put_data.sql` ausführt, konfigurieren:

```
ALTER PROCEDURE CDEL(resnm LONG VARCHAR, mediatype LONG VARCHAR)
URL 'http://localhost/resource/!resnm'
TYPE 'HTTP:DELETE:!mediatype';

CALL CDEL('hello', 'text/plain');
```

- **HTTP:HEAD** Die head-Methode ist identisch mit einer GET-Methode, aber der Server gibt keinen Hauptteil zurück. Ein Medientyp kann angegeben werden.

```
ALTER PROCEDURE CHEAD(resnm LONG VARCHAR)
URL 'http://localhost/resource/!resnm'
TYPE 'HTTP:HEAD';

CALL CHEAD('hello');
```

- **'SOAP:RPC'** Dieser Typ setzt den Content-Type-Header auf `'text/xml'`. SOAP-Vorgänge und-Parameter werden in SOAP Envelope-XML-Dokumenten gekapselt.
- **'SOAP:DOC'** Dieser Typ setzt den Content-Type-Header auf `'text/xml'`. Er ähnelt dem SOAP:RPC-Typ, ermöglicht jedoch das Senden umfangreicherer Datentypen. SOAP-Vorgänge und-Parameter werden in SOAP Envelope-XML-Dokumenten gekapselt.

Das Angeben eines MIME-Typs für die TYPE-Klausel setzt automatisch den Content-Type-Header auf diesen MIME-Typ.

HEADER-Klausel Wenn Sie HTTP-Webdienst-Clientprozeduren erstellen, verwenden Sie diese Klausel, um Headereinträge der HTTP-Anforderung hinzuzufügen, sie zu ändern oder zu löschen. Die Angabe von Headern ist dem Format sehr ähnlich, das im RFC2616 Hypertext Transfer Protocol - HTTP/1.1 und im RFC822-Standard für ARPA Internet-Textnachrichten verwendet wird, einschließlich der Tatsache, dass nur druckbare ASCII-Zeichen für HTTP-Header angegeben werden können, deren Groß- und Kleinschreibung nicht berücksichtigt wird.

Header können definiert werden als *header-name:value-name*-Paare. Jeder Header wird mit einem Doppelpunkt (:) von seinem Wert getrennt und darf daher keinen Doppelpunkt enthalten. Sie können mehrere Header definieren, indem Sie die Paare mit `\n`, `\x0d\n`, `<LF>` (Zeilenvorschub), oder `<CR><LF>` voneinander trennen. (Wagenrücklauf gefolgt von einem Zeilenvorschub)

Mehrere aufeinanderfolgende Leerzeichen innerhalb des Headers werden in ein einziges Leerzeichen konvertiert.

CERTIFICATE-Klausel Um eine sichere (HTTPS-)Anforderung durchführen zu können, muss der Client Zugriff auf das vom HTTPS-Server verwendete Zertifikat haben. Die benötigten Informationen werden in einer Zeichenfolge von durch Semikola getrennten Schlüssel/Werte-Paaren angegeben. Sie können mithilfe des Dateischlüssels den Dateinamen für das Zertifikat angeben oder mithilfe des Zertifikatschlüssels das Serverzertifikat in einer Zeichenfolge angeben. Sie können nicht sowohl einen Dateischlüssel als auch einen Zertifikatschlüssel angeben. Folgende Schlüssel sind verfügbar:

Taste	Abkürzung	Beschreibung
file		Der Dateiname des Zertifikats
certificate	cert	Das Zertifikat selbst
certificate_name	cert_name	Der Name eines in der Datenbank gespeicherten Zertifikats
company	co	Die im Zertifikat angegebene Organisation
unit		Die im Zertifikat angegebene Organisations-einheit
name		Der im Zertifikat angegebene allgemeine Name

Zertifikate sind nur bei Anforderungen erforderlich, die entweder an den HTTPS-Server gerichtet sind oder von einem nicht-sicheren zu einem sicheren Server umgeleitet werden können. Nur PEM-formatierte Zertifikate werden unterstützt.

CLIENTPORT-Klausel Kennzeichnet die Portnummer, auf der die HTTP-Clientprozedur mithilfe von TCP/IP kommuniziert. Sie wird nur für Verbindungen über Firewalls bereitgestellt und empfohlen, die "ausgehende" TCP/IP-Verbindungen filtern. Sie können eine einzelne Portnummer, Bereiche von Portnummern oder eine Kombination von beiden angeben, z.B. CLIENTPORT "85,90-97".

PROXY-Klausel Gibt den URI eines Proxyservers an. Wird verwendet, wenn der Client über einen Proxyserver auf das Netzwerk zugreifen muss. Zeigt an, dass die Prozedur eine Verbindung zum Proxyserver herstellen soll, um die Anforderung durch ihn zum Webdienst zu senden.

SET-Klausel Gibt Optionen für protokollspezifisches Verhalten für HTTP und SOAP an. Die folgende Liste beschreibt die unterstützten SET-Optionen. CHUNK, VERSION, REDIR und kto gelten für das HTTP-Protokoll und OPERATION gilt für das SOAP-Protokoll.

- **'HTTP(CH[UNK]=option)'** (HTTP oder SOAP) Mit dieser Option können Sie angeben, ob das Aufteilen in Abschnitte verwendet werden soll. Dies ermöglicht es, HTTP-Nachrichten in mehrere Abschnitte aufzuteilen. Mögliche Werte sind: ON (immer Abschnitte verwenden), OFF (keine Abschnitte) und AUTO (Abschnitte nur verwenden, wenn der Inhalt, ausgenommen automatisch generiertes Markup, 8196 Byte überschreitet). Die folgende SET-Klausel aktiviert z.B. das Aufteilen in Abschnitte:

```
SET 'HTTP(CHUNK=ON)'
```

Wenn die CHUNK-Option nicht angegeben ist, ist das Standardverhalten AUTO. Wenn eine in Abschnitte aufgeteilte Anforderung im AUTO-Modus mit dem Status 505 (HTTP Version Not Supported - HTTP-Version nicht unterstützt), mit 501 (Not Implemented - Nicht implementiert) oder mit 411 (Length Required - Länge erforderlich) fehlschlägt, wiederholt der Client die Anforderung ohne eine in Abschnitte aufgeteilte Übertragungskodierung.

Definieren Sie die CHUNK-Option mit OFF (keine Aufteilung), wenn der HTTP-Server in Abschnitte aufgeteilte übertragungskodierte Anforderungen nicht unterstützt.

Da der CHUNK-Modus die Übertragungskodierung ab HTTP-Version 1.1 unterstützt, erfordert die Definition von CHUNK mit ON, dass die Version (VER) auf 1.1 oder gar nicht gesetzt wird. Im letzten Fall wird 1.1 als die Standardversion verwendet.

- **'HTTP(VER[SION]=ver;kto=number-of-seconds)'** (HTTP oder SOAP) Mit dieser Option können Sie die Version des HTTP-Protokolls angeben, das für das Format der HTTP-Nachricht verwendet wird. Beispiel: Die folgende SET-Klausel setzt die HTTP-Version auf 1.1:

```
SET 'HTTP(VERSION=1.1)'
```

Mögliche Werte sind 1.0 und 1.1. Wenn VERSION nicht angegeben ist:

- Wenn CHUNK auf ON gesetzt ist, wird die HTTP-Version 1.1 verwendet.
- Wenn CHUNK auf OFF gesetzt ist, wird die HTTP-Version 1.0 verwendet.
- Wenn CHUNK auf AUTO gesetzt ist, wird entweder 1.0 oder 1.1 verwendet, abhängig davon, ob der Client im CHUNK-Modus sendet.
- **kto=number-of-seconds** Gibt die Kriterien für das Keepalive-Timeout an (**kto**) und ermöglicht es so einer Webclient-Prozedur, eine HTTP/HTTPS-Keepalive-Verbindung für einen bestimmten Zeitraum zu instanziiieren und im Cache abzulegen. Damit eine HTTP-Keepalive-Verbindung im Cache abgelegt werden kann, muss die HTTP-Version auf 1.1 gesetzt werden und kto auf einen anderen Wert als Null. kto kann besonders bei HTTPS-Verbindungen nützlich sein,

wenn Sie einen erheblichen Performanceunterschied zwischen HTTP- und HTTPS-Verbindungen feststellen. Eine Datenbankverbindung kann nur eine einzelne HTTP-Keepalive-Verbindung im Cache ablegen. Bei nachfolgenden Aufrufen einer Webclient-Prozedur mit demselben URI wird die Keepalive-Verbindung wiederverwendet. Der ausführende Webclient-Aufruf muss deshalb einen URI enthalten, dessen Schema, Zielhost und Port mit denjenigen des im Cache abgelegten URI übereinstimmen, und in der HEADER-Klausel darf nicht "Connection: close" angegeben werden. Wenn `kto` nicht angegeben oder auf Null gesetzt wird, werden HTTP/HTTPS-Verbindungen werden nicht im Cache abgelegt.

- **'REDIR(COUNT=count, STATUS=status-list)'** (HTTP oder SOAP) Die HTTP-Antwortstatuscodes wie **302 Found** und **303 See Other** werden verwendet, um Webanwendungen zu einem neuen URI umzuleiten, vor allem, wenn HTTP POST ausgeführt wurde. Eine Clientanforderung kann zum Beispiel folgendermaßen aussehen:

```
GET /people/alice HTTP/1.1
Host: www.example.com
Accept: text/html, application/xhtml+xml
Accept-Language: en, de
```

Die Webserverantwort kann folgendermaßen aussehen:

```
HTTP/1.1 302 Found
Location: http://www.example.com/people/alice.en.html
```

Als Antwort würde der Client eine andere HTTP-Anforderung an den neuen URI senden. Mit der REDIR-Option können Sie die maximale Anzahl der zulässigen Umleitungen steuern und festlegen, welche HTTP-Antwortstatuscodes automatisch umgeleitet werden.

`SET 'REDIR(count=3, status=301,307)'` ermöglicht beispielsweise eine maximale Anzahl von 3 Umleitungen und eine Umleitung für die Status 301 und 307. Wenn einer der anderen Umleitungsstatuscodes wie z.B. 302 oder 303 empfangen wird, wird ein Fehler ausgegeben (SQLCODE-983).

Die Anzahl (*count*) der Umleitungen ist standardmäßig auf 5 begrenzt. Standardmäßig wird eine HTTP-Clientprozedur automatisch als Antwort zu allen HTTP-Umleitungsstatuscodes (301, 302, 303, 307) umgeleitet. Um keine Umleitungsstatuscodes zuzulassen, verwenden Sie `SET REDIR(COUNT=0)`. In diesem Modus führt die Antwort auf eine Umleitung nicht zu einem Fehler (SQLCODE-983). Stattdessen wird eine Ergebnismenge mit dem HTTP-Status und Antwort-Headern zurückgegeben. Dies ermöglicht es dem Aufrufer, die Anforderung basierend auf dem URI im **Location**-Header bedingt neu auszugeben.

Eine Webdienstprozedur, die eine POST HTTP-Methode festlegt, die den Status **303 See Other** empfängt, gibt mit der GET HTTP-Methode eine Umleitungsanforderung aus.

Der **Location**-Header kann entweder einen absoluten Pfad oder einen relativen Pfad enthalten. Die HTTP-Clientprozedur verarbeitet beide. Der Header kann auch Abfrageparameter enthalten. Diese werden an den umgeleiteten Speicherort weitergeleitet. Beispiel: Wenn der Header Parameter wie z.B. die folgenden enthält, umfasst der nachfolgende GET- oder ein POST-Wert diese Parameter.

```
Location: alternate_service?a=1&b=2
```

Im oben gezeigten Beispiel lauten die Abfrageparameter `a=1&b=2`.

- **' SOAP(OP[ERATION]=soap-operation-name)** (Nur SOAP) Mit dieser Option können Sie den Namen des SOAP-Vorgangs angeben, falls er sich vom Namen der Prozedur unterscheidet, die Sie erstellen. Der Wert für OPERATION ist mit dem Namen eines entfernten Prozeduraufrufs vergleichbar. Beispiel: Wenn Sie eine Prozedur mit dem Namen accounts_login erstellen wollen, die einen SOAP-Vorgang mit dem Namen login aufruft, geben Sie in etwa Folgendes ein:

```
CREATE PROCEDURE accounts_login(
    name LONG VARCHAR,
    pwd LONG VARCHAR )
SET 'SOAP(OPERATION=login)'
```

Wenn die OPERATION-Option nicht angegeben ist, muss der Name des SOAP-Vorgangs mit dem Namen der Prozedur übereinstimmen, die Sie erstellen.

Die folgende Anweisung zeigt, wie mehrere *protocol-option*-Einstellungen in derselben SET-Klausel kombiniert werden:

```
CREATE PROCEDURE accounts_login(
    name LONG VARCHAR,
    pwd LONG VARCHAR )
SET 'HTTP ( CHUNK=ON; VERSION=1.1 ), SOAP( OPERATION=login )'
...
```

SOAPHEADER-Klausel (Nur SOAP-Format) Wenn Sie einen SOAP-Webdienst als Prozedur deklarieren, verwenden Sie diese Klausel, um einen oder mehrere Headereinträge der SOAP-Anforderung anzugeben. Ein SOAP-Header kann als statische Konstante deklariert oder dynamisch mithilfe des Parameterersatzungsmechanismus (IN-, OUT- oder INOUT-Parameter für hd1, hd2 etc. deklarieren) gesetzt werden. Eine Webdienstprozedur kann einen oder mehrere IN-Modus-Ersetzungsparameter und einen einzigen INOUT- oder OUT-Ersetzungsparameter definieren.

Das folgende Beispiel zeigt, wie ein Client das Senden von mehreren Headereinträgen mit Parameterersetzung und das Empfangen der Antwort-SOAP-Header-Daten angeben kann:

```
CREATE PROCEDURE soap_client
(INOUT hd1 LONG VARCHAR, IN hd2 LONG VARCHAR, IN hd3 LONG VARCHAR)
URL 'localhost/some_endpoint'
SOAPHEADER '!hd1!hd2!hd3';
```

NAMESPACE-Klausel (Nur SOAP-Format) Diese Klausel kennzeichnet den Methoden-Namespace, der üblicherweise sowohl bei SOAP:RPC- als auch bei SOAP:DOC-Anforderungen erforderlich ist. Der die Anforderung verarbeitende SOAP-Server verwendet diesen Namespace, um die Namen von Entitäten im Hauptteil der SOAP-Anforderung zu interpretieren. Der Namespace kann anhand der WSDL-Beschreibung (WSDL = Web Services Description Language) des SOAP-Diensts ermittelt werden, die der Webdienstserver zur Verfügung stellt. Der Standardwert ist die URL der Prozedur bis zur optionalen Pfadkomponente (die aber nicht eingeschlossen ist).

Bemerkungen

Parameterwerte werden als Teil der Anforderung weitergegeben. Die verwendete Syntax hängt vom Typ der Anforderung ab. Bei HTTP:GET werden die Parameter als Teil der URL weitergegeben. Bei HTTP:POST-Anforderungen werden die Werte in den Hauptteil der Anforderung platziert. Parameter von SOAP-Anforderungen werden immer im Anforderungshauptteil gebündelt.

Privilegien

Sie müssen das CREATE PROCEDURE-Systemprivileg haben, um Prozeduren erstellen zu können, deren Eigentümer Sie sind. Sie müssen das CREATE ANY PROCEDURE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben, um Prozeduren erstellen zu können, deren Eigentümer andere Benutzer sind.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „ALTER PROCEDURE-Anweisung“ auf Seite 483
- „CALL-Anweisung“ auf Seite 564
- „CREATE FUNCTION-Anweisung“ auf Seite 633
- „CREATE FUNCTION-Anweisung [Webdienst]“ auf Seite 624
- „CREATE PROCEDURE-Anweisung“ auf Seite 681
- „CREATE PROCEDURE-Anweisung [T-SQL]“ auf Seite 679
- „CREATE PROCEDURE-Anweisung [externer Aufruf]“ auf Seite 661
- „DROP PROCEDURE-Anweisung“ auf Seite 816
- „Strukturierte SOAP-Datentypen“ [*SQL Anywhere Server - Programmierung*]
- „HTTP- und SOAP-Anforderungsstrukturen“ [*SQL Anywhere Server - Programmierung*]
- „HTTP-Anforderungsheader verwalten“ [*SQL Anywhere Server - Programmierung*]
- „Praktische Einführung: Verwenden von SQL Anywhere für den Zugriff auf einen SOAP/DISH-Dienst“ [*SQL Anywhere Server - Programmierung*]
- „Praktische Einführung: Webserver erstellen und über einen Webclient darauf zugreifen“ [*SQL Anywhere Server - Programmierung*]
- „An Webdienste übergebene Variable“ [*SQL Anywhere Server - Programmierung*]
- „GRANT-Anweisung“ auf Seite 881
- „SQL Anywhere als HTTP-Webserver“ [*SQL Anywhere Server - Programmierung*]
- „Webclient-Anwendungsentwicklung“ [*SQL Anywhere Server - Programmierung*]
- „remote_idle_timeout-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Für Klauselwerte verwendete Ersetzungsparameter“ [*SQL Anywhere Server - Programmierung*]
- „Webclient-Anwendungsentwicklung“ [*SQL Anywhere Server - Programmierung*]
- „ClientPort-Protokolloption (CPORT) (nur clientseitig)“ [*SQL Anywhere Server - Datenbankadministration*]
- „HTTP-Anforderung fehlgeschlagen. Statuscode '%1'“ [*Fehlermeldungen*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** Wird von Adaptive Server Enterprise nicht unterstützt.

Beispiel

Im folgenden Beispiel wird eine Webdienst-Clientprozedur namens FtoC erstellt.

```
CREATE PROCEDURE FtoC( IN temperature FLOAT,  
    INOUT inouthead LONG VARCHAR,  
    IN inheader LONG VARCHAR )  
    URL 'http://localhost:8082/FtoCService'
```

```
TYPE 'SOAP:DOC'
SOAPHEADER '!inoutheader!inheader';
```

Im folgenden Beispiel eine sichere Webdienst-Clientprozedur namens SecureSendWithMimeType erstellt, die ein in der Datenbank gespeichertes Zertifikat verwendet.

```
CREATE CERTIFICATE client_cert
FROM FILE 'C:\\Users\\Public\\Documents\\SQL Anywhere 16\\Samples\\
\\Certificates\\rsaroot.crt';

CREATE PROCEDURE SecureSendWithMimeType(
    value LONG VARCHAR,
    mimeType LONG VARCHAR,
    urlSpec LONG VARCHAR
)
URL '!urlSpec'
CERTIFICATE 'certificate_name=client_cert'
TYPE 'HTTPS:POST:!mimeType';

CALL SecureSendWithMimeType('<hello>this is xml</hello>',
    'text/xml',
    'https://localhost:4043/EchoService'
);
```

CREATE PROCEDURE-Anweisung [T-SQL]

Erstellt eine neue Prozedur in der Datenbank so, dass sie mit Adaptive Server Enterprise kompatibel ist.

Syntax

Die folgende Teilmenge der Transact-SQL-Anweisung CREATE PROCEDURE wird in SQL Anywhere unterstützt.

```
CREATE [ OR REPLACE ]PROCEDURE [owner.]procedure_name
[ NO RESULT SET ]
[ [ ( ) @parameter-name data-type [ = default ] [ OUTPUT ], ... ( ) ] ]
[ WITH RECOMPILE ] AS statement-list
```

Parameter

OR REPLACE-Klausel Wenn Sie OR REPLACE angeben, wird eine neue Prozedur erstellt oder eine bestehende Prozedur mit demselben Namen ersetzt. Diese Klausel ändert die Definition der Prozedur, lässt aber vorhandene Privilegien unberührt. Ein Fehler wird zurückgegeben, wenn Sie eine Prozedur ersetzen, die gerade verwendet wird.

NO RESULT SET-Klausel Deklariert, dass von dieser Prozedur keine Ergebnismenge zurückgegeben wird. Das ist nützlich, wenn eine externe Umgebung wissen muss, dass eine Prozedur keine Ergebnismenge zurückgibt.

WITH RECOMPILE-Klausel Diese Klausel wird zur Kompatibilität mit Transact-SQL akzeptiert, aber ignoriert. SQL Anywhere kompiliert Prozeduren neu, wenn sie beim Start der Datenbank zum ersten Mal ausgeführt werden, und speichert die kompilierte Prozedur, bis die Datenbank angehalten wird.

Bemerkungen

Im Folgenden werden Unterschiede zwischen Transact-SQL- und SQL Anywhere-Anweisungen (Watcom-SQL) als Hilfe für alle aufgelistet, die in beiden Dialekten schreiben.

- **Variablenamen mit Präfix @** Das Zeichen "@" bezeichnet einen Transact-SQL-Variablenamen. Eine Watcom-SQL-Variable kann dagegen jeder gültige Bezeichner sein und das Präfix @ ist optional.
- **Eingabe- und Ausgabeparameter** Watcom-SQL-Prozedurparameter sind standardmäßig auf INOUT gesetzt. Sie können aber auch IN, OUT oder INOUT angeben. Transact-SQL-Prozedurparameter sind standardmäßig INPUT-Parameter. Sie können durch Hinzufügen des Schlüsselworts OUTPUT als INPUT/OUTPUT angegeben werden. Im Transact-SQL-Dialekt gibt es keine reinen OUTPUT-Parameter.

Wenn Sie im Watcom-SQL-Dialekt einen OUT-Parameter deklarieren, ist dies ein reiner OUTPUT-Parameter. Die Mischen von Dialekten wird nicht empfohlen, da es Probleme verursachen kann, wenn die Prozedurdeklaration entladen und zum Neuerstellen der Datenbank verwendet wird. In diesem Fall wird die neue Prozedurdeklaration im Transact-SQL-Dialekt erstellt, das Schlüsselwort OUTPUT wird verwendet und es handelt sich um einen INPUT/OUTPUT-Parameter.

- **Standardwerte für Parameter** In Watcom-SQL werden Prozedurparametern mit dem Schlüsselwort DEFAULT Standardwerte zugewiesen. In Transact-SQL wird ein Gleichheitszeichen (=) verwendet, um den Standardwert bereitzustellen.
- **Ergebnismengen liefern** Watcom-SQL verwendet eine RESULT-Klausel, um zurückgegebene Ergebnismengen anzugeben. In Transact-SQL-Prozeduren werden die Spaltennamen oder Aliasnamen der ersten Abfrage an die aufrufende Umgebung zurückgegeben.

Die folgende Transact-SQL-Prozedur veranschaulicht, wie Ergebnismengen von gespeicherten Transact-SQL-Prozeduren zurückgegeben werden:

```
CREATE PROCEDURE showdept @deptname varchar(30)
AS
    SELECT Employees.Surname, Employees.GivenName
    FROM Departments, Employees
    WHERE Departments.DepartmentName = @deptname
    AND Departments.DepartmentID = Employees.DepartmentID;
```

Nachfolgend wird die entsprechende Watcom-SQL-Prozedur aufgeführt:

```
CREATE PROCEDURE showdept2(in deptname
    varchar(30) )
RESULT ( lastname char(20), firstname char(20))
ON EXCEPTION RESUME
BEGIN
    SELECT Employees.Surname, Employees.GivenName
    FROM Departments, Employees
    WHERE Departments.DepartmentName = deptname
    AND Departments.DepartmentID = Employees.DepartmentID
END;
```

- **Hauptteil einer Prozedur** Der Hauptteil einer Transact-SQL-Prozedur besteht aus einer Liste von Transact-SQL-Anweisungen, denen das Schlüsselwort AS als Präfix vorausgeht. Der Hauptteil einer

Watcom-SQL-Prozedur umfasst eine zusammengesetzte Anweisung, die von den beiden Schlüsselwörtern BEGIN und END umschlossen wird.

Privilegien

Sie müssen das CREATE PROCEDURE-Privileg haben, um Prozeduren erstellen zu können, deren Eigentümer Sie sind. Sie müssen das CREATE ANY PROCEDURE-Privileg oder das CREATE ANY OBJECT-Privileg haben, um Prozeduren erstellen zu können, deren Eigentümer andere Benutzer sind.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE FUNCTION-Anweisung“ auf Seite 633
- „CREATE PROCEDURE-Anweisung“ auf Seite 681

Standards und Kompatibilität

- **SQL/2008** Transact-SQL-Erweiterung.
- **Transact-SQL** SQL Anywhere unterstützt eine Teilmenge der Syntax der CREATE PROCEDURE-Anweisung für Adaptive Server Enterprise.

Nur Transact-SQL-Prozeduren werden im Transact-SQL-Dialekt von SQL Anywhere unterstützt. Um eine externe Prozedur zu erstellen, müssen Sie die Watcom-SQL-Syntax verwenden. Adaptive Server Enterprise unterstützt nicht die NO RESULT SET-Klausel. Wenn die optionale Transact-SQL-Klausel WITH RECOMPILE angegeben ist, wird sie ignoriert. SQL Anywhere kompiliert Prozeduren neu, wenn sie beim Start der Datenbank zum ersten Mal ausgeführt werden, und speichert die kompilierte Prozedur, bis die Datenbank angehalten wird.

In SQL Anywhere werden Gruppen von Transact-SQL-Prozeduren nicht unterstützt.

CREATE PROCEDURE-Anweisung

Erstellt eine benutzerdefinierte SQL-Prozedur in der Datenbank.

Syntax

```
CREATE [ OR REPLACE | TEMPORARY ] PROCEDURE [ owner.]procedure-name
( [ parameter, ... ] )
[ SQL SECURITY { INVOKER | DEFINER } ]
[ RESULT ( result-column, ... ) | NO RESULT SET ]
[ ON EXCEPTION RESUME ]
compound-statement | AT location-string
```

```
parameter :
parameter-mode parameter-name data-type [ DEFAULT expression ]
| SQLCODE
| SQLSTATE
```

```
parameter-mode :
IN
```

| OUT
| INOUT

result-column : column-name data-type

Parameter

Sie können permanent gespeicherte Prozeduren erstellen, die externe oder native, mit unterschiedlichen Programmiersprachen erstellte Prozeduren aufrufen. PROC kann als Synonym für PROCEDURE verwendet werden.

OR REPLACE-Klausel Wenn Sie OR REPLACE angeben, wird eine neue Prozedur erstellt oder eine bestehende Prozedur mit demselben Namen ersetzt. Diese Klausel ändert die Definition der Prozedur, lässt aber vorhandene Privilegien unberührt. Ein Fehler wird zurückgegeben, wenn Sie eine Prozedur ersetzen, die gerade verwendet wird.

TEMPORARY-Klausel Die Angabe von CREATE TEMPORARY PROCEDURE bedeutet, dass die gespeicherte Prozedur lediglich für die Verbindung sichtbar ist, die sie erstellt hat, und dass sie automatisch gelöscht wird, wenn die Verbindung beendet wird. Temporäre gespeicherte Prozeduren können auch explizit gelöscht werden. Sie können kein ALTER, GRANT oder REVOKE für sie ausführen und im Gegensatz zu anderen gespeicherten Prozeduren werden temporäre gespeicherte Prozeduren nicht im Katalog oder Transaktionslog aufgezeichnet.

Temporäre Prozeduren werden mit den Privilegien ihres Erstellers (des aktuellen Benutzers) oder des angegebenen Eigentümers ausgeführt. Sie können unter folgenden Voraussetzungen einen Eigentümer für eine temporäre Prozedur festlegen:

- Wenn die temporäre Prozedur in einer permanent gespeicherten Prozedur erstellt wird
- Wenn der Eigentümer der temporären Prozedur mit dem der permanenten Prozedur identisch ist

Um den Eigentümer einer temporären Prozedur zu löschen, müssen Sie erst die temporäre Prozedur löschen.

Temporäre gespeicherte Prozeduren können erstellt und gelöscht werden, wenn sie mit einer schreibgeschützten Datenbank verbunden sind, und sie dürfen keine externen Prozeduren sein.

Die folgende temporäre Prozedur löscht beispielsweise die fiktive Tabelle "CustRank". In diesem Beispiel nimmt die Prozedur an, dass der Tabellename eindeutig ist und vom Prozedurersteller ohne Angabe eines Tabelleneigentümers referenziert werden kann:

```
CREATE TEMPORARY PROCEDURE drop_table( IN @TableName char(128) )
BEGIN
    IF EXISTS ( SELECT * FROM SYS.SYSTAB WHERE table_name = @TableName )
    THEN
        EXECUTE IMMEDIATE 'DROP TABLE ' || @TableName || ''';
        MESSAGE 'Table ' || @TableName || ' dropped' to client;
    END IF;
END;
CALL drop_table( 'CustRank' );
```

Parameter Parameternamen müssen den Regeln für andere Datenbankbezeichner, wie z.B. Spaltennamen, entsprechen. Es muss sich dabei um einen gültigen SQL-Datentyp handeln.

Parameter können eines der Schlüsselwörter IN, OUT oder INOUT vorangestellt haben. Wenn Sie keinen dieser Werte angeben, sind die Parameter standardmäßig INOUT. Die Schlüsselwörter haben die folgenden Bedeutungen:

- **IN** Dieser Parameter ist ein Ausdruck, der der Prozedur einen Wert zur Verfügung stellt.
- **OUT** Dieser Parameter ist eine Variable, die von der Prozedur einen Wert erhalten kann.
- **INOUT** Dieser Parameter ist eine Variable, die einen Wert für die Prozedur bereitstellt und von der Prozedur einen neuen Wert erhalten kann.

Wenn Prozeduren mit der CALL-Anweisung ausgeführt werden, müssen nicht alle Parameter angegeben werden. Wenn in der CREATE PROCEDURE-Anweisung ein Standardwert bereitgestellt wird, werden fehlenden Parametern die Standardwerte zugeordnet. Falls in der CALL-Anweisung kein Argument angegeben wurde und kein Standardwert gesetzt ist, wird ein Fehler ausgegeben.

SQLSTATE und SQLCODE sind spezielle OUT-Parameter, die den SQLSTATE- oder SQLCODE-Wert ausgeben, wenn die Prozedur beendet wird. Die Spezialwerte SQLSTATE und SQLCODE können sofort geprüft werden, nachdem ein Prozeduraufruf abgeschlossen wurde, um den Rückgabestatus der Prozedur zu testen.

Die Spezialwerte SQLSTATE und SQLCODE werden durch die nächste SQL-Anweisung geändert. Indem SQLSTATE oder SQLCODE als Prozedurargumente bereitgestellt werden, kann die Rückmeldung in einer Variablen gespeichert werden.

Wenn Sie CREATE OR REPLACE PROCEDURE angeben, wird eine neue Prozedur erstellt oder eine bestehende Prozedur mit demselben Namen ersetzt. Diese Klausel ändert die Definition der Prozedur, lässt aber vorhandene Privilegien unberührt. Sie können die OR REPLACE-Klausel nicht mit temporären Prozeduren verwenden. Ein Fehler wird zurückgegeben, wenn die zu ersetzende Prozedur bereits verwendet wird. Geöffnete Cursor für eine Verbindung werden geschlossen, wenn eine CREATE OR REPLACE PROCEDURE-Anweisung ausgeführt wird.

RESULT-Klausel Die RESULT-Klausel deklariert die Anzahl und den Typ der Spalten in der Ergebnismenge. Mit der Liste in Klammern nach dem Schlüsselwort RESULT werden die Namen und Typen der Ergebnisspalten festgelegt. Diese Angaben werden von Embedded SQL DESCRIBE oder von ODBC SQLDescribeCol zurückgegeben, wenn eine CALL-Anweisung beschrieben wird.

Einige Prozeduren können mehr als eine Ergebnismenge mit unterschiedlicher Spaltenanzahl zurückgeben, je nachdem, wie sie ausgeführt werden. Die folgende Prozedur gibt beispielsweise unter bestimmten Bedingungen zwei Spalten und in anderen Fällen nur eine Spalte zurück.

```
CREATE PROCEDURE names( IN formal char(1))
BEGIN
    IF formal = 'n' THEN
        SELECT GivenName
        FROM GROUPO.Employees
    ELSE
        SELECT Surname, GivenName
        FROM GROUPO.Employees
    END IF
END;
```

Prozeduren mit variablen Ergebnismengen müssen ohne RESULT-Klausel oder in Transact-SQL geschrieben werden. Ihre Verwendung ist den folgenden Einschränkungen unterworfen:

- **Embedded SQL** Sie müssen den Prozeduraufruf mithilfe einer DESCRIBE-Anweisung beschreiben, nachdem der Cursor für die Ergebnismenge geöffnet wurde, aber bevor Zeilen zurückgegeben werden, damit die richtige Form der Ergebnismenge bezogen wird. Die Klausel `CURSOR cursor-name` in der DESCRIBE-Anweisung ist erforderlich.
- **ODBC, OLE DB, ADO.NET** Prozeduren mit variablen Ergebnismengen können von Anwendungen verwendet werden, die diese Schnittstellen benutzen. Die richtige Beschreibung der Ergebnismengen wird vom Treiber oder Provider vorgenommen.
- **Open Client-Anwendungen** Prozeduren mit variablen Ergebnismengen können von Open Client-Anwendungen verwendet werden.
- **Webdienste** Webdienste verlassen sich auf die RESULTS-Klausel der gespeicherten Prozedur, um die Anzahl und die Typen der Spalte in der Ergebnismenge zu ermitteln. Webdienste unterstützen keine Prozeduren, die mehrere Ergebnismengen zurückgeben, und auch keine variablen Ergebnismengen durch die Verwendung von EXECUTE IMMEDIATE.

Hinweis

Wenn eine EXECUTE IMMEDIATE-Anweisung, die eine WITH RESULT SET ON-Klausel enthält, in der Prozedur verwendet wird und die Ergebnismenge, die von der Anweisung zurückgegeben wird, dieselbe ist wie die Ergebnismenge, die von der Prozedur zurückgegeben wird, wird nur die erste Spalte der Ergebnismenge der EXECUTE IMMEDIATE-Anweisung zurückgegeben.

Wenn Ihre Prozedur nur eine Ergebnismenge zurückgibt, sollten Sie eine RESULT-Klausel verwenden. Das Vorhandensein dieser Klausel verhindert, dass ODBC- und Open Client-Anwendungen die Ergebnismenge noch einmal beschreiben, nachdem der Cursor geöffnet wurde.

Um mehrere Ergebnismengen verarbeiten zu können, muss ODBC den aktuell ausgeführten Cursor beschreiben, nicht die definierte Ergebnismenge der Prozedur. Deshalb beschreibt ODBC die Spaltennamen nicht immer so, wie sie in der RESULT-Klausel der gespeicherten Prozedur definiert sind. Um dieses Problem zu vermeiden, verwenden Sie Spaltenaliasnamen in der SELECT-Anweisung, die die Ergebnismenge erzeugt.

NO RESULT SET-Klausel Deklariert, dass von dieser Prozedur keine Ergebnismenge zurückgegeben wird. Das ist nützlich, wenn eine externe Umgebung wissen muss, dass eine Prozedur keine Ergebnismenge zurückgibt.

SQL SECURITY-Klausel Die SQL SECURITY-Klausel legt fest, ob die Prozedur als INVOKER (der Benutzer, der die Prozedur aufruft) oder als DEFINER (der Eigentümer der Prozedur) aufgerufen wird. Standardwert ist DEFINER.

Wenn SQL SECURITY INVOKER angegeben ist, wird mehr Speicher verwendet, weil für jeden Benutzer, der die Prozedur aufruft, ein Vermerk erfolgen muss. Wenn SQL SECURITY INVOKER angegeben wird, erfolgt ebenfalls die Namensauflösung als Aufrufer. Achten Sie daher darauf, alle Objektnamen (Tabellen, Prozeduren, usw.) mit dem jeweiligen Eigentümer zu qualifizieren. Beispiel: Der Benutzer user1 erstellt die folgende Prozedur:

```
CREATE PROCEDURE user1.myProcedure()
    RESULT( columnA INT )
    SQL SECURITY INVOKER
BEGIN
    SELECT columnA FROM table1;
END;
```

Wenn der Benutzer user2 versucht, diese Prozedur auszuführen und eine Tabelle user2.table1 *nicht existiert*, kommt es zu einem Tabellensuchfehler. Wenn eine Tabelle user2.table1 *existiert*, wird diese Tabelle anstelle der beabsichtigten user1.table1 verwendet. Um dies zu verhindern, qualifizieren Sie die Tabellenreferenz in der Anweisung (user1.table1 anstelle von table1).

ON EXCEPTION RESUME-Klausel Diese Klausel aktiviert eine Transact-SQL-ähnliche Fehlerbehandlung, die innerhalb einer Prozedur mit Watcom-SQL-Syntax verwendet wird.

Wenn Sie ON EXCEPTION RESUME verwenden, übernimmt die Prozedur eine Aktion, die von der Einstellung der Option on_tsq_error abhängt. Wenn on_tsq_error auf Conditional (Standardwert) gesetzt ist, wird die Ausführung fortgesetzt, falls die nächste Anweisung den Fehler behebt. Andernfalls wird die Ausführung beendet.

Anweisungen für die Fehlerbehandlung enthalten folgende Elemente:

- IF
- SELECT @variable =
- CASE
- LOOP
- LEAVE
- CONTINUE
- CALL
- EXECUTE
- SIGNAL
- RESIGNAL
- DECLARE
- SET VARIABLE

Einen expliziten Fehlerbehandlungscode sollten Sie nicht mit der Klausel ON EXCEPTION RESUME verwenden.

Diese Klausel wird innerhalb des TRY-Blocks einer BEGIN...END-Anweisung ignoriert.

compound-statement Eine Gruppe von SQL-Anweisungen in BEGIN und END, getrennt durch Semikola.

AT-Klausel Erstellt eine gespeicherte Proxy-Prozedur in der aktuellen Datenbank für eine entfernte Prozedur, die durch *location-string* angegeben ist. Die AT-Klausel unterstützt das Semikolon (;) als Feldbegrenzer in *location-string*. Wenn kein Semikolon vorhanden ist, gilt der Punkt als Feldbegrenzer. Damit können Dateinamen oder Erweiterungen in den Feldern für Datenbank und Eigentümer verwendet werden.

Die Zeichenfolge in der AT-Klausel kann auch lokale oder globale Variablenamen in geschweiften Klammern enthalten ({*variable-name*}). Der SQL-Variablenname muss vom Typ CHAR, VARCHAR

oder LONG VARCHAR sein. Eine AT-Klausel mit 'bostonase.master.dbo.{@myprocedure}' zeigt beispielsweise an, dass *@myprocedure* eine SQL-Variable ist und dass der aktuelle Inhalt der *@myprocedure*-Variablen beim Anwenden der entfernten Prozedur ersetzt werden muss.

Wenn die entfernte Prozedur eine Ergebnismenge zurückgeben kann, selbst wenn sie dies nicht in allen Fällen tut, muss die lokale Prozedurdefinition eine RESULT-Klausel enthalten.

Bemerkungen

Die CREATE PROCEDURE-Anweisung erstellt eine Prozedur in der Datenbank. Eine Prozedur wird mit einer CALL-Anweisung aufgerufen.

Wenn eine gespeicherte Prozedur eine Ergebnismenge zurückgibt, kann sie nicht Ausgabeparameter setzen oder einen Rückgabewert zurückgeben.

Bei der Referenzierung einer temporären Tabelle durch mehrere Prozeduren kann ein Problem entstehen, wenn die Definitionen der temporären Tabelle nicht konsistent sind und Anweisungen, die die Tabelle referenzieren, im Cache abgelegt sind.

Privilegien

Sie müssen das CREATE PROCEDURE-Systemprivileg haben, um Prozeduren erstellen zu können, deren Eigentümer Sie sind. Sie müssen das CREATE ANY PROCEDURE-Privileg oder das CREATE ANY OBJECT-Privileg haben, um Prozeduren erstellen zu können, deren Eigentümer andere Benutzer sind. Wenn Sie externe Prozeduren erstellen möchten, müssen Sie außerdem das CREATE EXTERNAL REFERENCE-Systemprivileg haben.

Sie benötigen keine Privilegien, um temporäre Prozeduren zu erstellen.

Nebenwirkungen

Automatisches Festschreiben, auch für temporäre Prozeduren.

Siehe auch

- „ALTER PROCEDURE-Anweisung“ auf Seite 483
- „BEGIN-Anweisung“ auf Seite 557
- „CALL-Anweisung“ auf Seite 564
- „CREATE FUNCTION-Anweisung“ auf Seite 633
- „CREATE PROCEDURE-Anweisung [externer Aufruf]“ auf Seite 661
- „CREATE PROCEDURE-Anweisung [Webdienste]“ auf Seite 670
- „CREATE PROCEDURE-Anweisung [T-SQL]“ auf Seite 679
- „CREATE SERVER-Anweisung“ auf Seite 701
- „DROP PROCEDURE-Anweisung“ auf Seite 816
- „EXECUTE IMMEDIATE-Anweisung [SP]“ auf Seite 843
- „GRANT-Anweisung“ auf Seite 881
- „EXECUTE IMMEDIATE in Prozeduren, Triggern, benutzerdefinierten Funktionen und Batches“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „SQL-Datentypen“ auf Seite 95
- „on_tsql_error-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Ergebnismengen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Referenzen auf temporäre Tabellen innerhalb von Prozeduren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Erstellen von entfernten Prozeduren (Sybase Central)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Entfernte Prozeduraufrufe“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** CREATE PROCEDURE ist eine Kernfunktion des SQL/2008-Standards, aber einige der in SQL Anywhere unterstützten Komponenten sind optionale SQL-Sprachenfunktionen. Zu diesen Funktionen gehören:
 - Die SQL SECURITY-Klausel ist die optionale SQL/2008-Sprachenfunktion T324.
 - Die Möglichkeit zum Übergeben eines LONG VARCHAR-, LONG NVARCHAR- oder LONG BINARY-Werts an eine SQL-Prozedur ist SQL/2008-Sprachenfunktion T041.
 - Die Möglichkeit, ein Schema-Objekt innerhalb einer SQL-Prozedur mit Anweisungen wie CREATE TABLE und DROP TRIGGER zu erstellen oder zu ändern, ist SQL/2008-Sprachenfunktion T653.
 - Die Möglichkeit zum Verwenden einer Dynamic-SQL-Anweisung innerhalb einer SQL-Prozedur, einschließlich der Anweisungen CONNECT, EXECUTE IMMEDIATE, PREPARE und DESCRIBE, ist SQL/2008-Sprachenfunktion T652.

Einige Klauseln der CREATE PROCEDURE-Anweisung sind Erweiterungen des Herstellers. Es handelt sich dabei um die folgenden:

- Die TEMPORARY-Klausel.

- Die ON EXCEPTION RESUME-Klausel.
- Die AT-Klausel.
- Die optionale DEFAULT-Klausel für einen bestimmten Routinenparameter.
- Die Klauseln RESULT und NO RESULT SET. Der SQL/2008-Standard verwendet das RETURNS-Schlüsselwort.
- Die optionale OR REPLACE-Klausel.
- **Transact-SQL** CREATE PROCEDURE wird von Adaptive Server Enterprise unterstützt.

Beispiele

Die folgende Prozedur fragt die Tabelle Employees ab und gibt Gehälter zurück, die innerhalb des angegebenen Prozentsatzes (*percentage*) von einem angegebenen Gehalt (*sal*) liegen:

```
CREATE OR REPLACE PROCEDURE AverageEmployees( IN percentage NUMERIC( 5,3),
IN sal NUMERIC( 20, 3 ) )
RESULT( Department CHAR(40), GivenName person_name_t, Surname person_name_t,
Salary NUMERIC( 20, 3 ) )
BEGIN
    DECLARE maxS NUMERIC( 20, 3 );
    DECLARE minS NUMERIC( 20, 3 );

    IF percentage >= 1 THEN
        SET percentage = percentage / 100;
    ELSEIF percentage < 0 THEN
        SELECT 'Percentage error', 'Err', 'Err', -1;
        RETURN;
    END IF;

    SELECT MIN( E.Salary ), MAX( E.Salary ) INTO minS, maxS
    FROM GROUPO.Employees E;

    IF sal < minS OR sal > maxS THEN
        SELECT 'Salary out of bounds', 'Err', 'Err', -2;
        RETURN;
    END IF;

    SELECT D.DepartmentName, E.GivenName, E.Surname, E.Salary
    FROM GROUPO.Employees E JOIN Departments D ON E.DepartmentID =
D.DepartmentID
    WHERE E.Salary BETWEEN sal *( 1 - percentage ) AND sal * ( 1 +
percentage );
END;
```

Die folgende Prozedur verwendet eine CASE-Anweisung, um die Ergebnisse einer Abfrage zu klassifizieren.

```
CREATE PROCEDURE ProductType (IN product_ID INT, OUT type CHAR(10))
BEGIN
    DECLARE prod_name CHAR(20);
    SELECT name INTO prod_name FROM GROUPO.Products
    WHERE ID = product_ID;
    CASE prod_name
    WHEN 'Tee Shirt' THEN
        SET type = 'Shirt'
    WHEN 'Sweatshirt' THEN
```

```

        SET type = 'Shirt'
    WHEN 'Baseball Cap' THEN
        SET type = 'Hat'
    WHEN 'Visor' THEN
        SET type = 'Hat'
    WHEN 'Shorts' THEN
        SET type = 'Shorts'
    ELSE
        SET type = 'UNKNOWN'
    END CASE;
END;

```

Im folgenden Beispiel wird die Prozedur ProductType ersetzt, die im vorherigen Beispiel erstellt wurde. Nach dem Ersetzen der Prozedur werden die Parameter für Tee Shirt und Sweatshirt aktualisiert:

```

CREATE OR REPLACE PROCEDURE ProductType (IN product_ID INT, OUT type
CHAR(10))
BEGIN
    DECLARE prod_name CHAR(20);
    SELECT name INTO prod_name FROM GROUPO.Products
    WHERE ID = product_ID;
    CASE prod_name
    WHEN 'Tee Shirt' THEN
        SET type = 'T Shirt'
    WHEN 'Sweatshirt' THEN
        SET type = 'Long Sleeve Shirt'
    WHEN 'Baseball Cap' THEN
        SET type = 'Hat'
    WHEN 'Visor' THEN
        SET type = 'Hat'
    WHEN 'Shorts' THEN
        SET type = 'Shorts'
    ELSE
        SET type = 'UNKNOWN'
    END CASE;
END;

```

Die folgende Prozedur verwendet einen Cursor und eine Schleife über die Zeilen des Cursors, um einen Einzelwert zurückzugeben.

```

CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT TopValue INT)
BEGIN
    DECLARE err_notfound EXCEPTION
    FOR SQLSTATE '02000';
    DECLARE curThisCust CURSOR FOR
        SELECT CompanyName,
            CAST(SUM(SalesOrderItems.Quantity *
                Products.UnitPrice) AS INTEGER) VALUE
        FROM GROUPO.Customers
        LEFT OUTER JOIN SalesOrders
        LEFT OUTER JOIN SalesOrderItems
        LEFT OUTER JOIN Products
        GROUP BY CompanyName;
    DECLARE ThisValue INT;
    DECLARE ThisCompany CHAR(35);
    SET TopValue = 0;
    OPEN curThisCust;
    CustomerLoop:
    LOOP
        FETCH NEXT curThisCust
        INTO ThisCompany, ThisValue;
        IF SQLSTATE = err_notfound THEN

```

```
        LEAVE CustomerLoop;
    END IF;
    IF ThisValue > TopValue THEN
        SET TopValue = ThisValue;
        SET TopCompany = ThisCompany;
    END IF;
END LOOP CustomerLoop;
CLOSE curThisCust;
END;
```

Im folgenden Beispiel wird die NewDepartment-Prozedur erstellt, die einen INSERT-Vorgang in der Tabelle "Departments" der SQL Anywhere-Beispieldatenbank ausführt und so eine neue Abteilung erstellt.

```
CREATE PROCEDURE NewDepartment(
    IN id INT,
    IN name CHAR(35),
    IN head_id INT )
BEGIN
    INSERT
    INTO GROUPO.Departments ( DepartmentID,
        DepartmentName, DepartmentHeadID )
    VALUES ( id, name, head_id );
END;
```

Der Hauptteil einer Prozedur besteht aus einer zusammengesetzten Anweisung. Die zusammengesetzte Anweisung startet mit BEGIN und wird mit END abgeschlossen. Im Fall von NewDepartment ist die zusammengesetzte Anweisung eine einzelne INSERT-Anweisung, die zwischen die Anweisungen BEGIN und END gesetzt ist.

Parameter für Prozeduren können als IN, OUT oder INOUT markiert werden. Standardmäßig sind Parameter INOUT-Parameter. Alle Parameter der NewDepartment-Prozedur sind IN-Parameter, da sie nicht von der Prozedur geändert werden. Sie sollten Parameter auf IN setzen, wenn sie nicht benutzt werden, um Werte an den Aufrufer zu liefern.

CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote]

Erstellt eine Publikation In MobiLink kennzeichnet eine Publikation synchronisierte Daten in einer entfernten SQL Anywhere-Datenbank. In SQL Remote kennzeichnen Publikationen replizierte Daten sowohl in konsolidierten als auch in entfernten Datenbanken.

Syntax 1 (MobiLink, allgemeiner Gebrauch)

```
CREATE PUBLICATION [ IF NOT EXISTS ] [ owner. ] publication-name
( article-definition, ... )
```

article-definition :

```
TABLE table-name [ ( column-name, ... ) ]
[ WHERE search-condition ]
```

Syntax 2 (MobiLink, skriptgesteuerter Upload)

```
CREATE PUBLICATION [ IF NOT EXISTS ] [ owner. ] publication-name
WITH SCRIPTED UPLOAD
( article-definition, ... )
```

article-definition :

```
TABLE table-name [ ( column-name, ... ) ]
[ USING ( [ PROCEDURE ] [ owner. ] procedure-name
FOR UPLOAD { INSERT | DELETE | UPDATE }, ... ) ]
```

Syntax 3 (MobiLink, reine Downloadpublikationen)

```
CREATE PUBLICATION [ IF NOT EXISTS ] [ owner. ] publication-name
FOR DOWNLOAD ONLY
( article-definition, ... )
```

article-definition : **TABLE** *table-name* [(*column-name*, ...)]

Syntax 4 (SQL Remote)

```
CREATE PUBLICATION [ IF NOT EXISTS ] [ owner. ] publication-name
( article-definition, ... )
```

article-definition :

```
TABLE table-name [ ( column-name, ... ) ]
[ WHERE search-condition ]
[ SUBSCRIBE BY expression ]
```

Parameter

IF NOT EXISTS-Klausel Wenn die IF NOT EXISTS-Klausel angegeben wurde und die benannte Publikation bereits vorhanden ist, werden keine Änderungen vorgenommen und es wird kein Fehler zurückgegeben.

article-definition Publikationen werden aus Artikeln erstellt. Jeder Artikel kennzeichnet die Zeilen und Spalten einer einzelnen Tabelle, die in der Publikation enthalten sind. Eine Publikation kann nicht zwei Artikel enthalten, die sich auf dieselbe Tabelle beziehen.

Wenn eine Liste der Spaltennamen in den Artikel aufgenommen wird, sind nur die Spalten in der Publikation enthalten. Wenn keine Spaltennamen aufgelistet sind, werden alle Spalten in der Tabelle in die Publikation aufgenommen. Bei der MobiLink Synchronisation müssen, wenn die Spaltennamen aufgelistet werden, alle Spalten im Primärschlüssel der Tabelle in der Liste enthalten sein.

In Syntax 2, die für Publikationen verwendet wird, welche skriptgesteuerte Uploads durchführen, registriert die Artikeldefinition auch die Skripts, die für die Definition des Uploads verwendet werden.

In Syntax 3, die für reine Downloadpublikationen verwendet wird, gibt der Artikel nur die Tabellen und Spalten für den Download an.

WHERE-Klausel Mit der WHERE-Klausel können Sie die Teilmenge von Zeilen einer Tabelle definieren, die in einen Artikel aufgenommen werden soll.

In MobiLink-Anwendungen bezieht sich die WHERE-Klausel auf die im Upload enthaltenen Zeilen. (Der Download wird durch das download_cursor-Skript definiert.) In entfernten MobiLink SQL Anywhere-

Datenbanken kann die WHERE-Klausel nur im Artikel enthaltene Spalten referenzieren und darf keine Unterabfragen, Variablen oder nicht-deterministische Funktionen enthalten.

SUBSCRIBE BY-Klausel In SQL Remote kann mit dieser Klausel eine Teilmenge von Zeilen einer Tabelle definiert werden, die in einen Artikel aufgenommen werden soll. Diese Klausel ermöglicht es vielen verschiedenen Subskribenten, unterschiedliche Zeilen einer Tabelle in einer einzigen Publikationsdefinition zu erhalten.

Bemerkungen

Die CREATE PUBLICATION-Anweisung erstellt eine Publikation in der Datenbank. Eine Publikation kann für einen anderen Benutzer erstellt werden, indem ein Eigentümername angegeben wird.

In MobiLink werden Publikationen in entfernten SQL Anywhere-Datenbanken benötigt, in UltraLite-Datenbanken sind sie optional. Diese Publikationen und die entsprechenden Subskriptionen bestimmen, welche Daten an den MobiLink-Server übertragen werden.

Die Optionen für eine MobiLink-Publikation können Sie mit der ADD OPTION-Klausel in der CREATE SYNCHRONIZATION SUBSCRIPTION- oder ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung einstellen.

Syntax 2 erstellt eine Publikation für skriptgesteuerte Uploads. Verwenden Sie die USING-Klausel, um die gespeicherten Prozeduren anzugeben, die Sie zur Definition des Uploads verwenden wollen. Bei jeder Tabelle können Sie bis zu drei gespeicherte Prozeduren verwenden: jeweils eine für Einfügungen, Löschungen und Aktualisierungen.

Syntax 3 erstellt eine reine Downloadpublikation, die ohne Transaktionslogdatei synchronisiert werden kann. Wenn reine Downloadpublikationen synchronisiert werden, können Downloadzeilen Änderungen überschreiben, die an diesen Zeilen in der entfernten Datenbank durchgeführt wurden.

In SQL Remote ist das Publizieren ein Vorgang mit zwei Möglichkeiten, da Daten sowohl in konsolidierten als auch in entfernten Datenbanken eingegeben werden können. In einer SQL Remote-Installation müssen die konsolidierte Datenbank und alle entfernten Datenbanken dieselbe Publikation definiert haben. Wenn das SQL Remote-Extraktionsdienstprogramm aus einer konsolidierten Datenbank ausgeführt wird, führt das System automatisch die richtige CREATE PUBLICATION-Anweisung in der entfernten Datenbank aus.

Für jede Syntax benötigen Sie exklusiven Zugriff auf alle in der Anweisung referenzierten Tabellen, um die Anweisung ausführen zu können.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „ALTER PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ auf Seite 485
- „DROP PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ auf Seite 817
- „CREATE SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink]“ auf Seite 733
- „ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink]“ auf Seite 512
- SQL Anywhere MobiLink-Clients: „Publikationen“ [*MobiLink - Clientadministration*]
- UltraLite MobiLink-Clients: „CREATE PUBLICATION-Anweisung [UltraLite]“ [*UltraLite - Datenbankverwaltung*]
- SQL Remote: „Publikationen und Artikel“ [*SQL Remote*]
- „Skriptgesteuerter Upload“ [*MobiLink - Clientadministration*]
- „Reine Download-Publikationen“ [*MobiLink - Clientadministration*]
- „SYSSYNC-Systemansicht“ auf Seite 1489
- „Publikationen für skriptgesteuerte Uploads“ [*MobiLink - Clientadministration*]
- „Erstellen von Publikationen“ [*SQL Remote*]
- „Nur einige Spalten einer Tabelle publizieren“ [*SQL Remote*]
- „Publizieren von nur einigen Zeilen mit der SUBSCRIBE BY-Klausel“ [*SQL Remote*]
- „Nur einige Zeilen mittels einer WHERE-Klausel publizieren“ [*SQL Remote*]
- „Nur einige Zeilen einer Tabelle publizieren“ [*SQL Remote*]
- „Primärschlüsselpool replizieren“ [*SQL Remote*]
- „Partitionen mit Überlappung“ [*SQL Remote*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Mit der folgenden Anweisung werden alle Spalten und Zeilen von zwei Tabellen publiziert.

```
CREATE PUBLICATION pub_contact (
    TABLE GROUPO.Contacts,
    TABLE GROUPO.Customers
);
```

Mit der folgenden Anweisung werden nur einige Spalten einer Tabelle publiziert.

```
CREATE PUBLICATION pub_customer (
    TABLE GROUPO.Customers ( ID, CompanyName, City )
);
```

Die folgende Anweisung publiziert nur die Zeilen für Kunden in New York (NY), indem eine WHERE-Klausel einbezogen wird, die den Wert in der State-Spalte der Tabelle "Customers" prüft.

```
CREATE PUBLICATION pub_customer (
    TABLE GROUPO.Customers ( ID, CompanyName, City, State, Status )
    WHERE State = 'NY'
);
```

Mit der folgenden Anweisung werden nur einige Zeilen publiziert, und zwar durch Angabe eines SUBSCRIBE BY-Werts. Diese Methode kann nur mit SQL Remote benutzt werden.

```
CREATE PUBLICATION pub_customer (
    TABLE GROUPO.Customers ( ID, CompanyName, City, State )
    SUBSCRIBE BY State
);
```

Beim Erstellen einer SQL Remote-Subskription wird der SUBSCRIBE BY-Wert wie folgt verwendet.

```
CREATE SUBSCRIPTION TO pub_customer ( 'NY' )  
FOR jsmith;
```

Das folgende Beispiel erstellt eine MobiLink-Publikation, die skriptgesteuerte Uploads verwendet:

```
CREATE PUBLICATION pub WITH SCRIPTED UPLOAD (  
  GROUPO.TABLE t1 (a, b, c) USING (  
    PROCEDURE my.t1_ui FOR UPLOAD INSERT,  
    PROCEDURE my.t1_ud FOR UPLOAD DELETE,  
    PROCEDURE my.t1_uu FOR UPLOAD UPDATE  
  ),  
  GROUPO.TABLE t2 AS my_t2 USING (  
    PROCEDURE my.t2_ui FOR UPLOAD INSERT  
  )  
);
```

Das folgende Beispiel erstellt eine reine Downloadpublikation:

```
CREATE PUBLICATION p1 FOR DOWNLOAD ONLY (  
  GROUPO.TABLE t1  
);
```

CREATE REMOTE [MESSAGE] TYPE-Anweisung [SQL Remote]

Bestimmt eine Nachrichtenverbindung und eine Rücksendeadresse für ausgehende Nachrichten aus einer Datenbank.

Syntax

```
CREATE REMOTE [MESSAGE] TYPE message-system  
[ ADDRESS address-string ]
```

message-system :

```
FILE  
| FTP  
| HTTP  
| SMTP
```

Parameter

message-system Eines der Nachrichtensysteme, die von SQL Remote unterstützt werden. Hierfür muss einer der folgenden Werte verwendet werden: **FILE**, **FTP** oder **SMTP**.

address-string Eine Zeichenfolge, die eine gültige Adresse für das angegebene Nachrichtensystem enthält.

Bemerkungen

Der Nachrichtenagent versendet ausgehende Nachrichten von einer Datenbank über eine der unterstützten Nachrichtenverbindungen. Rückmeldungen für Benutzer, die die angegebene Verbindung benutzen, werden an die angegebene Adresse gesendet, wenn die entfernte Datenbank vom Extraktionsdienstprogramm erstellt wird. Der Nachrichtenagent startet Verbindungen nur, wenn es entfernte Benutzer für diese Verbindungen gibt.

Die Adresse ist die des Publikationseigentümers unter dem angegebenen Nachrichtensystem. Wenn es sich um ein E-Mail-System handelt, muss die Adressenzeichenfolge eine gültige E-Mail-Adresse sein. Wenn es sich um ein System mit gemeinsamer Dateinutzung handelt, ist die Adressenzeichenfolge ein Unterverzeichnis jenes Verzeichnisses, das durch die SQLREMOTE-Umgebungsvariable bestimmt ist, oder des aktuellen Verzeichnisses, wenn die Variable nicht entsprechend festgelegt ist. Sie können diese Einstellung für die GRANT CONSOLIDATE-Anweisung in der entfernten Datenbank aufheben.

Um die Adresse zu entfernen, führen Sie die Anweisung CREATE REMOTE MESSAGE TYPE ohne ADDRESS-Klausel aus.

Das Dienstprogramm dbinit erstellt Nachrichtentypen automatisch, ohne Adresse. Im Gegensatz zu anderen CREATE-Anweisungen gibt die CREATE REMOTE MESSAGE TYPE-Anweisung keinen Fehler aus, wenn der Typ bereits vorhanden ist, sondern sie ändert den Typ.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „GRANT PUBLISH-Anweisung [SQL Remote]“ auf Seite 894
- „GRANT REMOTE-Anweisung [SQL Remote]“ auf Seite 900
- „GRANT CONSOLIDATE-Anweisung [SQL Remote]“ auf Seite 889
- „DROP REMOTE MESSAGE TYPE-Anweisung [SQL Remote]“ auf Seite 818
- „ALTER REMOTE MESSAGE TYPE-Anweisung [SQL Remote]“ auf Seite 487
- „SQL Remote-Nachrichtensysteme“ [SQL Remote]
- „Einen Nachrichtentyp erstellen“ [SQL Remote]
- „Das FILE-Nachrichtensystem“ [SQL Remote]
- „Das FTP-Nachrichtensystem“ [SQL Remote]
- „Das SMTP-Nachrichtensystem“ [SQL Remote]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Wenn entfernte Datenbanken mit dem Extraktionsdienstprogramm extrahiert werden, legt die folgende Anweisung fest, dass alle Empfänger von Nachrichten aus dem FILE-Nachrichtensystem Meldungen zurück an das Unterverzeichnis *company* senden.

Außerdem wird dbremote angewiesen, im Unterverzeichnis *company* nach eintreffenden Nachrichten zu suchen.

```
CREATE REMOTE MESSAGE TYPE file  
ADDRESS 'company';
```

CREATE ROLE-Anweisung

Erstellt oder ersetzt eine Rolle, erstellt eine benutzererweiterte Rolle oder ändert Administratoren für eine Rolle.

Syntax

```
CREATE [ OR REPLACE ] ROLE { role-name | FOR USER userid }  
[ WITH ADMIN [ ONLY ] administrator-userid [...]]
```

Parameter

- **role-name** Verwenden Sie diese Variable, um der neuen Rolle einen Namen zuzuordnen. Dieser Name muss unter allen Benutzern und Rollen in der Datenbank eindeutig sein.
- **OR REPLACE-Klausel** Verwenden Sie diese Klausel, um die Rolle zu erstellen, wenn sie noch nicht vorhanden ist, bzw. um ihre Administratoren zu ändern, wenn sie bereits vorhanden ist.
- **FOR USER *userid*-Klausel** Verwenden Sie diese Klausel, um den angegebenen Benutzer in eine benutzererweiterte Rolle zu konvertieren, die anderen zugeordnet werden kann. Der Benutzer darf nicht bereits zu einer anderen Rolle erweitert sein.
- **Klausel WITH ADMIN und WITH ADMIN ONLY *administrator-userid*** Geben Sie optional Administratoren für die Rolle an. WITH ADMIN bedeutet, dass *administrator-userid* Ausübungs- und Administrationsrechte für die Rolle erhält. WITH ADMIN ONLY bedeutet, dass *administrator-userid* nur Administrationsrechte für die Rolle erhält. Wenn keine Klausel angegeben wird, kann jeder Benutzer mit MANAGE ROLES-Systemprivileg die Rolle verwalten.

Die `min_role_admins`-Datenbankoption steuert die Mindestanzahl von Administratoren, die für die einzelnen Rollen erforderlich ist. Wenn Sie beim Erstellen der Rolle nicht genug Administratoren angeben, gibt die Anweisung einen Fehler zurück.

Bemerkungen

Der Name der neuen Rolle darf nicht sowohl mit "SYS_" anfangen als auch mit "_ROLE" enden. SYS_MyBackup_ROLE ist beispielsweise kein zulässiger Name für eine benutzerdefinierte Rolle, während MyBackup_ROLE und SYS_MyBackup zulässig sind.

Wenn eine ADMIN-Klausel angegeben wird, können nur die festgelegten Benutzer die Rollen verwalten. Wenn keine ADMIN-Klausel angegeben wird, wird die Rolle standardmäßig dem MANAGE ROLES-Systemprivileg nur mit Administrationsrechten erteilt. Das bedeutet, dass globale Administratoren die Rolle verwalten können.

Wenn Sie eine benutzererweiterte Rolle erstellen möchten (d.h., einen Benutzer zu einer Rolle erweitern), verwenden Sie die Syntax `CREATE ROLE FOR USER userid`.

Verwenden Sie die GRANT-Anweisungen, um der Rolle Systemprivilegien zu erteilen.

Privilegien

Sie müssen das MANAGE ROLES-Systemprivileg haben, um eine neue Rolle erstellen zu können.

Wenn die OR REPLACE-Klausel angegeben wird und die Rolle bereits vorhanden ist, müssen Sie außerdem Administrationsrechte für die Rolle haben.

Nebenwirkungen

Keine.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung erstellt die Rolle Sales. Jeder Benutzer mit MANAGE ROLES-Systemprivileg kann die Rolle verwalten.

```
CREATE ROLE Sales;
```

Die folgende Anweisung erweitert den Benutzer JaneSmith zu einer Rolle, die anderen zugeordnet werden kann.

```
CREATE ROLE FOR USER JaneSmith;
```

Die folgende Anweisung erstellt die Rolle Finance und erteilt MaryJones und JeffTurkott als Rollenadministratoren (nur) Administrationsrechte für die Rolle.

```
CREATE ROLE Finance
WITH ADMIN ONLY MaryJones, JeffTurkott;
```

Das folgende Beispiel ersetzt die vorhandene Rolle Finance, die im vorherigen Beispiel erstellt wurde. Dabei werden MaryJones und JeffTurkott als Rollenadministratoren durch EllenChong und DaveLexx ersetzt, dieses Mal mit Ausübungsrechten für die Rolle.

```
CREATE OR REPLACE ROLE Finance
WITH ADMIN EllenChong, DaveLexx;
```

Siehe auch

- „Rollenadministratoren“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „min_role_admins-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „DROP ROLE-Anweisung“ auf Seite 820
- „Rollen erteilen (Sybase Central)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Benutzererweiterte Rollen“ [[SQL Anywhere Server - Datenbankadministration](#)]

CREATE SCHEMA-Anweisung

Erstellt eine Sammlung von Tabellen und Ansichten für einen Datenbankbenutzer.

Syntax

```
CREATE SCHEMA
AUTHORIZATION userid
[ create-table-statement
| create-view-statement
```

```
| grant-statement  
] ... ;
```

Bemerkungen

Die CREATE SCHEMA-Anweisung erstellt ein Schema. Ein Schema ist eine Sammlung von Tabellen und Ansichten sowie den diesen zugeordneten Privilegien.

Die *userid* muss die Benutzer-ID der aktuellen Verbindung sein. Sie können kein Schema für einen anderen Benutzer erstellen.

Wenn eine Anweisung in der CREATE SCHEMA-Anweisung fehlschlägt, wird die gesamte CREATE SCHEMA-Anweisung zurückgesetzt.

Die CREATE SCHEMA-Anweisung ist eine Möglichkeit, einzelne CREATE- und GRANT-Anweisungen in einem Vorgang zusammenzufassen. In der Datenbank gibt es kein erstelltes SCHEMA-Datenbankobjekt und Sie müssen einzelne DROP TABLE- oder DROP VIEW-Anweisungen verwenden, wenn Sie die Objekte löschen möchten. Wenn Sie Privilegien entziehen möchten, müssen Sie eine REVOKE-Anweisung für jedes erteilte Privileg verwenden.

Einzelne CREATE- oder GRANT-Anweisungen werden nicht durch Trennzeichen für Anweisungen getrennt. Das Trennzeichen für Anweisungen markiert das Ende der CREATE SCHEMA-Anweisung selbst.

Die einzelnen CREATE- oder GRANT-Anweisungen müssen so geordnet werden, dass die Objekte zuerst erstellt und dann die entsprechenden Privilegien erteilt werden.

Sie können zwar mehr als ein Schema pro Benutzer erstellen, aber dies wird nicht empfohlen.

Privilegien

Welche Systemprivilegien erforderlich sind, hängt von dem Vorgang ab, der in der zu definierenden CREATE SCHEMA-Anweisung angegeben ist. Weitere Hinweise zu den erforderlichen Privilegien finden Sie in den Abschnitten über die Systemprivilegien für die betreffenden Anweisungen (CREATE TABLE, CREATE VIEW und GRANT).

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE TABLE-Anweisung“ auf Seite 737
- „CREATE VIEW-Anweisung“ auf Seite 773
- „GRANT-Anweisung“ auf Seite 881

Standards und Kompatibilität

- **SQL/2008** CREATE SCHEMA ist eine Kernfunktion des SQL/2008-Standards. Die Möglichkeit zum Erstellen mehrerer Schemata für einen einzelnen Benutzer ist die optionale SQL/2008-Sprachenfunktion F171. SQL Anywhere unterstützt nicht die Verwendung von REVOKE-Anweisungen innerhalb der CREATE SCHEMA-Anweisung und lässt auch nicht deren Verwendung in Transact-SQL-Batchdateien oder -Prozeduren zu.

- **Transact-SQL** Wird von Adaptive Server Enterprise unterstützt, indem GRANT- und REVOKE-Anweisungen innerhalb der CREATE SCHEMA-Anweisung unterstützt werden.

Beispiel

Die folgende CREATE SCHEMA-Anweisung erstellt ein Schema aus zwei Tabellen. Die Anweisung muss von der Benutzer-ID sample_user ausgeführt werden, die das CREATE TABLE-Systemprivileg haben muss. Wenn die Anweisung zur Erstellung von Tabelle t2 fehlschlägt, wird keine der beiden Tabellen erstellt.

```
CREATE SCHEMA AUTHORIZATION sample_user
CREATE TABLE t1 ( id1 INT PRIMARY KEY )
CREATE TABLE t2 ( id2 INT PRIMARY KEY );
```

Das Trennzeichen für Anweisungen wird in der folgenden CREATE SCHEMA-Anweisung hinter die erste CREATE TABLE-Anweisung gesetzt. Da das Trennzeichen für Anweisungen das Ende der CREATE SCHEMA-Anweisung markiert, wird das Beispiel vom Datenbankserver als ein Batch aus zwei Anweisungen interpretiert. Folglich wird die Tabelle t1 erstellt, auch wenn die Anweisung zur Erstellung der Tabelle t2 fehlschlägt.

```
CREATE SCHEMA AUTHORIZATION sample_user
CREATE TABLE t1 ( id1 INT PRIMARY KEY );
CREATE TABLE t2 ( id2 INT PRIMARY KEY );
```

CREATE SEQUENCE-Anweisung

Erstellt eine Sequenz, die verwendet werden kann, um Primärschlüsselwerte zu erstellen, die über mehrere Tabellen hinweg eindeutig sind, und um Standardwerte für eine Tabelle zu generieren.

Syntax

```
CREATE [ OR REPLACE ] SEQUENCE [ owner.] sequence-name
[ INCREMENT BY signed-integer ]
[ START WITH signed-integer ]
[ MINVALUE signed-integer | NO MINVALUE ]
[ MAXVALUE signed-integer | NO MAXVALUE ]
[ CACHE integer | NO CACHE ]
[ CYCLE | NO CYCLE ]
```

Parameter

OR REPLACE-Klausel Wenn Sie OR REPLACE angeben, wird eine neue Sequenz erstellt oder eine bestehende Sequenz mit demselben Namen ersetzt. Wenn Sie nicht die OR REPLACE-Klausel verwenden, wird ein Fehler zurückgegeben, wenn Sie den Namen einer Sequenz angeben, die für den aktuellen Benutzer bereits vorhanden ist.

INCREMENT BY-Klausel Definiert den Betrag, um den der nächste Sequenzwert gegenüber dem letzten zugeordneten Wert erhöht wird. Standardwert ist "1". Geben Sie einen negativen Wert an, um für eine absteigende Sequenz zu erstellen. Wenn der INCREMENT BY-Wert 0 ist, wird ein Fehler zurückgegeben.

START WITH-Klausel Definiert den Startwert der Sequenz. Wenn Sie keinen Wert für die START WITH-Klausel angeben, wird MINVALUE für aufsteigende Sequenzen bzw. MAXVALUE für

absteigende Sequenzen verwendet. Es wird ein Fehler gemeldet, wenn der START WITH-Wert außerhalb des durch MINVALUE und MAXVALUE angegebenen Bereichs liegt.

MINVALUE-Klausel Definiert den kleinsten durch die Sequenz generierten Wert. Standardwert ist "1". Ein Fehler wird zurückgegeben, wenn MINVALUE größer ist als $(2^{63}-1)$ oder kleiner als $-(2^{63}-1)$. Außerdem wird ein Fehler zurückgegeben, wenn MINVALUE größer ist als MAXVALUE.

MAXVALUE-Klausel Definiert den größten durch die Sequenz generierten Wert. Standardwert ist $2^{63}-1$. Ein Fehler wird zurückgegeben, wenn MAXVALUE größer ist als $2^{63}-1$ oder kleiner als $-(2^{63}-1)$.

CACHE-Klausel Gibt die Anzahl der vorab zugewiesenen Sequenzwerte an, die für einen schnelleren Zugriff im Speicher aufbewahrt werden. Wenn der Cache belegt ist, wird der Sequenzcache neu gefüllt und ein entsprechender Eintrag in das Transaktionslog geschrieben. Zur Checkpoint-Zeit wird der aktuelle Wert des Cache an die ISYSEQUENCE-Systemtabelle weitergeleitet. Der Standardwert ist 100.

CYCLE-Klausel Legt fest, ob weitere Werte generiert werden sollen, nachdem der Höchst- oder Mindestwert erreicht wurde.

Der Standardwert ist NOCYCLE, sodass ein Fehler zurückgegeben wird, sobald der Höchst- oder Mindestwert erreicht ist.

Bemerkungen

Eine Sequenz ist ein Datenbankobjekt, das die automatische Generierung von numerischen Werten ermöglicht. Eine Sequenz ist nicht an eine bestimmte oder eindeutige Tabellenspalte gebunden, aber der Zugriff ist nur über die Tabellenspalte möglich, auf die sie angewendet wird.

Sequenzen können Werte auf eine der folgenden Arten generieren:

- Gleichmäßiges Erhöhen oder Vermindern ohne Grenze
- Erhöhen oder Vermindern bis zu einer vom Benutzer definierten Grenze und Stoppen
- Erhöhen oder Vermindern bis zu einer vom Benutzer definierten Grenze, zurück an den Anfang und erneuter Start

Sie steuern das Verhalten der Sequenz, wenn sie keine Werte mehr zur Verfügung hat, mithilfe der CYCLE-Klausel.

Wenn ein Sequenzwert erhöht wird und MAXVALUE überschreitet, wird MINVALUE als nächster Sequenzwert verwendet, sofern CYCLE angegeben ist. Wenn ein Sequenzwert vermindert wird und MINVALUE unterschreitet, wird MAXVALUE als nächster Sequenzwert verwendet, sofern CYCLE angegeben ist. Wenn CYCLE nicht angegeben ist, wird ein Fehler zurückgegeben.

Sequenzwerte können nicht mit Ansichten oder Definitionen von materialisierten Ansichten verwendet werden.

Privilegien

Sie müssen das CREATE ANY SEQUENCE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben, um Sequenzen erstellen zu können.

Nebenwirkungen

Keine

Siehe auch

- „ALTER SEQUENCE-Anweisung“ auf Seite 490
- „DROP SEQUENCE-Anweisung“ auf Seite 822
- Sequenzausdruck-Klausel, SELECT-Anweisung auf Seite 1028
- „Verwendung einer Sequenz zum Generieren von eindeutigen Werten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- Wählen zwischen Sequenzen und AUTOINCREMENT-Werten [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Sequenzen umfassen SQL/2008-Sprachenfunktion T176. SQL Anywhere lässt nicht die optionale Angabe des SEQUENCE-Datentyps zu. Dies kann mit einem CAST bei Verwendung der Sequenz erreicht werden.

Außerdem sind die folgenden Erweiterungen des Herstellers verfügbar:

- CACHE-Klausel
- OR REPLACE-Syntax
- CURRVAL-Ausdruck
- Verwendung des Sequenzgenerators in DEFAULT Ausdrücken

Beispiel

Im folgenden Beispiel wird eine Sequenz mit dem Namen Test erstellt, die mit 4 beginnt, um 2 erhöht wird, nicht als Zyklus durchlaufen wird und 15 Werte gleichzeitig im Cache speichert:

```
CREATE SEQUENCE Test
START WITH 4
INCREMENT BY 2
NO MAXVALUE
NO CYCLE
CACHE 15;
```

CREATE SERVER-Anweisung

Erstellt einen Fremdserver.

Syntax 1

```
CREATE SERVER server-name
CLASS server-class-string
USING connection-info-string
[ READ ONLY ]
```

```
server-class-string :
'ADSODBC'
| 'ASEODBC'
| 'DB2ODBC'
| 'HANAODBC'
```

```
'IQODBC'  
'MIRROR'  
'MSACCESSODBC'  
'MSSODBC'  
'MSQLODBC'  
'ODBC'  
'ORAODBC'  
'SAODBC'  
'ULODBC'
```

connection-info-string :
{ 'data-source-name' | 'sqlanywhere-connection-string' }

Syntax 2

```
CREATE SERVER server-name  
CLASS 'DIRECTORY'  
USING using-string
```

using-string :
'ROOT = *path* [;SUBDIRS = *n*] [;READONLY = { YES | NO }] [;CREATEDIRS = { YES | NO }]
[;DELIMITER = { / | \ }]'

Parameter

CLASS-Klausel Gibt die Serverklasse an, die Sie für eine Fernverbindung verwenden wollen. Serverklassen enthalten detaillierte Informationen über die Serverfähigkeiten.

Die DIRECTORY-Klasse wird in Syntax 2 verwendet, um auf ein Verzeichnis auf dem lokalen Computer zuzugreifen.

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

USING-Klausel Die USING-Klausel in Syntax 1 stellt eine Verbindungszeichenfolge für den Datenbankserver bereit. Die entsprechende Verbindungszeichenfolge hängt vom verwendeten Treiber ab, der wiederum von der *server-class-string* abhängt.

Die USING-Klausel ist eine ODBC-Verbindungszeichenfolge, die 'DSN=data-source-name' enthalten kann, um einen ODBC-Datenquellennamen anzugeben, bzw. 'DRIVER=driver-name', um eine Treiber-Binärdatei unter Unix oder einen Treibernamen unter Windows anzugeben.

Bei SQL Anywhere-Fremdservern (SAODBC-Serverklassen) kann der Parameter *connection-info-string* jede gültige SQL Anywhere-Verbindungszeichenfolge sein. Sie können jeden beliebigen SQL Anywhere-Verbindungsparameter verwenden. Wenn Sie z.B. Verbindungsprobleme haben, können Sie einen LOG-Verbindungsparameter zur Protokollierung des Verbindungsversuchs aufnehmen.

Die Zeichenfolge in der USING-Klausel kann auch lokale oder globale Variablennamen in geschweiften Klammern enthalten ({*variable-name*}). Der SQL-Variablenname muss vom Typ CHAR, VARCHAR oder LONG VARCHAR sein. Eine USING-Klausel mit 'DSN={@mydsn}' zeigt beispielsweise an, dass @mydsn eine SQL-Variable ist und dass der aktuelle Inhalt der @mydsn-Variablen beim Herstellen der Verbindung mit dem Ferndatenzugriffsserver ersetzt werden muss.

In Syntax 2 gibt die USING-Klausel die folgenden Werte für das lokale Verzeichnis an:

- **ROOT-Klausel** Gibt den Pfad an, relativ zum Datenbankserver, der das Stammverzeichnis der Verzeichniszugriffsklasse bildet. Wenn Sie eine Proxytabelle unter Verwendung des Namens des Verzeichniszugriffsservers erstellen, ist die Proxytabelle relativ zu diesem Stammpfad.

Wenn Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Hauptdatenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

- **SUBDIRS-Klausel** Gibt eine Zahl zwischen 0 und 10 an, die die Anzahl der Verzeichnisebenen innerhalb des Stammverzeichnisses darstellt, auf die der Datenbankserver zugreifen kann. Wenn SUBDIRS weggelassen oder auf 0 gesetzt wird, sind nur die Dateien im Stammverzeichnis über den Verzeichniszugriffsserver zugänglich. Sie können Proxytabellen für jedes über diesen Server verfügbare Verzeichnis oder Unterverzeichnis erstellen.
- **READONLY-Klausel** Gibt an, ob die Dateien, auf die über das Verzeichnis zugegriffen wird, schreibgeschützt sind und nicht geändert werden können. Standardmäßig ist READONLY auf NO gesetzt.
- **CREATEDIRS-Klausel** Gibt an, ob mithilfe des Verzeichniszugriffsservers Verzeichnisse erstellt werden können. Der Standardwert ist NO.
- **DELIMITER-Klausel** Gibt an, ob das Trennzeichen für Pfade in der file_name-Spalte der Schrägstrich (/) oder der Backslash (\) ist. Standardmäßig wird das native Pfadtrennzeichen verwendet.

Die Zeichenfolge in der USING-Klausel kann auch lokale oder globale Variablennamen in geschweiften Klammern enthalten (*{variable-name}*). Der SQL-Variablenname muss vom Typ CHAR, VARCHAR oder LONG VARCHAR sein. Eine USING-Klausel mit 'ROOT={@mypath}' zeigt beispielsweise an, dass @mypath eine SQL-Variable ist und dass der aktuelle Inhalt der @mypath-Variablen beim Herstellen der Verbindung mit dem Verzeichniszugriffsserver ersetzt werden muss. Dies gestattet die Erstellung von dynamischen Verzeichniszugriffsservern.

Weitere Hinweise zur Verwendung von Variablen der USING-Klausel, finden Sie unter in „[Erstellen von Verzeichniszugriffsservern \(SQL\)](#)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

READ ONLY-Klausel Gibt an, dass der Zugriff auf den Fremdserver im schreibgeschützten Modus erfolgt.

Bemerkungen

Diese Anweisung wird unter Windows Mobile nicht unterstützt.

Wenn Sie einen Fremdserver erstellen, wird er der ISYSSERVER-Systemtabelle hinzugefügt. Verwenden Sie die entsprechende Systemansicht SYSSERVER, um die Tabelle anzuzeigen.

- **Syntax 1** Die CREATE SERVER-Anweisung definiert einen Fremdserver.

Um den ODBC-Treibermanager bei der Definition eines SQL Anywhere-Fremdservers zu umgehen, verwenden Sie die nachstehende Syntax, gefolgt vom Rest der *connection-info-string*:

```
CREATE SERVER remote-server
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere Native;DSN=my-dsn;UID=my-username;PWD=my-pwd';
```

Mit dieser Syntax kann beim Ferndatenzugriff direkt der SQL Anywhere ODBC-Treiber geladen werden. Sie wird von Windows und Unix unterstützt. Das direkte Laden des SQL Anywhere ODBC-Treibers gewährleistet, dass der ODBC-Treiber der aktuellen Serverversion verwendet wird. Wenn der SQL Anywhere ODBC-Treiber nur für den Ferndatenzugriff verwendet wird, muss er nicht registriert werden.

Hinweis

Wenn die Anwendung auch Fremdserver verwendet, die keine SQL Anywhere-Server sind, oder wenn SQL Anywhere-Fremdserver vorhanden sind, die ohne 'DRIVER=SQL Anywhere Native' definiert wurden, benutzt der Ferndatenzugriff weiterhin einen Treibermanager der anderen Fremdserver.

Auf Unix-Plattformen können Sie auch den SQL Anywhere ODBC-Treiber referenzieren. Die Syntax lautet wie folgt:

```
USING 'DRIVER=SQL Anywhere 16;DSN=my-dsn'
```

- **Syntax 2** Mit der Anweisung CREATE SERVER können Sie einen Verzeichniszugriffsserver erstellen, mit dem Sie auf die lokale Verzeichnisstruktur auf dem Computer zugreifen können, auf dem der Datenbankserver läuft. Sie müssen ein externes Login für jeden Datenbankbenutzer erstellen, der den Verzeichniszugriffsserver verwenden wird. Unter Unix wird der Datenbankserver als spezifischer Benutzer ausgeführt. Daher basieren Dateiberechtigungen auf den Privilegien, die dem Datenbankserverbenutzer erteilt wurden.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „ALTER SERVER-Anweisung“ auf Seite 491
- „CREATE EXTERNLOGIN-Anweisung“ auf Seite 616
- „CREATE EXISTING TABLE-Anweisung“ auf Seite 613
- „DROP SERVER-Anweisung“ auf Seite 823
- „DROP REMOTE CONNECTION-Anweisung“ auf Seite 819
- „SYSSERVER-Systemansicht“ auf Seite 1484
- „Ferndatenzugriff“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Fremdserver“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Verzeichniszugriffsserver“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Serverklassen für den Ferndatenzugriff“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Verbindungsparameter“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird unter Verwendung des SQL Anywhere ODBC-Treibers ein SQL Anywhere-Fremdserver namens RemoteSA erstellt.

```
CREATE SERVER RemoteSA
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 16;DSN=RemoteDS';
```

Im folgenden Beispiel wird der SQL Anywhere ODBC-Treiber direkt ohne den ODBC-Treibermanager geladen:

```
CREATE SERVER RemoteSA
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere Native;DSN=RemoteDS';
```

Im folgenden Beispiel wird eine Variablenreferenz verwendet, um einen dynamischen Ferndatenzugriffserver zu erstellen. Sie müssen die Systemprivilegien **MANAGE ANY USER** und **CREATE TABLE** haben, um dieses Beispiel ausführen zu können.

```
CREATE SERVER RemoteSA
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere
16;DSN={dsn_string};Server=saremate;UID=DBA;PWD=sql';

CREATE VARIABLE dsn_string LONG VARCHAR;
SET dsn_string = 'Test16';

CREATE EXTERNLOGIN DBA TO RemoteSA;

CREATE EXISTING TABLE test_employees
AT 'RemoteSA...Employees';
SELECT * FROM test_employees;
DROP REMOTE CONNECTION TO RemoteSA CLOSE ALL;
```

Im folgenden Beispiel wird unter Verwendung der ASE ODBC-Treibers ein Adaptive Server Enterprise (ASE)-Fremdserver namens ase_prod erstellt.

```
CREATE SERVER ase_prod
CLASS 'ASEODBC'
USING 'DSN=remoteASE';
```

Im folgenden Beispiel wird ein Fremdserver für den Oracle-Server namens oracle723 erstellt. Sein ODBC-Datenquellennamen ist oracle723.

```
CREATE SERVER oracle723
CLASS 'ORAODBC'
USING 'oracle723';
```

Im folgenden Beispiel wird ein Verzeichniszugriffserver erstellt, der nur auf Dateien im Verzeichnis *c:\temp* zugreift.

```
CREATE SERVER diskserver0
CLASS 'DIRECTORY'
USING 'ROOT=c:\\temp';
```

```
CREATE EXTERNLOGIN DBA TO diskserver0;
CREATE EXISTING TABLE diskdir0 AT 'diskserver0;;;.';

-- Get a list of those files.
SELECT privileges, file_name, size FROM diskdir0;
```

Im folgenden Beispiel wird ein dynamischer Verzeichniszugriffsserver erstellt, mit dessen Hilfe zwei verschiedene Verzeichnisse durchsucht werden.

```
CREATE SERVER diskserver9
CLASS 'DIRECTORY'
USING '{dir_options}';

CREATE EXTERNLOGIN DBA TO diskserver9;
CREATE EXISTING TABLE diskdir9 AT 'diskserver9;;;.';

CREATE VARIABLE dir_options VARCHAR(256);
SET dir_options = 'ROOT=c:\\temp;SUBDIRS=9;DELIMITER=';
SELECT * FROM diskdir9;

DROP REMOTE CONNECTION TO diskserver9 CLOSE ALL;
SET dir_options = 'ROOT=c:\\ProgramData;SUBDIRS=9;DELIMITER=';
SELECT * FROM diskdir9;
```

CREATE SERVICE-Anweisung [HTTP-Webdienst]

Erstellt einen neuen HTTP-Webdienst.

Syntax

```
CREATE SERVICE service-name
TYPE { 'RAW' | 'HTML' | 'JSON' | 'XML' }
[ URL [PATH] { ON | OFF | ELEMENTS } ]
[ common-attributes ]
[ AS { statement | NULL } ]
```

common-attributes :

```
[ AUTHORIZATION { ON | OFF } ]
[ ENABLE | DISABLE ]
[ METHODS 'method,...' ]
[ SECURE { ON | OFF } ]
[ USER { user-name | NULL } ]
```

method :

```
DEFAULT
| POST
| GET
| HEAD
| PUT
| DELETE
| NONE
| *
```

Parameter

service-name Webdienstnamen können aus einer beliebigen Sequenz von alphanumerischen Zeichen bestehen sowie die Zeichen Schrägstrich (/), Bindestrich (-), Unterstrich (_), Punkt (.), Ausrufezeichen (!),

Tilde (~), Sternchen (*), Apostroph ('), öffnende Klammer (()) oder schließende Klammer (()) enthalten. Allerdings darf der Dienstname nicht mit einem Schrägstrich (/) beginnen oder enden oder zwei oder mehr aufeinander folgende Schrägstriche (z.B. //) enthalten.

Sie können Ihren Dienst **root** nennen, aber dieser Name hat eine spezielle Funktion.

TYPE-Klausel Gibt den Typ des Dienstes an, wobei für jeden Dienst ein bestimmtes Antwortformat definiert wird. Der Typ muss einer der aufgelisteten Typen sein. Es gibt keine Standardeinstellung.

- **'RAW'** Die Ergebnismenge wird ohne weitere Formatierung an den Client gesendet. Die Nutzung dieses Dienstes erfordert, dass alle Inhalts-Markups explizit zur Verfügung gestellt werden. Komplexe dynamische Inhalte, die aktuellen Inhalt mit Markup, JavaScript und Grafiken enthalten, können bei Bedarf generiert werden. Der Medientyp kann durch Einstellen des Antwort-Headers "Content-Type" unter Verwendung der Prozedur `sa_set_http_header` angegeben werden. Die Einstellung 'text/html' für den Content-Type-Header wird empfohlen, wenn Sie HTML-Markup generieren, damit alle Browser die Markup als HTML und nicht "text/plain" anzeigen.
- **'HTML'** Die Ergebnismenge wird als HTML-Darstellung einer Tabelle oder einer Ansicht zurückgegeben.
- **'JSON'** Die Ergebnismenge wird in JavaScript Object Notation (JSON) zurückgegeben. Ein JSON-Dienst verarbeitet nicht automatisch eine JSON-Eingabe. Er stellt lediglich Daten (in der Antwort) im JSON-Format bereit. JSON akzeptiert POST/PUT-Methoden, wobei **application/x-www-form-urlencoded** unterstützt wird. Wenn für eine POST/PUT METHODE **Content-Type: application/json** angegeben ist, kann die Anwendung `http_variable('body')` zum Abrufen des JSON-(Anforderung)-Inhalts verwenden. SQL Anywhere führt nicht automatisch eine syntaktische Analyse der JSON-Eingabe durch. Die Anwendung muss sie syntaktisch analysieren. Weitere Hinweise zu JSON finden Sie unter <http://www.json.org>
- **'XML'** Die Ergebnismenge wird als XML zurückgegeben. Wenn die Ergebnismenge bereits XML ist, wird keine weitere Formatierung angewendet. Andernfalls wird sie automatisch als XML formatiert. Als alternativer Ansatz könnte ein RAW-Dienst unter Verwendung der FOR XML RAW-Klausel eine Auswahl zurückgeben, nachdem Sie mit der Prozedur `sa_set_http_header` einen gültigen Content-Type wie zum Beispiel 'text/xml' festgelegt haben.

URL-Klausel Bestimmt, ob URL-Pfade akzeptiert werden und, falls ja, wie sie verarbeitet werden sollen. Wenn Sie URL PATH angeben, hat dies dieselbe Wirkung wie URL.

- **OFF** Gibt an, dass auf den Dienstnamen in einer URL-Anforderung kein Pfad folgen darf. OFF ist die Standardeinstellung. Zum Beispiel ist das folgende Format aufgrund der Pfadelemente `/aaa/bbb/ccc` nicht zulässig.

`http://host-name/service-name/aaa/bbb/ccc`

Nehmen wir an, bei der Erstellung des Webdiensts wurde `CREATE SERVICE echo URL PATH OFF` angegeben. Eine URL ähnlich wie `http://localhost/echo?id=1` erzeugt die folgenden Werte:

Function call	Result
HTTP_VARIABLE('id')	1
HTTP_HEADER('@HttpQueryString')	id=1

- **ON** Gibt an, dass auf den Dienstenamen in einer URL-Anforderung ein Pfad folgen darf. Der Pfadwert wird zurückgegeben durch Abfragen einer ausschließlich für ihn vorgesehenen HTTP-Variable mit dem Namen **URL**. Ein Dienst kann so definiert werden, dass er explizit den URL-Parameter bereitstellt, oder er kann mit der Funktion HTTP_VARIABLE abgerufen werden. Die folgende Form ist beispielsweise zulässig:

```
http://host-name/service-name/aaa/bbb/ccc
```

Nehmen wir an, bei der Erstellung des Webdiensts wurde `CREATE SERVICE echo URL PATH ON` angegeben. Eine URL ähnlich wie `http://localhost/echo/one/two?id=1` erzeugt die folgenden Werte:

Function call	Result
HTTP_VARIABLE('id')	1
HTTP_VARIABLE('URL')	one/two
HTTP_HEADER('@HttpQueryString')	id=1

- **ELEMENTS** Gibt an, dass auf den Dienstenamen in einer URL-Anforderung ein Pfad folgen darf. Der Pfad wird in Segmenten abgerufen durch die Angabe des einzelnen Parameter-Schlüsselworts **URL1**, **URL2** usw. Jeder Parameter kann mit der Funktion HTTP_VARIABLE oder NEXT_HTTP_VARIABLE abgerufen werden. Diese Wiederholfunktionen können in Anwendungen verwendet werden, in denen eine variable Anzahl von Pfadelementen bereitgestellt werden kann. Die folgende Form ist beispielsweise zulässig:

```
http://host-name/service-name/aaa/bbb/ccc
```

Nehmen wir an, bei der Erstellung des Webdiensts wurde `CREATE SERVICE echo URL PATH ELEMENTS` angegeben. Eine URL ähnlich wie `http://localhost/echo/one/two?id=1` erzeugt die folgenden Werte:

Function call	Result
HTTP_VARIABLE('id')	1
HTTP_VARIABLE('URL1')	one
HTTP_VARIABLE('URL2')	two
HTTP_VARIABLE('URL3')	NULL

Function call	Result
HTTP_HEADER('@HttpQueryString')	id=1

Bis zu 10 Elemente können abgerufen werden. NULL wird zurückgegeben, wenn das entsprechende Element nicht angegeben wurde. In diesem Beispiel gibt HTTP_VARIABLE ('URL3 ') NULL zurück, weil kein entsprechendes Element angegeben wurde.

Weitere Hinweise zu URLs finden Sie unter „[Auf einem SQL Anywhere-HTTP-Webserver navigieren](#)“ [*SQL Anywhere Server - Programmierung*] und „[Zugriff auf vom Client bereitgestellte HTTP-Variablen und -Header](#)“ [*SQL Anywhere Server - Programmierung*].

AUTHORIZATION-Klausel Bestimmt, ob die Benutzer einen Benutzernamen und ein Kennwort über die HTTP-Grundautorisierung angeben müssen, wenn sie eine Verbindung mit dem Dienst herstellen. Der Standardwert ist ON. Wenn AUTHORIZATION auf OFF gesetzt ist, muss die AS-Klausel für alle Dienste verwendet werden und ein Benutzer muss mit der USER-Klausel angegeben werden. Alle Anforderungen werden mit dem Konto und den Privilegien dieses Benutzers ausgeführt. Wenn AUTHORIZATION auf ON gesetzt ist, müssen alle Benutzer einen Benutzernamen und ein Kennwort angeben. Der Zugriff auf diesen Dienst kann durch Angabe eines Benutzer- oder Gruppennamens mithilfe der USER-Klausel eingeschränkt werden. Wenn der Benutzername NULL ist, haben alle bekannten Benutzer Zugriff auf den Dienst. Die AUTHORIZATION-Klausel ermöglicht es Ihren Webdiensten, mithilfe von Datenbankautorisierung und Privilegien den Zugriff auf die Daten in Ihrer Datenbank zu steuern.

Wenn der Autorisierungswert ON ist, benutzt ein HTTP-Client, der sich mit einem Webdienst verbindet, die Basisauthentifizierung (RFC 2617), die die Benutzer- und Kennwortdaten mit base-64-Kodierung verschleiert. Es wird empfohlen, für erweiterte Sicherheit das HTTPS-Protokoll zu verwenden.

ENABLE- und DISABLE-Klausel Legt fest, ob der Dienst verfügbar ist. Standardmäßig ist ein Dienst aktiviert, wenn er erstellt wird. Beim Erstellen oder Ändern eines Diensts können Sie eine ENABLE- oder DISABLE-Klausel einbauen. Durch das Deaktivieren eines Diensts wird dieser offline gesetzt. Später kann er mit ALTER SERVICE und einer ENABLE-Klausel aktiviert werden. Eine HTTP-Anforderung an einen deaktivierten Dienst gibt in der Regel den HTTP-Status 404 Not Found zurück.

METHODS-Klausel Gibt die HTTP-Methoden an, die vom Dienst unterstützt werden. Gültige Werte sind DEFAULT, POST, GET, HEAD, PUT, DELETE und NONE. Ein Sternchen (*) kann als Platzhalter für die Methoden POST, GET und HEAD verwendet werden, welche die Standard-Anforderungstypen für die Diensttypen RAW, HTML und XML sind. Nicht alle HTTP-Methoden sind für alle Diensttypen gültig. Die folgende Tabelle enthält eine Übersicht über die gültigen HTTP-Methoden, die auf die einzelnen Diensttypen angewendet werden können:

Methodenwert	Gilt für Dienst	Beschreibung
DEFAULT	alle	Verwenden Sie DEFAULT, um die Gruppe der standardmäßigen HTTP-Methoden für den angegebenen Dienstyp zurückzusetzen. Kann nicht in eine Liste mit anderen Methodenwerten aufgenommen werden.
POST	RAW, HTML, JSON, XML	Standardmäßig aktiviert.
GET	RAW, HTML, JSON, XML	Standardmäßig aktiviert.
HEAD	RAW, HTML, JSON, XML	Standardmäßig aktiviert.
PUT	RAW, HTML, JSON, XML	Standardmäßig nicht aktiviert.
DELETE	RAW, HTML, JSON, XML	Standardmäßig nicht aktiviert.
NONE	alle	Verwenden Sie NONE, um den Zugriff auf einen Dienst zu deaktivieren.
*	RAW, HTML, JSON, XML	Wie ' POST,GET,HEAD '.

Zum Beispiel können Sie eine der folgenden Klauseln verwenden, um festzulegen, dass ein Dienst alle HTTP-Methodentypen unterstützt:

```
METHODS ' *, PUT, DELETE '
METHODS ' POST, GET, HEAD, PUT, DELETE '
```

Um die Liste der Anforderungstypen für einen beliebigen Dienstyp auf den Standardwert zu setzen, können Sie die folgende Klausel verwenden:

```
METHODS ' DEFAULT '
```

SECURE-Klausel Gibt an, ob der Dienst auf einem sicheren oder nicht sicheren Listener zugänglich sein soll. ON bedeutet, dass nur HTTPS-Verbindungen akzeptiert werden und dass über den HTTP-Port empfangene Verbindungen automatisch zum HTTPS-Port umgeleitet werden. OFF bedeutet, dass sowohl HTTP- als auch HTTPS-Verbindungen akzeptiert werden, vorausgesetzt, die erforderlichen Ports werden beim Starten des Webserver angegeben. Der Standardwert ist OFF.

USER-Klausel Gibt einen Datenbankbenutzer oder eine Gruppe von Benutzern mit den erforderlichen Privilegien an, um die Webdienstanforderung auszuführen. Eine USER-Klausel muss angegeben werden, wenn der Dienst mit AUTHORIZATION OFF konfiguriert wird, und sollte angegeben werden, wenn AUTHORIZATION ON (Standardwert) verwendet wird. Eine HTTP-Anforderung an einen Dienst, der

Autorisierung erfordert, liefert den HTTP-Antwortstatus "401 Authorization Required". Basierend auf dieser Antwort fordert der Webbrowser den Benutzer auf, eine Benutzer-ID und ein Kennwort einzugeben.

Vorsicht

Es wird dringend empfohlen, dass Sie eine USER-Klausel angeben, wenn die Autorisierung aktiviert ist (Standardwert). Andernfalls wird die Autorisierung allen Benutzern erteilt.

Die USER-Klausel steuert, welche Datenbank-Benutzerkonten verwendet werden können, um Dienstanforderungen zu verarbeiten. Datenbankzugriffsberechtigungen werden auf die dem Benutzer des Dienstes zugewiesenen Berechtigungen beschränkt.

statement Gibt einen Befehl an, zum Beispiel einen Aufruf einer gespeicherten Prozedur, der beim Zugreifen auf den Dienst aufgerufen werden soll.

Eine HTTP-Anforderung an einen Nicht-DISH-Dienst ohne *statement* gibt den SQL-Ausdruck zurück, der innerhalb seiner URL ausgeführt werden soll. Obwohl eine Autorisierung erforderlich ist, darf diese Fähigkeit nicht in Produktionssystemen benutzt werden, weil dies den Server für SQL-Injektionen anfällig macht. Wenn eine Anweisung in dem Dienst definiert ist, kann die angegebene SQL-Anweisung als einzige Anweisung über den Dienst ausgeführt werden.

In einer typischen Webdienstanwendung verwenden Sie *statement* zum Aufrufen einer Funktion oder Prozedur. Sie können Hostvariablen als Parameter für den Zugriff auf vom Client gelieferte HTTP-Variablen übergeben.

Die folgende *statement* zeigt einen Prozeduraufruf, der zwei Hostvariablen an eine Prozedur mit dem Namen **AuthenticateUser** übergibt. Dieser Aufruf setzt voraus, dass ein Webclient die Variablen **user_name** und **user_password** liefert:

```
CALL AuthenticateUser ( :user_name, :user_password );
```

Weitere Hinweise zum Übergeben von Hostvariablen an eine Funktion oder Prozedur finden Sie unter „Zugriff auf vom Client bereitgestellte HTTP-Variablen und -Header“ [[SQL Anywhere Server - Programmierung](#)].

Bemerkungen

Dienstdefinitionen werden innerhalb der ISYSWEBSERVICE-Tabelle gespeichert und können über die SYSWEBSERVICE-Ansicht untersucht werden.

Falls eine USER-Klausel angegeben ist, wird der Dienst gelöscht, wenn *user-name* gelöscht wird.

Privilegien

Sie müssen das MANAGE ANY WEB SERVICE-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „ALTER SERVICE-Anweisung [HTTP-Webdienst]“ auf Seite 494
- „DROP SERVICE-Anweisung“ auf Seite 824
- „sp_parse_json-Systemprozedur“ auf Seite 1383
- „SYSWEBSERVICE-Systemansicht“ auf Seite 1513
- „SQL Anywhere als HTTP-Webserver“ [*SQL Anywhere Server - Programmierung*]
- „root-Webdienste erstellen und anpassen“ [*SQL Anywhere Server - Programmierung*]
- „sa_set_http_header-Systemprozedur“ auf Seite 1323
- „Webdienstanwendungen auf einem HTTP-Webserver entwickeln“ [*SQL Anywhere Server - Programmierung*]
- „Bezeichner“ auf Seite 4
- „ROW-Konstruktor [zusammengesetzt]“ auf Seite 371
- „ARRAY-Konstruktor [zusammengesetzt]“ auf Seite 168

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** CREATE SERVICE wird von Adaptive Server Enterprise unterstützt, allerdings nur für die Typen XML und RAW.

Beispiele

Das folgende Beispiel zeigt, wie ein JSON-Dienst erstellt wird.

Starten Sie einen Datenbankserver mit der Option -xs (http oder https) und führen Sie dann die folgende SQL-Anweisung aus, um den Dienst einzurichten:

```
CREATE PROCEDURE ListEmployees()  
RESULT (  
    EmployeeID          integer,  
    Surname              person_name_t,  
    GivenName            person_name_t,  
    StartDate            date,  
    TerminationDate      date )  
BEGIN  
    SELECT EmployeeID, Surname, GivenName, StartDate, TerminationDate  
    FROM GROUPO.Employees  
END;  
  
CREATE SERVICE "jsonEmployeeList"  
    TYPE 'JSON'  
    AUTHORIZATION OFF  
    SECURE OFF  
    USER DBA  
    AS CALL ListEmployees();
```

Der JSON-Dienst liefert Daten, die leicht von einem AJAX-Callback verarbeitet werden können.

Führen Sie die folgende SQL-Anweisung zur Erstellung eines HTML-Diensts aus, der den Dienst in einem lesbaren Format bereitstellt:

```
CREATE SERVICE "EmployeeList"  
    TYPE 'HTML'  
    AUTHORIZATION OFF  
    SECURE OFF
```

```

USER DBA
AS CALL ListEmployees();

```

Verwenden Sie einen Webbrowser, um über eine URL, ähnlich wie `http://localhost/EmployeeList`, auf den Dienst zuzugreifen.

CREATE SERVICE-Anweisung [SOAP-Webdienst]

Erstellt einen neuen SOAP über HTTP- oder DISH-Dienst.

Syntax 1 - SOAP über HTTP-Dienste

```

CREATE SERVICE service-name
TYPE 'SOAP'
[ DATATYPE { ON | OFF | IN | OUT } ]
[ FORMAT { 'DNET' | 'CONCRETE' [ EXPLICIT { ON | OFF } ] | 'XML' | NULL } ]
[ common-attributes ]
AS statement

```

common-attributes :

```

[ AUTHORIZATION { ON | OFF } ]
[ ENABLE | DISABLE ]
[ METHODS 'method,...' ]
[ SECURE { ON | OFF } ]
[ USER { user-name | NULL } ]

```

method :

```

DEFAULT
| POST
| HEAD
| NONE

```

Syntax 2 - DISH-Dienste

```

CREATE SERVICE service-name
TYPE 'DISH'
[ GROUP { group-name | NULL } ]
[ FORMAT { 'DNET' | 'CONCRETE' [ EXPLICIT { ON | OFF } ] | 'XML' | NULL } ]
[ common-attributes ]

```

common-attributes:

```

[ AUTHORIZATION { ON | OFF } ]
[ ENABLE | DISABLE ]
[ METHODS 'method,...' ]
[ SECURE { ON | OFF } ]
[ USER { user-name | NULL } ]

```

method :

```

DEFAULT
| POST
| GET
| HEAD
| NONE
| *

```

Parameter

service-name Webdienstnamen können aus einer beliebigen Sequenz von alphanumerischen Zeichen bestehen sowie die Zeichen Schrägstrich (/), Bindestrich (-), Unterstrich (_), Punkt (.), Ausrufezeichen (!), Tilde (~), Sternchen (*), Apostroph ('), öffnende Klammer (()) oder schließende Klammer (()) enthalten. Allerdings darf der Dienstname nicht mit einem Schrägstrich (/) beginnen oder enden oder zwei oder mehr aufeinander folgende Schrägstriche (z.B. //) enthalten.

Im Gegensatz zu anderen Diensten dürfen Sie in einem DISH-Dienstnamen keinen Schrägstrich (/) verwenden.

Sie können Ihren Dienst **root** nennen, aber dieser Name hat eine spezielle Funktion.

TYPE-Klausel Gibt den Typ des Dienstes an, wobei für jeden Dienst ein bestimmtes Antwortformat definiert wird. Der Typ muss einer der aufgelisteten Typen sein. Es gibt keine Standardeinstellung.

- **'SOAP'** Die Ergebnismenge wird im XML-Format zurückgegeben, das als SOAP-Envelope bezeichnet wird. Das Format der Daten wird durch die FORMAT-Klausel bestimmt. Eine Anforderung an einen SOAP-Dienst muss eine gültige SOAP-Anforderung und nicht nur eine einfache HTTP-Anforderung sein. Weitere Hinweise zu den SOAP-Standards finden Sie unter <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>.
- **'DISH'** Ein DISH-Dienst (Determine SOAP Handler) ist ein SOAP-Endpunkt, der alle SOAP-Dienste in seinem GROUP-Kontext referenziert. Außerdem exponiert er die Schnittstellen zu seinen SOAP-Diensten durch die Erstellung eines WSDL-Objekts (Web Services Description Language), das von SOAP-Client-Toolkits verwendet werden kann.

GROUP-Klausel Ein DISH Dienst ohne GROUP-Klausel exponiert alle in der Datenbank definierten SOAP-Dienste. Laut Konvention kann der SOAP-Dienstname aus einem GROUP- und einem NAME-Element bestehen. Der Name wird durch den letzten Schrägstrich von der Gruppe getrennt. Beispiel: Der Name eines als 'aaa/bbb/cc' definierten SOAP-Dienstes ist 'cc', und die Gruppe 'aaa/bbb'. Die Angabe eines DISH-Dienstes unter Verwendung dieser Konvention ist ungültig. Stattdessen wird eine GROUP-Klausel angewendet, um die Gruppe von SOAP-Diensten anzugeben, für die er als SOAP-Endpunkt fungieren soll.

Hinweis

Schrägstriche werden in WSDL in Unterstriche konvertiert, um gültigen XML-Code zu erzeugen. Lassen Sie bei der Verwendung von einem DISH-Dienst Vorsicht walten, dessen GROUP-Klausel nicht so angegeben ist, dass alle SOAP-Dienste exponiert werden, die möglicherweise Schrägstriche enthalten. Achten Sie darauf, Mehrdeutigkeiten zu vermeiden, wenn Sie Gruppen in Verbindung mit SOAP-Dienstnamen verwenden, die Unterstriche enthalten.

DATATYPE-Klausel Gilt nur für SOAP-Dienste. Wenn DATATYPE OFF angegeben ist, werden SOAP-Eingabeparameter und -Antwortdaten als XMLSchema-Zeichenfolgetypen definiert. In den meisten Fällen werden die tatsächlichen Datentypen bevorzugt, weil durch sie die Notwendigkeit entfällt, dass der SOAP-Client die Daten vor der Berechnung festlegt. Parameterdatentypen werden im Schema-Abschnitt der WDSL angezeigt, der vom DISH-Dienst erstellt wird. Ausgabedatentypen werden als XML-Schematypattribute für die jeweilige Datenspalte dargestellt.

Die folgenden Werte sind bei der DATATYPE-Klausel zulässig:

- **ON** Generiert die Datentypisierung für Eingabeparameter und zurückgegebene Ergebnismengen
- **OFF** Alle Eingabeparameter und Antwortdaten haben den Datentyp **XMLSchema**-Zeichenfolge (Standardwert).
- **IN** Generiert die tatsächlichen Datentypen nur für Eingabeparameter. Antwortdatentypen sind **XMLSchema**-Zeichenfolgen.
- **OUT** Generiert die tatsächlichen Datentypen nur für Antworten. Eingabeparameter werden als **XMLSchema**-Zeichenfolge eingegeben.

Weitere Hinweise zu SOAP-Diensten finden Sie unter „[Praktische Einführung: Verwenden von SQL Anywhere für den Zugriff auf einen SOAP/DISH-Dienst](#)“ [[SQL Anywhere Server - Programmierung](#)].

Weitere Hinweise zur Zuordnung von XML-Schematypen zu SQL-Datentypen finden Sie unter „[SOAP-Datentypen](#)“ [[SQL Anywhere Server - Programmierung](#)].

FORMAT-Klausel Diese Klausel gibt das Ausgabeformat an, das beim Senden von Antworten an SOAP-Clientanwendungen verwendet wird.

Das SOAP-Dienstformat wird durch die dazugehörige DISH-Dienstformatspezifikation bestimmt, wenn es nicht vom SOAP-Dienst angegeben wird. Das Standardformat ist DNET.

SOAP-Anforderungen sollten an den DISH-Dienst (den SQL Anywhere-SOAP-Endpunkt) gerichtet werden, um gemeinsame Formatierungsregeln für eine Gruppe von SOAP-Diensten (SOAP-Vorgängen) zu nutzen. Eine FORMAT-Spezifikation für einen SOAP-Dienst überschreibt diejenige eines DISH-Dienstes. Die Formatspezifikation des DISH-Dienstes wird verwendet, wenn für den SOAP-Dienst keine FORMAT-Klausel definiert ist. Wenn für keinen der beiden Dienste ein FORMAT-Objekt zur Verfügung gestellt wird, ist der Standardwert '**DNET**'.

Folgende Formate werden unterstützt:

- '**DNET**' Die Ausgabe erfolgt in einem System.Data.DataSet-kompatiblen Format zur Verwendung durch .NET-Clientanwendungen. (Standardwert)
- '**CONCRETE**' Dieses Ausgabeformat wird zur Unterstützung von Client-SOAP-Toolkits verwendet, die in der Lage sind, Schnittstellen zu generieren, die Arrays von Zeilen- und Spaltenobjekten darstellen, aber nicht in der Lage sind, das DNET-Format zu bearbeiten. Java- und .NET-Clients können dieses Ausgabeformat einfach bearbeiten.

Das spezifische Format wird innerhalb des WSDL-Codes als explizites DataSet-Objekt oder als **SimpleDataset** bereitgestellt. Beide DataSet-Darstellungen beschreiben eine Datenstruktur für einen Zeilen-Array, in dem jede Zeile ein Array von Spalten enthält. Ein explizites DataSet-Objekt hat den Vorteil, dass es die tatsächliche Form der Ergebnismenge darstellt, indem es Parameternamen und Datentypen für jede Spalte in der Zeile angibt. Im Gegensatz dazu stellt das **SimpleDataset** Zeilen mit einer unbegrenzten Anzahl der Spalten beliebigen Typs bereit.

FORMAT 'CONCRETE' EXPLICIT ON setzt voraus, dass die **Service**-Anweisung eine gespeicherte Prozedur aufruft, die wiederum eine **RESULT**-Klausel definiert. Wenn diese Bedingung erfüllt ist, stellt der SOAP-Dienst ein explizites DataSet bereit, dessen Name mit dem Dienstenamen und dem Zusatz **Dataset** beginnt.

Wenn die Bedingung nicht erfüllt ist, wird ein **SimpleDataset** verwendet.

- **'XML'** Die Ausgabe wird in einem XMLSchema-Zeichenfolgenformat generiert. Die Antwort ist ein XML-Dokument, das zum Extrahieren der Spaltendaten eine weitere Verarbeitung durch den SOAP-Client erfordert. Dieses Format ist für SOAP-Clients geeignet, die keine Zwischen-Schnittstellenobjekte generieren, welche Arrays von Zeilen und Spalten darstellen.
- **NULL** Im Falle eines NULL-Typs verwendet der SOAP- oder DISH-Dienst das Standardverhalten. Der Formattyp eines bestehenden Diensts wird überschrieben, wenn der NULL-Typ in einer ALTER SERVICE-Anweisung verwendet wird.

AUTHORIZATION-Klausel Bestimmt, ob die Benutzer einen Benutzernamen und ein Kennwort über die HTTP-Grundautorisierung angeben müssen, wenn sie eine Verbindung mit dem Dienst herstellen. Der Standardwert ist ON. Wenn AUTHORIZATION auf OFF gesetzt ist, muss die AS-Klausel für SOAP-Dienste verwendet werden und ein Benutzer muss mit der USER-Klausel angegeben werden. Alle Anforderungen werden mit dem Konto und den Privilegien dieses Benutzers ausgeführt. Wenn AUTHORIZATION auf ON gesetzt ist, müssen alle Benutzer einen Benutzernamen und ein Kennwort angeben. Der Zugriff auf diesen Dienst kann durch Angabe eines Benutzer- oder Gruppennamens mithilfe der USER-Klausel eingeschränkt werden. Wenn der Benutzername NULL ist, haben alle bekannten Benutzer Zugriff auf den Dienst. Die AUTHORIZATION-Klausel ermöglicht es Ihren Webdiensten, mithilfe von Datenbankautorisierung und Privilegien den Zugriff auf die Daten in Ihrer Datenbank zu steuern.

Wenn der Autorisierungswert ON ist, benutzt ein HTTP-Client, der sich mit einem Webdienst verbindet, die Basisauthentifizierung (RFC 2617), die die Benutzer- und Kennwortdaten mit base-64-Kodierung verschleiert. Es wird empfohlen, für erweiterte Sicherheit das HTTPS-Protokoll zu verwenden.

ENABLE- und DISABLE-Klausel Legt fest, ob der Dienst verfügbar ist. Standardmäßig ist ein Dienst aktiviert, wenn er erstellt wird. Beim Erstellen oder Ändern eines Diensts können Sie eine ENABLE- oder DISABLE-Klausel einbauen. Durch das Deaktivieren eines Diensts wird dieser offline gesetzt. Später kann er mit ALTER SERVICE und einer ENABLE-Klausel aktiviert werden. Eine HTTP-Anforderung an einen deaktivierten Dienst gibt in der Regel den HTTP Status "404 Not Found" zurück.

METHODS-Klausel Gibt die HTTP-Methoden an, die vom Dienst unterstützt werden. Gültige Werte sind DEFAULT, POST, GET, HEAD, und NONE. Ein Sternchen (*) kann als Kurzform für die Darstellung der POST-, GET- und HEAD-Methoden verwendet werden. Die Standard-Methodentypen für SOAP-Dienste sind POST und HEAD. Die Standard-Methodentypen für DISH-Dienste sind GET, POST und HEAD. Nicht alle HTTP-Methoden sind für alle Diensttypen gültig. Die folgende Tabelle enthält eine Übersicht über die gültigen HTTP-Methoden, die auf die einzelnen Diensttypen angewendet werden können:

Methodenwert	Gilt für Dienst	Beschreibung
DEFAULT	Beide	Verwenden Sie DEFAULT, um die Gruppe der standardmäßigen HTTP-Methoden für den angegebenen Dienstyp zurückzusetzen. Kann nicht in eine Liste mit anderen Methodenwerten aufgenommen werden.
POST	Beide	Standardmäßig aktiviert für SOAP.
GET	Nur DISH	Standardmäßig aktiviert für DISH.
HEAD	Beide	Standardmäßig aktiviert für SOAP und DISH.
NONE	Beide	Verwenden Sie NONE, um den Zugriff auf einen Dienst zu deaktivieren. Bei der Anwendung auf einen SOAP-Dienst kann auf den Dienst nicht direkt über eine SOAP-Anforderung zugegriffen werden. Dies erzwingt exklusiven Zugriff auf einen SOAP-Vorgang über einen als SOAP-Endpunkt fungierenden DISH-Dienst. Es wird empfohlen, für die einzelnen SOAP-Dienste METHOD NONE anzugeben.
*	Nur DISH	Wie 'POST,GET,HEAD'.

Zum Beispiel können Sie die folgende Klausel verwenden, um festzulegen, dass ein Dienst alle SOAP über HTTP-Methodentypen unterstützt:

```
METHODS 'POST,HEAD'
```

Um die Liste der Anforderungstypen für einen beliebigen Dienstyp auf den Standardwert zu setzen, können Sie die folgende Klausel verwenden:

```
METHODS 'DEFAULT'
```

SECURE-Klausel Gibt an, ob der Dienst auf einem sicheren oder nicht sicheren Listener zugänglich sein soll. ON bedeutet, dass nur HTTPS-Verbindungen akzeptiert werden und dass über den HTTP-Port empfangene Verbindungen automatisch zum HTTPS-Port umgeleitet werden. OFF bedeutet, dass sowohl HTTP- als auch HTTPS-Verbindungen akzeptiert werden, vorausgesetzt, die erforderlichen Ports werden beim Starten des Webserver angegeben. Der Standardwert ist OFF.

USER-Klausel Gibt einen Datenbankbenutzer oder eine Gruppe von Benutzern mit den erforderlichen Privilegien an, um die Webdienstanforderung auszuführen. Eine USER-Klausel muss angegeben werden,

wenn der Dienst mit AUTHORIZATION OFF konfiguriert wird, und sollte angegeben werden, wenn AUTHORIZATION ON (Standardwert) verwendet wird. Eine HTTP-Anforderung an einen Dienst, der Autorisierung erfordert, liefert den HTTP-Antwortstatus "401 Authorization Required". Basierend auf dieser Antwort fordert der Webbrowser den Benutzer auf, eine Benutzer-ID und ein Kennwort einzugeben.

Vorsicht

Es wird dringend empfohlen, dass Sie eine USER-Klausel angeben, wenn die Autorisierung aktiviert ist (Standardwert). Andernfalls wird die Autorisierung allen Benutzern erteilt.

Die USER-Klausel steuert, welche Datenbank-Benutzerkonten verwendet werden können, um Dienstanforderungen zu verarbeiten. Datenbankzugriffsberechtigungen werden auf die dem Benutzer des Dienstes zugewiesenen Berechtigungen beschränkt.

statement Gibt einen Befehl an, zum Beispiel einen Aufruf einer gespeicherten Prozedur, der beim Zugreifen auf den Dienst aufgerufen werden soll.

Ein DISH-Dienst ist der einzige Dienst, der entweder eine Null-Anweisung definieren oder keine Anweisung definieren muss. Ein SOAP-Dienst muss eine Anweisung definieren. Jeder andere SERVICE-Wert kann eine NULL-Anweisung definieren, jedoch nur, wenn er mit AUTHORIZATION ON konfiguriert ist.

Eine HTTP-Anforderung an einen Nicht-DISH-Dienst ohne *statement* gibt den SQL-Ausdruck zurück, der innerhalb seiner URL ausgeführt werden soll. Obwohl eine Autorisierung erforderlich ist, darf diese Fähigkeit nicht in Produktionssystemen benutzt werden, weil dies den Server für SQL-Injektionen anfällig macht. Wenn eine Anweisung in dem Dienst definiert ist, kann die angegebene SQL-Anweisung als einzige Anweisung über den Dienst ausgeführt werden.

In einer typischen Webdienstanwendung verwenden Sie *statement* zum Aufrufen einer Funktion oder Prozedur. Sie können Hostvariablen als Parameter für den Zugriff auf vom Client gelieferte HTTP-Variablen übergeben.

Die folgende *statement* zeigt einen Prozeduraufruf, der zwei Hostvariablen an eine Prozedur mit dem Namen **AuthenticateUser** übergibt. Dieser Aufruf setzt voraus, dass ein Webclient die Variablen **user_name** und **user_password** liefert:

```
CALL AuthenticateUser ( :user_name, :user_password );
```

Weitere Hinweise zum Übergeben von Hostvariablen an eine Funktion oder Prozedur finden Sie unter „Zugriff auf vom Client bereitgestellte HTTP-Variablen und -Header“ [[SQL Anywhere Server - Programmierung](#)].

Bemerkungen

Dienstdefinitionen werden innerhalb der ISYSWEBSERVICE-Tabelle gespeichert und können über die SYSWEBSERVICE-Ansicht untersucht werden.

Privilegien

Sie müssen das MANAGE ANY WEB SERVICE-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „ALTER SERVICE-Anweisung [SOAP-Webdienst]“ auf Seite 500
- „DROP SERVICE-Anweisung“ auf Seite 824
- „SYSWEBSERVICE-Systemansicht“ auf Seite 1513
- „SQL Anywhere als HTTP-Webserver“ [*SQL Anywhere Server - Programmierung*]
- „root-Webdienste erstellen und anpassen“ [*SQL Anywhere Server - Programmierung*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** CREATE SERVICE wird von Adaptive Server Enterprise nur für SOAP-Datentypen unterstützt.

CREATE SPATIAL REFERENCE SYSTEM-Anweisung

Erstellt oder ersetzt ein räumliches Bezugssystem.

Syntax

```
{ CREATE [ OR REPLACE ] SPATIAL REFERENCE SYSTEM
| CREATE SPATIAL REFERENCE SYSTEM IF NOT EXISTS }
srs-name
[ srs-attribute ] [ srs-attribute ... ]

srs-name : Zeichenfolge

srs-attribute :
IDENTIFIED BY srs-id
| DEFINITION { definition-string | NULL }
| ORGANIZATION { organization-name IDENTIFIED BY organization-srs-id | NULL }
| TRANSFORM DEFINITION { transform-definition-string | NULL }
| LINEAR UNIT OF MEASURE linear-unit-name
| ANGULAR UNIT OF MEASURE { angular-unit-name | NULL }
| TYPE { ROUND EARTH | PLANAR }
| COORDINATE coordinate-name { UNBOUNDED | BETWEEN low-number AND high-number }
| ELLIPSOID SEMI MAJOR AXIS semi-major-axis-length { SEMI MINOR AXIS semi-minor-axis-length |
INVERSE FLATTENING inverse-flattening-ratio }
| SNAP TO GRID { grid-size | DEFAULT }
| TOLERANCE { tolerance-distance | DEFAULT }
| POLYGON FORMAT polygon-format
| STORAGE FORMAT storage-format
```

srs-id : Ganzzahl

semi-major-axis-length : Zahl

semi-minor-axis-length : Zahl

inverse-flattening-ratio : Zahl

grid-size : *DOUBLE* : normalerweise zwischen 0 und 1

tolerance-distance : *Zahl*

axis-order : { 'x/y/z/m' | 'long/lat/z/m' | 'lat/long/z/m' }

polygon-format : { 'CounterClockWise' | 'Clockwise' | 'EvenOdd' }

storage-format : { 'Internal' | 'Original' | 'Mixed' }

Parameter

OR REPLACE-Klausel Mit OR REPLACE wird das räumliche Bezugssystem erstellt, wenn das System nicht bereits in der Datenbank vorhanden ist, und das System ersetzt, wenn es vorhanden ist. Ein Fehler wird zurückgegeben, wenn Sie ein räumliches Bezugssystem ersetzen, das gerade verwendet wird. Ein Fehler wird auch zurückgegeben, wenn Sie versuchen, ein räumliches Bezugssystem zu ersetzen, das bereits in der Datenbank vorhanden ist, und dabei die OR REPLACE-Klausel nicht einbeziehen.

CREATE SPATIAL REFERENCE IF NOT EXISTS CREATE SPATIAL REFERENCE IF NOT EXISTS prüft, ob bereits ein räumliches Bezugssystem mit diesem Namen vorhanden ist. Falls es nicht vorhanden ist, erstellt der Datenbankserver das räumliche Bezugssystem. Wenn es bereits vorhanden ist, wird keine weitere Aktion ausgeführt und kein Fehler zurückgegeben.

IDENTIFIED BY-Klausel Verwenden Sie diese Klausel zum Angeben der SRID (*srs-id*) für das räumliche Bezugssystem. Wenn das räumliche Bezugssystem von einer Organisation mit einer *organization-srs-id* definiert wird, sollte *srs-id* auf diesen Wert gesetzt werden.

Wenn die IDENTIFIED BY-Klausel nicht angegeben ist, wird die SRID standardmäßig auf die *organization-srs-id* gesetzt, die durch die ORGANIZATION-Klausel oder die DEFINITION-Klausel definiert ist. Wenn keine der beiden Klauseln eine *organization-srs-id* definiert, die als Standard-SRID verwendet werden könnte, wird ein Fehler zurückgegeben.

Wenn das räumliche Bezugssystem auf einem Well-Known-Koordinatensystem basiert, aber eine andere geodätische Interpretation aufweist, setzen Sie den *srs-id*-Wert 1000000000 (eine Milliarde) plus Well-Known-Wert. Zum Beispiel wäre die SRID für eine planare Interpretation des geodätischen räumlichen Bezugssystems WGS 84 (ID 4326) gleich 1000004326.

Mit Ausnahme von SRID 0 erhalten von SQL Anywhere bereitgestellte räumliche Bezugssysteme, die nicht auf Well-Known-Systemen basieren, SRIDs von 2000000000 (zwei Milliarden) und höher. Der Bereich der SRID-Werte von 2000000000 bis 2147483647 ist von SQL Anywhere reserviert und Sie sollten keine SRIDs in diesem Bereich erstellen.

Um die Möglichkeit zu reduzieren, dass Sie eine von einer Definitionsautorität wie OGC oder von anderen Herstellern reservierte SRID wählen, sollten Sie keine SRID im Bereich 0 bis 32767 (reserviert von EPSG) oder im Bereich 2147483547 bis 2147483647 wählen.

Außerdem darf die Zahl nicht größer sein als $2^{31}-1$ oder 2147483647, weil die SRID als 32-Bit-Ganzzahlwert mit Vorzeichen gespeichert wird.

DEFINITION-Klausel Verwenden Sie diese Klausel, um Standardeinstellungen für das Koordinatensystem festzulegen oder aufzuheben. Wenn ein Attribut in einer anderen Klausel als der

DEFINITION-Klausel gesetzt wird, verwendet es den in dieser anderen Klausel angegebenen Wert, unabhängig von der Angabe in der DEFINITION-Klausel.

definition-string ist eine Zeichenfolge in der Well Known Text-Syntax des räumlichen Bezugssystems gemäß der Definition von SQL/MM und OGC. Zum Beispiel liefert die folgende Abfrage die Definition für WGS 84.

```
SELECT ST_SpatialRefSys::ST_FormatWKT( definition )
FROM ST_SPATIAL_REFERENCE_SYSTEMS
WHERE srs_id=4326;
```

Wenn Sie in Interactive SQL auf den zurückgegebenen Wert doppelklicken, erscheint eine einfacher zu lesende Version des Werts.

Wenn die DEFINITION-Klausel angegeben ist, wird die *definition-string* syntaktisch analysiert und verwendet, um Standardwerte für Attribute zu wählen. Zum Beispiel kann die *definition-string* ein AUTHORITY-Element zum Definieren von *organization-name* und *organization-srs-id* enthalten.

Parameterwerte in der *definition-string* werden von Werten außer Kraft gesetzt, die mit Klauseln der SQL-Anweisung explizit eingestellt werden. Wenn beispielsweise die ORGANIZATION-Klausel angegeben ist, setzt sie den Wert für ORGANIZATION in der *definition-string* außer Kraft.

ORGANIZATION-Klausel Verwenden Sie diese Klausel zum Angeben von Informationen über die Organisation, die das räumliche Bezugssystem erstellt hat, auf dem das neue räumliche Bezugssystem basiert. *organization-name* ist der Name der Organisation, die das System erstellt hat; *organization-srs-id* der numerische Bezeichner, den die Organisation zur Identifizierung des räumlichen Bezugssystems verwendet.

TRANSFORM DEFINITION-Klausel Mit dieser Klausel geben Sie eine Beschreibung der Transformationsdefinition an, die für das räumliche Bezugssystem verwendet werden soll. Derzeit wird nur die PROJ.4-Transformationsdefinition unterstützt. Zum Beispiel lautet die *transform-definition-string* für WGS 84 '+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs'.

Wenn Sie eine nicht unterstützte Transformationsdefinition angeben, wird ein Fehler zurückgegeben.

Die Transformationsdefinition wird von der ST_Transform-Methode beim Transformieren von Daten zwischen räumlichen Bezugssystemen verwendet. Einige Transformationen können möglicherweise auch dann durchgeführt werden, wenn keine *transform-definition-string* definiert ist.

LINEAR UNIT OF MEASURE-Klausel Verwenden Sie diese Klausel zum Angeben der linearen Maßeinheit für das räumliche Bezugssystem. Der angegebene Wert muss einer linearen Maßeinheit gemäß der Definition in der ST_UNITS_OF_MEASURE-Systemansicht entsprechen.

Wenn diese Klausel nicht angegeben ist und nicht in der DEFINITION-Klausel definiert wird, ist der Standardwert METRE.

Verwenden Sie zum Hinzufügen von vordefinierten Maßeinheiten zur Datenbank die sa_install_feature-Systemprozedur.

Verwenden Sie zum Hinzufügen von benutzerdefinierten Maßeinheiten zur Datenbank die CREATE SPATIAL UNIT OF MEASURE-Anweisung.

Hinweis

Obwohl die Schreibweisen METRE und METER akzeptiert werden, wird METRE bevorzugt, da es dem SQL/MM-Standard entspricht.

ANGULAR UNIT OF MEASURE-Klausel Verwenden Sie diese Klausel zum Angeben der Winkelmaßeinheit für das räumliche Bezugssystem. Der angegebene Wert muss einer Winkelmaßeinheit gemäß der Definition in der ST_UNITS_OF_MEASURE-Systemansicht entsprechen.

Wenn diese Klausel nicht angegeben ist und nicht in der DEFINITION-Klausel definiert wird, ist der Standardwert DEGREE für geografische räumliche Bezugssysteme und NULL für nicht geografische räumliche Bezugssysteme.

Die Winkelmaßeinheit darf bei geografischen räumlichen Bezugssystemen nicht NULL sein und muss bei nicht geografischen räumlichen Bezugssystemen NULL sein.

Verwenden Sie zum Hinzufügen von vordefinierten Maßeinheiten zur Datenbank die sa_install_feature-Systemprozedur.

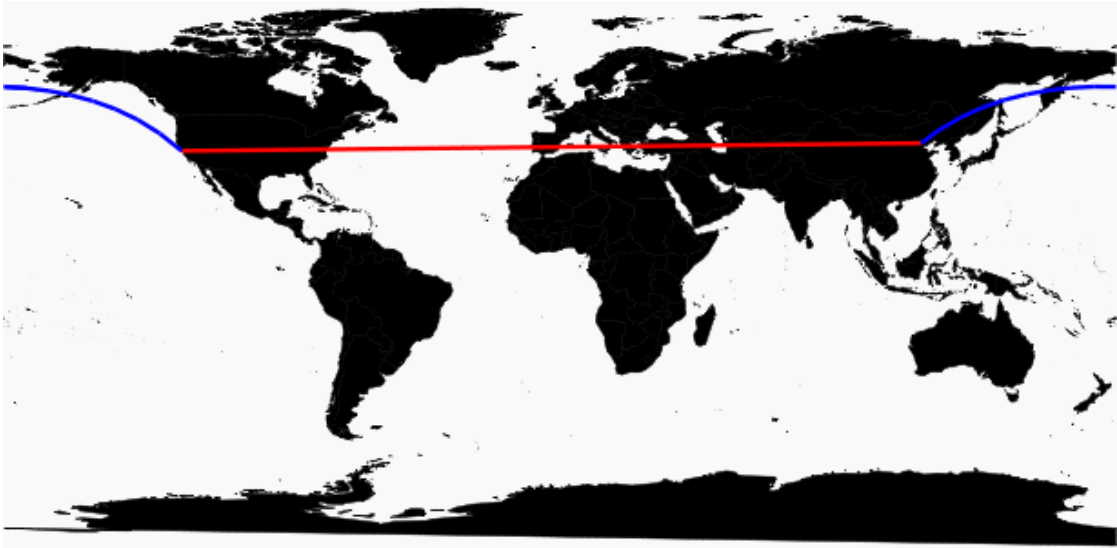
Verwenden Sie zum Hinzufügen von benutzerdefinierten Maßeinheiten zur Datenbank die CREATE SPATIAL UNIT OF MEASURE-Anweisung.

TYPE-Klausel Verwenden Sie die TYPE-Klausel, um zu steuern, wie das SRS Linien zwischen Punkten interpretiert. Bei geografischen räumlichen Bezugssystemen kann die TYPE-Klausel entweder ROUND EARTH (Standardwert) oder PLANAR lauten. Das ROUND EARTH-Modell interpretiert Linien zwischen Punkten als große elliptische Bögen. Bei zwei gegebenen Punkten auf der Erdoberfläche wird eine Ebene ausgewählt, die beide Punkte und den Mittelpunkt der Erddarstellung schneidet. Diese Ebene schneidet die Erde und die Linie zwischen den beiden Punkten ist der kürzeste Abstand entlang diesem Querschnitt.

Bei Punkten, die einander direkt gegenüberliegen, gibt es keine einzelne eindeutige Ebene, die beide Punkte und den Mittelpunkt der Erddarstellung schneidet. Liniensegmente, die diese antipodischen Punkte verbinden, sind nicht gültig und führen im ROUND EARTH-Modell zu einem Fehler.

Das ROUND EARTH-Modell behandelt die Erddarstellung als sphäroid und wählt Linien, die der Erdkrümmung folgen. In einigen Fällen kann es notwendig sein, ein planares Modell zu verwenden, in dem eine Linie zwischen zwei Punkten als gerade Linie in der sphärischen Projektion interpretiert wird, wobei $x=\text{long}$, $y=\text{lat}$.

Im folgenden Beispiel zeigt die blaue Linie die Linieninterpretation im ROUND EARTH-Modell und die rote Linie das entsprechende PLANAR-Modell.



Das PLANAR-Modell kann für eine Angleichung an die Interpretation durch andere Produkte verwendet werden. Außerdem kann das PLANAR-Modell hilfreich sein, weil es einige Einschränkungen für Methoden gibt, die im ROUND EARTH-Modell nicht unterstützt werden (z.B. ST_Area, ST_ConvexHull), und einige Methoden nur teilweise unterstützt werden (z.B. ST_Distance nur zwischen Punktgeometrien). Geometrien, die auf einer Kreisbogenfolge basieren, werden in räumlichen Bezugssystemen mit gewölbter Erddarstellung nicht unterstützt.

Bei nicht geografischen räumlichen Bezugssystemen muss der Typ PLANAR sein. (Dies ist auch der Standardwert, wenn die TYPE-Klausel nicht angegeben ist und die DEFINITION-Klausel entweder nicht angegeben ist oder eine nicht geografische Definition verwendet.)

COORDINATE-Klausel Mit dieser Klausel geben Sie die Grenzen für die Dimensionen des räumlichen Bezugssystems an. *coordinate-name* ist der Name des von dem räumlichen Bezugssystem verwendeten Koordinatensystems. Bei nicht geografischen Koordinatensystemen kann *coordinate-name* x, y oder m sein. Bei geografischen Koordinatensystemen kann *coordinate-name* LATITUDE, LONGITUDE, z oder m sein.

Geben Sie UNBOUNDED an, um keine Grenzen für die Dimensionen zu setzen. Verwenden Sie die BETWEEN-Klausel, um Unter- und Obergrenze zu setzen.

Den Koordinaten X und Y müssen Grenzen zugeordnet sein. Bei geografischen räumlichen Bezugssystemen sind standardmäßig die LONGITUDE-Koordinate zwischen -180 und 180 Grad und die LATITUDE-Koordinate zwischen -90 und 90 Grad begrenzt, es sei denn, diese Einstellungen werden durch die COORDINATE-Klausel aufgehoben. Für nicht geografische räumliche Bezugssysteme muss die CREATE-Anweisung Grenzen für die Koordinaten X und Y angeben.

LATITUDE und LONGITUDE werden bei geografischen Koordinatensystemen verwendet. Die Grenzen für LATITUDE und LONGITUDE entsprechen standardmäßig der gesamten Erddarstellung, wenn nichts anderes angegeben ist.

ELLIPSOID-Klausel Verwenden Sie die ELLIPSOID-Klausel, um die Werte anzugeben, die bei räumlichen Bezugssystemen vom Typ ROUND EARTH für die Erddarstellung als Ellipsoid verwendet werden sollen. Wenn die DEFINITION-Klausel vorliegt, kann sie eine Ellipsoiddefinition angeben. Wenn die ELLIPSOID-Klausel angegeben ist, hebt sie diesen Standard-Ellipsoid auf.

Die Erde ist keine perfekte Kugel, weil die Erdrotation eine Abplattung verursacht, sodass der Abstand vom Erdmittelpunkt zum Nord- oder Südpol kleiner ist als der Abstand vom Mittelpunkt zum Äquator. Aus diesem Grund wird die Erde als Ellipsoid modelliert, mit unterschiedlichen Werten für die große Halbachse (Abstand vom Mittelpunkt zum Äquator) und die kleine Halbachse (Abstand vom Mittelpunkt zum Pol). Die gängigste Methode besteht darin, ein Ellipsoid mit großer Halbachse und inverser Abplattung zu definieren, aber es kann auch unter Verwendung der kleinen Halbachse angegeben werden. (Dies ist beispielsweise erforderlich, wenn eine perfekte Kugel als angenäherte Erddarstellung verwendet wird.) Große und kleine Halbachse werden in den linearen Einheiten des räumlichen Bezugssystems definiert und die inverse Abplattung (1/f) ist ein Verhältnis:

$$1/f = (\text{semi-major-axis}) / (\text{semi-major-axis} - \text{semi-minor-axis})$$

SQL Anywhere verwendet die Ellipsoiddefinition beim Berechnen von Abständen in geografischen räumlichen Bezugssystemen.

Das Ellipsoid muss für geografische räumliche Bezugssysteme definiert werden (entweder in der DEFINITION-Klausel oder in der ELLIPSOID-Klausel) und darf für nicht geografische räumliche Bezugssysteme nicht angegeben werden.

SNAP TO GRID-Klausel Verwenden Sie bei räumlichen Bezugssystemen mit dem Modell "plane Erde" (PLANAR) die SNAP TO GRID-Klausel, um die Größe des Rasters zu definieren, das von SQL Anywhere beim Ausführen von Berechnungen verwendet wird. Standardmäßig wählt SQL Anywhere die Rastergröße so, dass an allen Punkten innerhalb der räumlichen Grenzen für X und Y 12 signifikante Stellen gespeichert werden können. Wenn beispielsweise ein räumliches Bezugssystem X zwischen -180 und 180 und Y zwischen -90 und 90 begrenzt, wird eine Rastergröße von 0.000000001 (1E-9) ausgewählt.

grid-size muss so groß sein, dass am Raster ausgerichtete Punkte überall innerhalb der räumlichen Grenzen mit derselben Gesamtstellenzahl dargestellt werden können. Wenn die *grid-size* zu klein ist, gibt der Server eine Fehlermeldung aus.

Wenn dieser Wert 0 ist, wird keine Ausrichtung am Raster durchgeführt.

Bei räumlichen Bezugssystemen mit dem Modell "gewölbte Erde" (ROUND EARTH) muss SNAP TO GRID auf 0 gesetzt sein.

Geben Sie SNAP TO GRID DEFAULT an, um die Rastergröße auf den Standardwert zu setzen, den der Datenbankserver verwenden würde.

TOLERANCE-Klausel Verwenden Sie bei räumlichen Bezugssystemen mit dem Modell "plane Erde" (PLANAR) die TOLERANCE-Klausel, um die Genauigkeit anzugeben, die beim Vergleichen von Punkten verwendet werden soll. Wenn der Abstand zwischen zwei Punkten kleiner ist als *tolerance-distance*, gelten die beiden Punkte als gleich. Mit der Einstellung *tolerance-distance* können Sie die Toleranz für Ungenauigkeiten in den Eingabedaten oder eine eingeschränkte interne Genauigkeit steuern. Standardmäßig wird der *tolerance-distance* auf denselben Wert gesetzt wie die *grid-size*.

Wenn dieser Wert 0 ist, müssen zwei Punkte genau übereinstimmen, um als gleich zu gelten.

Bei räumlichen Bezugssystemen mit dem Modell "gewölbte Erde" muss TOLERANCE auf 0 gesetzt sein.

POLYGON FORMAT-Klausel Intern interpretiert SQL Anywhere Polygone anhand der Ausrichtung der Ringe. Wenn Sie einem Ring in der Reihenfolge der definierten Punkte folgen, befindet sich die Innenseite des Polygons auf der linken Seite des Rings. Für räumliche Bezugssysteme mit planer und gewölbter Erddarstellung gelten dieselben Regeln.

Die von SQL Anywhere verwendete Interpretation ist gebräuchlich, aber nicht universell. Bei einigen Produkten wird genau die entgegengesetzte Ausrichtung verwendet und bei einigen Produkten wird zum Interpretieren von Polygonen nicht auf die Ringausrichtung zurückgegriffen. Die POLYGON FORMAT-Klausel kann bei Bedarf verwendet werden, um eine zu den Eingabedaten passende Polygoninterpretation auszuwählen. Folgende Werte werden unterstützt:

- **'CounterClockwise'** Die Eingabe folgt der internen Interpretation von SQL Anywhere: Die Innenseite des Polygons befindet sich auf der linken Seite, wenn Sie der Ringausrichtung folgen.
- **'Clockwise'** Die Eingabe folgt dem Gegenteil des Ansatzes von SQL Anywhere: Die Innenseite des Polygons befindet sich auf der rechten Seite, wenn Sie der Ringausrichtung folgen.
- **'EvenOdd'** EvenOdd ist das Standardformat. Bei EvenOdd wird die Ausrichtung der Ringe ignoriert und die Innenseite des Polygons wird anhand der Verschachtelung der Ringe bestimmt, wobei der äußere Ring der größte ist und die inneren Ringe kleinere Ringe innerhalb dieses Rings darstellen. Ein Strahl verläuft von einem Punkt innerhalb der Ringe nach außen und schneidet alle Ringe. Wenn die Anzahl der geschnittenen Ringe eine gerade Zahl ist, handelt es sich um einen äußeren Ring. Wenn die Zahl ungerade ist, handelt es sich um einen inneren Ring.

STORAGE FORMAT-Klausel Beim Einfügen räumlicher Daten in die Datenbank aus einem externen Format (z.B. WKT oder WKB) normalisiert der Datenbankserver die Daten, um Performance und Semantik bei Vorgängen mit räumlichen Daten zu verbessern. Die normalisierte Darstellung unterscheidet sich möglicherweise von der ursprünglichen Darstellung (z.B. in der Ausrichtung von Polygonringen oder der in einzelnen Koordinaten gespeicherten Genauigkeit). Obwohl die räumliche Gleichheit nach der Normalisierung erhalten bleibt, sind einige Merkmale der ursprünglichen Eingabe möglicherweise nicht reproduzierbar, z.B. Genauigkeit und Ringausrichtung. In manchen Fällen kann es sinnvoll sein, die ursprüngliche Darstellung entweder ausschließlich oder zusätzlich zur normalisierten Darstellung zu speichern.

Um zu steuern, was gespeichert wird, geben Sie die STORAGE FORMAT-Klausel an, gefolgt von einem der folgenden Werte:

- **'Internal'** SQL Anywhere speichert nur die normalisierte Darstellung. Geben Sie diesen Wert an, wenn die ursprünglichen Eingabemerkmale nicht reproduziert werden müssen. Dies ist die Standardeinstellung für planare räumliche Bezugssysteme (TYPE PLANAR).

Hinweis

Wenn Sie MobiLink zum Synchronisieren Ihrer räumlichen Daten verwenden, sollten Sie **Mixed** angeben. MobiLink testet die Gleichheit während der Synchronisation, was die Daten in ihrem ursprünglichen Format erfordert.

- **'Original'** SQL Anywhere speichert nur die ursprüngliche Darstellung. Die ursprünglichen Eingabemerkmale können reproduziert werden, aber bei allen Vorgängen mit den gespeicherten Werten müssen die Normalisierungsschritte wiederholt werden, wodurch Vorgänge mit den Daten möglicherweise verlangsamt werden.
- **'Mixed'** SQL Anywhere speichert die interne Version, und wenn diese von der ursprünglichen Version abweicht, speichert SQL Anywhere auch die Originalversion. Durch das Speichern beider Versionen können die ursprünglichen Darstellungsmerkmale reproduziert werden und bei Vorgängen mit den gespeicherten Werten müssen nicht die Normalisierungsschritte wiederholt werden. Die Speicheranforderungen werden jedoch möglicherweise signifikant erhöht, da potenziell für jede Geometrie zwei Darstellungen gespeichert werden.

"Mixed" ist das Standardformat für räumliche Bezugssysteme mit dem Modell "gewölbte Erde" (TYPE ROUND EARTH).

Bemerkungen

Bei einem geografischen räumlichen Bezugssystem können Sie eine lineare Maßeinheit (LINEAR) *und* eine Winkelmaßeinheit (ANGULAR) angeben. Andernfalls geben Sie für nicht geografische Systeme nur eine lineare Maßeinheit an. Die lineare Maßeinheit wird für die Berechnung des Abstands zwischen Punkten und Flächen verwendet. Die Winkelmaßeinheit gibt an, wie Breiten- und Längengrad interpretiert werden. Sie ist bei projizierten Koordinatensystemen NULL und bei geografischen Koordinatensystemen nicht NULL.

Alle von Vorgängen zurückgegebenen abgeleiteten Geometrien werden normalisiert.

Bei der Arbeit mit Daten, die mit einer nicht-SQL Anywhere-Datenbank synchronisiert werden, sollte STORAGE FORMAT entweder auf 'Original' oder auf 'Mixed' gesetzt werden, damit die ursprünglichen Merkmale der Daten erhalten bleiben.

Privilegien

Sie müssen das MANAGE ANY SPATIAL OBJECT-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „sa_install_feature-Systemprozedur“ auf Seite 1245
- „CREATE SPATIAL UNIT OF MEASURE-Anweisung“ auf Seite 727
- „Konsolidierte Ansicht ST_UNITS_OF_MEASURE“ auf Seite 1518
- „Konsolidierte Ansicht ST_SPATIAL_REFERENCE_SYSTEMS“ auf Seite 1515
- „ALTER SPATIAL REFERENCE SYSTEM-Anweisung“ auf Seite 506
- „Räumliche Bezugssysteme (SRS) und räumliche Referenz-IDs (SRID)“ [*SQL Anywhere Server - Unterstützung für räumliche Daten*]
- „Räumliche Daten“ [*SQL Anywhere Server - Unterstützung für räumliche Daten*]
- „sa_install_feature-Systemprozedur“ auf Seite 1245

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird ein räumliches Bezugssystem mit dem Namen mySpatialRS erstellt:

```
CREATE SPATIAL REFERENCE SYSTEM "mySpatialRS"
IDENTIFIED BY 1000026980
LINEAR UNIT OF MEASURE "metre"
TYPE PLANAR
COORDINATE X BETWEEN 171266.736269555 AND 831044.757769222
COORDINATE Y BETWEEN 524881.608973277 AND 691571.125115319
DEFINITION 'PROJCS["NAD83 / Kentucky South",
GEOGCS["NAD83",
DATUM["North_American_Datum_1983",
SPHEROID["GRS 1980",6378137,298.257222101,AUTHORITY["EPSG","7019"]],
AUTHORITY["EPSG","6269"]],
PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],
UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],
AUTHORITY["EPSG","4269"]],
UNIT["metre",1,AUTHORITY["EPSG","9001"]],
PROJECTION["Lambert_Conformal_Conic_2SP"],
PARAMETER["standard_parallel_1",37.93333333333333],
PARAMETER["standard_parallel_2",36.73333333333333],
PARAMETER["latitude_of_origin",36.33333333333334],
PARAMETER["central_meridian",-85.75],
PARAMETER["false_easting",500000],
PARAMETER["false_northing",500000],
AUTHORITY["EPSG","26980"]],
AXIS["X",EAST],
AXIS["Y",NORTH]]'
TRANSFORM DEFINITION '+proj=lcc
+lat_1=37.93333333333333+lat_2=36.73333333333333+lat_0=36.33333333333334+lon_
0=-85.75+x_0=500000+y_0=500000+ellps=GRS80+datum=NAD83+units=m+no_defs';
```

CREATE SPATIAL UNIT OF MEASURE-Anweisung

Erstellt oder ersetzt eine räumliche Maßeinheit.

Syntax

```
CREATE [ OR REPLACE ] SPATIAL UNIT OF MEASURE identifier
TYPE { LINEAR | ANGULAR }
[ CONVERT USING number ]
```

Parameter

OR REPLACE-Klausel Das Einbeziehen von OR REPLACE erstellt eine neue räumliche Maßeinheit oder ersetzt eine vorhandene räumliche Maßeinheit mit demselben Namen. Diese Klausel lässt vorhandene Privilegien unberührt. Ein Fehler wird zurückgegeben, wenn Sie eine räumliche Maßeinheit ersetzen, die gerade verwendet wird.

TYPE-Klausel Definiert, ob die Maßeinheit für Winkel (ANGULAR) oder Abstände (LINEAR) verwendet wird.

CONVERT USING Der Konvertierungsfaktor für die räumliche Einheit relativ zur Basiseinheit. Bei linearen Einheiten ist die Basiseinheit "METRE". Bei Winkleinheiten ist die Basiseinheit "RADIAN".

Bemerkungen

Mit der CONVERT USING-Klausel wird definiert, wie ein Messwert in der definierten Maßeinheit in die Basismaßeinheit (Bogenmaß oder Meter) konvertiert wird. Der Messwert wird mit dem angegebenen Konvertierungsfaktor multipliziert, was einen Wert in der Basismaßeinheit liefert. Ein Messwert von 512 Millimetern würde beispielsweise mit einem Konvertierungsfaktor von 0,001 multipliziert und liefert so einen Wert von 0,512 Metern.

Räumliche Bezugssysteme enthalten stets eine lineare Maßeinheit, die bei der Berechnung von Abständen (ST_Distance oder ST_Length) oder Flächen verwendet werden soll. Wenn zum Beispiel die lineare Maßeinheit für ein räumliches Bezugssystem Kilometer ist, wird für Flächen die Einheit Quadratkilometer verwendet. In einigen Fällen akzeptieren räumliche Methoden einen optionalen Parameter, der die zu verwendende lineare Maßeinheit angibt. Wenn beispielsweise die lineare Maßeinheit für ein räumliches Bezugssystem Meilen ist, können Sie den Abstand zwischen zwei Geometrien in Metern abrufen, indem Sie den optionalen Parameter "metre" angeben.

Bei projizierten Koordinatensystemen werden die Koordinaten X und Y in der linearen Einheit des räumlichen Bezugssystems angegeben. Bei geografischen Koordinatensystemen werden Breiten- und Längengrad in der mit dem räumlichen Bezugssystem verknüpften Winkelmaßeinheit angegeben. In vielen Fällen ist diese Winkelmaßeinheit Grad, aber jede gültige Winkelmaßeinheit kann verwendet werden.

Mit der sa_install_feature-Systemprozedur können Sie der Datenbank vordefinierte Maßeinheiten hinzufügen.

Privilegien

Sie müssen das MANAGE ANY SPATIAL OBJECT-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „sa_install_feature-Systemprozedur“ auf Seite 1245
- „DROP SPATIAL UNIT OF MEASURE-Anweisung“ auf Seite 825
- „Räumliche Daten“ [*SQL Anywhere Server - Unterstützung für räumliche Daten*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird eine räumliche Maßeinheit mit dem Namen "Test" erstellt.

```
CREATE SPATIAL UNIT OF MEASURE Test
TYPE LINEAR
CONVERT USING 15;
```

CREATE STATISTICS-Anweisung

Mit dieser Anweisung werden die vom Optimierer verwendeten Spaltenstatistiken neu erstellt und in der ISYSCOLSTAT-Systemtabelle gespeichert.

Syntax

CREATE STATISTICS *object-name* [(*column-list*)]

object-name :

table-name | *materialized-view-name* | *temp-table-name*

Bemerkungen

Die CREATE STATISTICS-Anweisung erstellt die Spaltenstatistiken neu, die SQL Anywhere verwendet, um Datenbankabfragen zu optimieren. Sie kann für Basistabellen, materialisierte Ansichten, lokale temporäre Tabellen und globale temporäre Tabellen durchgeführt werden. Für Proxytabellen können keine Statistiken erstellt werden. Spaltenstatistiken enthalten Histogramme, die für die angegebenen Spalten die Datenverteilung in der Datenbank repräsentieren. Standardmäßig werden Spaltenstatistiken für Tabellen mit fünf oder mehr Zeilen automatisch erstellt.

In Ausnahmefällen, wenn Ihre Datenbankabfragen stark variieren und wenn die Datenverteilung nicht gleichmäßig ist bzw. die Daten häufig geändert werden, können Sie die Performance verbessern, indem Sie die CREATE STATISTICS-Anweisung mit einer Tabelle bzw. Spalte ausführen.

Während der Ausführung aktualisiert die CREATE STATISTICS-Anweisung bestehende Spaltenstatistiken unabhängig von der Größe der Tabelle, außer die Tabelle ist leer. In diesem Fall wird keine Aktion durchgeführt. Wenn es Spaltenstatistiken für eine leere Tabelle gibt, werden sie durch die CREATE STATISTICS-Anweisung nicht geändert. Um Spaltenstatistiken für eine leere Tabelle zu löschen, führen Sie die DROP STATISTICS-Anweisung aus.

Bei der Ausführung von CREATE STATISTICS wird ein kompletter Scan der Tabelle durchgeführt. Aus diesem Grund sollten Sie CREATE STATISTICS-Anweisungen mit Bedacht verwenden.

Wenn Sie Statistiken löschen, wird empfohlen, sie mit der CREATE STATISTICS-Anweisung neu zu erstellen. Ohne Statistiken erzeugt der Optimierer möglicherweise unwirksame Datenzugriffspläne und verursacht damit eine schlechte Datenbank-Performance.

Privilegien

Sie müssen entweder Eigentümer der Tabelle sein oder das MANAGE ANY STATISTICS-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Ausführungspläne können sich ändern.

Siehe auch

- „Optimiererschätzungen und -statistiken“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „DROP STATISTICS-Anweisung“ auf Seite 827
- „LOAD TABLE-Anweisung“ auf Seite 931
- „SYSCOLSTAT-Systemansicht“ auf Seite 1441
- „Histogramm-Dienstprogramm (dbhist)“ [*SQL Anywhere Server - Datenbankadministration*]
- „sa_get_histogram-Systemprozedur“ auf Seite 1225

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung aktualisiert die Spaltenstatistiken für die Spalte ProductID in der Tabelle SalesOrderItems:

```
CREATE STATISTICS GROUP O.SalesOrderItems ( ProductID );
```

CREATE SUBSCRIPTION-Anweisung [SQL Remote]

Erstellt eine Subskription einer Publikation für einen Benutzer.

Syntax

```
CREATE SUBSCRIPTION  
TO publication-name [ ( subscription-value ) ]  
FOR subscriber-id
```

publication-name : *identifier*

subscription-value : *string*

subscriber-id : *string*

Parameter

publication-name Der Name der Publikation, die für den Benutzer subskribiert ist. Darin kann der Eigentümer der Publikation enthalten sein.

subscription-value Eine Zeichenfolge, die mit dem Subskriptionsausdruck der Publikation verglichen wird. Der Subskribent erhält alle Zeilen, für die der Subskriptionsausdruck mit dem Subskriptionswert übereinstimmt.

subscriber-id Die Benutzer-ID des Subskribenten für die Publikation. Wenn Sie eine Subskription für einen entfernten Benutzer erstellen, muss dem entfernten Benutzer in der konsolidierten Datenbank das REMOTE-Privileg erteilt worden sein. Wenn Sie eine Subskription für den konsolidierten Benutzer erstellen, muss diesem Benutzer in der entfernten Datenbank das CONSOLIDATED-Privileg erteilt worden sein.

Bemerkungen

In SQL Remote besteht zwischen Publikationen und Subskriptionen eine Zwei-Weg-Beziehung. Wenn Sie für einen entfernten Benutzer eine Subskription zu einer Publikation in einer konsolidierten Datenbank erstellen, sollten Sie auch für den konsolidierten Benutzer eine Subskription einer Publikation in der entfernten Datenbank erstellen. Standardmäßig erteilen das Extraktionsdienstprogramm (dbxtract) und der **Assistent zum Extrahieren einer Datenbank** Benutzern in den entfernten Datenbanken das entsprechende PUBLISH- und CONSOLIDATE-Privileg.

Wenn *subscription-value* angegeben ist, wird der entsprechende Wert mit jedem SUBSCRIBE BY-Ausdruck in der Publikation abgeglichen. Der Subskribent erhält alle Zeilen, für die der Wert des Ausdrucks mit der gelieferten Zeichenfolge übereinstimmt.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „DROP SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 828
- „GRANT REMOTE-Anweisung [SQLRemote]“ auf Seite 900
- „GRANT CONSOLIDATE-Anweisung [SQL Remote]“ auf Seite 889
- „SYNCHRONIZE SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 1086
- „START SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 1069
- „GRANT PUBLISH-Anweisung [SQL Remote]“ auf Seite 894
- „SYSSUBSCRIPTION-Systemansicht“ auf Seite 1489
- „Subskriptionen“ [SQL Remote]
- „Primärschlüsselpool replizieren“ [SQL Remote]
- „Nur einige Zeilen einer Tabelle publizieren“ [SQL Remote]
- „Publizieren von nur einigen Zeilen mit der SUBSCRIBE BY-Klausel“ [SQL Remote]
- „Nur einige Zeilen mittels einer WHERE-Klausel publizieren“ [SQL Remote]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung erstellt eine Subskription für den Benutzer p_chin für die Publikation pub_sales. Der Subskribent empfängt alle Zeilen, bei denen der Subskriptionsausdruck den Wert Eastern hat.

```
CREATE SUBSCRIPTION
TO pub_sales ( 'Eastern' )
FOR p_chin;
```

Die folgende Anweisung erstellt eine Subskription für den Benutzernamen SamS für die Publikation CustomerPub, die mit einer WHERE-Klausel erstellt wurde:

```
CREATE SUBSCRIPTION
TO CustomerPub
FOR Sam_Singer;
```

Die folgende Anweisung erstellt eine Subskription für den Benutzernamen SamS für die Publikation PubOrders, die mit dem Subskriptionsausdruck SalesRepresentative definiert ist, wobei die Zeilen für die eigenen Verkäufe von Samuel Singer angefordert werden:

```
CREATE SUBSCRIPTION
  TO PubOrders ( '856' )
  FOR Sam_Singer;
```

CREATE SYNCHRONIZATION PROFILE-Anweisung [MobiLink]

Erstellt ein SQL Anywhere-Synchronisationsprofil.

Syntax

```
CREATE [ OR REPLACE ] SYNCHRONIZATION PROFILE name string
```

Parameter

OR REPLACE-Klausel Durch Angeben von CREATE OR REPLACE SYNCHRONIZATION PROFILE wird die Definition des benannten Synchronisationsprofils ersetzt, falls sie bereits vorhanden ist.

name Der Name des zu erstellenden Synchronisationsprofils. Jedes Profil muss einen eindeutigen Namen haben.

string Geben Sie eine gültige Optionszeichenfolge wie unten beschrieben ein. Optionszeichenfolgen werden als durch Semikola getrennte Listen von Elementen in der Form *option-name=option-value* angegeben. Zum Beispiel `subscription=sl;verbosity=high`.

Bemerkungen

Synchronisationsprofile sind benannte Sammlungen von Synchronisationsoptionen, die zur Steuerung der Synchronisation verwendet werden können. Eine Liste der Synchronisationsprofiloptionen, die von dbmlsync unterstützt werden, finden Sie unter „MobiLink-Synchronisationsprofile“ [[MobiLink - Clientadministration](#)].

Bei Optionen, die mit einem Booleschen Wert eingegeben werden, entspricht die Einstellung des Werts auf TRUE der Angabe der entsprechenden Option in der Befehlszeile.

Folgende Werte können für die Angabe von TRUE verwendet werden: TRUE, ON, 1, YES.

Folgende Werte können für die Angabe von FALSE verwendet werden: FALSE, OFF, 0, NO.

Verwenden Sie beim Festlegen von erweiterten Optionen die folgende Syntax:

```
CREATE SYNCHRONIZATION PROFILE myprofile
  's=mysub;e={ctp=tcip;adr='host=localhost;port=2439'}'
```

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „ALTER SYNCHRONIZATION PROFILE-Anweisung [MobiLink]“ auf Seite 511
- „DROP SYNCHRONIZATION PROFILE-Anweisung [MobiLink]“ auf Seite 829
- „SYNCHRONIZE-Anweisung [MobiLink]“ auf Seite 1081

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

CREATE SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink]

Erstellt in einer entfernten SQL Anywhere-Datenbank eine Subskription zwischen einem MobiLink-Benutzer und einer Publikation.

Syntax

```

CREATE SYNCHRONIZATION SUBSCRIPTION[ subscription-name ]
TO publication-name
[ FOR ml-username, ... ]
[ TYPE network-protocol ]
[ ADDRESS protocol-options ]
[ OPTION option=value, ... ]
[ SCRIPT VERSION script-version ]

```

subscription-name : *identifier*

ml-username : *identifier*

network-protocol : **http** | **https** | **tls** | **tcpip**

protocol-options : *string*

value : *string* | *integer*

script-version : *string*

Parameter

subscription-name Ein eindeutiger Name, den Sie zum Identifizieren dieser Subskription verwenden können. Es wird dringend empfohlen, dass Sie alle Ihre Subskriptionen benennen.

TO-Klausel Diese Klausel gibt den Namen einer Publikation an.

FOR-Klausel Diese Klausel gibt einen oder mehrere MobiLink-Benutzernamen an. Wenn Sie mehr als einen Benutzernamen angeben, wird für jeden Benutzer eine separate Subskription erstellt. Wenn Sie einen Subskriptionsnamen angeben, kann nur ein MobiLink-Benutzername angegeben werden.

ml-username ist ein Benutzer, der für die Synchronisation mit dem MobiLink-Server berechtigt ist.

Lassen Sie die FOR-Klausel weg, wenn Sie den Protokolltyp, Protokolloptionen und erweiterte Optionen für eine Publikation festlegen möchten. Wenn die FOR-Klausel weggelassen wird, können Sie weder einen Subskriptionsnamen angeben noch die SCRIPT VERSION-Klausel verwenden.

TYPE-Klausel Mit dieser Klausel wird das Netzwerkprotokoll für die Synchronisation angegeben. Das Standardprotokoll ist tcpip.

ADDRESS-Klausel Die ADDRESS-Klausel gibt Netzwerkprotokolloptionen wie den Standort des MobiLink-Servers an. Mehrere Optionen müssen durch Semikola getrennt werden.

OPTION-Klausel Mit dieser Klausel können Sie erweiterte Optionen für die Subskription festlegen. Wenn keine FOR-Klausel angegeben ist, werden die erweiterten Optionen zu den Standardeinstellungen der Publikation.

SCRIPT VERSION-Klausel Mit dieser Klausel wird die Skriptversion für die Synchronisation angegeben. In der Regel müssen Sie für jede Schemaänderung, die Sie implementieren, eine neue Skriptversion angeben.

Sie können nicht mit der SCRIPT VERSION-Klausel arbeiten, wenn keine FOR-Klausel vorhanden ist.

Bemerkungen

Wenn *subscription-name* nicht angegeben ist, wird ein eindeutiger Name generiert. Der generierte Subskriptionsname stimmt mit dem Publikationsnamen überein, sofern dieser eindeutig ist. Andernfalls wird ein eindeutiger Name gebildet, indem am Ende des Publikationsnamens eine Nummer hinzugefügt wird, z.B. pub001, pub002 usw.

Die Werte für *network-protocol*, *protocol-options* und *option* können an mehreren Stellen festgelegt werden.

Diese Anweisung bewirkt, dass Optionen und andere Informationen in der SQL Anywhere-Systemtabelle ISYSSYNC gespeichert werden. Jeder Benutzer mit Privilegien zum Anzeigen von Daten in der SYSSYNC-Systemansicht kann die Informationen anzeigen, zu denen auch Kennwörter und Verschlüsselungszertifikate gehören können. Um dieses potenzielle Sicherheitsproblem zu umgehen, können Sie die Informationen in der dbmlsync-Befehlszeile angeben.

Sie benötigen exklusiven Zugriff auf alle in der Publikation referenzierten Tabellen, um die Anweisung ausführen zu können.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink]“ auf Seite 512
- „DROP SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink]“ auf Seite 830
- SQL Anywhere MobiLink-Clients: „Erstellen von Synchronisationssubskriptionen“ [*MobiLink - Clientadministration*]
- UltraLite MobiLink-Clients: „UltraLite-Clientsynchronisationsplanung“ [*UltraLite - Datenbankverwaltung*]
- „MobiLink-Benutzer“ [*MobiLink - Clientadministration*]
- „Netzwerkprotokolloptionen des MobiLink-Clients“ [*MobiLink - Clientadministration*]
- „Erweiterte Optionen von MobiLink SQL Anywhere-Clients“ [*MobiLink - Clientadministration*]
- „Erweiterte Option CommunicationType (ctp)“ [*MobiLink - Clientadministration*]
- „SYSSYNC-Systemansicht“ auf Seite 1489
- Prioritätenfolge [*MobiLink - Clientadministration*]
- „Skriptversionen“ [*MobiLink - Serveradministration*]
- „dbmlsync-Syntax“ [*MobiLink - Clientadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Im folgenden Beispiel wird die Subskription sales zwischen dem MobiLink-Benutzer SSinger und der Publikation sales_publication erstellt, deren Eigentümer user ist. Beim Synchronisieren der Subskription wird die Skriptversion sales_v1 verwendet und Tabellen werden im exklusiven Modus gesperrt:

```
CREATE SYNCHRONIZATION SUBSCRIPTION sales
TO user.sales_publication
FOR SSinger
OPTION locktables='exclusive'
SCRIPT VERSION 'sales_v1'
```

Im folgenden Beispiel wird die FOR-Klausel weggelassen und die Einstellungen für die Publikation sales_publication werden gespeichert:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
TO user.sales_publication
ADDRESS 'host=test.internal;port=2439;
OPTION locktables=exclusive';
```

CREATE SYNCHRONIZATION USER-Anweisung [MobiLink]

Erstellt einen MobiLink-Benutzer in einer entfernten SQL Anywhere-Datenbank.

Syntax

```
CREATE SYNCHRONIZATION USER ml-username
[ TYPE network-protocol ]
[ ADDRESS protocol-options ]
[ OPTION option=value, ... ]
```

ml-username : *identifizier*

network-protocol :

tcpip
| **http**
| **https**
| **tls**

protocol-options : *string*

value : *string* | *integer*

Parameter

ml_username Name, der einen MobiLink-Benutzer bezeichnet

TYPE-Klausel Mit dieser Klausel wird das Netzwerkprotokoll für die Synchronisation angegeben. Das Standardprotokoll ist tcpip.

ADDRESS-Klausel Diese Klausel gibt *protocol-options* in der Form *Schlüsselwort=Wert* an, durch Semikola getrennt. Welche Einstellungen Sie angeben, hängt von dem verwendeten Kommunikationsprotokoll ab (TCP/IP, TLS, HTTP oder HTTPS).

OPTION-Klausel Die OPTION-Klausel ermöglicht es Ihnen, erweiterte Optionen in der Form *Option=Wert* in einer durch Kommas getrennten Auflistung anzugeben.

Die Werte für die einzelnen Optionen können keine Gleichheitszeichen oder Semikola enthalten. Der Datenbankserver akzeptiert alle Optionen, die Sie angeben, ohne sie auf ihre Gültigkeit zu prüfen. Wenn Sie also eine Option falsch schreiben oder einen ungültigen Wert eingeben, erscheint erst eine Fehlermeldung, wenn Sie den Befehl dbmsync zum Starten der Synchronisation ausführen.

Für einen Synchronisationsbenutzer festgelegte Optionen können in einzelnen Subskriptionen oder in der dbmsync-Befehlszeile aufgehoben werden.

Die Parameter *network-protocol*, *protocol-options* und *options* können an mehreren Stellen festgelegt werden.

Diese Anweisung bewirkt, dass Optionen und andere Informationen in der SQL Anywhere-Systemtabelle ISYSSYNC gespeichert werden. Jeder Benutzer mit den richtigen Privilegien zum Anzeigen der SYSSYNC-Systemansicht kann die Informationen anzeigen, zu denen auch Kennwörter und Verschlüsselungszertifikate gehören können. Um dieses potenzielle Sicherheitsproblem zu umgehen, können Sie die Informationen in der dbmsync-Befehlszeile angeben.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „ALTER SYNCHRONIZATION USER-Anweisung [MobiLink]“ auf Seite 515
- „DROP SYNCHRONIZATION USER-Anweisung [MobiLink]“ auf Seite 831
- „Erweiterte Option CommunicationType (ctp)“ [MobiLink - Clientadministration]
- „Verschlüsselung der MobiLink-Client/Server-Kommunikation“ [SQL Anywhere Server - Datenbankadministration]
- „SYSSYNC-Systemansicht“ auf Seite 1489
- „dbmlsync-Syntax“ [MobiLink - Clientadministration]
- „MobiLink-Benutzer“ [MobiLink - Clientadministration]
- „Netzwerkprotokolloptionen des MobiLink-Clients“ [MobiLink - Clientadministration]
- „Erweiterte Optionen von MobiLink SQL Anywhere-Clients“ [MobiLink - Clientadministration]
- Prioritätenfolge [MobiLink - Clientadministration]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Mit dem folgenden Beispiel wird ein MobiLink-Benutzer mit dem Namen 'SSinger' erstellt, der über TCP/IP mit dem Servercomputer 'mlserver.mycompany.com' und mit dem Kennwort 'Sam' synchronisiert wird. Die Verwendung eines Kennwortes in der Benutzerdefinition ist *nicht* sicher.

```
CREATE SYNCHRONIZATION USER SSinger
TYPE http
ADDRESS 'host=mlserver.mycompany.com'
OPTION MobiLinkPwd='Sam';
```

CREATE TABLE-Anweisung

Erstellt eine neue Tabelle in der Datenbank und optional eine Tabelle auf einem Fremdserver.

Syntax

```
CREATE [ GLOBAL TEMPORARY ] TABLE [ IF NOT EXISTS ] [ owner. ] table-name
( { column-definition | table-constraint | pctfree }, ... )
[ { IN | ON } dbspace-name ]
[ ENCRYPTED ]
[ ON COMMIT { DELETE | PRESERVE } ROWS
  | NOT TRANSACTIONAL ]
[ AT location-string ]
[ SHARE BY ALL ]

column-definition :
column-name data-type
[ COMPRESSED ]
[ INLINE { inline-length | USE DEFAULT } ]
[ PREFIX { prefix-length | USE DEFAULT } ]
[ [ NO ] INDEX ]
[ [ NOT ] NULL ]
[ DEFAULT default-value | IDENTITY ]
[ column-constraint ... ]
```

default-value :
| *special-value*
| *string*
| *global-variable*
| [-] *number*
| (*constant-expression*)
| (*sequence-expression*)
| *built-in-function*(*constant-expression*)
AUTOINCREMENT
| **GLOBAL AUTOINCREMENT** [(*partition-size*)]

special-value :
CURRENT DATABASE
| **CURRENT DATE**
| **CURRENT TIME**
| [**CURRENT**] **TIMESTAMP**
| **CURRENT PUBLISHER**
| **CURRENT REMOTE USER**
| [**CURRENT**] **USER**
| [**CURRENT**] **UTC TIMESTAMP**
| **LAST USER**
| **NULL**

column-constraint :
| **CONSTRAINT** *constraint-name* {
| **UNIQUE** [**CLUSTERED**]
| **PRIMARY KEY** [**CLUSTERED**] [**ASC** | **DESC**]
| **REFERENCES** *table-name* [(*column-name*)]
| [**MATCH** [**UNIQUE**] { **SIMPLE** | **FULL** }]
| [*action-list*] [**CLUSTERED**]
| **CHECK** (*condition*)
| }
| **COMPUTE** (*expression*)

table-constraint :
| **CONSTRAINT** *constraint-name* {
| **UNIQUE** [**CLUSTERED**] (*column-name* [**ASC** | **DESC**], ...)
| **PRIMARY KEY** [**CLUSTERED**] (*column-name* [**ASC** | **DESC**], ...)
| **CHECK** (*condition*)
| *foreign-key-constraint*
| }

foreign-key-constraint :
| **NOT NULL** **FOREIGN KEY** [*role-name*]
| [(*column-name* [**ASC** | **DESC**], ...)]
| **REFERENCES** *table-name*
| [(*column-name*, ...)]
| [**MATCH** [**UNIQUE**] { **SIMPLE** | **FULL** }]
| [*action-list*] [**CHECK ON COMMIT**] [**CLUSTERED**] [**FOR OLAP WORKLOAD**]

action-list :
| **ON UPDATE** *action*]
| **ON DELETE** *action*]

action :
CASCADE
| **SET NULL**

SET DEFAULT RESTRICT

location-string :

remote-server-name.[*db-name*].[*owner*].*object-name*
| *remote-server-name*;*db-name*;*owner*;*object-name*

pctfree : **PCTFREE** *percent-free-space*

percent-free-space : *integer*

Parameter

IN-Klausel Verwenden Sie diese Klausel, um den DBSpace anzugeben, in dem sich die Basistabelle befindet. Wenn diese Klausel nicht angegeben ist, wird die materialisierte Ansicht in dem DBSpace erstellt, der in der Option `default_dbspace` angegeben wurde.

Temporäre Tabellen können nur im DBSpace "temporary" erstellt werden. Wenn Sie eine GLOBAL TEMPORARY-Tabelle erstellen und IN angeben, wird die Tabelle im DBSpace TEMPORARY erstellt. Wenn Sie einen benutzerdefinierten DBSpace angeben, wird ein Fehler zurückgegeben.

ENCRYPTED-Klausel Die Klausel ENCRYPTED gibt an, dass die Tabelle verschlüsselt werden soll. Wenn Sie vorhaben, Tabellen zu verschlüsseln, müssen Sie beim Erstellen einer Datenbank die Tabellenverschlüsselung aktivieren. Die Verschlüsselung erfolgt unter Verwendung des Chiffrierschlüssels und des Algorithmus, die bei der Datenbankerstellung festgelegt wurden.

ON COMMIT-Klausel Die ON COMMIT-Klausel ist nur für temporäre Tabellen zulässig. Standardmäßig werden die Zeilen einer temporären Tabelle bei COMMIT gelöscht. Wenn die SHARE BY ALL-Klausel angegeben ist, muss entweder ON COMMIT PRESERVE ROWS oder NOT TRANSACTIONAL angegeben werden.

NOT TRANSACTIONAL-Klausel Die NOT TRANSACTIONAL-Klausel ist zulässig, wenn Sie eine globale temporäre Tabelle erstellen. Eine mit NOT TRANSACTIONAL erstellte Tabelle wird weder durch COMMIT noch durch ROLLBACK verändert. Wenn die SHARE BY ALL-Klausel angegeben ist, muss entweder ON COMMIT PRESERVE ROWS oder NOT TRANSACTIONAL angegeben werden.

AT-Klausel Erstellt eine entfernte Tabelle auf einem anderen Server, der durch *location-string* angegeben wird, wie auch eine Proxytabelle in der aktuellen Datenbank, die der entfernten Tabelle zugeordnet ist. Die AT-Klausel unterstützt das Semikolon (;) als Feldbegrenzer in *location-string*. Wenn kein Semikolon vorhanden ist, gilt der Punkt als Feldbegrenzer. Mit dieser Syntax können Dateinamen oder Erweiterungen in den Datenbank- und Eigentümerfeldern verwendet werden.

Die folgende Anweisung ordnet beispielsweise die Tabelle "proxy_Customers" einer neuen Tabelle "Customers" in einer fiktiven Microsoft Access-Datenbank namens MyAccessDB zu:

```
CREATE TABLE proxy_Customers
(
    ID                INTEGER CONSTRAINT PRIMARY KEY,
    ClientName        TEXT,
    ClientAddress     TEXT,
    Telephone         TEXT
)
AT 'MyAccessDB;;;Customers';
```

Die Zeichenfolge in der AT-Klausel kann auch lokale oder globale Variablennamen in geschweiften Klammern enthalten (*{variable-name}*). Der SQL-Variablenname muss vom Typ CHAR, VARCHAR oder LONG VARCHAR sein. Eine AT-Klausel mit 'access;*{@myfile}*;ial' zeigt beispielsweise an, dass *@myfile* eine SQL-Variable ist und dass der aktuelle Inhalt der *@myfile*-Variablen beim Erstellen der Proxy-Tabelle ersetzt werden muss.

Die AT-Klausel wird unter Windows Mobile nicht unterstützt.

Fremdschlüsseldefinitionen werden in entfernten Tabellen ignoriert. Fremdschlüsseldefinitionen für lokale Tabellen, die entfernte Tabellen referenzieren, werden ebenfalls ignoriert.

Primärschlüsseldefinitionen werden an den Fremdserver geschickt, wenn der Datenbankservers Primärschlüssel unterstützt.

SHARE BY ALL-Klausel Verwenden Sie diese Klausel nur, wenn Sie globale temporäre Tabellen erstellen, um die Tabelle für alle Verbindungen zur Datenbank verfügbar zu machen. Wenn die SHARE BY ALL-Klausel angegeben ist, muss entweder ON COMMIT PRESERVE ROWS oder NOT TRANSACTIONAL angegeben werden.

IF NOT EXISTS-Klausel Existiert die angegebene Tabelle bereits, werden keine Änderungen durchgeführt und keine Fehlermeldung ausgegeben.

column-definition Definiert eine Spalte in der Tabelle. Nachfolgend werden die Bestandteile von Spaltendefinitionen aufgeführt.

- **column-name** Der Spaltenname ist ein Bezeichner. Mehrere Spalten in derselben Tabelle können nicht denselben Namen haben.
- **data-type** Der in der Spalte gespeicherte Datentyp.
- **COMPRESSED-Klausel** Komprimiert die Spalte.

INLINE- und PREFIX-Klausel Die INLINE-Klausel gibt in Byte die maximale BLOB-Größe an, die in einer Zeile gespeichert werden kann. BLOBs, die kleiner oder gleich dem in der INLINE-Klausel festgelegten Wert sind, werden in der Zeile gespeichert. BLOBs, die den in der INLINE-Klausel definierten Wert überschreiten, werden außerhalb der Zeile in Tabellenerweiterungsseiten gespeichert. Außerdem kann eine Kopie einiger Byte vom Beginn des BLOBs in der Zeile gespeichert werden, wenn ein BLOB größer als der INLINE-Wert ist. Verwenden Sie die PREFIX-Klausel, um anzugeben, wie viele Byte in der Zeile gehalten werden können. Die PREFIX-Klausel kann die Performance von Anforderungen verbessern, die die Präfixbyte eines BLOBs benötigen, um zu ermitteln, ob eine Zeile akzeptiert oder zurückgewiesen wird.

Die Präfixdaten für eine komprimierte Spalte werden nicht komprimiert gespeichert, daher ist keine Dekomprimierung erforderlich, wenn alle Daten, die zur Erfüllung einer Anforderung erforderlich sind, im Präfix gespeichert sind.

Wenn weder INLINE noch PREFIX angegeben sind oder wenn USE DEFAULT angegeben ist, werden Standardwerte wie folgt angewendet:

- Bei Spalten vom Zeichendaten-Typ wie CHAR, NCHAR und LONG VARCHAR ist der Standardwert für INLINE 256 und der Standardwert für PREFIX ist 8.

- Bei Spalten vom Typ Binärdaten wie BINARY, LONG BINARY, VARBINARY, BIT, VARBIT, LONG VARBIT, BIT VARYING und UUID ist der Standardwert für INLINE 256 und der Standardwert für PREFIX 0.

Hinweis

Es wird dringend empfohlen, die Standardwerte zu verwenden, es sei denn, spezielle Umstände erfordern eine andere Einstellung. Die Standardwerte wurden ausgewählt, um die Performance und den Festplattenspeicherbedarf möglichst gut aneinander anzupassen. Wenn Sie z.B. INLINE auf einen großen Wert setzen und alle BLOBs in der Zeile gespeichert werden, kann sich die Performance bei der Zeilenverarbeitung verschlechtern. Wenn Sie PREFIX zu hoch einstellen, erhöhen Sie den erforderlichen Speicherplatz zum Speichern von BLOBs, da die PREFIX-Daten ein Duplikat eines BLOB-Abschnitts sind.

Wenn nur ein Wert angegeben wird, wird der andere Wert automatisch auf die maximale Größe gesetzt, die nicht mit dem angegebenen Wert in Konflikt steht. Weder der INLINE- noch der PREFIX-Wert dürfen die Seitengröße der Datenbank überschreiten. Auch gibt es in einer Tabellenseite eine geringe Menge an reserviertem Overhead, der nicht zum Speichern von Zeilendaten verwendet werden kann. Daher führt das Angeben eines INLINE-Werts, der fast die Seitengröße der Datenbank erreicht, möglicherweise zu einer etwas niedrigeren Anzahl der in der Zeile gespeicherten Bytes.

INDEX- und NO INDEX-Klauseln Wenn Sie BLOBs speichern (nur Zeichen- oder Binärdatentypen), geben Sie INDEX an, um BLOB-Indizes für eingefügte Werte zu erstellen, die den internen BLOB-Schwellenwert übersteigen (etwa acht Datenbankseiten). Dies ist die Standardeinstellung.

BLOB-Indizes können die Performance verbessern, wenn Suchvorgänge mit wahlfreiem Zugriff innerhalb der BLOBs erforderlich sind. Allerdings kann bei BLOB-Werten wie Bildern und Multimediadateien, bei denen ein zufälliger Zugriff nicht erforderlich ist, die Performance gesteigert werden, wenn die BLOB-Indizierung deaktiviert ist. Um die BLOB-Indizierung für eine Spalte zu deaktivieren, geben Sie NO INDEX ein.

Hinweis

Ein BLOB-Index unterscheidet sich von einem Tabellenindex. Ein Tabellenindex wird erstellt, um Werte in einer oder mehreren Spalten zu indizieren.

NULL- und NOT NULL-Klauseln Wenn NULL angegeben ist, werden NULL-Werte in der Spalte zugelassen. Dies ist die Standardeinstellung. Diese Einstellung wird mit der Datenbankoption `allow_nulls_by_default` gesteuert.

Standardmäßig sind Spalten, die als BIT deklariert sind, nicht nullwertfähig, können aber explizit nullwertfähig gemacht werden.

Wenn NOT NULL angegeben ist, werden NULL-Werte nicht zugelassen.

Wenn die Spalte Teil einer UNIQUE- oder PRIMARY KEY-Integritätsregel ist, kann die Spalte keine NULL-Werte enthalten, auch wenn die NULL-Klausel angegeben wurde.

DEFAULT-Klausel Wenn ein DEFAULT-Wert angegeben ist, wird er als Wert für die Spalte in jeder INSERT-Anweisung benutzt, die für diese Spalte keinen Wert angibt. Wenn kein DEFAULT-Wert angegeben wird, entspricht dies DEFAULT NULL.

Die folgende Liste enthält mögliche Werte für DEFAULT:

- **special-value** Sie verwenden eine von mehreren Spezialwerten in der DEFAULT-Klausel.
- **[CURRENT] TIMESTAMP** Damit kann angezeigt werden, wann die einzelnen Zeilen in der Tabelle zuletzt geändert wurden. Ist eine Spalte mit DEFAULT TIMESTAMP deklariert, dann wird bei Einfügungen ein Standardwert bereitgestellt, und der Wert wird bei jeder Aktualisierung der Zeile mit Datum und Tageszeit aktualisiert.

Um einen Standardwert beim Einfügen bereitzustellen, die Spalte aber nicht bei jeder Zeilenaktualisierung zu aktualisieren, verwenden Sie DEFAULT CURRENT TIMESTAMP anstatt DEFAULT TIMESTAMP.

Spalten, die mit DEFAULT TIMESTAMP deklariert wurden, enthalten eindeutige Werte. Damit können Anwendungen fast gleichzeitige Aktualisierungen derselben Zeile ermitteln. Wenn der aktuelle TIMESTAMP-Wert mit dem letzten Wert übereinstimmt, wird er durch den Wert der Option default_timestamp_increment hochgezählt.

In SQL Anywhere können Sie TIMESTAMP-Werte automatisch kürzen, indem Sie die Option default_timestamp_increment verwenden. Dies ist hilfreich, wenn Sie die Kompatibilität mit anderen Datenbankprogrammen sicherstellen möchten, die timestamp-Werte weniger genau erfassen.

Die globale Variable @@dbts gibt einen TIMESTAMP-Wert zurück, der den zuletzt für eine Spalte mit DEFAULT TIMESTAMP generierten Wert repräsentiert.

- **[CURRENT] UTC TIMESTAMP** Damit kann angezeigt werden, wann die einzelnen Zeilen in der Tabelle zuletzt geändert wurden. Ist eine Spalte mit DEFAULT UTC TIMESTAMP deklariert, dann wird bei Einfügungen ein Standardwert bereitgestellt, und der Wert wird bei jeder Aktualisierung der Zeile mit der aktuellen Coordinated Universal Time (UTC) aktualisiert, wenn die Zeile aktualisiert wird.

Um einen Standardwert beim Einfügen bereitzustellen, die Spalte aber nicht bei jeder Zeilenaktualisierung zu aktualisieren, verwenden Sie DEFAULT CURRENT UTC TIMESTAMP anstatt DEFAULT UTC TIMESTAMP.

Das Verhalten dieses Standardwerts ist dasselbe wie TIMESTAMP und CURRENT TIMESTAMP mit dem Unterschied, dass das Datum und die Uhrzeit in der Coordinated Universal Time (UTC) angezeigt werden.

- **string** Siehe „Zeichenfolgen“ auf Seite 6.
- **global-variable** Siehe „Globale Variablen“ auf Seite 88.
- **constant-expression** Konstante Ausdrücke, die keine Datenbankobjekte referenzieren, sind in einer DEFAULT-Klausel zulässig. Daher können Funktionen wie GETDATE oder DATEADD verwendet werden. Wenn der Ausdruck keine Funktion oder kein einfacher Wert ist, muss er in Klammern eingeschlossen werden.
- **sequence-expression** Sie können DEFAULT auf den aktuellen oder nächsten Wert aus einer Sequenz in der Datenbank setzen.

- **AUTOINCREMENT** Wenn AUTOINCREMENT verwendet wird, muss die Spalte einer der Ganzzahl-Datentypen oder ein numerisch exakter Typ sein.

Ist beim Einfügen in die Tabelle kein Wert für die AUTOINCREMENT-Spalte vorgegeben, wird ein eindeutiger Wert erstellt, der größer ist als alle Werte in der Spalte. Wenn INSERT einen Wert für die Spalte angibt, der größer ist als der derzeitige Höchstwert für die Spalte, wird dieser Wert eingefügt und als Startpunkt für nachfolgende Einfügungen verwendet.

Das Löschen von Zeilen setzt den AUTOINCREMENT-Zähler nicht herunter. Lücken durch gelöschte Zeilen können nur durch explizite Zuweisungen wieder gefüllt werden, wenn eine Einfügung verwendet wird. Nach der expliziten Einfügung eines Spaltenwerts unter dem Maximum werden nachfolgende Zeilen ohne explizite Zuordnung weiterhin automatisch mit einem um 1 höheren Wert als das vorherige Maximum erhöht.

Sie können den zuletzt eingefügten Wert der Spalte herausfinden, indem Sie die globale Variable @@identity heranziehen.

AUTOINCREMENT-Werte werden als 64-Bit-Ganzzahlwerte mit Vorzeichen aufrechterhalten, entsprechend dem Datentyp der max_identity-Spalte in der SYSTABCOL-Systemansicht. Wenn der nächste zu generierende Wert den Höchstwert überschreitet, der in der Spalte mit der AUTOINCREMENT-Zuordnung gespeichert werden kann, wird NULL zurückgegeben. Wenn in der Spalte NULL nicht zulässig ist, wie im Fall von Primärschlüsselspalten, wird ein SQL-Fehler generiert.

Der nächste Wert für eine Spalte kann unter Verwendung der Prozedur sa_reset_identity zurückgesetzt werden.

- **GLOBAL AUTOINCREMENT** Dieser Standardwert ist vorgesehen, wenn mehrere Datenbanken in einer MobiLink-Synchronisationsumgebung oder einer SQL Remote-Replikation verwendet werden.

Diese Option ähnelt AUTOINCREMENT, außer dass die Domäne partitioniert ist. Jede Teilmenge enthält dieselbe Anzahl von Werten. Sie ordnen jeder Kopie der Datenbank eine eindeutige Datenbank-Identifizierungsnummer zu. SQL Anywhere liefert Standardwerte in einer Datenbank nur von der Partition, die eindeutig durch diese Datenbanknummer gekennzeichnet ist.

Die Partitionsgröße kann unmittelbar nach dem Schlüsselwort AUTOINCREMENT in Klammern angegeben werden. Die Partitionsgröße kann jede positive Ganzzahl sein, obwohl dieser Wert normalerweise so eingeteilt wird, dass seine Größe kaum jemals überschritten werden kann.

Wenn die Spalte vom Typ BIGINT oder UNSIGNED BIGINT ist, beträgt die Standard-Partitionsgröße $2^{32} = 4294967296$. Bei Spalten aller anderen Typen ist der Standardwert $2^{16} = 65536$. Da diese Standardwerte nicht immer sinnvoll sind, vor allem wenn die Spalte nicht vom Typ INT oder BIGINT ist, empfiehlt es sich, die Partitionsgröße explizit festzulegen.

Wenn Sie diesen Standardwert verwenden, muss der Wert der öffentlichen Option global_database_id in jeder Datenbank auf eine eindeutige, nicht-negative Ganzzahl gesetzt werden. Dieser Wert kennzeichnet die Datenbank eindeutig und zeigt an, von welcher Partition Standardwerte zugeordnet werden sollen. Der Bereich der zulässigen Werte geht von $np + 1$ bis $p(n + 1)$, wobei n der Wert der öffentlichen Option global_database_id und p die Partitionsgröße ist. Wenn Sie z.B. die

Partitionsgröße 1000 festlegen und `global_database_id` auf 3 gesetzt ist, liegt der Bereich zwischen 3001 und 4000.

Wenn der vorherige Wert kleiner ist als $p(n+1)$, wird der nächste Standardwert um eins größer sein als der vorherige größte Wert in der Spalte. Wenn die Spalte keine Werte enthält, ist der erste Standardwert $np+1$. Standardspaltenwerte sind von Werten, die sich außerhalb der aktuellen Partition befinden, nicht betroffen, d.h. von Nummern kleiner als $np+1$ oder größer als $p(n+1)$. Solche Werte können auftreten, wenn sie von einer anderen Datenbank über MobiLink oder SQL Remote repliziert wurden.

Sie können den zuletzt eingefügten Wert der Spalte herausfinden, indem Sie die globale Variable `@@identity` heranziehen.

GLOBAL AUTOINCREMENT-Werte werden als 64-Bit-Ganzzahlwerte mit Vorzeichen aufrechterhalten, entsprechend dem Datentyp der `max_identity`-Spalte in der SYSTABCOL-Systemansicht. Wenn der Wertevorrat innerhalb der Partition aufgebraucht ist, wird NULL zurückgegeben. Wenn in der Spalte NULL nicht zulässig ist, wie im Fall von Primärschlüsselspalten, wird ein SQL-Fehler generiert. In diesem Fall sollte der Datenbank ein neuer `global_database_id`-Wert zugewiesen werden, damit Standardwerte aus einer anderen Partition gewählt werden können. Um festzustellen, ob der Vorrat von ungenutzten Werten zu Ende geht, und um diese Situation zu beheben, erstellen Sie ein Ereignis vom Typ GlobalAutoincrement.

Da die öffentliche Option `global_database_id` nicht auf einen negativen Wert gesetzt werden kann, sind die ausgewählten Werte immer positiv. Die maximale Identifizierungsnummer wird nur durch den Spaltendatentyp und die Partitionsgröße beschränkt.

Wenn die öffentliche Option `global_database_id` auf den Standardwert 2147483647 gesetzt ist, wird NULL in die Spalte eingefügt. Falls NULL nicht zulässig ist, wird beim Versuch, die Zeile einzufügen, ein Fehler erzeugt.

Der nächste Wert für eine Spalte kann unter Verwendung der Prozedur `sa_reset_identity` zurückgesetzt werden.

- **LAST USER** LAST USER ist die Benutzer-ID des Benutzers, der die Zeile zuletzt geändert hat.

LAST USER kann als Standardwert in Spalten mit Zeichendatentypen verwendet werden.

Dieser Standardwert hat bei INSERT dieselbe Wirkung wie CURRENT USER.

Wenn Sie eine Spalte mit einem Standardwert LAST USER bei einer UPDATE-Anweisung nicht ausdrücklich ändern, wird sie auf den Namen des aktuellen Benutzers abgeändert.

In Kombination mit DEFAULT TIMESTAMP oder DEFAULT UTC TIMESTAMP kann der Standardwert LAST USER dazu verwendet werden, sowohl den Benutzer als auch Datum und Uhrzeit der letzten Änderung an einer Zeile zu erfassen (in getrennten Spalten).

IDENTITY-Klausel IDENTITY ist eine Transact-SQL-kompatible Alternative zur Verwendung von DEFAULT AUTOINCREMENT. In SQL Anywhere wird eine mit IDENTITY definierte Spalte als DEFAULT AUTOINCREMENT implementiert.

Klauseln der Spalten-Integritätsregel und Tabellen-Integritätsregel Spalten- und Tabellen-Integritätsregeln helfen, die Integrität der Daten in der Datenbank sicherzustellen. Wenn eine Anweisung die Verletzung einer Integritätsregel verursacht, wird die Ausführung der Anweisung abgebrochen, alle Änderungen, die vor dem Auftreten des Fehlers vorgenommen wurden, werden rückgängig gemacht und ein Fehler wird protokolliert. Es können zwei Klassen von Integritätsregeln erstellt werden: **Prüf-Integritätsregel** und **Regeln zur Erhaltung der referenziellen Integrität (RI)**. CHECK-Integritätsregeln werden verwendet, um Bedingungen anzugeben, die von Spaltenwerten, die in die Datenbank eingegeben werden, erfüllt werden müssen. RI-Integritätsregeln stellen eine Beziehung zwischen Daten in verschiedenen Tabellen her, die erhalten bleiben müssen, und geben überdies Eindeutigkeitsanforderungen für Daten an.

Es gibt drei Arten von RI-Integritätsregeln: Primärschlüssel, Fremdschlüssel und Eindeutigkeits-Integritätsregeln. Wenn Sie eine RI-Integritätsregel (Primärschlüssel, Fremdschlüssel oder Eindeutigkeits-Integritätsregel) erstellen, erzwingt der Datenbankserver die Integritätsregel, indem er implizit einen Index für die Spalten erstellt, die den Schlüssel der Integritätsregel bilden. Der Index wird auf dem Schlüssel für die Integritätsregel, wie angegeben, erstellt. Ein Schlüssel besteht aus einer sortierten Spaltenliste und einer Reihenfolge der Werte (aufsteigend oder absteigend) für alle Spalten.

Integritätsregeln können für Spalten oder Tabellen angegeben werden. Eine Spalten-Integritätsregel ist eine Spalte in einer Tabelle, während sich eine Tabellen-Integritätsregel auf eine oder mehrere Spalten in der Tabelle beziehen kann.

- **PRIMARY KEY-Klausel** Ein Primärschlüssel definiert eindeutig jede Zeile in der Tabelle. Primärschlüssel umfassen eine oder mehrere Spalten. Eine Tabelle kann nicht mehr als einen Primärschlüssel besitzen. In einer *column-constraint*-Klausel gibt die Verwendung von PRIMARY KEY an, dass die Spalte der Primärschlüssel für die Tabelle ist. In einer *table-constraint*-Klausel verwenden Sie die PRIMARY KEY-Klausel, um eine oder mehrere Spalten anzugeben, die, wenn sie in der angegebenen Reihenfolge kombiniert werden, den Primärschlüssel für die Tabelle darstellen.

Die Reihenfolge der Spalten in einem Primärschlüssel muss nicht mit den entsprechenden Ordinalnummern der Spalte übereinstimmen. Das heißt, dass die Spalten in einem Primärschlüssel nicht dieselbe physische Reihenfolge in der Zeile haben müssen. Außerdem dürfen Sie keine doppelten Spaltennamen angeben.

Wenn Sie einen Primärschlüssel erstellen, wird automatisch ein Index für den Schlüssel erstellt. Sie können die Reihenfolge der Werte in einem Index festlegen, indem Sie bei jeder Spalte ASC (ascending, aufsteigend) oder DESC (descending, absteigend) angeben. Sie können auch angeben, ob der Index als Clustered-Index erstellt werden soll, indem Sie das Schlüsselwort CLUSTERED verwenden.

Spalten, die in Primärschlüsseln enthalten sind, können NULL nicht zulassen. Jede Zeile in der Tabelle besitzt einen eindeutigen Primärschlüsselwert.

Es wird empfohlen, keine angenäherten Datentypen wie FLOAT und DOUBLE für Primärschlüssel zu verwenden. Bei angenäherten numerischen Datentypen können nach arithmetischen Vorgängen Rundungsfehler auftreten.

Spalten mit räumlichen Daten können nicht in einen Primärschlüssel einbezogen werden.

- **FOREIGN KEY-Klausel** Ein Fremdschlüssel beschränkt die Werte für eine Reihe von Spalten, sodass sie mit den Werten in einem Primärschlüssel oder einer Eindeutigkeits-Integritätsregel einer anderen Tabelle (der Primärtabelle) übereinstimmen. Eine Fremdschlüssel-Integritätsregel kann beispielsweise verwendet werden, um zu gewährleisten, dass eine Kundennummer in einer Rechnungstabelle mit der Kundennummer in der Customers-Tabelle übereinstimmt.

Die Reihenfolge der Fremdschlüsselspalten muss nicht der Reihenfolge der Spalten in der Tabelle entsprechen.

Doppelte Spaltennamen sind in der Fremdschlüsselspezifikation nicht zulässig.

Die Standardaktion (*action*) ist RESTRICT, wenn keine Aktion für einen UPDATE- oder DELETE-Vorgang angegeben ist.

Wenn Sie einen Fremdschlüssel erstellen, wird automatisch ein Index für den Schlüssel erstellt. Sie können die Reihenfolge der Werte in einem Index festlegen, indem Sie bei jeder Spalte ASC (ascending, aufsteigend) oder DESC (descending, absteigend) angeben. Sie können auch angeben, ob der Index als Clustered-Index erstellt werden soll, indem Sie das Schlüsselwort CLUSTERED verwenden.

Eine globale temporäre Tabelle kann keinen Fremdschlüssel besitzen, der eine Basistabelle referenziert, und eine Basistabelle kann keinen Fremdschlüssel besitzen, der eine globale temporäre Tabelle referenziert.

Wenn Sie versuchen, einen Fremdschlüssel zu einer nicht vorhandenen Spalte hinzuzufügen, wird die Spalte automatisch erstellt.

- **NOT NULL-Klausel** Lässt NULL nicht in den Fremdschlüsselspalten zu. NULL in einem Fremdschlüssel bedeutet, dass dieser Zeile in der Fremdtabelle keine Zeile in der Primärtabelle entspricht.
- **role-name-Klausel** Der Rollename ist der Name des Fremdschlüssels. Die Hauptfunktion des Rollennamens liegt in der Unterscheidung von zwei Fremdschlüsseln für dieselbe Tabelle. Wenn kein Rollename angegeben ist, wird der Rollename wie folgt zugeordnet:
 1. Wenn es keinen Fremdschlüssel mit einem Rollennamen gibt, der genauso lautet wie der Tabellename, wird der Tabellename als Rollename zugeordnet.
 2. Wenn der Tabellename bereits vergeben ist, wird für den Rollennamen ein Tabellename verwendet, der aus einer für diese Tabelle eindeutigen dreistelligen und mit vorangestellten Nullen versehenen Zahl zusammengesetzt ist.
- **REFERENCES-Klausel** Eine Fremdschlüssel-Integritätsregel kann mithilfe einer REFERENCES-Spalten-Integritätsregel (nur eine einzige Spalte) oder einer FOREIGN KEY-Tabellen-Integritätsregel implementiert werden, wobei die Integritätsregel eine oder mehrere Spalten angeben kann. Wenn Sie einen *column-name*-Wert in einer REFERENCES-Spalten-Integritätsregel angeben, muss es sich um eine Spalte in der Primärtabelle handeln. Diese muss einer Eindeutigkeits-Integritätsregel oder Primärschlüssel-Integritätsregel unterliegen und diese Integritätsregel darf nur aus dieser einen Spalte bestehen. Wenn Sie keinen *column-name*-Wert

angeben, referenziert die Fremdschlüsselspalte die einzelne Primärschlüsselspalte der Primärtabelle.

- **MATCH-Klausel** Die MATCH-Klausel legt fest, was als Übereinstimmung gilt, wenn ein Mehrspalten-Fremdschlüssel verwendet wird, indem Sie einstellen können, was als verwaiste Zeile und was als Verletzung der referenziellen Integrität zu behandeln ist. Die MATCH-Klausel ermöglicht es Ihnen außerdem, Eindeutigkeit für den Schlüssel anzugeben, wodurch sich ein separates Deklarieren der Eindeutigkeit erübrigt.

Es folgt eine Liste der MATCH-Typen, die Sie angeben können. Im Abschnitt "Beispiele" am Ende dieses Themas finden Sie eine Beschreibung dazu, wie sich der MATCH-Typ auf die Suche nach Übereinstimmungen auswirkt.

- **MATCH [UNIQUE] SIMPLE** Eine Übereinstimmung tritt für eine Zeile in der Fremdschlüsseltabelle auf, wenn alle Spaltenwerte mit den entsprechenden Spaltenwerten in einer Zeile der Primärschlüsseltabelle übereinstimmen. Eine Zeile wird in der Fremdschlüsseltabelle zur Waisen, wenn mindestens ein Spaltenwert im Fremdschlüssel NULL ist.

MATCH SIMPLE ist das Standardverhalten.

Wenn das UNIQUE-Schlüsselwort angegeben ist, kann die referenzierende Tabelle bei Nicht-NULL-Schlüsselwerten jeweils nur eine Übereinstimmung enthalten.

- **MATCH [UNIQUE] FULL** Eine Übereinstimmung tritt für eine Zeile in der Fremdschlüsseltabelle auf, wenn keiner der Werte NULL ist und die Werte mit den entsprechenden Spaltenwerten in einer Zeile der Primärschlüsseltabelle übereinstimmen. Eine Zeile wird zur Waisen, wenn alle Spaltenwerte im Fremdschlüssel NULL sind.

Wenn das UNIQUE-Schlüsselwort angegeben ist, kann die referenzierende Tabelle bei Nicht-NULL-Schlüsselwerten jeweils nur eine Übereinstimmung enthalten.

- **UNIQUE-Klausel** In einer *column-constraint*-Klausel gibt eine UNIQUE-Integritätsregel an, dass die Werte in der Spalte eindeutig sein müssen. In einer *table-constraint*-Klausel kennzeichnet die UNIQUE-Integritätsregel eine oder mehrere Spalten, die jede Zeile in der Tabelle eindeutig identifizieren. Es kann in einer Tabelle nicht mehrere Zeilen mit denselben Werten in der bzw. allen benannten Spalte(n) geben. Eine Tabelle kann mehr als eine UNIQUE-Integritätsregel besitzen.

Eine UNIQUE-Integritätsregel unterscheidet sich von einem eindeutigen Index. Spalten mit einem eindeutigen Index lassen NULL zu, nicht aber Spalten in einer Eindeutigkeits-Integritätsregel. Auch kann ein Fremdschlüssel entweder einen Primärschlüssel oder eine UNIQUE-Integritätsregel referenzieren, aber keinen eindeutigen Index, da dieser mehrere Instanzen von NULL enthalten kann.

Spalten in einer UNIQUE-Integritätsregel können in beliebiger Reihenfolge angegeben werden. Zusätzlich können Sie die Reihenfolge der Werte im entsprechenden Index, der automatisch erstellt wird, festlegen, indem Sie bei jeder Spalte ASC (ascending, aufsteigend) oder DESC (descending, absteigend) angeben. Sie dürfen allerdings keine doppelten Spaltennamen angeben.

Es wird empfohlen, keine angenäherten Datentypen wie FLOAT und DOUBLE für Spalten mit Eindeutigkeits-Integritätsregeln zu verwenden. Bei angenäherten numerischen Datentypen können nach arithmetischen Vorgängen Rundungsfehler auftreten.

Sie können auch angeben, ob die Integritätsregel als Clustered-Integritätsregel erstellt werden soll, indem Sie das Schlüsselwort CLUSTERED verwenden.

- **CHECK-Klausel** Diese Integritätsregel ermöglicht die Überprüfung beliebiger Bedingungen. Zum Beispiel kann mit einer CHECK-Integritätsregel gewährleistet werden, dass eine Spalte mit dem Namen Geschlecht nur die Werte M oder W enthält.

Wenn Sie eine Prüf-Integritätsregel erstellen müssen, die eine Beziehung zwischen zwei oder mehr Spalten in der Tabelle umfasst (beispielsweise muss Spalte A kleiner sein als Spalte B), definieren Sie anstelle dessen eine Tabellenintegritätsregel.

Keine Zeile in einer Tabelle darf gegen eine CHECK-Integritätsregel verstoßen. Wenn eine INSERT- oder UPDATE-Anweisung dazu führen sollte, dass eine Zeile gegen die Integritätsregel verstößt, wird der Vorgang nicht zugelassen und die Auswirkungen der Anweisung werden rückgängig gemacht. Die Änderung wird nur zurückgewiesen, wenn die Bedingung für die CHECK-Integritätsregel FALSE ist. Dagegen ist die Änderung zulässig, wenn diese Bedingung TRUE oder UNKNOWN ist.

- **COMPUTE-Klausel** Die COMPUTE-Klausel ist nur zur Verwendung in einer *column-constraint*-Klausel bestimmt.

Wenn eine Spalte unter Verwendung einer COMPUTE-Klausel erstellt wurde, entspricht ihr Wert in jeder Zeile dem Wert des angegebenen Ausdrucks. Mit dieser Integritätsregel erstellte Spalten sind schreibgeschützte Spalten für Anwendungen: Der Wert wird vom Datenbankserver geändert, wenn die Zeile geändert wird. Der COMPUTE-Ausdruck darf keinen nicht deterministischen Wert zurückgeben. Er darf z.B. nicht einen Spezialwert wie CURRENT_TIMESTAMP oder eine nicht deterministische Funktion enthalten. Wenn ein COMPUTE-Ausdruck einen nicht deterministischen Wert zurückgibt, kann er nicht verwendet werden, um eine Übereinstimmung in einer Abfrage zu suchen.

Die COMPUTE-Klausel wird für entfernte Tabellen ignoriert.

Eine UPDATE-Anweisung, die versucht, den Wert einer berechneten Spalte zu ändern, löst alle Trigger aus, die mit der Spalte verbunden sind.

CHECK ON COMMIT-Klausel Die CHECK ON COMMIT-Option hebt die Datenbankoption wait_for_commit auf und veranlasst den Datenbankserver, auf ein COMMIT zu warten, bevor er RESTRICT-Aktionen auf einen Fremdschlüssel überprüft. Die CHECK ON COMMIT-Option verzögert die Fremdschlüsselüberprüfung, nicht aber andere Aktionen, z.B. CASCADE, SET NULL, SET DEFAULT, oder CHECK-Integritätsregeln.

FOR OLAP WORKLOAD-Klausel Wenn Sie FOR OLAP WORKLOAD in der REFERENCES-Klausel einer Fremdschlüsseldefinition angeben, führt der Datenbankserver bestimmte Optimierungen durch und sammelt Statistiken, um die Verbesserung der Performance für OLAP-Arbeitslasten zu unterstützen, vor allem wenn die Option optimization_workload auf OLAP gesetzt ist.

PCTFREE-Klausel Gibt den Prozentsatz des freien Speicherplatzes an, der für jede Tabellenseite reserviert werden soll. Der freie Speicherplatz wird verwendet, wenn die Größe der Zeilen durch die

Datenaktualisierung zunimmt. Wenn in einer Tabellenseite kein freier Speicherplatz verfügbar ist, führt jede Vergrößerung einer Zeile dieser Seite zu einer Aufteilung der Zeile auf mehrere Tabellenseiten, was Zeilenfragmentierung und eventuell Performanceverlust nach sich zieht.

Der Wert von *percent-free-space* ist eine Ganzzahl zwischen 0 und 100. Der erste Wert bedeutet, dass auf den einzelnen Seiten kein freier Platz zur Verfügung stehen darf, d.h. jede Seite wird vollgeschrieben. Ein hoher Wert führt dazu, dass jede Zeile auf eine eigene Seite geschrieben wird. Wenn PCTFREE nicht festgelegt oder später gelöscht wurde, wird der PCTFREE-Standardwert entsprechend der Seitengröße der Datenbank angewendet (200 Byte für eine Seitengröße von 4 kB und mehr). Der Wert für PCTFREE wird in der Systemtabelle ISYSTAB gespeichert.

Bemerkungen

Die CREATE TABLE-Anweisung erstellt eine neue Tabelle. Eine Tabelle kann für einen anderen Benutzer erstellt werden, indem ein Eigentümername angegeben wird. Wenn GLOBAL TEMPORARY angegeben wird, handelt es sich bei der Tabelle um eine temporäre Tabelle. Andernfalls ist es eine Basistabelle.

Tabellen, die erstellt werden, indem dem Tabellennamen in der CREATE TABLE-Anweisung ein Rautenzeichen (#) vorangestellt wird, werden als temporäre Tabellen deklariert, die nur in der aktuellen Verbindung verfügbar sind. Temporäre Tabellen, die mit dem Rautenzeichen (#) erstellt wurden, sind identisch mit denen, die mit der Klausel ON COMMIT PRESERVE ROWS erstellt wurden.

Zwei lokale temporäre Tabellen in demselben Geltungsbereich können nicht den gleichen Namen haben. Falls Sie eine temporäre Tabelle mit demselben Namen wie eine Basistabelle erstellen, wird die Basistabelle innerhalb der Verbindung erst sichtbar, wenn der Bereich der lokalen temporären Tabelle endet. Eine Verbindung kann keine Basistabelle mit dem Namen einer vorhandenen temporären Tabelle erstellen.

Spalten in SQL Anywhere lassen standardmäßig NULL zu. Diese Einstellung kann mit der Datenbankoption `allow_nulls_by_default` gesteuert werden.

Privilegien

Sie müssen das CREATE TABLE-Systemprivileg haben, um Tabellen erstellen zu können, deren Eigentümer Sie sind. Sie müssen das CREATE ANY TABLE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben, um Tabellen erstellen zu können, deren Eigentümer andere Benutzer sind.

Wenn Sie Proxy-Tabellen erstellen möchten, deren Eigentümer Sie sind, müssen Sie das CREATE PROXY TABLE-Systemprivileg haben. Sie müssen das CREATE ANY TABLE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben, um Proxy-Tabellen erstellen zu können, deren Eigentümer andere Benutzer sind.

Nebenwirkungen

Automatisches Festschreiben (auch, wenn Sie globale temporäre Tabellen erstellen).

Siehe auch

- „OLAP-Unterstützung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „allow_nulls_by_default-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Ereignisse“ [*SQL Anywhere Server - Datenbankadministration*]
- „sa_reset_identity-Systemprozedur“ auf Seite 1299
- Wählen zwischen Sequenzen und AUTOINCREMENT-Werten [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- Neuladen von Tabellen mit AUTOINCREMENT-Spalten [*SQL Anywhere 16 - Änderungen und Upgrades*]
- „CREATE LOCAL TEMPORARY TABLE-Anweisung“ auf Seite 645
- „Globale Variable @@identity“ auf Seite 91
- „Die spezielle IDENTITY-Spalte“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Tabellen verschlüsseln“ [*SQL Anywhere Server - Datenbankadministration*]
- „ALTER TABLE-Anweisung“ auf Seite 516
- „CREATE DBSPACE-Anweisung“ auf Seite 593
- „CREATE EXISTING TABLE-Anweisung“ auf Seite 613
- „DECLARE LOCAL TEMPORARY TABLE-Anweisung“ auf Seite 784
- „DROP TABLE-Anweisung“ auf Seite 832
- „optimization_workload-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Spezialwerte“ auf Seite 70
- „SQL-Datentypen“ auf Seite 95
- „Erstellen einer Tabelle“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „allow_nulls_by_default-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Temporäre Tabellen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Berechnete Spalten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „CREATE DBSPACE-Anweisung“
- „Zusätzliche Hinweise zu DBSpaces“ [*SQL Anywhere Server - Datenbankadministration*]
- „Verwendung einer Sequenz zum Generieren von eindeutigen Werten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „default_dbspace-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „DECLARE LOCAL TEMPORARY TABLE-Anweisung“ auf Seite 784

Standards und Kompatibilität

- **SQL/2008** CREATE TABLE ist eine Kernfunktion des SQL/2008-Standards, obwohl einige der in SQL Anywhere unterstützten Komponenten optionale SQL-Sprachenfunktionen sind. Zu diesen Funktionen gehören folgende:
 - Die Unterstützung für temporäre Tabellen ist SQL-Sprachenfunktion F531.
 - Die Unterstützung für IDENTITY-Spalten ist SQL-Funktion T174, obwohl SQL Anywhere eine etwas andere Syntax verwendet als der Standard.
 - Die Unterstützung für Fremdschlüssel-Integritätsregeln umfasst die SQL-Sprachenfunktion T191 (referenzielle Aktion RESTRICT), F741 (referenzielle MATCH-Typen), F191 (referenzielle Löschaktionen) und F701 (referenzielle UPDATE-Aktionen). SQL Anywhere bietet keine Unterstützung für MATCH PARTIAL.

SQL Anywhere bietet keine Unterstützung für SQL-Sprachenfunktion T591 (UNIQUE-Integritätsregeln potenzieller NULL-Spalten). In SQL Anywhere müssen alle Spalten, die Teil einer UNIQUE- oder PRIMARY KEY-Integritätsregel sind, als NOT NULL deklariert werden.

Die folgenden Komponenten von CREATE TABLE sind Erweiterungen des Herstellers:

- Die { IN | ON } *dbspace-name*-Klausel
- Die Klauseln ENCRYPTED, NOT TRANSACTIONAL und SHARE BY ALL.
- Die Klauseln COMPRESSED, INLINE, PREFIX und NO INDEX einer Spaltendefinition.
- Verschiedene in der Implementierung definierte DEFAULT-Werte, einschließlich AUTOINCREMENT, GLOBAL AUTOINCREMENT, CURRENT DATABASE, CURRENT REMOTE USER, CURRENT UTC TIMESTAMP, sowie die meisten Spezialwerte. Eine DEFAULT-Klausel, die einen Sequenzgenerator referenziert, ist ebenfalls eine Erweiterung des Herstellers.
- Die Spezifikation von MATCH UNIQUE.
- Die Sortierungsspezifikation (ASC oder DESC) in einer PRIMARY KEY oder FOREIGN KEY-Klausel.
- Die Möglichkeit zum Angeben von FOREIGN KEY-Spalten in einer anderen Reihenfolge als in der PRIMARY KEY-Klausel der referenzierten Tabelle angegeben.

Beispiele

Die folgende Anweisung erstellt die Tabelle "file_table" mit zwei Spalten: file_name und file_contents. Die Spalte contents ist LONG BINARY und komprimiert:

```
CREATE TABLE file_table (
    file_name VARCHAR(255),
    file_contents LONG BINARY COMPRESSED
);
```

Mit dem folgenden Beispiel wird eine Tabelle für eine Bibliotheksdatenbank zur Aufnahme von Bücherdaten erstellt.

```
CREATE TABLE library_books (
    -- NOT NULL is assumed for primary key columns
    isbn CHAR(20) PRIMARY KEY,
    copyright_date DATE,
    title CHAR(100),
    author CHAR(50),
    -- column(s) corresponding to primary key of room
    -- are created automatically
);
```

Mit dem folgenden Beispiel wird eine Tabelle für eine Bibliotheksdatenbank zur Aufnahme von Angaben über ausgeliehene Bücher erstellt. Der Standardwert für date_borrowed zeigt an, dass das Buch an dem Tag ausgeliehen wurde, an dem der Eintrag erfolgte. Die Spalte date_returned ist NULL, bis das Buch zurückgegeben wird.

```
CREATE TABLE borrowed_book (
    date_borrowed DATE NOT NULL DEFAULT CURRENT DATE,
```

```
    date_returned DATE,
    book CHAR(20)
    REFERENCES library_books (isbn),
    -- The check condition is UNKNOWN until
    -- the book is returned, which is allowed
    CHECK( date_returned >= date_borrowed )
);
```

Mit dem folgenden Beispiel werden Tabellen für die Datenbank einer Verkaufsabteilung zur Aufnahme von Angaben zu Aufträgen und Auftragspositionen erstellt.

```
CREATE TABLE Orders (
    order_num INTEGER NOT NULL PRIMARY KEY,
    date_ordered DATE,
    name CHAR(80)
);
CREATE TABLE Order_item (
    order_num INTEGER NOT NULL,
    item_num SMALLINT NOT NULL,
    PRIMARY KEY ( order_num, item_num ),
    -- When an order is deleted, delete all of its
    -- items.
    FOREIGN KEY ( order_num )
    REFERENCES Orders ( order_num )
    ON DELETE CASCADE
);
```

Im folgenden Beispiel wird die Tabelle "t1" auf dem **fiktiven** Fremdserver SERVER_A erstellt und die Proxytabelle "t1", die der entfernten Tabelle zugeordnet ist.

```
CREATE TABLE t1
( a INT,
  b CHAR(10) )
AT 'SERVER_A.db1.joe.t1';
```

Im folgenden Beispiel werden zwei Tabellen namens Table1 und Table2 erstellt, Fremdschlüssel zu Table2 hinzugefügt und Werte in Table1 eingefügt. In der letzten Anweisung wird versucht, Werte in Table2 einzufügen. Ein Fehler wird zurückgegeben, weil die Werte, die Sie einzufügen versuchen, keine einfache Übereinstimmung mit Table1 darstellen.

```
CREATE TABLE Table1 ( P1 INT, P2 INT, P3 INT, P4 INT, P5 INT, P6 INT,
    PRIMARY KEY ( P1, P2 ) );
CREATE TABLE Table2 ( F1 INT, F2 INT, F3 INT, PRIMARY KEY ( F1, F2 ) );
ALTER TABLE Table2
    ADD FOREIGN KEY fk2( F1,F2 )
    REFERENCES Table1( P1, P2 )
    MATCH SIMPLE;
INSERT INTO Table1 (P1, P2, P3, P4, P5, P6) VALUES ( 1,2,3,4,5,6 );
INSERT INTO Table2 (F1,F2) VALUES ( 3,4 );
```

Die folgenden Anweisungen zeigen, wie MATCH SIMPLE und MATCH SIMPLE UNIQUE sich in der Art unterscheiden, wie Fremdschlüssel mit mehreren Spalten verarbeitet werden, wenn einige, aber nicht alle Spalten im Schlüssel NULL sind. Diese Anweisung schlägt fehl, weil die Werte für die zweite Spalte nicht eindeutig sind.

```
CREATE TABLE pt( pk INT PRIMARY KEY, str VARCHAR(10));
INSERT INTO pt VALUES(1,'one'), (2,'two');
COMMIT;

CREATE TABLE ft1( fpk INT PRIMARY KEY, FOREIGN KEY (ref) REFERENCES pt MATCH
```

```

SIMPLE);
INSERT INTO ft1 VALUES(100,1), (200,1); //Diese Anweisung fügt 2 Zeilen ein.

CREATE TABLE ft2( fpk INT PRIMARY KEY, FOREIGN KEY (ref) REFERENCES pt MATCH
UNIQUE SIMPLE);
INSERT INTO ft2 VALUES(100,1), (200,1);

```

Die folgenden Anweisungen zeigen, wie sich MATCH SIMPLE und MATCH UNIQUE SIMPLE unterscheiden:

```

CREATE TABLE pt2(
    pk1 INT NOT NULL,
    pk2 INT NOT NULL,
    str VARCHAR(10),
    PRIMARY KEY (pk1,pk2));

INSERT INTO pt2 VALUES(1,10,'one-ten'), (2,20,'two-twenty');
COMMIT;

CREATE TABLE ft3(
    fpk INT PRIMARY KEY,
    ref1 INT,
    ref2 INT );

ALTER TABLE ft3 ADD FOREIGN KEY (ref1,ref2)
REFERENCES pt2 (pk1,pk2) MATCH SIMPLE;

CREATE TABLE ft4(
    fpk INT PRIMARY KEY,
    ref1 INT,
    ref2 INT );

INSERT INTO ft3 VALUES(100,1,10);
// MATCH SIMPLE-Test ist erfolgreich, weil alle Spaltenwerte mit den
entsprechenden Werten in pt2 übereinstimmen.
INSERT INTO ft3 VALUES(200,null,null); // MATCH SIMPLE-Test ist erfolgreich,
weil mindestens eine Spalte im Schlüssel NULL ist.
INSERT INTO ft3 VALUES(300,2,null); // MATCH SIMPLE-Test ist erfolgreich,
weil mindestens eine Spalte im Schlüssel NULL ist.
INSERT INTO ft4 VALUES(100,1,10); // MATCH SIMPLE-Test ist erfolgreich, weil
alle Spaltenwerte mit den entsprechenden Werten in pt2 übereinstimmen.
INSERT INTO ft4 VALUES(200,null,null); // MATCH SIMPLE-Test ist erfolgreich,
weil mindestens eine Spalte im Schlüssel NULL ist.
INSERT INTO ft4 VALUES(300,2,null);
// MATCH FULL-Test schlägt fehl, weil beide Spalten im Schlüssel NULL sein
oder mit den entsprechenden Werten in pt2 übereinstimmen müssen.

```

CREATE TEMPORARY TRACE EVENT-Anweisung

Erstellt ein Benutzer-Trace-Ereignis, das bestehen bleibt, bis die Datenbank gestoppt wird.

Syntax

```

CREATE [ OR REPLACE ] TEMPORARY TRACE EVENT trace-event-name
[ WITH SEVERITY {
    0-255
    | ALWAYS
    | CRITICAL
    | ERROR
    | WARNING
}

```

```
| INFORMATION
| DEBUG } ]
[ field-name field-type [ , ... ] ]
```

Parameter

trace-event-name Geben Sie einen Namen für das Benutzer-Trace-Ereignis an. Namen von Benutzer-Trace-Ereignissen dürfen nicht mit dem Präfix **SYS_** beginnen. Namen von System-Trace-Ereignissen können nicht angegeben werden.

OR REPLACE-Klausel Erstellen Sie ein neues Trace-Ereignis oder ersetzen Sie ein vorhandenes Trace-Ereignis mit demselben Namen.

WITH SEVERITY-Klausel Wenn kein Schweregrad angegeben wurde, wird der Standard-Schweregrad (DEBUG) verwendet. Benutzer-Trace-Ereignisse sind Eigentum der Datenbank, mit der der Benutzer beim Erstellen des Trace-Ereignisses verbunden war. Die folgenden Schweregradwerte werden unterstützt:

Ebene	Schweregrad-Wertebereich
ALWAYS	0
CRITICAL	1-50
ERROR	51-100
WARNING	101-150
INFORMATION	151-200
DEBUG	201-255

field-name Das Feld, das Informationen eines bestimmten Typs aus dem Benutzer Trace-Ereignis sammelt. Das Feld muss ein gültiger Bezeichner sein.

field-type Sie können jeden für eine Spalte unterstützten Datentyp verwenden, außer Arraytypen.

Bemerkungen

Ein Trace-Ereignis wird im Speicher gespeichert und wird beim Stoppen des Datenbankservers gelöscht, sofern es nicht explizit gelöscht wurde.

Systemprivilegien

Sie müssen das **MANAGE ANY TRACE SESSION**-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Ereignisprotokollierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE TEMPORARY TRACE EVENT SESSION-Anweisung“ auf Seite 755
- „ALTER TRACE EVENT SESSION-Anweisung“ auf Seite 538
- „DROP TRACE EVENT-Anweisung“ auf Seite 835
- „DROP TRACE EVENT SESSION-Anweisung“ auf Seite 836
- „NOTIFY TRACE EVENT-Anweisung“ auf Seite 963
- „sp_trace_events-Systemprozedur“ auf Seite 1407
- „sp_trace_event_fields-Systemprozedur“ auf Seite 1401
- „sp_trace_event_sessions-Systemprozedur“ auf Seite 1406
- „sp_trace_event_session_events-Systemprozedur“ auf Seite 1402
- „sp_trace_event_session_targets-Systemprozedur“ auf Seite 1405
- „sp_trace_event_session_target_options-Systemprozedur“ auf Seite 1403
- „Dienstprogramm für die Verwaltung von ETD-Dateien (dbmanageetd)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung erstellt ein Benutzer-Trace-Ereignis:

```
CREATE TEMPORARY TRACE EVENT my_event( id INTEGER, information LONG
VARCHAR );
```

CREATE TEMPORARY TRACE EVENT SESSION- Anweisung

Erstellt eine Benutzer-Trace-Ereignissitzung.

Syntax

```
CREATE [ OR REPLACE ] TEMPORARY TRACE EVENT SESSION session-name
[ event-definition [ , ... ] ]
[ target-definition [ , ... ] ]
```

event-definition :

```
ADD TRACE EVENT event-name
```

target-definition :

```
ADD TARGET target-name
```

```
[ SET target-parameter-name=target-parameter-value [ ,... ] ]
```

target-name : FILE

target-parameter-name :

```
{ filename_prefix
| max_size
| num_files
```

```
| flush_on_write  
| compressed }
```

Parameter

- session-name** Der Name der Trace-Ereignissitzung.
- event-name** Der Name des Trace-Ereignisses, das zur Sitzung hinzugefügt werden soll. System- und benutzerdefinierte Trace-Ereignisse werden unterstützt. Rufen Sie die sp_trace_events-Systemprozedur auf, um eine Liste der systemdefinierten Trace-Ereignisse abzurufen.
- target-name** Der einzige unterstützte Wert ist FILE.
- target-parameter-name** Die folgenden Zielparameter werden unterstützt:

target-parameter-name	target-parameter-value
filename_prefix	Ein ETD-Dateinamenpräfix mit oder ohne Pfad. Alle ETD-Dateien haben die Erweiterung .etd. Dieser Parameter ist erforderlich.
max_size	Die maximale Größe der Datei in Byte. Der Standardwert ist 0, was bedeutet, dass es keine Grenze für die Dateigröße gibt und die Datei größer wird, solange Speicherplatz verfügbar ist. Sobald die angegebene Größe erreicht ist, wird eine neue Datei gestartet.
num_files	Die Anzahl der Dateien, in die Informationen zur Ereignisprotokollierung geschrieben werden. Diese Option wird nur verwendet, wenn max_size eingestellt ist. Wenn alle Dateien die maximale angegebene Größe erreichen, beginnt der Datenbankserver, die älteste Datei zu überschreiben.
flush_on_write	Ein Wert, der steuert, ob Festplattenpuffer für jedes protokollierte Ereignis geleert werden. Die Werte YES, TRUE, NO und FALSE werden akzeptiert. Der Standardwert ist FALSE. Bei aktiviertem Parameter kann die Performance des Datenbankservers vermindert sein, wenn viele Trace-Ereignisse protokolliert werden.
compressed	Ein Wert, der die Komprimierung der ETD-Datei zum Einsparen von Speicherplatz steuert. Die Werte "on" und "off" werden akzeptiert. Die Option ist standardmäßig deaktiviert.

OR REPLACE-Klausel Diese Klausel erstellt eine Trace-Ereignissitzung oder ersetzt eine vorhandene Trace-Ereignissitzung mit demselben Namen.

Bemerkungen

Trace-Ereignissitzungen werden erst ausgeführt, wenn sie mit der ALTER TRACE EVENT SESSION-Anweisung explizit gestartet werden. Trace-Ereignissitzungen können verwendet werden, um Trace-Ereignisse in Bezug auf das Systemverhalten oder für einen bestimmten Benutzer zu erfassen. Trace-Ereignissitzungen werden im Speicher gespeichert und werden beim Stoppen des Datenbankservers gelöscht, sofern sie nicht explizit gelöscht wurden.

Systemprivilegien

Sie müssen das `MANAGE ANY TRACE SESSION`-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Ereignisprotokollierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE TEMPORARY TRACE EVENT-Anweisung“ auf Seite 753
- „ALTER TRACE EVENT SESSION-Anweisung“ auf Seite 538
- „DROP TRACE EVENT-Anweisung“ auf Seite 835
- „DROP TRACE EVENT SESSION-Anweisung“ auf Seite 836
- „NOTIFY TRACE EVENT-Anweisung“ auf Seite 963
- „sp_trace_events-Systemprozedur“ auf Seite 1407
- „sp_trace_event_fields-Systemprozedur“ auf Seite 1401
- „sp_trace_event_sessions-Systemprozedur“ auf Seite 1406
- „sp_trace_event_session_events-Systemprozedur“ auf Seite 1402
- „sp_trace_event_session_targets-Systemprozedur“ auf Seite 1405
- „sp_trace_event_session_target_options-Systemprozedur“ auf Seite 1403
- „Dienstprogramm für die Verwaltung von ETD-Dateien (dbmanageetd)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -sf“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung erstellt eine Ereignisprotokollierungssitzung, durch die Informationen zum benutzerdefinierten Ereignis `my_event` und zum systemdefinierten Ereignis `ConsoleLogInformation` in einer Datei namens `my_trace_file` erfasst werden:

```
CREATE TEMPORARY TRACE EVENT SESSION my_session
ADD TRACE EVENT my_event, -- user event
ADD TRACE EVENT SYS_ConsoleLog_Information -- system event
ADD TARGET FILE ( SET filename_prefix='my_trace_file' ); -- add a target
```

CREATE TEXT CONFIGURATION-Anweisung

Erstellt ein Textkonfigurationsobjekt für die Verwendung bei der Erstellung und Aktualisierung von Textindizes.

Syntax

```
CREATE TEXT CONFIGURATION [ owner.]new-config-name
FROM [ owner.]existing-config-name
```

Parameter

FROM-Klausel Damit wird der Name eines Textkonfigurationsobjekts angegeben, das als Vorlage für ein neues verwendet werden soll. Die Namen der Standard-Textkonfigurationsobjekte sind `default_char` und `default_nchar`.

Wenn Sie ein Textkonfigurationsobjekt erstellen, werden die Datenbankoptionen, die sich darauf auswirken, wie Datums- und Zeitwerte in Zeichenfolgen konvertiert werden, aus den Textkonfigurationsobjektvorlagen `default_char` und `default_nchar` kopiert.

Bemerkungen

Sie erstellen ein Textkonfigurationsobjekt mit einem anderen Textkonfigurationsobjekt als Vorlage und ändern die Optionen beliebig mit der `ALTER TEXT CONFIGURATION`-Anweisung.

Um die Liste aller Textkonfigurationsobjekte in der Datenbank einschließlich ihrer Einstellungen anzuzeigen, fragen Sie die `SYSTEXTCONFIG`-Systemansicht ab.

Privilegien

Sie müssen das `CREATE TEXT CONFIGURATION`-Systemprivileg haben, um Textkonfigurationsobjekte erstellen zu können, deren Eigentümer Sie sind. Sie müssen das `CREATE ANY TEXT CONFIGURATION`-Systemprivileg oder das `CREATE ANY OBJECT`-Systemprivileg haben, um Textkonfigurationsobjekte erstellen zu können, deren Eigentümer andere Benutzer sind.

Alle Textkonfigurationsobjekte haben `PUBLIC`-Zugang. Jeder Benutzer mit dem Privileg zum Erstellen von Textindizes kann auch jedes beliebige Textkonfigurationsobjekt verwenden.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Volltextsuche“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- Standard-Textkonfigurationsobjekte [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Datenbankoptionen mit Auswirkungen auf Textkonfigurationsobjekte“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „SYSTEXTCONFIG-Systemansicht“ auf Seite 1499
- „Praktische Einführung: Volltextsuche in einem `GENERIC`-Textindex durchführen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Praktische Einführung: Eine Fuzzy-Volltextsuche durchführen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „`ALTER TEXT CONFIGURATION`-Anweisung“ auf Seite 532
- „`DROP TEXT CONFIGURATION`-Anweisung“ auf Seite 833
- „`sa_refresh_text_indexes`-Systemprozedur“ auf Seite 1295

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende CREATE TEXT CONFIGURATION-Anweisung erstellt ein Textkonfigurationsobjekt namens max_term_sixteen mit dem Textkonfigurationsobjekt default_char. Die folgende ALTER TEXT CONFIGURATION-Anweisung ändert die Begriff-Höchstlänge für max_term_sixteen auf 16.

```
CREATE TEXT CONFIGURATION max_term_sixteen FROM default_char;
ALTER TEXT CONFIGURATION max_term_sixteen
    MAXIMUM TERM LENGTH 16;
```

CREATE TEXT INDEX-Anweisung

Erstellt einen Textindex.

Syntax

```
CREATE TEXT INDEX [ IF NOT EXISTS ] text-index-name
ON [ owner. ] { table-name | mv-name } ( column-name, ... )
[ IN dbspace-name ]
[ CONFIGURATION [ owner. ] text-configuration-name ]
[ { IMMEDIATE REFRESH
  | MANUAL REFRESH
  | AUTO REFRESH [ EVERY integer { MINUTES | HOURS } ] } ]
```

Parameter

IF NOT EXISTS-Klausel Wenn die IF NOT EXISTS-Klausel angegeben wurde und der benannte Textindex vorhanden ist, werden keine Änderungen vorgenommen und es wird kein Fehler zurückgegeben.

ON-Klausel Geben Sie die Tabelle und die Spalten an, für die der Textindex erstellt werden soll.

IN-Klausel Geben Sie den DBSpace an, in dem sich der Textindex befindet. Wenn diese Klausel nicht angegeben ist, wird der Textindex in demselben DBSpace erstellt wie die von ihm referenzierte Tabelle.

CONFIGURATION-Klausel Geben Sie das Textkonfigurationsobjekt an, das beim Erstellen des Textindexes verwendet werden soll. Wenn diese Klausel nicht angegeben ist, wird das Textkonfigurationsobjekt default_nchar verwendet, sofern eine der Spalten im Index NCHAR ist, sonst wird das Textkonfigurationsobjekt default_char verwendet.

REFRESH-Klausel Geben Sie den Aktualisierungstyp für den Textindex an. Wenn Sie keine REFRESH-Klausel angeben, wird IMMEDIATE REFRESH als Standardwert verwendet. Sie können folgende Aktualisierungstypen festlegen:

- **IMMEDIATE REFRESH** Aktualisiert den Textindex jedes Mal, wenn Änderungen an der Basistabelle oder der materialisierten Ansicht Auswirkungen auf Daten im Textindex haben.
- **AUTO REFRESH** Aktualisiert den Textindex automatisch mit einem internen Serverereignis. Verwenden Sie die EVERY-Unterklausel, um das Aktualisierungsintervall in Minuten oder Stunden anzugeben. Wenn Sie AUTO REFRESH ohne Intervallinformationen angeben, aktualisiert der Datenbankserver den Textindex jeweils nach 60 Minuten. Ein Textindex kann früher als durch die AUTO REFRESH-Klausel angegeben aktualisiert werden, wenn der pending_size-Wert, der von der sa_text_index_stats-Systemprozedur zurückgegeben wird, 20% der Textindexgröße bei der letzten

Aktualisierung überschreitet oder wenn der deleted_lenght-Wert 50% der Textindexgröße überschreitet. Ein internes Ereignis wird einmal pro Minute durchgeführt, um diesen Zustand für alle AUTO REFRESH-Textindizes zu prüfen.

- **MANUAL REFRESH** Der Textindex wird manuell aktualisiert.

Weitere Hinweise zu Aktualisierungstypen finden Sie unter „[Textindex-Aktualisierungstypen](#)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Bemerkungen

Sie können keinen Textindex für eine reguläre Ansicht oder eine temporäre Tabelle erstellen.

Nachdem ein Textindex für eine materialisierte Ansicht erstellt wurde, kann er nicht aktualisiert oder gekürzt, sondern nur gelöscht werden. Der Textindex für eine materialisierte Ansicht wird vom Datenbankserver gepflegt, wenn die zugrunde liegende materialisierte Ansicht aktualisiert oder aktualisiert wird.

Diese Anweisung kann nicht ausgeführt werden, wenn mit der WITH HOLD-Klausel geöffnete Cursor vorhanden sind, die entweder Anweisungs- oder Transaktions-Snapshots verwenden.

Ein IMMEDIATE REFRESH-Textindex wird während der Erstellung mit Daten gefüllt und für die Tabelle wird während dieser ersten Aktualisierung eine Exklusivsperrung aufrechterhalten. IMMEDIATE REFRESH-Textindizes bieten umfassende Unterstützung für Abfragen, die die Snapshot-Isolation verwenden.

Ein IMMEDIATE REFRESH-Textindex für eine materialisierte Ansicht wird während der Erstellung mit Daten gefüllt, wenn die Ansicht mit Daten gefüllt ist.

MANUAL- und AUTO REFRESH-Textindizes müssen initialisiert (aktualisiert) werden, nachdem sie erstellt wurden.

Die Aktualisierung von AUTO REFRESH-Textindizes durchsucht die Tabelle unter Verwendung der Isolationsstufe 0.

Nachdem ein Textindex erstellt wurde, können Sie ihn nicht mehr von oder auf IMMEDIATE REFRESH ändern. Wenn eine dieser Änderungen erforderlich ist, müssen Sie den Textindex löschen und wiederherstellen.

Sie können einen AUTO REFRESH-Textindex manuell aktualisieren, indem Sie die REFRESH TEXT INDEX-Anweisung ausführen.

Hinweise zur Anzeige von Textindizes und der darauf Bezug nehmenden Textkonfigurationsobjekte finden Sie unter „[Begriffe und Einstellungen für Textindizes anzeigen \(Sybase Central\)](#)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Privilegien

Wenn Sie einen Textindex für eine Tabelle erstellen möchten, müssen Sie der Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- REFERENCES-Privileg für die Tabelle
- CREATE ANY INDEX-Systemprivileg
- CREATE ANY OBJECT-Systemprivileg

Wenn Sie einen Textindex für eine materialisierte Ansicht erstellen möchten, müssen Sie Eigentümer der materialisierten Ansicht sein oder eines der folgenden Privilegien haben:

- CREATE ANY INDEX-Systemprivileg
- CREATE ANY OBJECT-Systemprivileg

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Praktische Einführung: Volltextsuche in einem GENERIC-Textindex durchführen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „isolation_level-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Snapshot-Isolation“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „REFRESH TEXT INDEX-Anweisung“ auf Seite 990
- „Praktische Einführung: Eine Fuzzy-Volltextsuche durchführen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Konzepte und Referenz zu Textindizes“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „SYSTEXTCONFIG-Systemansicht“ auf Seite 1499
- „ALTER TEXT INDEX-Anweisung“ auf Seite 536
- „DROP TEXT INDEX-Anweisung“ auf Seite 834
- „REFRESH TEXT INDEX-Anweisung“ auf Seite 990
- „TRUNCATE TEXT INDEX-Anweisung“ auf Seite 1091
- „sa_char_terms-Systemprozedur“ auf Seite 1171
- „sa_nchar_terms-Systemprozedur“ auf Seite 1277
- „sa_refresh_text_indexes-Systemprozedur“ auf Seite 1295
- „sa_text_index_stats-Systemprozedur“ auf Seite 1342

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird der Textindex myTxtIdx für die Spalte Description der Tabelle MarketingInformation in der Beispieldatenbank erstellt. Das MarketingTextConfig-Textkonfigurationsobjekt wird verwendet und das Aktualisierungsintervall ist auf alle 24 Stunden gesetzt.

```
CREATE TEXT INDEX myTxtIdx ON GROUP0.MarketingInformation ( Description )
CONFIGURATION default_char
AUTO REFRESH EVERY 24 HOURS;
```

CREATE TRIGGER-Anweisung

Erstellt einen Trigger für eine Tabelle.

Syntax

```
CREATE [ OR REPLACE ] TRIGGER trigger-name trigger-type
{ trigger-event-list | UPDATE OF column-list }
[ ORDER integer ] ON table-name
[ REFERENCING [ OLD AS old-name ]
  [ NEW AS new-name ]
  [ REMOTE AS remote-name ] ]
[ FOR EACH { ROW | STATEMENT } ]
[ WHEN ( search-condition ) ]
trigger-body
```

column-list : *column-name*[, ...]

trigger-type :
BEFORE
| AFTER
| INSTEAD OF
| RESOLVE

trigger-event-list : *trigger-event*[, ...]

trigger-event :
DELETE
| INSERT
| UPDATE (*column-name*)
| UPDATING [(*column-name-string*)]

trigger-body : eine BEGIN-Anweisung. Siehe „[BEGIN-Anweisung](#)“ auf Seite 557.

Parameter

OR REPLACE-Klausel Wenn Sie OR REPLACE angeben, wird ein neuer Trigger erstellt oder ein bestehender Trigger mit demselben Namen ersetzt.

trigger-event Trigger können durch die folgenden Ereignisse ausgelöst werden. Sie können mehrfache Trigger für DELETE-, INSERT- oder UPDATE-Ereignisse oder einen Trigger für das Ereignis UPDATE OF *column-list* festlegen:

- **DELETE-Klausel** Wird jedes Mal aufgerufen, wenn eine Zeile in der zugeordneten Tabelle gelöscht wird.
- **INSERT-Klausel** Wird jedes Mal aufgerufen, wenn eine neue Zeile in die dem Trigger zugeordnete Tabelle eingefügt wird.
- **UPDATE-Klausel** Wird jedes Mal aufgerufen, wenn eine Zeile in der zugeordneten Tabelle aktualisiert wird.

Wenn Sie eine UPDATE-Klausel verwenden, müssen Sie auch eine REFERENCING-Klausel angeben, um Syntaxfehler zu vermeiden.

- **UPDATE OF *column-list*-Klausel** Wird jedes Mal aufgerufen, wenn eine Zeile der zugeordneten Tabelle aktualisiert und eine Spalte in *column-list* geändert wird. Dieser Typ eines Triggerereignisses kann nicht in einer *trigger-event-list* verwendet werden. Es muss sich um das einzige für den Trigger definierte Ereignis handeln. Diese Klausel darf nicht in einem INSTEAD OF-Trigger verwendet werden.

Sie können unterschiedliche Trigger für jedes zu behandelnde Ereignis schreiben. Wenn Sie einige gemeinsame Aktionen und von einem Ereignis abhängige Aktionen verwenden, können Sie auch einen Trigger für alle Ereignisse erstellen und eine IF-Anweisung zur Unterscheidung der einzelnen durchgeführten Aktionen einsetzen.

- **UPDATING-Klausel** Das Argument für UPDATING ist eine Zeichenfolge in Apostrophen (zum Beispiel: UPDATING('mycolumn')). Das Argument für UPDATE ist ein Bezeichner (zum Beispiel UPDATE(mycolumn)). Beide Versionen sind vollständig kompatibel und werden zur Kompatibilität mit dem SQL-Dialekt der Datenbankmanagement-Systeme anderer Anbieter einbezogen.

Wenn Sie eine UPDATING-Klausel verwenden, müssen Sie auch eine REFERENCING-Klausel angeben, um Syntaxfehler zu vermeiden.

trigger-type Trigger auf Zeilenebene können so definiert werden, dass sie vor (BEFORE), nach (AFTER) oder als Ersatz für (INSTEAD OF) einen Einfügings-, Aktualisierungs- oder Löschvorgang ausgeführt werden. Trigger auf Anweisungsebene können so definiert werden, dass sie als Ersatz für (INSTEAD OF) oder nach (AFTER) der Anweisung ausgeführt werden.

BEFORE UPDATE-Trigger werden jedes Mal ausgelöst, wenn eine UPDATE-Anweisung für eine Zeile ausgeführt wird, ganz gleich, ob sich der neue Wert vom alten unterscheidet. Das bedeutet: Wenn eine *column-list* für einen BEFORE UPDATE-Trigger angegeben wurde, wird der Trigger ausgelöst, sobald Spalten aus der *column-list* in der SET-Klausel der UPDATE-Anweisung erscheinen. Wenn eine *column-list* für einen AFTER UPDATE-Trigger angegeben wurde, wird der Trigger nur ausgelöst, wenn der Wert einer Spalte aus der *column-list* durch die UPDATE-Anweisung *geändert* wird.

INSTEAD OF-Trigger sind die einzige Art von Triggern, die Sie für eine reguläre Ansicht definieren können. INSTEAD OF-Trigger ersetzen die Trigger-Aktion durch eine andere Aktion. Bei Auslösen eines INSTEAD OF-Triggers wird die Trigger-auslösende Aktion übersprungen und stattdessen die angegebene Aktion durchgeführt. INSTEAD OF-Trigger können als Trigger auf Zeilenebene oder auf Anweisungsebene definiert werden. Ein auf Anweisungsebene definierter INSTEAD OF-Trigger ersetzt die gesamte Anweisung, einschließlich aller Vorgänge auf Zeilenebene. Wenn ein INSTEAD OF-Trigger auf Anweisungsebene ausgelöst wird, werden als Folge dieser Anweisung keine Trigger auf Zeilenebene ausgelöst. Der Hauptteil des Triggers auf Anweisungsebene könnte jedoch andere Vorgänge durchführen, die wiederum bewirken könnten, dass andere Trigger auf Zeilenebene ausgelöst werden.

Wenn Sie einen INSTEAD OF-Trigger definieren, können Sie nicht die Klausel UPDATE OF *column-list*, die ORDER-Klausel oder die WHEN-Klausel verwenden.

Der Triggertyp RESOLVE wird mit SQL Remote verwendet: Er wird nur vor UPDATE oder UPDATE OF *column-list* auf Zeilenebene ausgelöst.

ORDER-Klausel Wenn Sie zusätzliche Trigger desselben Typs (INSERT, UPDATE oder DELETE) definieren, die zu demselben Zeitpunkt auslösen sollen (BEFORE, AFTER oder RESOLVE), müssen Sie

eine ORDER-Klausel angeben, um den Datenbankserver anzuweisen, in welcher Reihenfolge er die Trigger auslösen soll. Die Ordnungsnummern müssen bei Triggern desselben Typs, die zum gleichzeitigen Auslösen konfiguriert wurden, eindeutig sein. Wenn Sie eine Ordnungsnummer eingeben, die nicht eindeutig ist, wird ein Fehler zurückgegeben. Ordnungsnummern müssen nicht aufeinanderfolgend sein (Eingabe von 1, 12, 30 ist zulässig). Der Datenbankserver löst den Trigger mit der niedrigsten Ordnungsnummer zuerst aus.

Wenn Sie die ORDER-Klausel weglassen oder 0 angeben, weist der Datenbankserver die Ordnungsnummer 1 zu. Wenn jedoch bereits ein anderer Trigger desselben Typs auf 1 gesetzt ist, wird ein Fehler zurückgegeben.

Wenn Sie zusätzliche Trigger hinzufügen, müssen Sie unter Umständen die bestehenden Trigger desselben Typs für das Ereignis ändern, je nachdem, ob die Aktionen der Trigger miteinander interagieren. Wenn sie nicht interagieren, muss der neue Trigger einen ORDER-Wert haben, der höher ist als diejenigen der vorhandenen Trigger. Wenn sie interagieren, müssen Sie überlegen, was die anderen Trigger tun, und gegebenenfalls die Reihenfolge ihres Auslösens ändern.

Die ORDER-Klausel wird für INSTEAD OF-Trigger nicht unterstützt, da nur ein INSTEAD OF-Trigger eines Typs (INSERT, UPDATE, DELETE) für eine Tabelle der Ansicht definiert werden kann.

REFERENCING-Klausel Mit den Klauseln REFERENCING OLD und REFERENCING NEW können Sie eingefügte, gelöschte oder aktualisierte Zeilen referenzieren. Bei dieser Klausel wird ein UPDATE wie ein Löschvorgang behandelt, dem ein Einfügevorgang folgt.

INSERT übernimmt die REFERENCING NEW-Klausel, welche die eine eingefügte Zeile darstellt. Es gibt keine REFERENCING OLD-Klausel.

DELETE übernimmt die REFERENCING OLD-Klausel, welche die gelöschte Zeile darstellt. Es gibt keine REFERENCING NEW-Klausel.

UPDATE übernimmt die REFERENCING OLD-Klausel, die die Zeile vor der Aktualisierung darstellt, sowie die REFERENCING NEW-Klausel, die die Zeile nach der Aktualisierung darstellt.

Die Bedeutung von REFERENCING OLD und REFERENCING NEW ist unterschiedlich, je nachdem, ob es sich um einen Trigger auf Zeilenebene oder auf Anweisungsebene handelt. Für Trigger auf Zeilenebene können Sie mit der REFERENCING OLD-Klausel Werte in einer Zeile vor einem Aktualisierungs- oder Löschvorgang referenzieren und mit der REFERENCING NEW-Klausel können Sie eingefügte oder aktualisierte Werte referenzieren. Die OLD- und NEW-Zeilen können in BEFORE- und AFTER-Trigger referenziert werden. Mit der REFERENCING NEW-Klausel können Sie die neue Zeile in einem BEFORE-Trigger ändern, bevor Einfügings- oder Aktualisierungsvorgänge stattfinden.

Wenn Trigger auf Anweisungsebene verwendet werden, beziehen sich die Klauseln REFERENCING OLD und REFERENCING NEW auf deklarierte temporäre Tabellen, welche die alten und neuen Werte der Zeilen enthalten.

Die REFERENCING REMOTE-Klausel wird mit SQL Remote verwendet. Mit dieser Klausel können Sie die Werte in der VERIFY-Klausel einer UPDATE-Anweisung referenzieren. Sie sollte nur mit Trigger für RESOLVE UPDATE- oder RESOLVE UPDATE OF-Spaltenlisten verwendet werden.

FOR EACH-Klausel Mit der FOR EACH ROW-Klausel können Sie einen Trigger als Trigger auf Zeilenebene deklarieren. Um einen Trigger auf Anweisungsebene zu deklarieren, können Sie entweder

eine FOR EACH STATEMENT-Klausel verwenden oder die FOR EACH-Klausel weglassen. Aus Gründen der Übersichtlichkeit wird empfohlen, die FOR EACH STATEMENT-Klausel anzugeben, wenn Sie einen Trigger auf Anweisungsebene deklarieren.

WHEN-Klausel Der Trigger wird nur für Zeilen ausgelöst, in denen die Suchbedingung mit TRUE ausgewertet wird. Die WHEN-Klausel kann nur mit Trigger auf Zeilenebene verwendet werden. Diese Klausel darf nicht in einem INSTEAD OF-Trigger verwendet werden.

trigger-body Der Trigger-Hauptteil enthält die Aktionen, die ausgeführt werden, wenn das Trigger-auslösende Ereignis eintritt, und besteht aus einer BEGIN-Anweisung.

Sie können Bedingungen für die Triggervorgänge in die BEGIN-Anweisung aufnehmen. Bedingungen für Triggervorgänge führen die Aktionen je nach dem Trigger-Ereignis durch, das den Trigger ausgelöst hat. Beispiel: Wenn der Trigger bei Aktualisierungen und Löschung auslösen soll, können Sie für diese beiden Bedingungen unterschiedliche Aktionen definieren.

Bemerkungen

Die CREATE TRIGGER-Anweisung erstellt einen Trigger, der einer Tabelle in der Datenbank zugeordnet ist, und speichert den Trigger in der Datenbank.

Sie können keinen Trigger für eine materialisierte Ansicht erstellen. Wenn Sie es versuchen, wird ein SQLE_INVALID_TRIGGER_MATVIEW-Fehler zurückgegeben.

Der Trigger wird entweder als Trigger auf Zeilenebene deklariert (und dann vor bzw. nach Änderung jeder Zeile ausgeführt) oder als Trigger auf Anweisungsebene deklariert (und dann nach Abschluss der gesamten Trigger-auslösenden Anweisung ausgeführt).

CREATE TRIGGER setzt eine Tabellensperre für die betreffende Tabelle und erfordert eine exklusive Verwendung der Tabelle.

Privilegien

Sie müssen das CREATE ANY TRIGGER-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben. Außerdem müssen Sie der Eigentümer der Tabelle sein, aus der der Trigger erstellt wird, oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Tabelle
- ALTER ANY TABLE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Wenn Sie einen Trigger in einer Ansicht eines anderen Eigentümers erstellen möchten, benötigen Sie entweder das CREATE ANY TRIGGER-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg sowie entweder das ALTER ANY VIEW-Systemprivileg oder das ALTER ANY OBJECT-Systemprivileg.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „BEGIN-Anweisung“ auf Seite 557
- „CREATE TRIGGER-Anweisung [T-SQL]“ auf Seite 769
- „DROP TRIGGER-Anweisung“ auf Seite 837
- „ROLLBACK TRIGGER-Anweisung“ auf Seite 1017
- „UPDATE-Anweisung“ auf Seite 1109
- „ALTER TRIGGER-Anweisung“ auf Seite 539
- „RAISERROR-Anweisung“ auf Seite 982
- „CONFLICT-Funktion [Verschiedene]“ auf Seite 194
- „INSTEAD OF-Trigger“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „SQL-Anweisungen für die Implementierung von Integritätsregeln“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Trigger für Tabellen erstellen (Sybase Central)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Trigger“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** CREATE TRIGGER ist Teil der optionalen SQL-Sprachenfunktion T211 (Basis-Triggerfähigkeit) des SQL/2008-Standards. ROW-Trigger sind die optionale SQL-Sprachenfunktion T212, INSTEAD OF-Trigger die optionale SQL-Sprachenfunktion T213.

Einige Funktionen von SQL Anywhere-Triggern sind Erweiterungen des Herstellers. Es handelt sich dabei um die folgenden:

- Die optionale OR REPLACE-Syntax. Beim Ersetzen eines bestehenden Triggers wird die Autorisierung zur Erstellung der neuen Trigger-Instanz umgangen.
- Die ORDER-Klausel. In SQL/2008 werden Trigger in der Reihenfolge ausgelöst, in der sie erstellt wurden.
- RESOLVE-Trigger sind eine Erweiterung des Herstellers.
- **Transact-SQL** ROW- und RESOLVE-Trigger werden von Adaptive Server Enterprise nicht unterstützt. Der Transact-SQL-Dialekt von SQL Anywhere bietet keine Unterstützung für INSTEAD OF-Trigger aus Transact-SQL, obwohl diese von Adaptive Server Enterprise unterstützt werden. Transact-SQL-Trigger werden mit einer anderen Syntax definiert.

Beispiel

Mit diesem Beispiel wird ein Trigger auf Anweisungsebene erstellt. Erstellen Sie zuerst eine Tabelle wie in dieser CREATE TABLE-Anweisung (erfordert das CREATE TABLE-Systemprivileg):

```
CREATE TABLE t0
( id INTEGER NOT NULL,
  times TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
  remarks TEXT NULL,
  PRIMARY KEY ( id )
);
```

Als Nächstes erstellen Sie einen Trigger auf Anweisungsebene für diese Tabelle:

```

CREATE TRIGGER myTrig AFTER INSERT ORDER 4 ON t0
REFERENCING NEW AS new_name
FOR EACH STATEMENT
BEGIN
    DECLARE @idl INTEGER;
    DECLARE @times1 TIMESTAMP;
    DECLARE @remarks1 LONG VARCHAR;
    DECLARE @err_notfound EXCEPTION FOR SQLSTATE VALUE '02000';
    //declare a cursor for table new_name
    DECLARE new1 CURSOR FOR
        SELECT id, times, remarks FROM new_name;
    OPEN new1;
    //Open the cursor, and get the value
    LoopGetRow:
    LOOP
        FETCH NEXT new1 INTO @idl, @times1,@remarks1;
        IF SQLSTATE = @err_notfound THEN
            LEAVE LoopGetRow
        END IF;
        //print the value or for other use
        PRINT (@remarks1);
    END LOOP LoopGetRow;
    CLOSE new1
END;

```

Im folgenden Beispiel wird der Trigger MyTrig ersetzt, der im vorherigen Beispiel erstellt wurde.

```

CREATE OR REPLACE TRIGGER myTrig AFTER INSERT ORDER 4 ON t0
REFERENCING NEW AS new_name
FOR EACH STATEMENT
BEGIN
    FOR L1 AS new1 CURSOR FOR
        SELECT id, times, remarks FROM new_name
    DO
        //print the value or for other use
        PRINT (@remarks1);
    END FOR;
END;

```

Das folgende Beispiel zeigt, wie Sie REFERENCING NEW in einem BEFORE UPDATE-Tigger verwenden können. Hier wird gewährleistet, dass die Postleitzahlen in der neuen Employees-Tabelle in Großbuchstaben geschrieben werden. Sie müssen die Privilegien SELECT, ALTER und UPDATE für GROUPO.Employees haben, um diese Anweisung ausführen zu können:

```

CREATE TRIGGER emp_upper_postal_code
BEFORE UPDATE OF PostalCode
ON GROUPO.Employees
REFERENCING NEW AS new_emp
FOR EACH ROW
WHEN ( ISNUMERIC( new_emp.PostalCode ) = 0 )
BEGIN
    -- Ensure postal code is uppercase (employee might be
    -- in Canada where postal codes contain letters)
    SET new_emp.PostalCode = UPPER(new_emp.PostalCode)
END;

UPDATE GROUPO.Employees SET state='ON', PostalCode='n2x 4y7' WHERE
EmployeeID=191;
SELECT PostalCode FROM GROUPO.Employees WHERE EmployeeID = 191;

```

Das folgende Beispiel zeigt, wie Sie REFERENCING OLD in einem BEFORE DELETE-Trigger verwenden können. Dieses Beispiel verhindert, dass ein Mitarbeiter aus der Tabelle "Employees" gelöscht wird, mit dem das Arbeitsverhältnis noch besteht.

```
CREATE TRIGGER TR_check_delete_employee
BEFORE DELETE
ON Employees
REFERENCING OLD AS current_employee
FOR EACH ROW WHEN ( current_employee.Terminate IS NULL )
BEGIN
    RAISERROR 30001 'You cannot delete an employee who has not been fired';
END;
```

Das folgende Beispiel zeigt, wie Sie REFERENCING NEW und REFERENCING OLD in einem BEFORE UPDATE-Trigger verwenden können. Dieses Beispiel verhindert eine Gehaltsminderung bei einem Mitarbeiter.

```
CREATE TRIGGER TR_check_salary_decrease
BEFORE UPDATE
ON GROUPO.Employees
REFERENCING OLD AS before_update
NEW AS after_update
FOR EACH ROW
BEGIN
    IF after_update.salary < before_update.salary THEN
        RAISERROR 30002 'You cannot decrease a salary';
    END IF;
END;
```

Das nächste Beispiel zeigt, wie Sie REFERENCING NEW in einem BEFORE INSERT- und BEFORE UPDATE-Trigger verwenden können. Im folgenden Beispiel wird ein Trigger erstellt, der ausgelöst wird, bevor eine Zeile in der Tabelle "SalesOrderItems" eingefügt oder aktualisiert wird.

```
CREATE TRIGGER TR_update_date
BEFORE INSERT, UPDATE
ON GROUPO.SalesOrderItems
REFERENCING NEW AS new_row
FOR EACH ROW
BEGIN
    SET new_row.ShipDate = CURRENT_TIMESTAMP;
END;
```

Der folgende Trigger gibt im Interactive SQL-Fensterausschnitt **Ergebnisse** auf der Registerkarte **Meldungen** eine Meldung aus, in der angezeigt wird, welche Aktion den Trigger ausgelöst hat.

```
CREATE TRIGGER tr BEFORE INSERT, UPDATE, DELETE
ON sample_table
REFERENCING OLD AS t1old
FOR EACH ROW
BEGIN
    DECLARE msg varchar(255);
    SET msg = 'This trigger was fired by an ';
    IF INSERTING THEN
        SET msg = msg || 'insert'
    ELSEIF DELETING THEN
        set msg = msg || 'delete'
    ELSEIF UPDATING THEN
        set msg = msg || 'update'
    END IF;
    MESSAGE msg TO CLIENT
END;
```

CREATE TRIGGER-Anweisung [T-SQL]

Erstellt einen neuen Trigger in der Datenbank so, dass er mit Adaptive Server Enterprise kompatibel ist.

Syntax 1

```
CREATE TRIGGER [owner.]trigger_name
ON [owner.]table_name
FOR { INSERT, UPDATE, DELETE }
AS statement-list
```

Syntax 2

```
CREATE TRIGGER [owner.]trigger_name
ON [owner.]table_name
FOR {INSERT, UPDATE}
AS
[ IF UPDATE ( column-name )
[ { AND | OR } UPDATE ( column-name ) ] ... ]
statement-list
[ IF UPDATE ( column-name )
[ { AND | OR } UPDATE ( column-name ) ] ... ]
statement-list
```

Bemerkungen

CREATE TRIGGER setzt eine exklusive Tabellensperre für die Tabelle.

Die gelöschten oder eingefügten Zeilen werden in zwei temporären Tabellen gespeichert. Wenn Trigger in Transact-SQL-Form verwendet werden, kann mit den Tabellennamen "deleted" und "inserted" darauf zugegriffen werden, ebenso wie in Adaptive Server Enterprise. In der Watcom-SQL-Anweisung CREATE TRIGGER können Sie mit der REFERENCING-Klausel auf diese Zeilen zugreifen.

Triggernamen müssen in der Datenbank eindeutig sein.

Transact-SQL-Trigger werden NACH der Trigger-auslösenden Anweisung ausgeführt.

Da die ORDER-Klausel beim Erstellen von Transact-SQL Triggern nicht unterstützt wird, wird der Wert von trigger_order auf 1 gesetzt. Die SYSTRIGGER-Systemtabelle enthält einen eindeutigen Index für table_id, event, trigger_time und trigger_order. Bei einem bestimmten Ereignis (INSERT, UPDATE, DELETE) sind Trigger auf Anweisungsebene immer AFTER und trigger_order kann nicht festgelegt werden. Daher kann es nur einen jedes Typs pro Tabelle geben, unter der Annahme, dass kein anderer Trigger eine andere Reihenfolge als 1 setzt.

Privilegien

Sie müssen Eigentümer der Tabelle sein, das ALTER-Privileg für die Tabelle haben oder das ALTER ANY TABLE-Systemprivileg haben. Außerdem benötigen Sie das CREATE ANY TRIGGER-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE TRIGGER-Anweisung“ auf Seite 762
- „Transact-SQLTrigger“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** ROW-Trigger werden von Adaptive Server Enterprise nicht unterstützt. Der Transact-SQL-Dialekt von SQL Anywhere bietet keine Unterstützung für INSTEAD OF-Trigger aus Transact-SQL, obwohl diese von Adaptive Server Enterprise unterstützt werden.

CREATE USER-Anweisung

Erstellt einen Datenbankbenutzer oder eine Gruppe.

Syntax

```
CREATE USER user-name [ IDENTIFIED BY password ]  
[ LOGIN POLICY policy-name ]  
[ FORCE PASSWORD CHANGE { ON | OFF } ]
```

Parameter

user-name Der Name des Benutzers, den Sie erstellen.

IDENTIFIED BY-Klausel Das Kennwort des Benutzers. Ein Benutzer ohne Kennwort kann keine Verbindung zur Datenbank herstellen.

policy-name Der Name der Login-Richtlinie, die dem Benutzer zugewiesen werden soll. Wenn keine Login-Richtlinie definiert ist, wird die DEFAULT-Login-Richtlinie verwendet.

FORCE PASSWORD CHANGE-Klausel Steuert, ob der Benutzer beim Anmelden ein neues Kennwort eingeben muss. Diese Einstellung setzt die Einstellung der password_expiry_on_next_login-Option in der Richtlinie des Benutzers außer Kraft.

Bemerkungen

Sie brauchen kein Kennwort für den Benutzer anzugeben. Ein Benutzer ohne Kennwort kann keine Verbindung zur Datenbank herstellen. Das ist sinnvoll, wenn Sie eine Gruppe erstellen und nicht möchten, dass jemand mit der Benutzer-ID der Gruppe eine Verbindung zur Datenbank herstellt. Eine Benutzer-ID muss ein gültiger Bezeichner sein.

- Benutzer-IDs dürfen Folgendes nicht:
 - Sie dürfen nicht mit Leerstellen, Apostrophen oder Anführungszeichen beginnen.
 - Sie dürfen nicht mit Leerstellen enden.
 - Sie dürfen keine Semikola enthalten.

- Kennwörter berücksichtigen die Groß- und Kleinschreibung. Im Übrigen gilt Folgendes:
 - Sie dürfen nicht mit Leerstellen, Apostrophen oder Anführungszeichen beginnen.
 - Sie dürfen nicht mit Leerstellen enden.
 - Sie dürfen keine Semikola enthalten.
 - länger als 255 Byte sein

Privilegien

Sie müssen das MANAGE ANY USER-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „CREATE LOGIN POLICY-Anweisung“ auf Seite 647
- „ALTER LOGIN POLICY-Anweisung“ auf Seite 471
- „ALTER USER-Anweisung“ auf Seite 540
- „COMMENT-Anweisung“ auf Seite 573
- „DROP LOGIN POLICY-Anweisung“ auf Seite 812
- „DROP USER-Anweisung“ auf Seite 838
- „Login-Richtlinien“ [*SQL Anywhere Server - Datenbankadministration*]
- „Benutzer erstellen (Sybase Central)“ [*SQL Anywhere Server - Datenbankadministration*]
- „GRANT-Anweisung“ auf Seite 881
- „min_password_length-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel erstellt einen Benutzer namens SQLTester mit dem Kennwort "welcome". Der Benutzer SQLTester wird aufgefordert, ein Kennwort anzugeben, wenn er sich anmeldet.

```
CREATE USER SQLTester IDENTIFIED BY welcome
FORCE PASSWORD CHANGE ON;
```

CREATE VARIABLE-Anweisung

Erstellt eine SQL-Variable.

Syntax

```
CREATE [ OR REPLACE ] VARIABLE identifier data-type [ { = | DEFAULT } initial-value ]
```

initial-value :

special-value

| *string*

| [-] *number*

| (*constant-expression*)

```
| built-in-function ( constant-expression )  
| NULL
```

```
special-value :  
CURRENT {  
DATABASE  
| DATE  
| PUBLISHER  
| TIME  
| TIMESTAMP  
| USER  
| UTC TIMESTAMP }  
| USER
```

Parameter

OR REPLACE-Klausel Durch Angeben der OR REPLACE-Klausel wird die benannte Variable gelöscht, falls sie bereits vorhanden ist, und ihre Definition wird ersetzt. Sie können die OR REPLACE-Klausel in SQL-Skripten als Alternative zur VAREXISTS-Funktion verwenden.

DEFAULT-Klausel Wenn Sie *initial-value* angeben, muss der Datentyp dem durch *data-type* definierten Typ entsprechen.

Bemerkungen

Die CREATE VARIABLE-Anweisung erstellt eine neue Variable eines angegebenen Datentyps. Wenn Sie einen *initial-value* angeben, wird die Variable auf diesen Wert gesetzt. Wenn Sie keinen *initial-value* angeben, enthält die Variable NULL, bis mit der SET-Anweisung ein anderer Wert zugewiesen wird.

Eine Variable kann in einem SQL-Ausdruck überall dort verwendet werden, wo ein Spaltenname zugelassen ist. Die Namensauflösung wird folgendermaßen durchgeführt:

1. Etwaige in der SELECT-Liste der Abfrage angegebenen Aliasnamen auf Übereinstimmung überprüfen
2. Spaltennamen für referenzierte Tabellen auf Übereinstimmung überprüfen
3. Annehmen, dass der Name eine Variable ist

Variablen gehören zur aktuellen Verbindung und bleiben erhalten, bis Sie die Verbindung zur Datenbank trennen oder die DROP VARIABLE-Anweisung verwenden. Variablen sind für andere Verbindungen nicht sichtbar. Variablen werden von den Anweisungen COMMIT oder ROLLBACK nicht beeinflusst.

Variablen sind für die Erstellung von umfangreichem Text oder binären Objekten für die INSERT- oder UPDATE-Anweisungen von Embedded SQL-Programmen hilfreich.

Lokale Variablen in Prozeduren und Triggern werden in einer zusammengesetzten Anweisung deklariert.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „Zusammengesetzte Anweisungen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „BEGIN-Anweisung“ auf Seite 557
- „SQL-Datentypen“ auf Seite 95
- „DROP VARIABLE-Anweisung“ auf Seite 839
- „SET-Anweisung“ auf Seite 1054
- „VAREXISTS-Funktion [Verschiedene]“ auf Seite 429
- „Spezialwerte“ auf Seite 70

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird eine Variable namens 'first_name' mit dem Datentyp VARCHAR(50) erstellt.

```
CREATE VARIABLE first_name VARCHAR(50);
```

In diesem Beispiel wird eine Variable namens 'birthday' mit dem Datentyp DATE erstellt.

```
CREATE VARIABLE birthday DATE;
```

In diesem Beispiel wird eine Variable mit dem Namen v1 als INT mit der Anfangseinstellung 5 erstellt.

```
CREATE VARIABLE v1 INT = 5;
```

In diesem Beispiel wird eine Variable mit dem Namen v1 erstellt und der Wert auf 10 gesetzt, unabhängig davon, ob die Variable v1 bereits vorhanden ist.

```
CREATE OR REPLACE VARIABLE v1 INT = 10;
```

CREATE VIEW-Anweisung

Erstellt eine Ansicht in der Datenbank.

Syntax

```
CREATE [ OR REPLACE ] VIEW  
[ owner.]view-name [ ( column-name, ... ) ]  
AS query-expression  
[ WITH CHECK OPTION ]
```

Parameter

OR REPLACE-Klausel Wenn Sie OR REPLACE (CREATE OR REPLACE VIEW) angeben, wird eine neue Ansicht erstellt oder eine bestehende Ansicht mit demselben Namen ersetzt. Vorhandene Privilegien bleiben unberührt, wenn Sie die OR REPLACE-Klausel verwenden. INSTEAD OF-Trigger für die Ansicht werden jedoch gelöscht.

Wenn Sie eine CREATE OR REPLACE VIEW-Anweisung für eine Ansicht ausführen, die einen oder mehrere INSTEAD OF-Trigger hat, wird ein Fehler zurückgegeben. Löschen Sie den Trigger, bevor Sie die Ansicht ändern oder löschen.

AS-Klausel Die SELECT-Anweisung, auf der die Ansicht basiert. Die SELECT-Anweisung darf sich nicht auf lokale temporäre Tabellen beziehen. Außerdem kann *query-expression* eine GROUP BY-, HAVING-, WINDOW- oder ORDER BY-Klausel enthalten sowie UNION, EXCEPT, INTERSECT oder einen allgemeinen Tabellenausdruck.

Die Semantik von Abfragen bestimmt, dass die Reihenfolge der zurückgegebenen Zeilen nicht definiert ist, es sei denn, die Abfrage kombiniert eine ORDER BY-Klausel mit einer TOP- oder FIRST-Klausel in der SELECT-Anweisung.

WITH CHECK OPTION-Klausel Die WITH CHECK OPTION-Klausel weist alle Aktualisierungen und Einfügungen der Ansicht zurück, die nicht den Kriterien der von den für *query-expression* festgelegten Werten entsprechen.

Bemerkungen

Ansichten existieren in der Datenbank nicht physisch als Tabellen. Sie werden jedes Mal abgeleitet, wenn sie verwendet werden. Eine Ansicht wird als das Ergebnis einer SELECT-Anweisung abgeleitet, die in einer CREATE VIEW-Anweisung angegeben wird. In einer Ansicht wird empfohlen, die Benutzer-ID des Tabelleneigentümers anzugeben, um zwischen Tabellen mit demselben Namen unterscheiden zu können.

Ein Ansichtsname kann in den Anweisungen SELECT, DELETE, UPDATE und INSERT statt eines Tabellennamens verwendet werden.

SELECT * kann in der Hauptabfrage verwendet werden sowie in einer Unterabfrage, einer abgeleiteten Tabelle oder einer Unterabfrage-Bedingung einer CREATE VIEW-Anweisung.

In einer Abfrage kann eine TOP n-, FIRST- oder LIMIT-Klausel angegeben werden, auch wenn er keine ORDER BY-Klausel enthält. Es wird höchstens die angegebene Zeilenanzahl zurückgegeben, aber die Reihenfolge der zurückgegebenen Zeilen ist nicht definiert. Eine ORDER BY-Klausel kann also angegeben werden, ist aber nicht erforderlich. Wenn in einer Abfrage eine Ansicht verwendet wird, bestimmt ORDER BY in der Ansicht nicht die Reihenfolge der Zeilen in der Abfrage, auch wenn die FROM-Klausel keine anderen Tabellen enthält. Das bedeutet, dass ORDER BY nur dann für eine Ansicht verwendet werden sollte, wenn es benötigt wird, um die Zeilen auszuwählen, die in eine TOP n-, FIRST- oder LIMIT-Klausel einbezogen werden sollen. Andernfalls hat die ORDER BY-Klausel keine Auswirkungen und wird vom Datenbankserver ignoriert.

Ansichten können aktualisiert werden, es sei denn, *query-expression* zur Definition der Ansicht enthält eine GROUP BY-Klausel, eine WINDOW-Klausel oder eine Aggregatfunktion bzw. erfordert einen Mengenoperator (UNION, INTERSECT, EXCEPT). Durch eine Aktualisierung der Ansicht werden auch deren Basistabellen aktualisiert.

Den Spalten in der Ansicht werden die Namen zugeordnet, die in der *column-name*-Liste angegeben sind. Wenn die Spaltennamenliste nicht festgelegt ist, werden den Spalten Namen aus den Elementen der Select-Liste gegeben. Alle Elemente in der Select-Liste müssen eindeutige Namen haben. Um Namen aus den Elementen der SELECT-Liste verwenden zu können, muss jedes Element ein einfacher Spaltenname sein oder einen Aliasnamen besitzen.

SQL Anywhere erlaubt unbenannte Ausdrücke in der Select-Liste von *query-expression*, die in der CREATE VIEW-Anweisung referenziert wird. Unbenannte Ausdrücke in der SELECT-Liste von *query-expression* erhalten den Namen **expression**, verbunden mit einem Ganzzahlwert, wenn mehr als ein

solcher Ausdruck vorhanden ist. Die folgende Anweisung würde Ansicht "V" mit drei Spalten (expression, expression1 und expression2) definieren. Diese Namen würden in der SYSCOLUMN-Systemansicht für die erstellte Ansicht "V" angezeigt.

```
CREATE VIEW V AS
  SELECT DATEADD( DAY, 1, NOW() ), DATEADD( DAY, 2, NOW() ), DATEADD( DAY,
2, NOW() )
  FROM SYS.DUMMY;
```

Es wird nicht empfohlen, dass Sie sich auf diese generierten Namen verlassen, da andere Ansichten mit unbenannten Ausdrücken in der Select-Liste dieselben zugewiesenen Namen haben.

Üblicherweise referenziert eine Ansicht Tabellen und Ansichten (sowie ihre entsprechenden Attribute), die im Katalog definiert sind. Allerdings kann eine Ansicht auch SQL-Variablen referenzieren. Wenn in diesem Fall eine Abfrage, die die Ansicht referenziert, ausgeführt wird, wird der Wert der SQL-Variablen verwendet. Ansichten, die SQL-Variablen referenzieren, werden **parametrisierte Ansichten** genannt, da die Variablen als Parameter der Ausführung der Ansicht agieren.

Parametrisierte Ansichten bieten eine Alternative zum Einbetten des Hauptteils eines äquivalenten SELECT-Blocks in einer Abfrage als abgeleitete Tabelle in der FROM-Klausel der Abfrage. Parametrisierte Ansichten können bei eingebetteten Abfragen in gespeicherten Prozeduren nützlich sein, bei denen die in der Ansicht referenzierten SQL-Variablen Eingabeparameter für die Prozedur sind.

Es ist nicht notwendig, dass die SQL-Variable vorhanden ist, wenn die CREATE VIEW-Anweisung ausgeführt wird. Wenn jedoch eine Abfrage, die die Ansicht referenziert, ausgeführt wird, ohne dass die SQL-Variable definiert ist, wird ein Fehler zurückgegeben, der anzeigt, dass die Spalte (Variable) nicht gefunden werden konnte.

Privilegien

Sie müssen das CREATE VIEW-Systemprivileg haben, um Ansichten erstellen zu können, deren Eigentümer Sie sind. Sie müssen das CREATE ANY VIEW-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben, um Ansichten erstellen zu können, deren Eigentümer andere Benutzer sind.

Wenn die von der Ansicht referenzierten Tabellen Eigentum anderer Benutzer sind, benötigen Sie SELECT-Privilegien für diese Tabellen.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Reguläre Ansichten erstellen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „ALTER VIEW-Anweisung“ auf Seite 543
- „INSTEAD OF-Trigger“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „SELECT-Anweisung“ auf Seite 1020

Standards und Kompatibilität

- **SQL/2008** CREATE VIEW ist eine Kernfunktion des SQL/2008-Standards, aber einige Funktionen einer in eine Ansicht eingebetteten SELECT-Anweisung sind optionale Sprachfunktionen. Die

Möglichkeit zum Angeben einer ORDER BY-Klausel mit der SELECT-Anweisung auf der obersten Ebene in der Ansichtsdefinition ist die optionale SQL/2008-Sprachenfunktion F852. Die Möglichkeit, die Ergebnismenge einer Ansicht mit SELECT TOP oder LIMIT einzuschränken, ist die optionale SQL/2008-Sprachenfunktion F859 (obwohl der SQL/2008-Standard die FETCH-Klausel für diesen Zweck verwendet). Die Möglichkeit, WITH CHECK OPTION für eine Ansicht anzugeben, die nicht aktualisierbar ist, z.B. weil die SELECT-Anweisung der Ansicht eine abgeleitete Tabelle mit Aggregaten oder DISTINCT enthält oder einen Mengenoperator (INTERSECT, EXCEPT oder UNION), ist die optionale SQL/2008-Sprachenfunktion T111.

Einige Funktionen von CREATE VIEW sind Erweiterungen des Herstellers. Parametrisierte Ansichten sind eine Erweiterung des Herstellers, wie auch die optionale OR REPLACE-Syntax und die automatische Generierung von Namen für unbenannte Ausdrücke in der SELECT-Liste.

Beispiel

Im folgenden Beispiel wird eine Ansicht erstellt, die nur Angaben zu den männlichen Mitarbeitern enthält. Die Ansicht verfügt über die gleichen Spaltennamen wie die Basistabelle:

```
CREATE VIEW MaleEmployees
AS SELECT *
FROM GROUPO.Employees
WHERE Sex = 'M';
```

Im folgenden Beispiel wird eine Ansicht mit den Mitarbeitern und ihren jeweiligen Abteilungen erstellt:

```
CREATE VIEW EmployeesAndDepartments
AS SELECT Surname, GivenName, DepartmentName
FROM GROUPO.Employees JOIN GROUPO.Departments
ON Employees.DepartmentID = Departments.DepartmentID;
```

Im folgenden Beispiel wird die Ansicht EmployeesAndDepartments ersetzt, die im vorherigen Beispiel erstellt wurde. Nachdem die Ansicht ersetzt wurde, zeigt sie "city", "state" und "country" für jeden Angestellten:

```
CREATE OR REPLACE VIEW EmployeesAndDepartments
AS SELECT Surname, GivenName, City, State, Country
FROM GROUPO.Employees JOIN GROUPO.Departments
ON Employees.DepartmentID = Departments.DepartmentID;
```

Im folgenden Beispiel wird eine parametrisierte Ansicht auf Basis der Variablen var1 und var2 erstellt, die keine Attribute der Tabellen "Employees" und "Departments" sind:

```
CREATE VIEW EmployeesByState
AS SELECT Surname, GivenName, DepartmentName
FROM GROUPO.Employees JOIN GROUPO.Departments
ON Employees.DepartmentID = Departments.DepartmentID
WHERE Employees.State = var1 and Employees.Status = var2;
```

Variablen können in der SELECT-Anweisung der Ansicht in jedem Kontext auftreten, bei dem eine Variable ein zulässiger Ausdruck ist. Beispiel: Die folgende parametrisierte Ansicht verwendet den Parameter var1 als Muster für ein LIKE-Prädikat:

```
CREATE VIEW ProductsByDescription
AS SELECT *
FROM GROUPO.Products
WHERE Products.Description LIKE var1;
```

Um diese Ansicht verwenden zu können, definieren Sie die Variable var1, bevor Sie die Abfrage ausführen, die die Ansicht referenziert. Folgende BEGIN-Anweisung könnte z.B. Bestandteil einer Prozedur, einer Funktion oder einer Batchanweisung sein:

```
BEGIN
DECLARE var1 CHAR(20);
SET var1 = '%cap%';
SELECT * FROM ProductsByDescription
END
```

DEALLOCATE DESCRIPTOR-Anweisung [ESQL]

Gibt Speicher frei, der einem SQL-Deskriptorbereich zugeordnet ist.

Syntax

DEALLOCATE DESCRIPTOR *descriptor-name*

descriptor-name : *identifizier*

Bemerkungen

Gibt allen Speicher frei, der einem Deskriptorbereich zugeordnet wurde, einschließlich Datenelementen, Indikatorvariablen und der Struktur selbst

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „[ALLOCATE DESCRIPTOR-Anweisung \[ESQL\]](#)“ auf Seite 449
- „[Der SQL-Deskriptor-Bereich \(SQLDA\)](#)“ [[SQL Anywhere Server - Programmierung](#)]
- „[SET DESCRIPTOR-Anweisung \[ESQL\]](#)“ auf Seite 1033

Standards und Kompatibilität

- **SQL/2008** DEALLOCATE DESCRIPTOR ist Teil der optionalen SQL/2008-Sprachenfunktion B031 (Basic Dynamic SQL).

Beispiel

Ein Beispiel finden Sie unter „[ALLOCATE DESCRIPTOR-Anweisung \[ESQL\]](#)“ auf Seite 449.

DEALLOCATE-Anweisung

Diese Anweisung hat in SQL Anywhere keine Auswirkung und wird ignoriert. Sie wird zur Gewährleistung der Kompatibilität von Adaptive Server Enterprise und Microsoft SQL Server verwendet.

Weitere Hinweise zu dieser Anweisung finden Sie in Ihrer Adaptive Server Enterprise- bzw. Microsoft SQL Server-Dokumentation.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Deklarationsabschnitt [ESQL]

Deklariert Hostvariablen in einem Embedded SQL-Programm. Hostvariablen werden benutzt, um Daten mit der Datenbank auszutauschen.

Syntax

```
EXEC SQL BEGIN DECLARE SECTION;  
C declarations  
EXEC SQL END DECLARE SECTION;
```

Bemerkungen

Ein Declaration-Abschnitt ist einfach ein Abschnitt der C-Variablendeklarationen in BEGIN DECLARE SECTION- und END DECLARE SECTION-Anweisungen. Ein Deklarationsabschnitt macht den SQL-Präprozessor auf C-Variablen aufmerksam, die als Hostvariablen verwendet werden. Nicht alle C-Deklarationen sind in einem Deklarationsabschnitt gültig.

Privilegien

Keine.

Siehe auch

- „Hostvariablen in Embedded SQL“ [[SQL Anywhere Server - Programmierung](#)]
- „BEGIN-Anweisung“ auf Seite 557

Standards und Kompatibilität

- **SQL/2008** Kernfunktion.

Beispiel

```
EXEC SQL BEGIN DECLARE SECTION;  
char *surname, initials[5];  
int dept;  
EXEC SQL END DECLARE SECTION;
```

DECLARE CURSOR-Anweisung [ESQL] [SP]

Deklariert einen Cursor.

Syntax 1 [ESQL]

```
DECLARE cursor-name  
[ UNIQUE ]  
[ NO SCROLL
```

```

| DYNAMIC SCROLL
| SCROLL
| INSENSITIVE
| SENSITIVE
|
CURSOR FOR
{ select-statement
| statement-name
| call-statement }

```

Syntax 2 [SP]

```

DECLARE cursor-name
[ NO SCROLL
| DYNAMIC SCROLL
| SCROLL
| INSENSITIVE
| SENSITIVE
|
CURSOR
{ FOR select-statement
| FOR call-statement
| USING variable-name }

```

cursor-name : *identifier*

statement-name : *identifier* | *hostvar*

variable-name : *identifier*

Parameter

UNIQUE-Klausel Wenn ein Cursor als UNIQUE deklariert wird, ist die Abfrage gezwungen, alle Spalten zurückzugeben, die für eine eindeutige Identifizierung jeder einzelnen Zeile notwendig sind. Meist bedeutet dies, dass alle Spalten im Primärschlüssel oder eine Eindeutigkeits-Integritätsregel für die Tabelle zurückgegeben werden. Spalten, die erforderlich sind, aber in der Abfrage nicht angegeben wurden, werden der Ergebnismenge hinzugefügt.

Eine DESCRIBE-Anweisung für einen UNIQUE-Cursor legt die folgenden zusätzlichen Optionen in den Indikatorvariablen fest:

- **DT_KEY_COLUMN** Die Spalte ist Teil des Schlüssels für die Zeile.
- **DT_HIDDEN_COLUMN** Die Spalte wurde der Abfrage hinzugefügt, weil sie erforderlich war, um die Zeilen eindeutig zu identifizieren.

NO SCROLL-Klausel Ein als NO SCROLL deklariert Cursor ist auf die Vorwärtsbewegung durch die Ergebnismenge mit den Suchvorgängen FETCH NEXT und FETCH RELATIVE 0 beschränkt.

Da er nicht mehr zu der Zeile zurückkehren kann, wenn der Cursor die Zeile verlassen hat, gibt es keine Sensitivitätseinschränkungen für den Cursor. Wenn ein NO SCROLL-Cursor angefordert wird, liefert SQL Anywhere den wirksamsten Cursor, nämlich einen asensitiven Cursor.

DYNAMIC SCROLL-Klausel DYNAMIC SCROLL ist der Standard-Cursortyp. DYNAMIC SCROLL-Cursor können alle Formate der FETCH-Anweisung verwenden.

Wenn ein DYNAMIC SCROLL-Cursor angefordert wird, liefert SQL Anywhere einen asensitiven Cursor. Wenn Sie Cursor verwenden, muss immer zwischen Effizienz und Konsistenz abgewogen werden. Asensitive Cursor bieten eine hohe Performance, allerdings bei geringerer Konsistenz.

SCROLL-Klausel Ein als SCROLL deklarierter Cursor kann alle Formate der FETCH-Anweisung verwenden. Wenn ein SCROLL-Cursor angefordert wird, liefert SQL Anywhere einen wertsensitiven Cursor. Bei einem wertsensitiven Cursor kann ein nachfolgendes Abrufen einer zuvor abgerufenen Ergebniszeile eine Warnung oder einen Fehler zurückgeben, wenn die zugrunde liegende Zeile geändert oder gelöscht wurde.

SQL Anywhere muss wertsensitive Cursor so ausführen, dass die Zugehörigkeit zur Ergebnismenge garantiert ist. DYNAMIC SCROLL-Cursor haben einen höheren Wirkungsgrad und sollten benutzt werden, wenn nicht ein konsistentes Verhalten von SCROLL-Cursors erforderlich ist.

INSENSITIVE-Klausel Ein als INSENSITIVE deklarierter Cursor hat beim Öffnen eine feste Zugehörigkeit; eine temporäre Tabelle wird mit einer Kopie aller Originalzeilen erstellt. Ein FETCH-Abruf aus einem INSENSITIVE-Cursor berücksichtigt nicht die Auswirkungen anderer INSERT-, UPDATE- oder DELETE-Anweisungen aus gleichzeitig ausgeführten Transaktionen sowie anderer UPDATE-Vorgänge innerhalb derselben Transaktion. INSENSITIVE-Cursor können nicht aktualisiert werden.

SENSITIVE-Klausel Ein als SENSITIVE (empfindlich) deklarierter Cursor erkennt Änderungen der Zugehörigkeit oder der Ergebnismengenwerte.

FOR *statement-name*-Klausel Anweisungen werden mit der PREPARE-Anweisung benannt. Cursor können nur für eine vorbereitete SELECT- oder CALL-Anweisung deklariert werden. Die in der PREPARE-Anweisung angegebene Cursor-Aktualisierbarkeit wird für den Cursor verwendet, es sei denn, die SQL-Präprozessoroption -m HISTORICAL ist angegeben.

USING *variable-name*-Klausel Nur für die Verwendung innerhalb von gespeicherten Prozeduren. Die Variable ist eine Zeichenfolge, die eine SELECT-Anweisung für den Cursor enthält. Die Variable muss beim Verarbeiten der DECLARE-Anweisung verfügbar sein und muss deshalb entweder ein Parameter für die Prozedur sein oder innerhalb einer anderen BEGIN...END-Anweisung verschachtelt werden, nachdem der Variablen ein Wert zugeordnet wurde.

Bemerkungen

Cursor sind das wichtigste Instrument, um die Ergebnisse von Abfragen zu beeinflussen. Die DECLARE CURSOR-Anweisung deklariert einen Cursor mit dem angegebenen Namen für eine SELECT- oder CALL-Anweisung. In einer Prozedur, einem Trigger oder einem Batch in Watcom-SQL muss eine DECLARE CURSOR-Anweisung mit anderen Deklarationen auftreten, und zwar unmittelbar nach dem BEGIN-Schlüsselwort. Cursornamen müssen eindeutig sein.

Wenn ein Cursor in einer zusammengesetzten Anweisung deklariert wird, besteht er nur für die Dauer dieser zusammengesetzten Anweisung (ganz gleich, ob die zusammengesetzte Anweisung in Watcom-SQL oder Transact-SQL geschrieben wurde).

Wenn eine einzige Anweisung verarbeitet wird, müssen alle DECLARE CURSOR-Anweisungen unterschiedliche Namen verwenden, auch wenn die Cursor in nicht überlappenden Bereichen deklariert werden. Der Cursor kann jedoch nur innerhalb der zusammengesetzten Anweisung verwendet werden, die ihn deklariert.

Der Typ des Cursors in einer DECLARE CURSOR-Anweisung kann den Ausführungsplan festlegen, der vom Abfrageoptimierer für diese Anweisung ausgewählt wird. Beispiel: Ein INSENSITIVE-Cursor über einer SELECT-Anweisung erfordert die vollständige Materialisierung von der Ergebnismenge der SELECT-Anweisung, wenn der Cursor geöffnet wird. Die Art des Cursors muss mit den Merkmalen der zugrunde liegenden Anweisung übereinstimmen. Wenn es keine Übereinstimmung zwischen dem Cursortyp und der Anweisung gibt, kann der Cursortyp automatisch geändert werden. Beispiel: Ein INSENSITIVE-Cursor steht mit einer aktualisierbaren SELECT-Anweisung in Konflikt, die FOR UPDATE angibt, weil per definitionem die INSENSITIVE-Cursor schreibgeschützt sind. In diesem Fall wird der Cursortyp automatisch von INSENSITIVE auf einen aktualisierbaren, wertempfindlichen Cursor geändert, wenn der Cursor geöffnet wird.

Wenn die Aktualisierbarkeit einer SELECT-Anweisung, die in eine Cursordeklaration eingebettet ist, nicht angegeben wird, wird sie aus der Einstellung der Option `ansi_update_constraints` ermittelt.

Privilegien

Keine.

Nebenwirkungen

Wenn der Cursortyp geändert werden muss, um die Anforderungen der zugrunde liegenden Anweisung zu erfüllen, wird beim Öffnen des Cursors eine Warnung zurückgegeben.

Siehe auch

- „PREPARE-Anweisung [ESQL]“ auf Seite 976
- „OPEN-Anweisung [ESQL] [SP]“ auf Seite 964
- „EXPLAIN-Anweisung [ESQL]“ auf Seite 851
- „SELECT-Anweisung“ auf Seite 1020
- „CALL-Anweisung“ auf Seite 564
- „FOR-Anweisung“ auf Seite 857
- FOR UPDATE- oder FOR READ ONLY-Klausel, SELECT-Anweisung auf Seite 1025
- WITH Tabellen-Hint-Klausel, FROM-Klausel auf Seite 869
- „Der SQL-Präprozessor“ [*SQL Anywhere Server - Programmierung*]
- „Sensitive Cursor“ [*SQL Anywhere Server - Programmierung*]
- „Insensitive Cursor“ [*SQL Anywhere Server - Programmierung*]
- „Wertsensitive Cursor“ [*SQL Anywhere Server - Programmierung*]
- „Asensitive Cursor“ [*SQL Anywhere Server - Programmierung*]
- „ansi_update_constraints-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** DECLARE CURSOR ist eine Kernfunktion des SQL/2008-Standards. Die Möglichkeit zum Angeben von FOR UPDATE mit SCROLL oder NO SCROLL ist die optionale SQL-Sprachenfunktion F831 (vollständige Cursoraktualisierung). Die Verwendung von DECLARE CURSOR in einem Embedded SQL-Programm ist die optionale SQL-Sprachenfunktion B031. Einige Cursortypen sind auch optionale SQL-Funktionen. Es handelt sich dabei um die folgenden:
 - INSENSITIVE-Cursor sind die optionale SQL-Sprachenfunktion F791 des SQL/2008-Standards.
 - SENSITIVE-Cursor sind die optionale SQL-Sprachenfunktion F231 des SQL/2008-Standards.

- Scrollfähige Cursor sind die optionale SQL-Sprachenfunktion F431 des SQL/2008-Standards.

SQL Anywhere unterstützt eine Reihe von Erweiterungen des Herstellers zu DECLARE CURSOR, einschließlich der folgenden:

- SQL Anywhere unterstützt mehrere Erweiterungen zur FOR UPDATE-Klausel, die SQL/2008 als Klausel der DECLARE CURSOR-Anweisung definiert.
- WITH HOLD wird als Klausel der OPEN-Anweisung angegeben statt als Klausel der DECLARE CURSOR-Anweisung, wie in SQL/2008 definiert.
- Der SQL/2008 Standard trennt zwischen Cursorempfindlichkeit und SCROLL-Fähigkeit, während SQL Anywhere beides aus historischen Gründen kombiniert. In SQL Anywhere sind alle Cursor vorwärts und rückwärts scrollfähig, außer denjenigen, die als NO SCROLL deklariert wurden.
- DYNAMIC SCROLL und UNIQUE sind Erweiterungen des Herstellers. DYNAMIC SCROLL weist ein ähnliches Verhalten auf wie Cursor, die im SQL/2008-Standard als ASENSITIVE deklariert wurden.
- Die Möglichkeit zum Deklarieren eines Cursors über eine CALL-Anweisung, oder mit einer USING-Klausel, ist eine Erweiterung des Herstellers.
- **Transact-SQL** DECLARE CURSOR wird von Adaptive Server Enterprise unterstützt, aber im Verhalten gibt es mehrere Unterschiede. Adaptive Server Enterprise unterscheidet, wie in SQL/2008, zwischen SCROLL-Fähigkeit und Empfindlichkeit. In Adaptive Server Enterprise sind die Cursorempfindlichkeitsoptionen SEMI-SENSITIVE, INSENSITIVE oder DEFAULT (ähnlich ASENSITIVE). In Adaptive Server Enterprise sind NO SCROLL-Cursor die Standardeinstellung und alle scrollfähigen Cursor sind schreibgeschützt. Einige Funktionen der DECLARE CURSOR-Anweisung werden von Adaptive Server Enterprise nicht unterstützt. Es handelt sich dabei um die folgenden:
 - Adaptive Server Enterprise unterstützt die Cursorparallelitätsklausel von SQL Anywhere nicht. Um eine Sperre auf eine abgerufene Zeile zu setzen, müssen Sie den HOLDLOCK-Tabellen-Hint verwenden.
 - Adaptive Server Enterprise unterstützt keine DYNAMIC SCROLL- oder UNIQUE-Cursor. DYNAMIC SCROLL ist ähnlich dem Standard-Cursorverhalten von Adaptive Server Enterprise.
 - Die Möglichkeit zum Deklarieren eines Cursors über eine CALL-Anweisung, oder mit einer USING-Klausel, wird von Adaptive Server Enterprise nicht unterstützt.

In Adaptive Server Enterprise können Transact-SQL-Prozeduren und -Funktionen mehrere DECLARE CURSOR-Anweisungen mit demselben Cursornamen enthalten. In Adaptive Server Enterprise wird die DEALLOCATE CURSOR-Anweisung verwendet, um einen Cursor aus dem aktuellen Bereich zu eliminieren, damit eine nachfolgende OPEN-Anweisung den richtigen, zuvor deklarierten Cursor referenzieren kann. Diese Funktion wird in SQL Anywhere nicht unterstützt. In SQL Anywhere müssen alle Cursor in einem gegebenen Bereich eindeutige Namen haben. Wenn eine Transact-SQL-Dialektprozedur mehrere Cursordeklarationen mit demselben Namen enthält, wird die Prozedur ohne Fehler syntaktisch analysiert. Wenn jedoch zur Ausführungszeit eine zweite DECLARE CURSOR-Anweisung mit demselben Cursornamen ausgeführt wird, tritt ein Fehler auf.

Beachten Sie, dass das TDS Wire Protocol für Open Client- und jConnect-Verbindungen keine tatsächlich scrollfähigen Ergebnismengen implementiert. Wenn Sie rückwärts durch einen Cursor blättern müssen, kann die FETCH-Anforderung sofort erfüllt werden, sofern die gewünschte Zeile sich in einem Fenster mit vorab abgerufenen Zeilen befindet, die bereits vom TDS-Client abgerufen wurden. Wenn die gewünschte Zeile außerhalb dieses Fensters liegt, kann jedoch die SELECT-Anweisung des Cursors erneut ausgeführt werden.

Beispiel

Das folgende Beispiel veranschaulicht, wie ein Scroll-Cursor in Embedded SQL deklariert wird:

```
EXEC SQL DECLARE cur_employee SCROLL CURSOR
FOR SELECT * FROM GROUPO.Employees;
```

Das folgende Beispiel veranschaulicht, wie ein Cursor für eine vorbereitete Anweisung in Embedded SQL deklariert wird:

```
EXEC SQL PREPARE employee_statement
FROM 'SELECT Surname FROM GROUPO.Employees' FOR READ ONLY;
EXEC SQL DECLARE cur_employee CURSOR
FOR employee_statement;
```

Das folgende Beispiel veranschaulicht die Cursorverwendung in einer gespeicherten Prozedur:

```
BEGIN
  DECLARE cur_employee CURSOR FOR
    SELECT Surname
    FROM GROUPO.Employees;
  DECLARE name CHAR(40);
  OPEN cur_employee;
  lp: LOOP
    FETCH NEXT cur_employee INTO name;
    IF SQLCODE <> 0 THEN LEAVE lp END IF;
    ...
  END LOOP;
  CLOSE cur_employee;
END
```

Dieses Beispiel zeigt, wie die USING-Klausel als Parameter für die Prozedur verwendet wird:

```
CREATE FUNCTION GetRowCount( IN qry LONG VARCHAR )
RETURNS INT
BEGIN
  DECLARE crsr CURSOR USING qry;
  DECLARE rowcnt INT;

  SET rowcnt = 0;
  OPEN crsr;
  lp: LOOP
    FETCH crsr;
    IF SQLCODE <> 0 THEN LEAVE lp END IF;
    SET rowcnt = rowcnt + 1;
  END LOOP;
  CLOSE crsr;
  RETURN rowcnt;
END;
```

Dieses Beispiel zeigt, wie die USING-Klausel innerhalb einer BEGIN...END-Anweisung verschachtelt wird, nachdem *variable-name* ein Wert zugewiesen wurde.

```
CREATE PROCEDURE get_table_name(  
    IN id_value INT, OUT tabname CHAR(128)  
)  
BEGIN  
    DECLARE qry LONG VARCHAR;  
  
    SET qry = 'SELECT table_name FROM SYS.SYSTAB ' ||  
              'WHERE table_id=' || string(id_value);  
  
    BEGIN  
        DECLARE crsr CURSOR USING qry;  
        OPEN crsr;  
        FETCH crsr INTO tabname;  
        CLOSE crsr;  
    END  
END;
```

Im folgenden Beispiel wird ein Fehler zurückgegeben, da die beiden Cursornamen in der Anweisung nicht eindeutig sind:

```
BEGIN  
    BEGIN  
        DECLARE MyCursor DYNAMIC SCROLL CURSOR FOR SELECT 1;  
    END;  
    BEGIN  
        DECLARE MYCursor DYNAMIC SCROLL CURSOR FOR SELECT 2;  
    END;  
END;
```

Im folgenden Beispiel wird ein Fehler zurückgegeben, da der Cursor nicht innerhalb der Anweisung deklariert wurde, die versucht, ihn zu öffnen.

```
BEGIN  
    BEGIN  
        DECLARE MyCursor DYNAMIC SCROLL CURSOR FOR SELECT 1;  
    END;  
    BEGIN  
        OPEN MyCursor;  
    END;  
END;
```

DECLARE LOCAL TEMPORARY TABLE-Anweisung

Deklariert eine lokale temporäre Tabelle.

Syntax

```
DECLARE LOCAL TEMPORARY TABLE table-name  
( { column-definition [ column-constraint ... ] | table-constraint | pctfree }, ... )  
[ ON COMMIT { DELETE | PRESERVE } ROWS  
  | NOT TRANSACTIONAL ]
```

pctfree : **PCTFREE** *percent-free-space*

percent-free-space : *integer*

Parameter

ON COMMIT-Klausel Standardmäßig werden die Zeilen einer temporären Tabelle bei COMMIT gelöscht. Sie können die ON COMMIT-Klausel benutzen, um die Zeilen bei COMMIT zu erhalten.

NOT TRANSACTIONAL-Klausel Eine mit dieser Klausel erstellte Tabelle wird weder durch COMMIT noch durch ROLLBACK verändert. Die NOT TRANSACTIONAL-Klausel bietet Performanceverbesserungen unter bestimmten Umständen, da Vorgänge in nicht-transaktionalen temporären Tabellen keine Einträge im Rollback-Log bewirken. NOT TRANSACTIONAL kann z.B. sinnvoll sein, wenn Prozeduren, die die temporäre Tabelle verwenden, ohne dazwischen liegende COMMITs oder ROLLBACKs wiederholt aufgerufen werden.

Bemerkungen

Bei einer lokalen temporären Tabelle können Sie nicht mit der *column-constraint* REFERENCES oder der FOREIGN KEY-Tabellen-Integritätsregel arbeiten.

Die DECLARE LOCAL TEMPORARY TABLE-Anweisung deklariert eine temporäre Tabelle.

Tabellen, die mit DECLARE LOCAL TEMPORARY TABLE erstellt wurden, erscheinen nicht in der SYSTABLE-Ansicht des Systemkatalogs.

Die Zeilen einer deklarierten temporären Tabelle werden gelöscht, wenn die Tabelle explizit gelöscht wird oder den Bereich überschreitet. Sie können Zeilen auch explizit mit TRUNCATE oder DELETE löschen.

Deklarierte lokale temporäre Tabellen in zusammengesetzten Anweisungen bestehen innerhalb der zusammengesetzten Anweisung. Sonst existiert die deklarierte lokale temporäre Tabelle bis zum Ende der Verbindung.

Zwei lokale temporäre Tabellen in demselben Geltungsbereich können nicht den gleichen Namen haben. Wenn Sie eine temporäre Tabelle erstellen, die den gleichen Namen hat wie eine Basistabelle, wird die Basistabelle innerhalb der Verbindung erst sichtbar, wenn der Geltungsbereich der lokalen temporären Tabelle endet. Eine Verbindung kann keine Basistabelle mit dem Namen einer vorhandenen temporären Tabelle erstellen.

Wenn eine Prozedur eine lokale temporäre Tabelle erstellen soll, die erhalten bleibt, nachdem die Prozedur abgeschlossen ist, verwenden Sie die CREATE LOCAL TEMPORARY TABLE-Anweisung.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „CREATE TABLE-Anweisung“ auf Seite 737
- „CREATE LOCAL TEMPORARY TABLE-Anweisung“ auf Seite 645
- „Zusammengesetzte Anweisungen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** DECLARE LOCAL TEMPORARY TABLE ist Teil der optionalen Sprachenfunktion F531 des SQL/2008-Standards. Die Klauseln PCTFREE und NOT TRANSACTIONAL sind Erweiterungen des Herstellers. Die in der Anweisung festgelegten Definitionen für Spalten und

Integritätsregeln können ebenfalls Syntax enthalten, die eine Erweiterung des Herstellers darstellt. Im SQL/2008-Standard ist festgelegt, dass mit der DECLARE LOCAL TEMPORARY TABLE-Anweisung erstellte Tabellen im Systemkatalog erscheinen. Dies ist bei SQL Anywhere nicht der Fall.

- **Transact-SQL** DECLARE LOCAL TEMPORARY TABLE wird von Adaptive Server Enterprise nicht unterstützt. In Sybase Adaptive Server Enterprise wird eine temporäre Tabelle erstellt unter Verwendung der CREATE TABLE-Anweisung mit einem Tabellennamen, der mit dem Sonderzeichen # beginnt.

Beispiel

Das folgende Beispiel veranschaulicht, wie eine temporäre Tabelle in einer gespeicherten Prozedur deklariert wird:

```
BEGIN
  DECLARE LOCAL TEMPORARY TABLE TempTab ( number INT );
  ...
END
```

DECLARE-Anweisung

Deklariert eine SQL-Variable oder eine Ausnahmebedingung in einer zusammengesetzten Anweisung (BEGIN...END).

Syntax 1 - Variable deklarieren

```
DECLARE variable-name [, ... ] data-type
[ { = | DEFAULT } initial-value ]
```

initial-value :

special-value

| *string*

| [-] *number*

| (*constant-expression*)

| *built-in-function* (*constant-expression*)

| **NULL**

special-value :

CURRENT {

DATABASE

 | **DATE**

 | **PUBLISHER**

 | **TIME**

 | **TIMESTAMP**

 | **USER**

 | **UTC TIMESTAMP** }

| **USER**

Syntax 2 - Ausnahmebedingung deklarieren

```
DECLARE exception-name EXCEPTION
FOR SQLSTATE [ VALUE ] string
```

Bemerkungen

DECLARE *variable-name*: Variablen, die im Hauptteil einer Prozedur oder eines Triggers oder in einem Batch verwendet werden, können mit der DECLARE-Anweisung mithilfe von Syntax 1 deklariert werden. Die Variable bleibt für die Dauer der zusammengesetzten Anweisung, in der sie deklariert wurde, erhalten. Wenn Sie einen *initial-value* angeben, wird die Variable auf diesen Wert gesetzt. Wenn Sie keinen *initial-value* angeben, enthält die Variable NULL, bis mit der SET-Anweisung ein anderer Wert zugewiesen wird.

Der Hauptteil einer Prozedur oder eines Triggers in Watcom-SQL ist eine zusammengesetzte Anweisung und Variablen müssen mit anderen Deklarationen, wie z.B. einer Cursordeklaration (DECLARE CURSOR), sofort nach dem BEGIN-Schlüsselwort deklariert werden. In einer Prozedur oder einem Trigger in Transact-SQL gibt es keine solche Einschränkung.

Wenn Sie *initial-value* angeben, muss der Datentyp dem durch *data-type* definierten Typ entsprechen.

DECLARE *exception-name* EXCEPTION: Verwenden Sie diese Syntax, um Variablen für Ausnahmebedingungen der SQL-Sprache innerhalb einer zusammengesetzten Anweisung (BEGIN...END) zu deklarieren. Die Variablen können für den Vergleich mit den während der Ausführung erhaltenen Werten für SQLSTATE, mit einer SIGNAL-Anweisung oder als Teil der Ausnahmebedingung in der Ausnahmeroutine verwendet werden.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „SQL-Datentypen“ auf Seite 95
- „DECLARE-Anweisung“ auf Seite 786
- „DECLARE CURSOR-Anweisung [ESQL] [SP]“ auf Seite 778
- „BEGIN-Anweisung“ auf Seite 557
- „SQLSTATE-Spezialwert“ auf Seite 80
- „Spezialwerte“ auf Seite 70
- „Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008 Syntax 1** (Variablen deklarieren) - Persistent Stored Module-Funktion. **Syntax 2** (Ausnahmebedingungen deklarieren) - das von SQL Anywhere unterstützte Format für die Ausnahmendeklaration, nämlich die DECLARE EXCEPTION-Anweisung, ist eine Erweiterung des Herstellers. In SQL/2008 werden Ausnahmen in Form einer Routinendeklaration mit den Schlüsselwörtern DECLARE HANDLER angegeben. Die DECLARE...EXCEPTION-Syntax ist in T-SQL-Prozeduren nicht zulässig.
- **Transact-SQL Syntax 2** (Ausnahmebedingungen deklarieren) kann nicht in zusammengesetzten Anweisungen und Prozeduren in Transact-SQL verwendet werden.

Beispiel

Der folgende Batch veranschaulicht die Verwendung der DECLARE-Anweisung und gibt im Meldungsfenster des Datenbankservers eine Meldung aus:

```
BEGIN
  DECLARE varname CHAR(61);
  SET varname = 'Test name';
  MESSAGE varname;
END
```

In diesem Beispiel werden die folgenden Variablen deklariert:

- v1 als INT mit dem Anfangswert 5.
- v2 und v3 als CHAR(10), beide mit dem Anfangswert abc.

```
BEGIN
  DECLARE v1 INT = 5;
  DECLARE v2, v3 CHAR(10) = 'abc';
  // ...
END
```

Die folgende Prozedur deklariert eine Ausnahmebedingung für die Verwendung mit dem SQLSTATE-Vergleich:

```
CREATE PROCEDURE HighSales (IN cutoff INT, OUT HighValues INT)
BEGIN
  DECLARE err_notfound EXCEPTION FOR
    SQLSTATE '02000';
  DECLARE curThisCust CURSOR FOR
    SELECT CAST( sum( SalesOrderItems.Quantity *
      Products.UnitPrice ) AS INTEGER) VALUE
    FROM Customers
      LEFT OUTER JOIN SalesOrders
      LEFT OUTER JOIN SalesOrderItems
      LEFT OUTER JOIN Products
    GROUP BY CompanyName;
  DECLARE ThisValue INT;
  SET HighValues = 0;
  OPEN curThisCust;
  CustomerLoop:
  LOOP
    FETCH NEXT curThisCust
      INTO ThisValue;
    IF SQLSTATE = err_notfound THEN
      LEAVE CustomerLoop;
    END IF;
    IF ThisValue > cutoff THEN
      SET HighValues = HighValues + ThisValue;
    END IF;
  END LOOP CustomerLoop;
  CLOSE curThisCust;
END;
```

Die folgende zusammengesetzte Anweisung deklariert eine Ausnahmebedingung für die Verwendung mit SIGNAL und eine Ausnahmeroutine:

```
BEGIN
  DECLARE err_div_by_0 EXCEPTION FOR
    SQLSTATE '22012';
  DECLARE curQuantity CURSOR FOR
```

```

        SELECT Quantity
        FROM SalesOrderItems
        WHERE ProductID = 300;
DECLARE Quantities INT;
DECLARE altogether INT;
SET Quantities = 0;
SET altogether = 0;
OPEN curQuantity;
LOOP
    FETCH NEXT curQuantity
    INTO Quantities;
    IF SQLSTATE = '02000' THEN
        SIGNAL err_div_by_0;
    END IF;
    SET altogether = altogether + Quantities;
END LOOP;
EXCEPTION
    WHEN err_div_by_0 THEN
        CLOSE curQuantity;
        SELECT altogether;
        return;
    WHEN OTHERS THEN
        RESIGNAL;
END;
```

DELETE-Anweisung (positionsbasiert) [ESQL] [SP]

Löscht die Daten an der aktuellen Cursorposition.

Syntax

DELETE [**[FROM]***table*] **WHERE CURRENT OF** *cursor-name*

cursor-name : *identifier* | *hostvar*

table : [*owner*.]*table-or-view* [[**AS**] *correlation-name*]

owner : *identifier*

table-or-view : *identifier*

correlation-name : *identifier*

Bemerkungen

Diese Form der DELETE-Anweisung löscht die aktuelle Zeile des angegebenen Cursors. Die aktuelle Zeile wird als die zuletzt vom Cursor abgerufene Zeile definiert.

Die Tabelle, aus der die Zeilen gelöscht werden, wird wie folgt definiert:

- Wenn keine FROM-Klausel eingeschlossen ist, muss es sich um einen Cursor auf nur eine Tabelle handeln.
- Wenn der Cursor für eine Join-Abfrage (einschließlich zum Benutzen einer Ansicht mit enthaltenem Join) gesetzt ist, muss die FROM-Klausel verwendet werden. Nur die aktuelle Zeile der angegebenen Tabelle wird gelöscht. Die anderen Tabellen des Joins sind nicht betroffen.

- Wenn eine FROM-Klausel enthalten ist, muss *table* eine aktualisierbare Tabelle im Cursor eindeutig identifizieren. Wenn ein *correlation-name* angegeben ist, versucht der Server im zugrunde liegenden Cursor einen übereinstimmenden Korrelationsnamen zu finden. Wenn in der DELETE-Anweisung kein Korrelationsname angegeben ist und auch kein Tabelleneigentümer angegeben wurde, versucht der Server, eine Übereinstimmung zwischen *table-or-view* und einer aktualisierbaren Tabelle im zugrunde liegenden Cursor zu finden. *table-or-view* wird zuerst mit allen Korrelationsnamen abgeglichen.
 - Wenn im zugrunde liegenden Cursor ein Korrelationsname vorhanden ist, kann *table-or-view* dem entsprechenden Korrelationsnamen zugeordnet werden.
 - Wenn kein Korrelationsname existiert, muss der Wert *table-or-view* eindeutig mit einem Tabellennamen im Cursor übereinstimmen.
- Wenn eine FROM-Klausel enthalten und auch ein Tabelleneigentümer angegeben ist, dann muss der Wert *table* eindeutig mit einer aktualisierbaren Tabelle im Cursor übereinstimmen.
- Die positionsbasierte DELETE-Anweisung kann für einen Cursor verwendet werden, der für eine Ansicht geöffnet ist, solange die Ansicht aktualisierbar ist.

Privilegien

Sie müssen Eigentümer der Tabelle sein oder das DELETE-Privileg für die Tabelle haben.

Nebenwirkungen

Keine.

Siehe auch

- „UPDATE-Anweisung“ auf Seite 1109
- „UPDATE-Anweisung (positionsbasiert) [ESQL] [SP]“ auf Seite 1103
- „INSERT-Anweisung“ auf Seite 917
- „PUT-Anweisung [ESQL]“ auf Seite 981

Standards und Kompatibilität

- **SQL/2008** Die (positionsbasierte) DELETE-Anweisung ist eine Kernfunktion des SQL/2008-Standards. Die Möglichkeit zum Verwenden einer positionsbasierten DELETE-Anweisung innerhalb eines Embedded SQL-Programms ist Teil der optionalen SQL-Sprachenfunktion B031 (Basic Dynamic SQL).

Das FROM-Schlüsselwort ist in SQL/2008 obligatorisch, in SQL Anywhere jedoch optional. Der Bereich der aktualisierbaren Cursors kann Erweiterungen des Herstellers enthalten, wenn die Option `ansi_update_constraints` auf OFF gesetzt ist.

Beispiel

Die folgende Anweisung entfernt die aktuelle Zeile im Cursor `cur_employee` aus der Datenbank.

```
DELETE  
WHERE CURRENT OF cur_employee;
```

DELETE-Anweisung

Löscht Zeilen aus der Datenbank.

Syntax 1

```
DELETE [ row-limitation ]
[ FROM ] [ owner. ] table-or-view [ [ AS ] correlation-name ]
[ WHERE search-condition ]
[ ORDER BY { expression | integer } [ ASC | DESC ], ... ]
[ OPTION( query-hint, ... ) ]
```

Syntax 2 - Transact-SQL

```
DELETE [ row-limitation ]
[ FROM ] [ owner. ] table-or-view [ [ AS ] correlation-name ]
[ FROM table-expression ]
[ WHERE search-condition ]
[ ORDER BY { expression | integer } [ ASC | DESC ], ... ]
[ OPTION( query-hint, ... ) ]
```

table-or-view : *identifier*

row-limitation :

```
FIRST
| TOP { ALL | limit-expression } [ START AT startat-expression ]
```

limit-expression : *simple-expression*

startat-expression : *simple-expression*

simple-expression :

```
integer
| variable
| ( simple-expression )
| ( simple-expression { + | - | * } simple-expression )
```

query-hint :

```
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| FORCE NO OPTIMIZATION
| option-name = option-value
```

table-expression: Ein vollständiger Tabellenausdruck, der Joins enthalten kann. Siehe „[FROM-Klausel](#)“ auf Seite 863.

option-name : *Bezeichner*

option-value :

```
host-variable (Bezeichner zulässig)
| string
| identifier
| number
```

Parameter

Zeilenbeschränkungsklausel Die Zeilenbeschränkungsklausel ermöglicht es Ihnen, die zu löschenden Zeilen auf eine Teilmenge der Zeilen zu beschränken, die von der WHERE-Klausel erfasst

werden. Die Argumente TOP und START AT können einfache arithmetische Ausdrücke über Hostvariablen, Ganzzahlkonstanten oder Ganzzahlvariablen sein. Das TOP-Argument muss jedoch mit einem Wert größer oder gleich 0 ausgewertet werden. Das START AT-Argument mit einem Wert größer als 0 ausgewertet werden. Wenn diese Klauseln angegeben werden, ist auch eine ORDER BY-Klausel erforderlich, um die Zeilen sinnvoll zu ordnen.

FROM-Klausel Die FROM-Klausel gibt die Tabelle an, aus der die Zeilen gelöscht werden sollen. Bei Syntax 2 bestimmt die zweite FROM-Klausel in der DELETE-Anweisung die Zeilen, die auf der Grundlage von Joins mit anderen Tabellen aus der angegebenen Tabelle gelöscht werden sollen. *table-expression* kann beliebig komplexe Tabellenausdrücke enthalten, einschließlich abgeleiteter Tabellen sowie KEY- und NATURAL-Joins.

Die folgenden Beispiele veranschaulichen, wie Korrelationsnamen bei Syntax 2 abgeglichen werden. Mit dieser Anweisung:

```
DELETE
FROM table_1
FROM table_1 AS alias_1, table_2 AS alias_2
WHERE ...
```

Die Tabelle table_1 hat keinen Korrelationsnamen in der ersten FROM-Klausel, wohl aber in der zweiten FROM-Klausel. In diesem Fall wird Tabelle_1 in der ersten Klausel mit Alias_1 in der zweiten Klausel identifiziert: es gibt nur eine Instanz von Tabelle_1 in dieser Anweisung. Hierbei handelt es sich um eine zulässige Ausnahme der allgemeinen Regel, dass dort, wo eine Tabelle mit einem Korrelationsnamen in derselben Anweisung identifiziert wird, zwei Instanzen der Tabelle berücksichtigt werden.

Im folgenden Beispiel gibt es aber zwei Instanzen von table_1 in der zweiten FROM-Klausel. Die Anweisung schlägt mit einem Syntaxfehler fehl, da es unklar ist, welche Instanz von table_1 in der zweiten FROM-Klausel mit der ersten Instanz von table_1 in der ersten FROM-Klausel übereinstimmt.

```
DELETE
FROM table_1
FROM table_1 AS alias_1, table_1 AS alias_2
WHERE ...
```

WHERE-Klausel Die DELETE-Anweisung löscht alle Zeilen, welche die Bedingungen in der WHERE-Klausel erfüllen. Wenn keine WHERE-Klausel angegeben ist, werden alle Zeilen der benannten Tabelle gelöscht. Wenn eine zweite FROM-Klausel vorliegt, qualifiziert die WHERE-Klausel die Zeilen im *table-expression* dieser zweiten FROM-Klausel.

ORDER BY-Klausel Gibt die Sortierreihenfolge für die zu löschenden Zeilen an. Normalerweise spielt die Reihenfolge, in der Zeilen aktualisiert werden, keine Rolle. In Verbindung mit der FIRST- oder TOP-Klausel kann die Reihenfolge jedoch wichtig sein.

Sie können Spalten-Ordinalnummern in der ORDER BY-Klausel nicht verwenden.

Jedes Element in der ORDER BY-Liste kann als ASC für aufsteigende Sortierfolge (Standardwert) oder DESC für absteigende Sortierfolge benannt werden.

OPTION-Klausel Verwenden Sie diese Klausel, um Hints für das Ausführen der Anweisung anzugeben. Die folgenden Hints werden unterstützt:

- MATERIALIZED VIEW OPTIMIZATION *option-value*
- FORCE OPTIMIZATION
- FORCE NO OPTIMIZATION
- *option-name* = *option-value*. Die Spezifikation `OPTION(isolation_level = ...)` im Abfragetext hebt alle anderen Festlegungen der Isolationsstufe für eine Abfrage auf.

Bemerkungen

Durch das Löschen großer Datenmengen mit der DELETE-Anweisung werden die Spaltenstatistiken aktualisiert.

Wenn Sie alle Zeilen einer Tabelle löschen möchten, sollten Sie die effizientere TRUNCATE TABLE-Anweisung verwenden.

DELETE-Vorgänge können in Ansichten ausgeführt werden, wenn die Abfragespezifikation, die die Ansicht definiert, aktualisierbar ist. Eine Ansicht ist unter der Voraussetzung aktualisierbar, dass die definierende SELECT-Anweisung nur eine Tabelle in der FROM-Klausel enthält, keine DISTINCT-Klausel, GROUP BY-Klausel, WINDOW-Klausel, oder Aggregatfunktion enthält sowie keine Mengenoperatoren wie UNION oder INTERSECT erfordert.

Privilegien

Sie müssen Eigentümer der Tabelle sein oder SELECT- und DELETE-Privilegien für die Tabelle haben.

Nebenwirkungen

Keine.

Siehe auch

- „TRUNCATE-Anweisung“ auf Seite 1089
- „Zeilenbeschränkungsklauseln in SELECT-, UPDATE- und DELETE-Abfrageblöcken“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „INSERT-Anweisung“ auf Seite 917
- „INPUT-Anweisung [Interactive SQL]“ auf Seite 910
- „FROM-Klausel“ auf Seite 863
- OPTION-Klausel, SELECT-Anweisung auf Seite 1027
- „Reguläre Ansichten“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Sperren während des Löschens“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Syntax 1 ist eine Kernfunktion des SQL/2008-Standards, Syntax 2 dagegen eine Transact-SQL-Erweiterung des Herstellers. Die folgenden Funktionen von Syntax 1 sind Erweiterungen des Herstellers:
 - Das optionale FROM Schlüsselwort.
 - Die *row-limitation*-Klausel und die ORDER BY-Klausel.

- Die OPTION-Klausel.

Beispiel

Alle Daten vor 2000 werden aus der Tabelle FinancialData entfernt.

```
DELETE
FROM GROUPO.FinancialData
WHERE Year < 2000;
```

Die ersten 10 Aufträge werden aus der Tabelle SalesOrderItems entfernt, wenn das Lieferdatum älter als 2001-01-01 ist und sie aus dem Gebiet "Central" stammen.

```
DELETE TOP 10
FROM GROUPO.SalesOrderItems
FROM GROUPO.SalesOrders
WHERE SalesOrderItems.ID = SalesOrders.ID
      and ShipDate < '2001-01-01' and Region = 'Central'
ORDER BY ShipDate ASC;
```

Abteilung 600 wird aus der Datenbank entfernt, wobei die Anweisung auf Isolationsstufe 3 ausgeführt wird.

```
DELETE FROM GROUPO.Departments
WHERE DepartmentID = 600
OPTION( isolation_level = 3 );
```

DESCRIBE-Anweisung [ESQL]

Ruft Informationen zu den Hostvariablen ab, die zum Speichern von aus der Datenbank abgerufenen Daten erforderlich sind, oder zu Hostvariablen, die zum Übergeben von Daten an die Datenbank erforderlich sind.

Syntax

```
DESCRIBE
[ USER TYPES ]
[ ALL | BIND VARIABLES FOR | INPUT | OUTPUT ]
[ SELECT LIST FOR ]
[ LONG NAMES [ long-name-spec ] | WITH VARIABLE RESULT ]
[ FOR ] { statement-name | CURSOR cursor-name }
INTO sqlda-name
```

long-name-spec :
OWNER.TABLE.COLUMN
| **TABLE.COLUMN**
| **COLUMN**

statement-name : *identifier* | *hostvar*

cursor-name : *deklarierter Cursor*

sqlda-name : *identifier*

Parameter

USER TYPES-Klausel Eine DESCRIBE-Anweisung mit der USER TYPES-Klausel liefert Informationen über die Domänen einer Spalte. Normalerweise wird eine solche DESCRIBE-Anweisung angegeben, wenn eine vorhergehende DESCRIBE-Anweisung einen Indikator mit DT_HAS_USERTYPE_INFO zurückgibt.

Diese zurückgegebenen Informationen sind dieselben wie für eine DESCRIBE-Anweisung ohne die Schlüsselwörter USER TYPES, außer dass das Feld sqlname den Namen der Domäne enthält anstatt den Namen der Spalte.

Wenn die DESCRIBE-Anweisung die LONG NAMES-Klausel verwendet, dann enthält das Feld sqldata diese Informationen.

ALL-Klausel Mit DESCRIBE ALL können Sie INPUT und OUTPUT mit nur einer Anforderung an den Datenbankserver beschreiben. Das wirkt sich vorteilhaft auf die Performance aus. Zuerst werden die OUTPUT-Informationen in den SQLDA eingetragen, anschließend die INPUT-Informationen. Das Feld sqld enthält die Gesamtzahl der INPUT- und OUTPUT-Variablen. Das Bit DT_DESCRIBE_INPUT in der Indikatorvariablen wird für INPUT-Variablen gesetzt und für OUTPUT-Variablen gelöscht.

BIND VARIABLES FOR-Klausel Entspricht der INPUT-Klausel.

SELECT LIST FOR-Klausel Entspricht der OUTPUT-Klausel.

INPUT-Klausel Eine Bindevariable ist ein Wert, der von der Anwendung bereitgestellt wird, wenn die Datenbank die Anweisungen ausführt. Bindevariablen können als Parameter für die Anweisung angesehen werden. DESCRIBE INPUT trägt die Namen der Bindevariablen in die Namensfelder des SQLDA ein. DESCRIBE INPUT trägt außerdem die Anzahl der Bindevariablen in das Feld sqlda des SQLDA ein.

DESCRIBE verwendet die Indikatorvariablen im SQLDA, um zusätzliche Informationen bereitzustellen. DT_PROCEDURE_IN und DT_PROCEDURE_OUT sind Bits, die in der Indikatorvariablen gesetzt werden, wenn eine CALL-Anweisung beschrieben wird. DT_PROCEDURE_IN gibt einen IN- oder INOUT-Parameter an, und DT_PROCEDURE_OUT gibt einen INOUT- oder OUT-Parameter an. RESULT-Spalten für Prozeduren haben beide Bits gelöscht. Nach einem Beschreibungs-OUTPUT können diese Bits benutzt werden, um zwischen Anweisungen zu unterscheiden, die Ergebnismengen umfassen (müssen OPEN, FETCH, RESUME, CLOSE verwenden), und Anweisungen, die keine Ergebnismengen umfassen (müssen EXECUTE verwenden). DESCRIBE INPUT stellt DT_PROCEDURE_IN und DT_PROCEDURE_OUT entsprechend ein, wenn eine Bindevariable ein Argument für eine CALL-Anweisung ist. Bindevariablen innerhalb eines Ausdrucks, der ein Argument in einer CALL-Anweisung ist, setzen keine Bits.

OUTPUT-Klausel Die DESCRIBE OUTPUT-Anweisung trägt den Datentyp und die Länge für jedes Element der Auswahlliste in den SQLDA ein. In das Namensfeld wird ebenfalls ein Name für das Element der Auswahlliste eingetragen. Wenn ein Alias für ein Element der Auswahlliste angegeben wird, ist der Name dieser Alias. Andernfalls wird der Name aus dem Element der Auswahlliste hergeleitet: Wenn das Element ein einfacher Spaltenname ist, wird dieser verwendet; sonst wird eine Teilzeichenfolge des Ausdrucks verwendet. DESCRIBE setzt außerdem die Anzahl der Auswahllistenelemente in das Feld sqld des SQLDA.

Wenn die beschriebene Anweisung ein UNION-Vorgang von zwei oder mehreren SELECT-Anweisungen ist, sind die für DESCRIBE OUTPUT zurückgegebenen Spaltennamen dieselben Spaltennamen, die auch für die erste SELECT-Anweisung zurückgegeben werden würden.

Wenn Sie eine CALL-Anweisung beschreiben, trägt die DESCRIBE OUTPUT-Anweisung den Datentyp, die Länge und den Namen im SQLDA für jeden INOUT- oder OUT-Parameter in der Prozedur ein. DESCRIBE OUTPUT setzt auch die Anzahl der INOUT- oder OUT-Parameter in das Feld sqld des SQLDA.

Wenn Sie eine CALL-Anweisung mit einer Ergebnismenge beschreiben, trägt die DESCRIBE OUTPUT-Anweisung den Datentyp, die Länge und den Namen im SQLDA für jede RESULT-Spalte in der Prozedurdefinition ein. DESCRIBE OUTPUT trägt außerdem die Anzahl der Ergebnisspalten in das Feld sqld des SQLDA ein.

LONG NAMES-Klausel Die LONG NAMES-Klausel dient zum Abrufen von Spaltennamen für eine Anweisung oder einen Cursor. Ohne diese Klausel ist die Länge von Spaltennamen auf 29 Zeichen begrenzt. Mit der Klausel werden beliebige Längen unterstützt.

Bei Verwendung von LONG NAMES werden die langen Namen im SQLDATA-Feld des SQLDA abgelegt, als ob Sie sie von einem Cursor abrufen würden. Keines der anderen Felder (SQLEN, SQLTYPE usw.) wird ausgefüllt. Der SQLDA muss so eingestellt werden wie ein FETCH-SQLDA: Er muss einen Eintrag für jede Spalte enthalten und der Eintrag muss ein Zeichenfolgentyp sein. Wenn eine Indikatorvariable vorhanden ist, wird die Kürzung in gewohnter Weise angezeigt.

Die Standardspezifikation für die langen Namen ist **TABLE.COLUMN**.

WITH VARIABLE RESULT-Klausel Diese Klausel wird verwendet, um Prozeduren zu beschreiben, die mehr als eine Ergebnismenge mit unterschiedlichen Anzahlen und Typen von Spalten haben können.

Wenn WITH VARIABLE RESULT verwendet wird, legt der Datenbankserver nach der DESCRIBE-Anweisung den Wert SQLCOUNT mit einer der folgenden Einstellungen fest:

- **0** Die Ergebnismenge kann sich ändern. Der Prozeduraufruf sollte nach jeder OPEN-Anweisung erneut beschrieben werden.
- **1** Die Ergebnismenge ist unveränderlich. Eine erneute Beschreibung ist nicht erforderlich.

Bemerkungen

Die DESCRIBE-Anweisung richtet den benannten SQLDA so ein, dass er entweder OUTPUT (entsprechend SELECT LIST FOR) oder INPUT (BIND VARIABLES FOR) für die benannte Anweisung beschreibt.

DESCRIBE BIND VARIABLES stellt bei INPUT die Datentypen im SQLDA nicht ein: Dieser Vorgang muss von der Anwendung durchgeführt werden. Mit dem Schlüsselwort ALL können Sie INPUT und OUTPUT in einem SQLDA beschreiben.

Wenn Sie einen Anweisungsnamen angeben, muss die Anweisung zuvor unter Verwendung der PREPARE-Anweisung mit demselben Anweisungsnamen vorbereitet worden sein und der SQLDA muss bereits zugeordnet worden sein.

Wenn Sie einen Cursornamen angeben, muss der Cursor zuvor deklariert und geöffnet worden sein. Die Standardaktion ist die Beschreibung von OUTPUT. Nur SELECT-Anweisungen und CALL-Anweisungen haben OUTPUT. DESCRIBE OUTPUT für jede andere Anweisung oder für einen nicht-dynamischen Cursor zeigt keine Ausgabe an, wenn das Feld sqlda des SQLDA auf Null gesetzt wird.

In Embedded SQL werden NCHAR, NVARCHAR und LONG NVARCHAR standardmäßig als DT_FIXCHAR, DT_VARCHAR bzw. DT_LONGVARCHAR beschrieben. Wenn die Funktion db_change_nchar_charset aufgerufen wurde, werden diese Datentypen als DT_NFIXCHAR, DT_NVARCHAR bzw. DT_LONGNVARCHAR beschrieben.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „ALLOCATE DESCRIPTOR-Anweisung [ESQL]“ auf Seite 449
- „DECLARE CURSOR-Anweisung [ESQL] [SP]“ auf Seite 778
- „OPEN-Anweisung [ESQL] [SP]“ auf Seite 964
- „PREPARE-Anweisung [ESQL]“ auf Seite 976
- „db_change_nchar_charset-Funktion“ [*SQL Anywhere Server - Programmierung*]
- „Der SQL-Deskriptor-Bereich (SQLDA)“ [*SQL Anywhere Server - Programmierung*]
- „LONG NVARCHAR-Datentyp“ auf Seite 97
- „NCHAR-Datentyp“ auf Seite 98
- „NVARCHAR-Datentyp“ auf Seite 99

Standards und Kompatibilität

- **SQL/2008** Die DESCRIBE OUTPUT-Anweisung ist die optionale SQL-Sprachenfunktion B031 (Basic Dynamic SQL) des SQL/2008-Standards. Die DESCRIBE INPUT-Anweisung ist die optionale SQL-Sprachenfunktion B031 (Extended Dynamic SQL) des SQL/2008-Standards. Viele der anderen Klauseln der DESCRIBE-Anweisung sind Erweiterungen des Herstellers. Es handelt sich dabei um die folgenden:
 - Die Klauseln USER TYPES, ALL, BIND VARIABLES FOR, LONG NAMES und WITH VARIABLE RESULT.
 - DESCRIBE verwendet die INTO-Klausel zur Identifizierung der SQLDA. Im SQL/2008-Standard wird stattdessen das Schlüsselwort USING verwendet.
 - Im SQL/2008-Standard endet die CURSOR-Klausel mit dem Schlüsselwort STRUCTURE. STRUCTURE wird von SQL Anywhere nicht unterstützt.

Beispiel

Das folgende Beispiel zeigt die Verwendung der DESCRIBE-Anweisung:

```
sqlda = alloc_sqlda( 3 );  
EXEC SQL DESCRIBE OUTPUT
```

```
FOR employee_statement
INTO sqlda;
if( sqlda->sqld > sqlda->sqln ) {
  actual_size = sqlda->sqld;
  free_sqlda( sqlda );
  sqlda = alloc_sqlda( actual_size );
  EXEC SQL DESCRIBE OUTPUT
    FOR employee_statement
    INTO sqlda;
}
```

DESCRIBE-Anweisung [Interactive SQL]

Gibt Informationen über ein bestimmtes Datenbankobjekt zurück.

Syntax 1 - Beschreiben von Datenbankobjekten

DESCRIBE [[INDEX FOR] TABLE | PROCEDURE] [owner.]object-name

object-name :

- table
- | view
- | materialized view
- | procedure
- | function

Syntax 2 - Beschreiben der laufenden Verbindung

DESCRIBE CONNECTION

Parameter

INDEX FOR-Klausel Gibt an, dass Sie die Indizes für den angegebenen *object-name* anzeigen möchten

TABLE-Klausel Gibt an, dass *object-name* eine Tabelle oder Ansicht ist.

PROCEDURE-Klausel Gibt an, dass *object-name* eine Prozedur oder eine Funktion ist.

Bemerkungen

Verwenden Sie DESCRIBE TABLE, um alle Spalten in der angegebenen Tabelle oder Ansicht aufzulisten. Die DESCRIBE TABLE-Anweisung gibt eine Zeile pro Tabellenspalte zurück, die Folgendes enthält:

- **Spalte** Der Name der Spalte.
- **Typ** Der Datentyp in der Spalte
- **Nullwertfähig** Gibt an, ob NULL zulässig ist (1=ja, 0=nein)
- **Primärschlüssel** Gibt an, ob die Spalte im Primärschlüssel enthalten ist (1=ja, 0=nein).

Verwenden Sie DESCRIBE INDEX FOR TABLE, um alle Indizes für die angegebene Tabelle aufzulisten. Die DESCRIBE TABLE-Anweisung gibt eine Zeile pro Index zurück, die Folgendes enthält:

- **Indexname** Der Name des Indexes.
- **Spalten** Die Spalten im Index
- **Eindeutig** Gibt an, ob der Index eindeutig ist (1=ja, 0=nein)
- **Typ** Der Indextyp. Mögliche Angaben sind: Clustered, Statistik, Hashed und Andere.

Verwenden Sie `DESCRIBE PROCEDURE`, um alle Parameter aufzulisten, die von der angegebenen Prozedur oder Funktion verwendet werden. Die `DESCRIBE PROCEDURE`-Anweisung gibt eine Zeile pro Parameter zurück, die Folgendes enthält:

- **Parameter** Der Name des Parameters.
- **Typ** Der Datentyp des Parameters
- **Eingabe/Ausgabe** Informationen darüber, was an den Parameter übergeben bzw. von ihm zurückgegeben wird. Die möglichen Werte sind:
 - **Eingabe** Der Parameter wird an die Prozedur übergeben, aber nicht geändert.
 - **Ausgabe** Die Prozedur ignoriert den Anfangswert des Parameters und stellt den Wert ein, wenn die Prozedur zurückgibt.
 - **Eingabe/Ausgabe** Der Parameter wird an die Prozedur übergeben und die Prozedur stellt den Wert des Parameters ein, wenn die Prozedur zurückgibt.
 - **Ergebnis** Der Parameter gibt eine Ergebnismenge zurück.
 - **Rückgabe** Der Parameter gibt einen deklarierten Rückgabewert zurück.

Wenn Sie weder `TABLE` noch `PROCEDURE` angeben (z.B. `DESCRIBE object-name`), nimmt Interactive SQL an, dass das Objekt eine Tabelle ist. Wenn jedoch die Tabelle nicht existiert, versucht Interactive SQL, das Objekt entweder als Prozedur oder als Funktion zu beschreiben.

Benutzen Sie Syntax 2, um Informationen über die Datenbank oder den Datenbankserver aufzulisten, mit der bzw. dem Interactive SQL verbunden ist. Folgende Eigenschaften werden zurückgegeben:

- **Datenbankprodukt** Der Name und die Versionsnummer der Datenbank, mit der Interactive SQL verbunden ist (z.B. SQL Anywhere 16.0.0.1403).
- **Hostname** Der Netzwerkname des Computers, auf dem der Datenbankserver läuft.
- **Host-TCP/IP-Adresse** Die IP-Adresse des Computers, auf dem der Datenbankserver läuft.
- **Host-Betriebssystem** Der Name und die Versionsnummer des Betriebssystems des Computers, auf dem der Datenbankserver läuft.
- **Servename** Der Name des Datenbankservers.
- **TCP/IP-Port des Servers** Die Portnummer des Datenbankservers für die laufende Verbindung.

- **Datenbankname** Der Name der Datenbank, mit der Interactive SQL verbunden ist.
- **Zeichensatz der Datenbank** Der Zeichensatz, der für CHAR-Spalten in der Datenbank verwendet wird.
- **Verbindungszeichenfolge** Die Verbindungszeichenfolge, die benutzt wurde, um die Verbindung mit dem Datenbankserver herzustellen. Drei Sternchen ersetzen Kennwörter.

Eigenschaften, die nicht für die laufende Verbindung gelten, werden weggelassen. Beispiel: Wenn Sie sich mit einer Datenbank mit Shared Memory verbinden, wird der TCP/IP-Port weggelassen.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Beschreiben der Spalten in der Tabelle Departments:

```
DESCRIBE TABLE GROUPO.Departments;
```

Nachfolgend wird ein Ergebnisbeispiel für diese Anweisung gezeigt:

Column	Type	Nullable	Primary key
DepartmentID	integer	0	1
DepartmentName	char(40)	0	0
DepartmentHeadID	integer	1	0

Auflistung der Indizes für die Customers-Tabelle:

```
DESCRIBE INDEX FOR TABLE GROUPO.Customers;
```

Nachfolgend wird ein Ergebnisbeispiel für diese Anweisung gezeigt:

Index Name	Columns	Unique	Type
IX_customer_name	Surname,GivenName	0	Clustered

DETACH TRACING-Anweisung

Beendet eine Diagnoseprotokollierungssitzung.

Syntax

DETACH TRACING { WITH | WITHOUT } SAVE

Parameter

WITH SAVE-Klausel Geben Sie WITH SAVE an, um nicht gespeicherte Diagnosedaten in den Diagnosetabellen zu speichern.

WITHOUT SAVE-Klausel Geben Sie WITHOUT SAVE an, wenn nicht gespeicherte Protokollierungsdaten nicht gespeichert werden sollen.

Bemerkungen

Führen Sie diese Anweisung aus der Datenbank aus, deren Profil erstellt wird, um das Senden von Diagnosedaten an die Diagnosetabellen zu stoppen. Wenn Sie die WITHOUT SAVE-Klausel angeben, können Sie die Daten immer noch später speichern - falls die Protokollierungsdatenbank weiterhin läuft und keine weitere Protokollierungssitzung gestartet wurde - indem Sie die Systemprozedur `sa_save_trace_data` verwenden.

Die aktuellen für eine Datenbank gesetzten Protokollierungsstufen finden Sie in der `sa_diagnostic_tracing_level`-Tabelle.

Hinweis

Protokollierungsdaten werden *nicht* als Teil eines Datenbank-Entladungs- oder -Aktualisierungsvorgangs entladen. Wenn Sie Protokollierungsinformationen aus einer Datenbank in eine andere übertragen möchten, müssen Sie dies manuell tun, indem Sie den Inhalt der `sa_diagnostic_*`-Tabellen kopieren. Dies wird jedoch nicht empfohlen.

Privilegien

Sie müssen das `MANAGE PROFILING`-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „[ATTACH TRACING-Anweisung](#)“ auf Seite 546
- „[REFRESH TRACING LEVEL-Anweisung](#)“ auf Seite 993
- „[Diagnoseprotokollierung](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „[sa_diagnostic_tracing_level-Tabelle](#)“ auf Seite 1156
- „[sa_save_trace_data-Systemprozedur](#)“ auf Seite 1302

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

DISCONNECT-Anweisung [ESQL] [Interactive SQL]

Löscht eine Verbindung mit einer Datenbank.

Syntax

DISCONNECT [*connection-name* | **CURRENT** | **ALL**]

connection-name :
identifier
| *string*
| *hostvar*

Bemerkungen

Die DISCONNECT-Anweisung trennt eine Verbindung mit dem Datenbankserver und gibt alle von ihr benutzten Ressourcen frei. Wenn die zu beendende Verbindung in einer CONNECT-Anweisung benannt wurde, kann der Name angegeben werden. Wenn Sie ALL angeben, werden alle Verbindungen der Anwendung zu allen Datenbankumgebungen beendet. CURRENT ist der Standardwert, sodass die aktuelle Verbindung gelöscht wird.

Bevor es die Datenbankverbindung beendet, führt Interactive SQL automatisch eine COMMIT-Anweisung aus, falls die Option commit_on_exit auf ON gesetzt ist. Wenn diese Option auf OFF gesetzt ist, führt Interactive SQL ein implizites ROLLBACK durch. Standardmäßig ist die Option commit_on_exit auf ON gesetzt.

Diese Anweisung wird in Prozeduren, Triggern, Ereignissen oder Batches nicht unterstützt.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „DROP CONNECTION-Anweisung“ auf Seite 803
- „CONNECT-Anweisung [ESQL] [Interactive SQL]“ auf Seite 578
- „SET CONNECTION-Anweisung [Interactive SQL] [ESQL]“ auf Seite 1032
- „Interactive SQL“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** DISCONNECT umfasst die optionale SQL-Sprachenfunktion F771 des SQL/2008-Standards. Die Möglichkeit zum Angeben von DISCONNECT ohne Parameter ist eine Erweiterung des Herstellers. Die Option commit_on_exit ist eine Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung zeigt, wie DISCONNECT in Embedded SQL verwendet wird:

```
EXEC SQL DISCONNECT :conn_name
```

Die folgende Anweisung zeigt, wie DISCONNECT in Interactive SQL verwendet wird, um alle Verbindungen zu trennen:

```
DISCONNECT ALL;
```

DROP CERTIFICATE-Anweisung

Löscht ein Zertifikat aus der Datenbank.

Syntax

```
DROP CERTIFICATE certificate-name
```

Bemerkungen

DROP CERTIFICATE löscht ein Zertifikat aus der ISYSCERTIFICATE-Systemtabelle.

Privilegien

Sie müssen das MANAGE CERTIFICATES-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- [„CREATE CERTIFICATE-Anweisung“ auf Seite 582](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

```
DROP CERTIFICATE mycert;
```

DROP CONNECTION-Anweisung

Trennt die Verbindung eines Benutzers mit einer Datenbank.

Syntax

```
DROP CONNECTION connection-id
```

Bemerkungen

Die DROP CONNECTION-Anweisung trennt die Verbindung eines Benutzers mit der Datenbank, indem diese Verbindung gelöscht wird.

Der *connection-id*-Parameter ist eine Ganzzahlkonstante. Sie können die *connection-id* abrufen, indem Sie die sa_conn_info-Systemprozedur verwenden.

Diese Anweisung wird in Prozeduren, Triggern, Ereignissen oder Batches nicht unterstützt.

Privilegien

Sie müssen das DROP CONNECTION-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „CONNECT-Anweisung [ESQL] [Interactive SQL]“ auf Seite 578
- „sa_conn_info-Systemprozedur“ auf Seite 1183
- „Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Prozedur trennt eine Verbindung, die anhand ihrer Verbindungsnummer identifiziert wird. Wenn Sie die DROP CONNECTION-Anweisung aus einer Prozedur heraus ausführen, verwenden Sie dafür die EXECUTE IMMEDIATE-Anweisung, wie in diesem Beispiel gezeigt:

```
CREATE PROCEDURE drop_connection_by_id( IN conn_number INTEGER )
BEGIN
    EXECUTE IMMEDIATE 'DROP CONNECTION ' || conn_number;
END;
```

Die folgende Anweisung löscht die Verbindung mit der ID '4'.

```
DROP CONNECTION 4;
```

DROP DATABASE-Anweisung

Löscht alle Datenbankdateien, die einer Datenbank zugeordnet sind.

Syntax

```
DROP DATABASE database-name [ KEY key ]
```

Bemerkungen

Die Anweisung DROP DATABASE löscht alle zugehörigen Datenbankdateien physisch von der Festplatte. Wenn die Datenbankdatei nicht existiert oder nicht so beschaffen ist, dass die Datenbank gestartet werden kann, wird ein Fehler ausgegeben.

DROP DATABASE kann nicht in gespeicherten Prozeduren, Triggern, Ereignissen oder Batches verwendet werden.

Die zu löschende Datenbank darf nicht laufen, wenn die DROP DATABASE-Anweisung ausgeführt wird. Sie können nicht mit der Datenbank verbunden sein, den Sie löschen möchten. Sie müssen mit einer anderen Datenbank verbunden sein, z.B. mit der Dienstprogrammdateiabank.

Wenn Sie eine stark verschlüsselte Datenbank löschen möchten, müssen Sie einen Schlüssel angeben. Der Schlüssel kann entweder eine Zeichenfolge oder ein Variablenname sein.

Diese Anweisung wird unter Windows Mobile nicht unterstützt.

Privilegien

Ob Sie diese Anweisung ausführen können, hängt von der Einstellung der Datenbankoption `-gu` ab und davon, ob Sie das `SERVER OPERATOR`-Systemprivileg haben.

Nebenwirkungen

Zusätzlich zur Datenbankdatei wird auch jede zugehörige Transaktionslog- oder Transaktionslogspiegeldatei von der Festplatte gelöscht.

Siehe auch

- „Datenbankserveroption `-gu`“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Löschen-Dienstprogramm (dberase)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Windows Mobile-Datenbanken über die Benutzeroberfläche des Geräts löschen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Verbindung mit der Dienstprogrammdatenbank herstellen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „CREATE DATABASE-Anweisung“ auf Seite 583
- „Verbindungsparameter DatabaseKey (DBKEY)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Führen Sie die folgende Anweisung aus, um die Datenbank `temp.db` zu löschen:

```
DROP DATABASE 'c:\temp\temp.db';
```

DROP DATATYPE-Anweisung

Entfernt einen Datentyp aus der Datenbank.

Syntax

```
DROP DATATYPE datatype-name
```

Bemerkungen

Es wird empfohlen, `DROP DOMAIN` gegenüber `DROP DATATYPE` vorzuziehen, da `DROP DOMAIN` die im SQL/2008-Standard verwendete Syntax ist. Sie können keine systemdefinierten Datentypen (wie `MONEY` oder `UNIQUEIDENTIFIER`) aus einer Datenbank löschen.

Privilegien

Sie müssen entweder Eigentümer des Datentyps sein oder das `DROP DATATYPE`-Systemprivileg oder das `DROP ANY OBJECT`-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Löscht den Inhalt der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** in Interactive SQL.

Siehe auch

- „DROP DOMAIN-Anweisung“ auf Seite 807
- „CREATE DOMAIN-Anweisung“ auf Seite 598
- „ALTER DOMAIN-Anweisung“ auf Seite 460

Standards und Kompatibilität

- **SQL/2008** Die Domänenunterstützung ist die optionale SQL-Sprachenfunktion F251 des SQL/2008-Standards. Die DROP DATATYPE-Anweisung ist eine Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird ein Datentyp namens PhoneNum erstellt und anschließend gelöscht:

```
CREATE DATATYPE PhoneNum CHAR(12) NULL;  
DROP DATATYPE PhoneNum;
```

DROP DBSPACE-Anweisung

Entfernt einen DBSpace aus der Datenbank

Syntax

DROP DBSPACE *dbspace-name*

Bemerkungen

Sie müssen alle Tabellen im DBSpace löschen, bevor Sie den DBSpace löschen. Sie können die DROP DBSPACE-Anweisung nicht verwenden, um die vordefinierten DBSpaces SYSTEM, TEMPORARY, TEMP, TRANSLOG oder TRANSLOGMIRROR zu löschen.

DROP DBSPACE wird verhindert, wenn die Anweisung ein Objekt betrifft, das zeitgleich von einer anderen Verbindung benutzt wird.

Sie müssen als einziger Benutzer mit der Datenbank verbunden sein, um diese Anweisung ausführen zu können.

Privilegien

Sie müssen das MANAGE ANY DBSPACE-Systemprivileg haben.

Nebenwirkungen

Bewirkt ein automatisches Festschreiben und setzt einen impliziten Checkpoint. Löscht den Inhalt der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** in Interactive SQL.

Siehe auch

- „CREATE DBSPACE-Anweisung“ auf Seite 593
- „ALTER DBSPACE-Anweisung“ auf Seite 457
- „DBSpaces löschen (Sybase Central)“ [*SQL Anywhere Server - Datenbankadministration*]
- „DBSpaces löschen (SQL)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Vordefinierte DBSpaces“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird ein fiktiver DBSpace, MyDBSpace, aus der Datenbank gelöscht.

```
DROP DBSPACE MyDBSpace;
```

DROP DOMAIN-Anweisung

Entfernt eine Domäne (einen Datentyp) aus der Datenbank.

Syntax

```
DROP DOMAIN domain-name
```

Bemerkungen

DROP DOMAIN wird nicht durchgeführt, wenn der Datentyp in einer Tabellenspalte oder einem Prozedur- bzw. Funktionsargument verwendet wird. Sie müssen die Datentypen für alle Spalten ändern, die für die Domäne definiert sind, um den Datentyp zu löschen. Es wird empfohlen, DROP DOMAIN gegenüber DROP DATATYPE vorzuziehen, da DROP DOMAIN die im SQL/2008-Standard verwendete Syntax ist. Sie können keine systemdefinierten Datentypen (wie MONEY oder UNIQUEIDENTIFIER) aus einer Datenbank löschen.

Privilegien

Sie müssen entweder Eigentümer der Domäne sein oder das DROP DATATYPE-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Löscht den Inhalt der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** in Interactive SQL.

Siehe auch

- „CREATE DOMAIN-Anweisung“ auf Seite 598
- „ALTER DOMAIN-Anweisung“ auf Seite 460

Standards und Kompatibilität

- **SQL/2008** Die Domänenunterstützung ist die optionale SQL-Sprachenfunktion F251 des SQL/2008-Standards.

Beispiel

Im folgenden Beispiel wird die Domäne CustPhoneNumber erstellt und anschließend gelöscht:

```
CREATE DOMAIN CustPhoneNumber CHAR(12) NULL;  
DROP DOMAIN CustPhoneNumber;
```

DROP EVENT-Anweisung

Löscht ein Ereignis aus der Datenbank.

Syntax

```
DROP EVENT [ IF EXISTS ] [ owner.]event-name
```

Bemerkungen

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP EVENT-Anweisung versucht, ein Ereignis zu entfernen, das nicht existiert.

Privilegien

Sie müssen entweder das MANAGE ANY EVENT-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Löscht den Inhalt der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** in Interactive SQL.

Siehe auch

- [„CREATE EVENT-Anweisung“ auf Seite 606](#)
- [„ALTER EVENT-Anweisung“ auf Seite 461](#)
- [„TRIGGER EVENT-Anweisung“ auf Seite 1088](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird ein fiktives Ereignis, MyEvent, aus der Datenbank gelöscht.

```
DROP EVENT MyEvent;
```

DROP EXTERNLOGIN-Anweisung

Löscht ein externes Login aus der Datenbank.

Syntax

```
DROP EXTERNLOGIN login-name TO remote-server
```

Parameter

DROP-Klausel Gibt die lokale Login-ID des Benutzers an.

TO-Klausel Gibt den Namen des Fremdservers an. Der alternative Loginname des lokalen Benutzers und sein Kennwort für diesen Server sind das externe Login, das gelöscht wird.

Bemerkungen

DROP EXTERNLOGIN löscht ein externes Login aus der Datenbank.

Privilegien

Sie müssen das MANAGE ANY USER-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- [„CREATE EXTERNLOGIN-Anweisung“ auf Seite 616](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird das externe Login DBA für den fiktiven Fremdservers sybase1 gelöscht.

```
DROP EXTERNLOGIN DBA TO sybase1;
```

DROP FUNCTION-Anweisung

Entfernt eine Funktion aus der Datenbank

Syntax

```
DROP FUNCTION [ IF EXISTS ] [ owner.]function-name
```

Bemerkungen

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP FUNCTION-Anweisung versucht, eine Funktion zu entfernen, die nicht existiert.

DROP FUNCTION wird verhindert, wenn die Anweisung ein Objekt betrifft, das zeitgleich von einer anderen Verbindung benutzt wird.

Privilegien

Sie müssen entweder Eigentümer der Funktion sein oder das DROP ANY PROCEDURE-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Löscht den Inhalt der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** in Interactive SQL.

Siehe auch

- „CREATE FUNCTION-Anweisung“ auf Seite 633
- „CREATE FUNCTION-Anweisung [externer Aufruf]“ auf Seite 617
- „CREATE FUNCTION-Anweisung [Webdienst]“ auf Seite 624
- „ALTER FUNCTION-Anweisung“ auf Seite 465

Standards und Kompatibilität

- **SQL/2008** Kernfunktion. Die IF EXISTS-Klausel ist eine Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird die fiktive Funktion MyFunction aus der Datenbank gelöscht.

```
DROP FUNCTION MyFunction;
```

DROP INDEX-Anweisung

Entfernt einen Index aus der Datenbank.

Syntax

```
DROP INDEX [ IF EXISTS ] { [ [ owner.]table-name.]index-name | [ [ owner.]materialized-view-name. ]index-name }
```

Bemerkungen

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP INDEX-Anweisung versucht, einen Index zu entfernen, der nicht existiert.

Wenn Sie die IF EXISTS-Klausel angeben und die benannte Tabelle nicht gefunden werden kann, wird ein Fehler zurückgegeben.

DROP INDEX wird verhindert, wenn die Anweisung ein Objekt betrifft, das zeitgleich von einer anderen Verbindung benutzt wird.

Die DROP INDEX-Anweisung kann nicht ausgeführt werden, wenn mit der WITH HOLD-Klausel geöffnete Cursor vorhanden sind, die Anweisungs- oder Transaktions-Snapshots verwenden.

Privilegien

Wenn Sie einen Index für eine Tabelle löschen möchten, müssen Sie der Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- REFERENCES-Privileg für die Tabelle
- DROP ANY INDEX-Systemprivileg
- DROP ANY OBJECT-Systemprivileg

Wenn Sie einen Index für eine materialisierte Ansicht löschen möchten, müssen Sie Eigentümer der materialisierten Ansicht sein oder eines der folgenden Privilegien haben:

- DROP ANY INDEX-Systemprivileg
- DROP ANY OBJECT-Systemprivileg

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Löscht den Inhalt der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** in Interactive SQL. Die DROP INDEX-Anweisung schließt alle Cursor für die aktuelle Verbindung.

Wenn Sie die DROP INDEX-Anweisung verwenden, um einen Index für eine lokale temporäre Tabelle zu löschen, wird ein Fehler zurückgegeben, wonach der Index nicht gefunden werden konnte. Verwenden Sie die DROP TABLE-Anweisung, um eine lokale temporäre Tabelle zu löschen. Indizes für lokale, temporäre Tabellen werden automatisch gelöscht, wenn die lokale, temporäre Tabelle gelöscht wird.

Siehe auch

- „CREATE INDEX-Anweisung“ auf Seite 638
- „ALTER INDEX-Anweisung“ auf Seite 466
- „Snapshot-Isolation“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird ein fiktiver Index, MyIndex, aus der Datenbank gelöscht.

```
DROP INDEX MyIndex;
```

DROP LDAP SERVER-Anweisung

Löscht ein LDAP-Serverkonfigurationsobjekt.

Syntax

```
DROP LDAP SERVER ldapua-server-name  
[ WITH DROP ALL REFERENCES ]  
[ WITH SUSPEND ]
```

Parameter

WITH DROP ALL REFERENCES-Klausel Geben Sie die DROP ALL REFERENCES-Klausel an, wenn Sie ein LDAP-Serverkonfigurationsobjekt löschen möchten, das aus einer Login-Richtlinie heraus referenziert wird.

WITH SUSPEND-Klausel Geben Sie die WITH SUSPEND-Klausel an, wenn Sie ein LDAP-Serverkonfigurationsobjekt löschen möchten, das sich im Zustand READY oder ACTIVE befindet.

Bemerkungen

Diese Anweisung entfernt das LDAP-Serverkonfigurationsobjekt aus der SYSLDAPSERVER-Systemansicht, nachdem sie geprüft hat, ob sich der LDAP-Server im Zustand READY oder ACTIVE befindet. Die Anweisung schlägt fehl, wenn der Zustand READY oder ACTIVE lautet, um zu gewährleisten, dass der Server nicht aktiv in Gebrauch ist. Wenn Sie diese Überprüfung aufheben möchten, geben Sie die WITH SUSPEND-Klausel an.

Standardmäßig führt eine Referenz auf das LDAP-Serverkonfigurationsobjekt in einer Login-Richtlinie ebenfalls zum Fehlschlagen dieser Anweisung. Wenn Sie LDAP-Serverkonfigurationsobjekt entfernen möchten, das in einer Login-Richtlinie referenziert wird, fügen Sie die DROP ALL REFERENCES-Klausel hinzu. Durch das Hinzufügen von DROP ALL REFERENCES wird nicht die Referenz aus der Login-Richtlinie gelöscht, sondern lediglich zugelassen, dass Sie das referenzierte Konfigurationsobjekt löschen. Sie müssen trotzdem die Referenz auf das LDAP-Serverkonfigurationsobjekt aus der Login-Richtlinie entfernen.

Privilegien

Sie müssen das MANAGE ANY LDAP SERVER-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „LDAP-Benutzerauthentifizierung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „CREATE LDAP SERVER-Anweisung“ auf Seite 642
- „ALTER LDAP SERVER-Anweisung“ auf Seite 468
- „VALIDATE LDAP SERVER-Anweisung“ auf Seite 1116
- „ALTER LOGIN POLICY-Anweisung“ auf Seite 471

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird ein in einer Login-Richtlinie referenziertes LDAP-Serverkonfigurationsobjekt namens apps_primary unterbrochen und anschließend gelöscht.

```
DROP LDAP SERVER apps_primary WITH DROP ALL REFERENCES WITH SUSPEND;
```

DROP LOGIN POLICY-Anweisung

Löscht eine Login-Richtlinie.

Syntax

```
DROP LOGIN POLICY policy-name
```

Parameter

policy-name Der Name der Login-Richtlinie

Bemerkungen

Diese Anweisung schlägt fehl, wenn Sie eine Richtlinie löschen, die einem Benutzer zugewiesen ist. Sie können die Root-Login-Richtlinie nicht löschen. Benutzen Sie die ALTER USER-Anweisung, um die Zuweisung der Richtlinie eines Benutzers zu ändern.

Privilegien

Sie müssen das MANAGE ANY LOGIN POLICY-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „ALTER LOGIN POLICY-Anweisung“ auf Seite 471
- „ALTER USER-Anweisung“ auf Seite 540
- „COMMENT-Anweisung“ auf Seite 573
- „CREATE LOGIN POLICY-Anweisung“ auf Seite 647
- „CREATE USER-Anweisung“ auf Seite 770
- „DROP USER-Anweisung“ auf Seite 838
- „Login-Richtlinien“ [*SQL Anywhere Server - Datenbankadministration*]
- „Login-Richtlinien löschen“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Das folgende Beispiel erstellt eine Login-Richtlinie Test11 und löscht sie.

```
CREATE LOGIN POLICY Test11;  
DROP LOGIN POLICY Test11;
```

DROP MATERIALIZED VIEW-Anweisung

Entfernt eine materialisierte Ansicht aus der Datenbank.

Syntax

```
DROP MATERIALIZED VIEW [ IF EXISTS ] [ owner.]materialized-view-name
```

Bemerkungen

Bei diesem Vorgang werden alle Daten in der materialisierten Ansicht automatisch gelöscht. Alle Indizes und Schlüssel für die materialisierte Ansicht werden ebenfalls gelöscht.

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP MATERIALIZED VIEW-Anweisung versucht, eine materialisierte Ansicht zu entfernen, die nicht existiert.

Sie können keine DROP MATERIALIZED VIEW-Anweisung für ein Objekt ausführen, das aktuell von einer anderen Verbindung benutzt wird.

Durch das Ausführen einer DROP MATERIALIZED VIEW-Anweisung wird der Status aller abhängigen regulären Ansichten auf INVALID gesetzt. Um Ansichtsabhängigkeiten vor dem Löschen einer materialisierten Ansicht zu ermitteln, verwenden Sie die Systemprozedur `sa_dependent_views`.

Privilegien

Sie müssen entweder Eigentümer der materialisierten Ansicht sein oder das DROP ANY MATERIALIZED VIEW-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Falls die materialisierte Ansicht mit Daten gefüllt wurde, löst DROP MATERIALIZED VIEW einen automatischen Checkpoint aus. Löscht den Inhalt der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** in Interactive SQL. Schließt alle Cursor für die aktuelle Verbindung.

Wird eine Ansicht gelöscht, werden alle Prozeduren und Trigger im Speicher entladen. Auf diese Weise berücksichtigen Prozeduren und Trigger, die die Ansicht referenzieren, dass die Ansicht nicht existiert. Das Entladen und Laden von Prozeduren und Triggern kann sich auf die Performance auswirken, wenn Sie Ansichten häufig löschen und erstellen.

Siehe auch

- „CREATE MATERIALIZED VIEW-Anweisung“ auf Seite 652
- „ALTER MATERIALIZED VIEW-Anweisung“ auf Seite 476
- „REFRESH MATERIALIZED VIEW-Anweisung“ auf Seite 987
- „Fortgeschrittene Aufgaben: Status und Eigenschaften von materialisierten Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „sa_dependent_views-Systemprozedur“ auf Seite 1202

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird eine fiktive materialisierte Ansicht, `MyMaterializedView`, aus der Datenbank gelöscht.

```
DROP MATERIALIZED VIEW MyMaterializedView;
```

DROP MESSAGE-Anweisung

Entfernt eine Nachricht aus der Datenbank.

Syntax

```
DROP MESSAGE msgnum
```

Bemerkungen

Keine.

Privilegien

Sie müssen entweder Eigentümer sein oder das DROP MESSAGE-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Löscht den Inhalt der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** in Interactive SQL.

Siehe auch

- „PRINT-Anweisung [T-SQL]“ auf Seite 980
- „CREATE MESSAGE-Anweisung [T-SQL]“ auf Seite 655
- „SYSUSERMESSAGE-Systemansicht“ auf Seite 1510

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** DROP MESSAGE liefert die von der sp_dropmessage()-Systemprozedur in Adaptive Server Enterprise bereitgestellte Funktionalität.

Beispiel

Im folgenden Beispiel wird eine neue Nachricht erstellt und anschließend gelöscht. Um dieses Beispiel ausführen zu können, benötigen Sie auch das CREATE MESSAGE-Systemprivileg:

```
CREATE MESSAGE 20000 AS 'End of line reached';
DROP MESSAGE 20000;
```

DROP MIRROR SERVER-Anweisung

Hinweis

Scale-Out mit Schreibschutz und Datenbankspiegelung erfordern jeweils eine getrennte Lizenz. Siehe „Getrennt lizenzierte Komponenten“ [SQL Anywhere 16 - Einführung].

Löscht einen Spiegelserver.

Syntax

```
DROP MIRROR SERVER mirror-server-name
```

Bemerkungen

Entfernt die angegebene Spiegelserverdefinition aus der Datenbank.

Die Spiegeldatenbank wird gestoppt. Wenn die Spiegeldatenbank die einzige laufende Datenbank auf dem Server ist, wird der Server ebenfalls gestoppt.

Privilegien

Sie müssen das **MANAGE ANY MIRROR SERVER**-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Datenbankspiegelung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „CREATE MIRROR SERVER-Anweisung“ auf Seite 656
- „ALTER MIRROR SERVER-Anweisung“ auf Seite 480
- „COMMENT-Anweisung“ auf Seite 573

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers

Beispiel

In diesem Beispiel wird ein Spiegelserver namens `scaleout_primary2` erstellt und anschließend gelöscht:

```
CREATE MIRROR SERVER "scaleout_primary2"  
  AS PRIMARY  
  connection_string =  
'server=scaleout_primary1;host=winxp-2:6871,winxp-3:6872';  
DROP MIRROR SERVER "scaleout_primary2";
```

DROP PROCEDURE-Anweisung

Entfernt eine Prozedur aus der Datenbank.

Syntax

```
DROP PROCEDURE [ IF EXISTS ] [ owner.]procedure-name
```

Bemerkungen

Verwenden Sie die **IF EXISTS**-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die **DROP PROCEDURE**-Anweisung versucht, eine Prozedur zu entfernen, die nicht existiert.

Sie können eine **DROP PROCEDURE**-Anweisung nicht ausführen, wenn die Anweisung ein Objekt betrifft, das aktuell von einer anderen Verbindung benutzt wird.

Privilegien

Sie müssen entweder Eigentümer der Prozedur sein oder das **DROP ANY PROCEDURE**-Systemprivileg oder das **DROP ANY OBJECT**-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Löscht den Inhalt der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** in Interactive SQL.

Siehe auch

- „CREATE PROCEDURE-Anweisung“ auf Seite 681
- „CREATE PROCEDURE-Anweisung [externer Aufruf]“ auf Seite 661
- „CREATE PROCEDURE-Anweisung [Webdienste]“ auf Seite 670
- „ALTER PROCEDURE-Anweisung“ auf Seite 483

Standards und Kompatibilität

- **SQL/2008** Kernfunktion. Die IF EXISTS-Klausel ist eine Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird eine Prozedur namens NewDepartment erstellt und anschließend gelöscht. Um dieses Beispiel ausführen zu können, benötigen Sie auch das CREATE PROCEDURE-Systemprivileg:

```
CREATE PROCEDURE NewDepartment(  
    IN id INT,  
    IN name CHAR(35),  
    IN head_id INT )  
BEGIN  
    INSERT  
    INTO GROUPO.Departments ( DepartmentID,  
        DepartmentName, DepartmentHeadID )  
    VALUES ( id, name, head_id );  
END;  
DROP PROCEDURE NewDepartment;
```

DROP PUBLICATION-Anweisung [MobiLink] [SQL Remote]

Löscht eine Publikation.

Syntax

DROP PUBLICATION [IF EXISTS] [*owner*.]*publication-name*

owner, publication-name : *identifier*

Bemerkungen

Diese Anweisung gilt nur für MobiLink und SQL Remote.

In MobiLink kennzeichnet eine Publikation synchronisierte Daten in einer entfernten SQL Anywhere-Datenbank. In SQL Remote kennzeichnen Publikationen replizierte Daten sowohl in konsolidierten als auch in entfernten Datenbanken.

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP PUBLICATION-Anweisung versucht, eine Publikation zu entfernen, die nicht existiert.

DROP PUBLICATION erfordert exklusiven Zugriff auf alle in der Publikation referenzierten Tabellen.

Privilegien

Sie müssen Eigentümer der Publikation sein oder die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Alle Subskriptionen für die Publikation werden gelöscht.

Siehe auch

- „ALTER PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ auf Seite 485
- „CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ auf Seite 690
- SQL Anywhere MobiLink-Clients: „Publikationen“ [*MobiLink - Clientadministration*]
- UltraLite MobiLink-Clients: „DROP PUBLICATION-Anweisung [UltraLite]“ [*UltraLite - Datenbankverwaltung*]
- SQL Anywhere MobiLink-Clients: „Publikation löschen“ [*MobiLink - Clientadministration*]
- SQL Remote: „Publikation löschen“ [*SQL Remote*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Mit der folgenden Anweisung wird die Publikation pub_contact gelöscht.

```
DROP PUBLICATION pub_contact;
```

DROP REMOTE MESSAGE TYPE-Anweisung [SQL Remote]

Löscht eine Nachrichtentypdefinition aus einer Datenbank.

Syntax

DROP REMOTE MESSAGE TYPE *message-system*

message-system :

FILE

| **FTP**

| **SMTP**

Bemerkungen

Die Anweisung entfernt einen Nachrichtentyp aus der Datenbank.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE REMOTE [MESSAGE] TYPE-Anweisung [SQL Remote]“ auf Seite 694
- „SQL Remote-Nachrichtensysteme“ [SQL Remote]
- „Einen Nachrichtentyp löschen“ [SQL Remote]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung löscht den FILE-Nachrichtentyp aus einer Datenbank.

```
DROP REMOTE MESSAGE TYPE FILE;
```

DROP REMOTE CONNECTION-Anweisung

Löscht Ferndatenzugriffsverbindungen mit einem Fremdserver.

Syntax

```
DROP REMOTE CONNECTION TO server-name  
CLOSE { CURRENT | ALL | connection-id }  
[ FOR { EFFECTIVE USER | LOGIN USER | USER user-name } ]
```

Parameter

server-name Der Ferndatenzugriffsserver, der in der CREATE SERVER-Anweisung angegeben wurde.

CLOSE-Klausel CLOSE CURRENT löscht entfernte Verbindungen für die aktuelle lokale Verbindung.

CLOSE ALL löscht entfernte Verbindungen für alle lokalen Verbindungen.

CLOSE *connection-id* löscht entfernte Verbindungen für die lokale Verbindung mit der angegebenen ID.

FOR-Klausel FOR EFFECTIVE USER löscht entfernte Verbindungen, die mit den Anmeldeinformationen für das externe Login des derzeit effektiven Benutzers erstellt wurden.

FOR LOGIN USER löscht entfernte Verbindungen, die mit den Anmeldeinformationen für das externe Login des derzeit angemeldeten Benutzers erstellt wurden.

FOR USER *user-name* löscht entfernte Verbindungen, die mit den Anmeldeinformationen für das externe Login von *user-name* erstellt wurden.

Wenn die FOR-Klausel weggelassen wird, werden entfernte Verbindungen für alle Benutzer gelöscht.

Bemerkungen

Mit der DROP REMOTE CONNECTION-Anweisung können Sie explizit Verbindungen mit einem Fremdserver schließen. Das ist nützlich, wenn eine entfernte Verbindung inaktiv oder nicht mehr benötigt wird.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „CREATE SERVER-Anweisung“ auf Seite 701
- „ALTER SERVER-Anweisung“ auf Seite 491

Beispiel

Löschen Sie alle entfernten Verbindungen des effektiven Benutzers mit dem Server myServer, unabhängig davon, ob es sich um die aktuelle Verbindung handelt:

```
DROP REMOTE CONNECTION TO myServer CLOSE ALL FOR EFFECTIVE USER;
```

Löschen sie die entfernte Verbindung mit dem Server myServer für die aktuelle lokale Verbindung von User2:

```
DROP REMOTE CONNECTION TO myServer CLOSE CURRENT FOR USER user2;
```

Löschen Sie alle entfernten Verbindungen von User2 mit dem Server myServer, unabhängig davon, ob es sich um die aktuelle Verbindung handelt:

```
DROP REMOTE CONNECTION TO myServer CLOSE ALL FOR USER user2;
```

Löschen Sie die entfernte Verbindung mit dem Server myServer für die aktuelle lokale Verbindung mit dem Login-Benutzer der aktuellen Verbindung:

```
DROP REMOTE CONNECTION TO myServer CLOSE CURRENT FOR LOGIN USER;
```

Löschen Sie die entfernte Verbindung mit dem Server myServer für die Verbindung mit der Verbindungs-ID "ID" und dem derzeit effektiven Benutzer:

```
DROP REMOTE CONNECTION TO myServer CLOSE connectionId FOR EFFECTIVE USER;
```

DROP ROLE-Anweisung

Entfernt eine Rolle aus der Datenbank oder konvertiert eine benutzererweiterte Rolle in einen normalen Benutzer zurück.

Syntax

```
DROP ROLE [ FROM USER ] role-name  
[ WITH { REVOKE | DROP OBJECTS } ]
```

Parameter

- ***role-name*** Geben Sie den Namen der Rolle an, die Sie löschen bzw. konvertieren möchten.

- **FROM USER-Klausel** Geben Sie diese Klausel an, um eine benutzererweiterte Rolle in einen normalen Benutzer zurückzukonvertieren. Der Benutzer behält alle Login-Privilegien, Systemprivilegien und Rollen, die er hatte.
- **WITH REVOKE-Klausel** Geben Sie WITH REVOKE an, wenn *role-name* anderen Benutzern erteilt wurde.
- **WITH DROP OBJECTS-Klausel** Geben Sie WITH DROP OBJECTS an, um die Objekte zu löschen, deren Eigentümer *role-name* ist. Wenn eines der Objekte nicht gelöscht werden kann, z.B. weil das Objekt verwendet wird, gibt die Anweisung einen Fehler zurück. Sie können diese Klausel nicht angeben, wenn *role-name* eine benutzererweiterte Rolle ist.

Bemerkungen

Eine benutzerdefinierte Rolle kann aus der Datenbank gelöscht oder in einen normalen Benutzer zurückkonvertiert werden, solange alle abhängigen Rollen die erforderliche minimale Anzahl von Administratorbenutzern mit aktiven Kennwörtern erfüllen, die durch die `min_role_admin-`Datenbankoption festgelegt ist.

Wenn Sie eine benutzererweiterte Rolle in einen normalen Benutzer zurückkonvertieren, bleibt das Eigentum an Objekten bei dem Benutzer, der in einen normalen Benutzer zurückkonvertiert wird.

Wenn Sie eine benutzererweiterte Rolle in einen normalen Benutzer zurückkonvertieren, bleiben alle Privilegien, die *role-name* erteilt wurden, nach dem Konvertieren bei dem Benutzer.

Wenn Sie eine benutzererweiterte Rolle in einen normalen Benutzer zurückkonvertieren und die benutzererweiterte Rolle anderen Rollen bzw. Benutzern erteilt wurde, müssen Sie die WITH REVOKE-Klausel angeben. Andernfalls gibt die Anweisung eine Fehlermeldung zurück und schlägt fehl.

Falls von dem Löschvorgang betroffene Objekte in Gebrauch sind, gibt die Anweisung eine Fehlermeldung zurück und schlägt fehl.

Privilegien

Sie benötigen Administrationsrechte für die zu löschende Rolle.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung konvertiert eine benutzererweiterte Rolle namens Joe in einen normalen Benutzer zurück. Objekte, die Eigentum der benutzererweiterten Rolle waren, gehören nun dem normalen Benutzer Joe. Benutzer oder Rollen, denen die Rolle Joe erteilt war, behalten die der Rolle zugrunde liegenden Privilegien.

```
DROP ROLE FROM USER Joe;
```

Die folgende Anweisung löscht eine benutzererweiterte Rolle namens Jack aus der Datenbank. Wenn Objekte Eigentum der Rolle Jack waren, wird der Benutzer Jack wieder Eigentümer der Objekte. Benutzer oder Rollen, denen die Rolle Jack erteilt war, behalten die der Rolle zugrunde liegenden Privilegien.

```
DROP ROLE Jack;
```

Die folgende Anweisung konvertiert eine benutzererweiterte Rolle namens Sam in einen normalen Benutzer zurück. Benutzern oder Rollen, denen die Rolle Sam erteilt war, werden die Privilegien von Jack entzogen.

```
DROP ROLE FROM USER Sam WITH REVOKE;
```

Die folgende Anweisung löscht eine Rolle namens Sales1. Benutzer oder Rollen, denen die Rolle Sales1 erteilt war, behalten die der Rolle zugrunde liegenden Privilegien.

```
DROP ROLE Sales1;
```

Die folgende Anweisung löscht eine Rolle namens Sales2. Benutzer oder Rollen, denen die Rolle Sales2 erteilt war, verlieren alle der Rolle zugrunde liegenden Privilegien.

```
DROP ROLE Sales2 WITH REVOKE;
```

Die folgende Anweisung konvertiert eine benutzererweiterte Rolle namens Marketing1 in einen normalen Benutzer namens Marketing1 und löscht alle Objekte, deren Eigentümerin die Rolle war.

```
DROP ROLE FROM USER Marketing1 WITH DROP OBJECTS;
```

Die folgende Anweisung löscht eine Rolle namens Marketing2, löscht die Objekte, deren Eigentümerin sie war, und entzieht die zugrunde liegenden Privilegien denjenigen, denen die Rolle erteilt war.

```
DROP ROLE Marketing2 WITH REVOKE WITH DROP OBJECTS;
```

Siehe auch

- „ALTER ROLE-Anweisung“ auf Seite 488
- „CREATE ROLE-Anweisung“ auf Seite 696
- „min_role_admins-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Benutzererweiterte Rollen“ [*SQL Anywhere Server - Datenbankadministration*]

DROP SEQUENCE-Anweisung

Löscht eine Sequenz.

Syntax

```
DROP SEQUENCE [ owner.] sequence-name
```

Bemerkungen

Wenn die benannte Sequenz nicht gefunden werden kann, wird eine Fehlermeldung zurückgegeben. Wenn Sie eine Sequenz löschen, werden alle Synonyme für den Namen der Sequenz vom Datenbankserver automatisch gelöscht.

Privilegien

Sie müssen entweder Eigentümer der Sequenz sein oder das DROP ANY SEQUENCE-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Verwendung einer Sequenz zum Generieren von eindeutigen Werten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „ALTER SEQUENCE-Anweisung“ auf Seite 490
- „CREATE SEQUENCE-Anweisung“ auf Seite 699

Standards und Kompatibilität

- **SQL/2008** Sequenzen umfassen die optionale SQL/2008-Sprachenfunktion T176.

Beispiel

Im folgenden Beispiel wird eine Sequenz namens Test erstellt und anschließend gelöscht:

```
CREATE SEQUENCE Test
START WITH 4
INCREMENT BY 2
NO MAXVALUE
NO CYCLE
CACHE 15;
DROP SEQUENCE Test;
```

DROP SERVER-Anweisung

Löscht einen Fremdserver aus dem SQL Anywhere-Katalog.

Syntax

DROP SERVER *server-name*

Bemerkungen

DROP SERVER löscht einen Fremdserver aus den SQL Anywhere-Katalogen. Sie müssen alle Proxytabellen löschen, die für den Fremdserver definiert wurden, bevor diese Anweisung erfolgreich abgeschlossen werden kann.

Diese Anweisung wird unter Windows Mobile nicht unterstützt.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „ALTER SERVER-Anweisung“ auf Seite 491
- „CREATE SERVER-Anweisung“ auf Seite 701
- „DROP REMOTE CONNECTION-Anweisung“ auf Seite 819
- „Ferndatenzugriff“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Fremdserver“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Verzeichniszugriffsserver“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird ein fiktiver Server namens ase_prod gelöscht:

```
DROP SERVER ase_prod;
```

DROP SERVICE-Anweisung

Löscht einen Webdienst.

Syntax

```
DROP SERVICE service-name
```

Bemerkungen

Diese Anweisung löscht einen Webdienst, der in der Systemtabelle ISYSWEBSERVICE enthalten ist.

Privilegien

Sie müssen Eigentümer des Diensts sein oder das MANAGE ANY WEB SERVICE-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „CREATE SERVICE-Anweisung [HTTP-Webdienst]“ auf Seite 706
- „CREATE SERVICE-Anweisung [SOAP-Webdienst]“ auf Seite 713
- „ALTER SERVICE-Anweisung [HTTP-Webdienst]“ auf Seite 494
- „ALTER SERVICE-Anweisung [SOAP-Webdienst]“ auf Seite 500
- „SYSWEBSERVICE-Systemansicht“ auf Seite 1513

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende SQL-Anweisung löscht einen fiktiven Webdienst namens WebServiceTable:

```
DROP SERVICE WebServiceTable;
```

DROP SPATIAL REFERENCE SYSTEM-Anweisung

Löscht ein räumliches Bezugssystem.

Syntax

```
DROP SPATIAL REFERENCE SYSTEM [ IF EXISTS ] name
```

Bemerkungen

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP SPATIAL REFERENCE SYSTEM-Anweisung versucht, ein räumliches Bezugssystem zu entfernen, das nicht existiert.

Privilegien

Sie müssen entweder Eigentümer sein oder das MANAGE ANY SPATIAL OBJECT-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „CREATE SPATIAL REFERENCE SYSTEM-Anweisung“ auf Seite 719
- „ALTER SPATIAL REFERENCE SYSTEM-Anweisung“ auf Seite 506
- „Räumliche Daten“ [[SQL Anywhere Server - Unterstützung für räumliche Daten](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird ein fiktives räumliches Bezugssystem namens Test gelöscht.

```
DROP SPATIAL REFERENCE SYSTEM Test;
```

DROP SPATIAL UNIT OF MEASURE-Anweisung

Löscht eine räumliche Maßeinheit.

Syntax

```
DROP SPATIAL UNIT OF MEASURE [ IF EXISTS ] identifier
```

Bemerkungen

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP SPATIAL UNIT OF MEASURE-Anweisung versucht, eine räumliche Maßeinheit zu entfernen, die nicht existiert.

Privilegien

Sie müssen entweder Eigentümer der räumlichen Maßeinheit sein oder das MANAGE ANY SPATIAL OBJECT-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „CREATE SPATIAL UNIT OF MEASURE-Anweisung“ auf Seite 727
- „Räumliche Daten“ [[SQL Anywhere Server - Unterstützung für räumliche Daten](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird eine fiktive räumliche Maßeinheit namens Test gelöscht.

```
DROP SPATIAL UNIT OF MEASURE Test;
```

DROP STATEMENT-Anweisung [ESQL]

Gibt Anweisungsressourcen frei.

Syntax

```
DROP STATEMENT [ owner.]statement-name
```

statement-name :

identifier

| *hostvar*

Bemerkungen

Die DROP STATEMENT-Anweisung gibt Ressourcen frei, die von der benannten vorbereiteten Anweisung verwendet werden. Diese Ressourcen werden durch eine erfolgreiche PREPARE-Anweisung belegt und normalerweise nicht freigegeben, bis die Datenbankverbindung freigegeben wird.

Um die Anweisung löschen zu können, müssen Sie zuerst die Anweisung vorbereitet haben.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „PREPARE-Anweisung [ESQL]“ auf Seite 976

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers. Im SQL/2008-Standard wird diese Funktion von der DEALLOCATE PREPARE-Anweisung bereitgestellt, die Teil der optionalen SQL-Sprachenfunktion B032 ist (Extended Dynamic SQL).

Beispiel

Nachfolgend werden Beispiele für die Verwendung der DROP STATEMENT-Anweisung gezeigt:

```
EXEC SQL DROP STATEMENT S1;
EXEC SQL DROP STATEMENT :stmt;
```

DROP STATISTICS-Anweisung

Löscht alle Spaltenstatistiken für die angegebenen Spalten.

Syntax

```
DROP STATISTICS [ ON ] [owner.]object-name [ ( column-list ) ]
```

```
object-name :
table-name
| materialized-view-name
| temp-table-name
```

Bemerkungen

Der Optimierer in SQL Anywhere verwendet Spaltenstatistiken, um die beste Strategie für das Ausführen der einzelnen Anweisungen festzulegen. SQL Anywhere stellt diese Statistiken automatisch zusammen und aktualisiert sie. Spaltenstatistiken werden in der Datenbank in der Systemtabelle ISYSCOLSTAT permanent gespeichert. Spaltenstatistiken, die während der Verarbeitung einer Anweisung erstellt werden, stehen bei der Suche nach effizienten Verfahren zur Ausführung nachfolgender Anweisungen zur Verfügung.

Es kommt gelegentlich vor, dass die Spaltenstatistiken ungenau werden oder dass relevante Statistiken nicht verfügbar sind. Dieser Fall tritt üblicherweise dann ein, wenn eine große Menge von Daten hinzugefügt, aktualisiert oder gelöscht wurde und seitdem nur wenige Abfragen ausgeführt wurden.

Mit der Anweisung DROP STATISTICS werden alle internen statistischen Daten aus der Systemtabelle ISYSCOLSTAT für die angegebenen Spalten gelöscht. Dieser drastische Schritt entzieht dem Optimierer jeden Zugriff auf wichtige statistische Informationen. Ohne diese Statistiken erzeugt der Optimierer möglicherweise unwirksame Datenzugriffspläne und verursacht damit eine schlechte Datenbank-Performance.

Die DROP STATISTICS-Anweisung erfordert eine Exklusivsperrung für die Tabelle, in der sie ausgeführt wird. Die Anweisung kann erst dann ausgeführt werden, wenn alle anderen Verbindungen, die die Tabelle referenzieren, die referenzierenden Transaktionen festgeschrieben oder zurückgesetzt haben oder alle geöffneten Cursor geschlossen haben, die die Tabelle referenzieren.

Diese Anweisung sollte nur beim Ermitteln von Problemen oder beim Neuladen von Daten in eine Datenbank verwendet werden, die sich stark von den Originaldaten unterscheiden.

Privilegien

Sie müssen entweder Eigentümer der Tabelle sein oder das `MANAGE ANY STATISTICS`-Systemprivileg oder das `DROP ANY OBJECT`-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „`CREATE STATISTICS`-Anweisung“ auf Seite 729
- „Optimiererschätzungen und -statistiken“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „`SYSCOLSTAT`-Systemansicht“ auf Seite 1441

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

DROP SUBSCRIPTION-Anweisung [SQL Remote]

Löscht eine Subskription einer Publikation für einen Benutzer.

Syntax

```
DROP SUBSCRIPTION TO publication-name [ ( subscription-value ) ]  
FOR subscriber-id, ...
```

subscription-value : *string*

subscriber-id : *string*

Parameter

publication-name Der Name der Publikation, die für den Benutzer subskribiert ist. Darin kann der Eigentümer der Publikation enthalten sein.

subscription-value Eine Zeichenfolge, die mit dem Subskriptionsausdruck der Publikation verglichen wird. Dieser Wert ist erforderlich, da ein Benutzer mehr als eine Subskription für eine Publikation haben kann.

subscriber-id Die Benutzer-ID des Subskribenten für die Publikation.

Bemerkungen

Löscht eine SQL Remote-Subskription einer Publikation für eine Benutzer-ID in der aktuellen Datenbank. Die Benutzer-ID wird keine weiteren Aktualisierungen erhalten, wenn Daten in der Publikation geändert werden.

In SQL Remote besteht zwischen Publikationen und Subskriptionen eine Zwei-Weg-Beziehung. Wenn Sie eine Subskription zu einer Publikation in einer konsolidierten Datenbank für einen entfernten Benutzer löschen, sollten Sie auch die Subskription in der entfernten Datenbank für die konsolidierte Datenbank löschen, um zu verhindern, dass Aktualisierungen in der entfernten Datenbank an die konsolidierte Datenbank gesendet werden.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 730
- „STOP SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 1078
- „SYSSUBSCRIPTION-Systemansicht“ auf Seite 1489

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung löscht eine Subskription der Benutzer-ID 'SamS' für die Publikation 'pub_contact'.

```
DROP SUBSCRIPTION TO pub_contact  
FOR Sam_Singer;
```

DROP SYNCHRONIZATION PROFILE-Anweisung [MobiLink]

Löscht ein SQL Anywhere-Synchronisationsprofil.

Syntax

DROP SYNCHRONIZATION PROFILE [IF EXISTS] *name*

Parameter

name Der Name des zu löschenden Synchronisationsprofils.

Bemerkungen

Synchronisationsprofile sind benannte Sammlungen von Synchronisationsoptionen, die zur Steuerung der Synchronisation verwendet werden können. Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP SYNCHRONIZATION PROFILE-Anweisung versucht, ein Synchronisationsprofil zu entfernen, das nicht existiert.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE SYNCHRONIZATION PROFILE-Anweisung [MobiLink]“ auf Seite 732
- „ALTER SYNCHRONIZATION PROFILE-Anweisung [MobiLink]“ auf Seite 511

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

DROP SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink]

Löscht eine Synchronisationssubskription in einer entfernten Datenbank.

Syntax

```
DROP SYNCHRONIZATION SUBSCRIPTION { subscription-name |  
TO publication-name  
[ FOR ml-username, ... ] }
```

Parameter

subscription-name Gibt den Namen der Subskription an, die gelöscht werden soll

TO-Klausel Gibt den Namen einer Publikation an.

FOR-Klausel Gibt einen oder mehrere Benutzer an.

Wenn Sie diese Klausel auslassen, werden die Standardeinstellungen für die Publikation gelöscht.

Bemerkungen

Erfordert exklusiven Zugriff auf alle in der Publikation referenzierten Tabellen.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink]“ auf Seite 512
- „CREATE SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink]“ auf Seite 733
- „SYSSYNC-Systemansicht“ auf Seite 1489
- „MobiLink-Subskriptionen löschen“ [*MobiLink - Clientadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Im folgenden Beispiel wird die Subskription sales gelöscht:

```
DROP SYNCHRONIZATION SUBSCRIPTION sales;
```

Im folgenden Beispiel wird die Subskription zwischen dem MobiLink-Benutzer SSinger und der Publikation sales_publication gelöscht:

```
DROP SYNCHRONIZATION SUBSCRIPTION  
  TO user.sales_publication  
  FOR "SSinger";
```

Das folgende Beispiel lässt die FOR-Klausel weg und löscht daher die Standardeinstellungen für die Publikation 'sales_publication':

```
DROP SYNCHRONIZATION SUBSCRIPTION  
  TO user.sales_publication;
```

DROP SYNCHRONIZATION USER-Anweisung [MobiLink]

Löscht einen oder mehrere Synchronisationsbenutzer aus einer entfernten SQL Anywhere-Datenbank.

Syntax

```
DROP SYNCHRONIZATION USER ml-username, ...
```

ml-username : *identifizier*

Bemerkungen

Löschen Sie einen oder mehrere Synchronisationsbenutzer aus einer entfernten MobiLink-Datenbank.

Sie benötigen exklusiven Zugriff auf alle Tabellen, die in vom Benutzer subskribierten Publikationen referenziert werden.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Alle dem Benutzer zugeordneten Subskriptionen werden ebenfalls gelöscht.

Siehe auch

- „ALTER SYNCHRONIZATION USER-Anweisung [MobiLink]“ auf Seite 515
- „CREATE SYNCHRONIZATION USER-Anweisung [MobiLink]“ auf Seite 735
- „SYSSYNC-Systemansicht“ auf Seite 1489

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Löschen Sie den MobiLink-Benutzer SSinger aus der Datenbank.

```
DROP SYNCHRONIZATION USER SSinger;
```

DROP TABLE-Anweisung

Entfernt eine Tabelle aus der Datenbank.

Syntax

```
DROP TABLE [ IF EXISTS ] [ owner.]table-name
```

Bemerkungen

Wenn Sie eine Tabelle entfernen, werden alle Daten in der Tabelle während des Löschvorgangs automatisch gelöscht. Alle Indizes und Schlüssel für die Tabelle werden ebenfalls gelöscht.

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP TABLE-Anweisung versucht, eine Tabelle zu entfernen, die nicht existiert.

Sie können eine DROP TABLE-Anweisung nicht ausführen, wenn die Anweisung eine Tabelle betrifft, die aktuell von einer anderen Verbindung benutzt wird. Das Ausführen einer DROP TABLE-Anweisung wird auch verhindert, wenn eine materialisierte Ansicht von der Tabelle abhängt.

Wenn Sie eine DROP TABLE-Anweisung ausführen, ändert sich der Status aller abhängigen regulären Ansichten auf INVALID. Um Ansichtsabhängigkeiten vor dem Löschen einer Tabelle zu ermitteln, verwenden Sie die Systemprozedur sa_dependent_views.

Globale temporäre Tabellen können erst gelöscht werden, wenn alle Benutzer, die die Tabelle referenzieren, ihre Verbindungen getrennt haben.

Privilegien

Sie müssen entweder Eigentümer der Tabelle sein oder das DROP ANY TABLE-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). DROP TABLE kann auch einen automatischen Checkpoint setzen. Löscht den Inhalt der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** in Interactive SQL. Beim Ausführen einer DROP TABLE-Anweisung werden alle Cursor für die aktuelle Verbindung geschlossen.

Sie können die DROP TABLE-Anweisung verwenden, um eine lokale temporäre Tabelle zu löschen.

Siehe auch

- „Löschen einer Tabelle“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „CREATE TABLE-Anweisung“ auf Seite 737
- „ALTER TABLE-Anweisung“ auf Seite 516
- „sa_dependent_views-Systemprozedur“ auf Seite 1202

Standards und Kompatibilität

- **SQL/2008** DROP TABLE ist eine Kernfunktion des SQL/2008-Standards. Die IF EXISTS-Klausel ist eine Erweiterung des Herstellers. Die Möglichkeit zum Löschen einer deklarierten lokalen temporären Tabelle mit der DROP TABLE-Anweisung ist eine Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird die fiktive Tabelle "MyTable" aus der Datenbank gelöscht.

```
DROP TABLE MyTable;
```

In diesem Beispiel wird die fiktive Tabelle "MyTable" aus der Datenbank gelöscht. Da IF EXISTS angegeben wurde, wird *kein* Fehler zurückgegeben, wenn die Tabelle nicht vorhanden ist.

```
DROP TABLE IF EXISTS MyTable;
```

DROP TEXT CONFIGURATION-Anweisung

Löscht ein Textkonfigurationsobjekt.

Syntax

```
DROP TEXT CONFIGURATION [ owner. ] text-config-name
```

Bemerkungen

Wenn versucht wird, ein Textkonfigurationsobjekt mit abhängigen Textindizes zu löschen, tritt ein Fehler auf. Sie müssen die abhängigen Textindizes löschen, bevor Sie das Textkonfigurationsobjekt löschen.

Textkonfigurationsobjekte werden in der Systemtabelle ISYSTETEXTCONFIG gespeichert.

Privilegien

Sie müssen entweder Eigentümer des Textkonfigurationsobjekts sein oder das DROP ANY TEXT CONFIGURATION-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „DROP TEXT INDEX-Anweisung“ auf Seite 834
- „Volltextsuche“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Konzepte und Referenz zu Textkonfigurationsobjekten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „SYSTETEXTCONFIG-Systemansicht“ auf Seite 1499
- „CREATE TEXT CONFIGURATION-Anweisung“ auf Seite 757
- „ALTER TEXT CONFIGURATION-Anweisung“ auf Seite 532

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Mit den folgenden Anweisungen wird das Textkonfigurationsobjekt mytextconfig erstellt und gelöscht:

```
CREATE TEXT CONFIGURATION mytextconfig FROM default_char;  
DROP TEXT CONFIGURATION mytextconfig;
```

DROP TEXT INDEX-Anweisung

Entfernt einen Textindex aus der Datenbank.

Syntax

```
DROP TEXT INDEX text-index-name  
ON [ owner.]table-name
```

Parameter

ON-Klausel Verwenden Sie diese Klausel, um die Tabelle oder materialisierte Ansicht anzugeben, für die der Textindex erstellt wurde.

Bemerkungen

Sie müssen abhängige Textindizes löschen, bevor Sie ein Textkonfigurationsobjekt löschen können.

Diese Anweisung kann nicht ausgeführt werden, wenn mit der WITH HOLD-Klausel geöffnete Cursor vorhanden sind, die entweder Anweisungs- oder Transaktions-Snapshots verwenden.

Privilegien

Wenn Sie einen Textindex für eine Tabelle löschen möchten, müssen Sie der Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- REFERENCES-Privileg für die Tabelle
- DROP ANY INDEX-Systemprivileg
- DROP ANY OBJECT-Systemprivileg

Wenn Sie einen Textindex für eine materialisierte Ansicht löschen möchten, müssen Sie Eigentümer der materialisierten Ansicht sein oder eines der folgenden Privilegien haben:

- DROP ANY INDEX-Systemprivileg
- DROP ANY OBJECT-Systemprivileg

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Volltextsuche“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Konzepte und Referenz zu Textindizes“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Snapshot-Isolation“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „SYSTEXTCONFIG-Systemansicht“ auf Seite 1499
- „CREATE TEXT INDEX-Anweisung“ auf Seite 759
- „ALTER TEXT INDEX-Anweisung“ auf Seite 536
- „DROP TEXT INDEX-Anweisung“ auf Seite 834
- „REFRESH TEXT INDEX-Anweisung“ auf Seite 990
- „TRUNCATE TEXT INDEX-Anweisung“ auf Seite 1091

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Mit den folgenden Anweisungen wird der Textindex TextIdx erstellt und gelöscht:

```
CREATE TEXT INDEX TextIdx ON GROUPO.MarketingInformation ( Description )
DROP TEXT INDEX TextIdx ON GROUPO.MarketingInformation;
```

DROP TRACE EVENT-Anweisung

Löscht ein benutzerdefiniertes Trace-Ereignis.

Syntax

```
DROP TRACE EVENT [ IF EXISTS ] trace-event-name
```

Bemerkungen

Diese Anweisung löscht nur benutzerdefinierte Trace-Ereignisse. Wenn Sie nicht möchten, dass ein Fehler zurückgegeben wird, wenn die DROP TRACE EVENT-Anweisung versucht, ein nicht vorhandenes Trace-Ereignis zu entfernen, verwenden Sie die IF EXISTS-Klausel. Wenn eine oder mehrere Ereignisprotokollierungssitzungen das Trace-Ereignis referenzieren, kann das Trace-Ereignis erst nach dem Löschen aller referenzierenden Trace-Sitzungen gelöscht werden.

Systemprivilegien

Sie müssen das MANAGE ANY TRACE SESSION-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Ereignisprotokollierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE TEMPORARY TRACE EVENT-Anweisung“ auf Seite 753
- „CREATE TEMPORARY TRACE EVENT SESSION-Anweisung“ auf Seite 755
- „ALTER TRACE EVENT SESSION-Anweisung“ auf Seite 538
- „DROP TRACE EVENT SESSION-Anweisung“ auf Seite 836
- „NOTIFY TRACE EVENT-Anweisung“ auf Seite 963
- „sp_trace_events-Systemprozedur“ auf Seite 1407
- „sp_trace_event_fields-Systemprozedur“ auf Seite 1401
- „sp_trace_events-Systemprozedur“ auf Seite 1407
- „sp_trace_event_session_events-Systemprozedur“ auf Seite 1402
- „sp_trace_event_session_targets-Systemprozedur“ auf Seite 1405
- „sp_trace_event_session_target_options-Systemprozedur“ auf Seite 1403
- „Dienstprogramm für die Verwaltung von ETD-Dateien (dbmanagetd)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Löschen Sie das Trace-Ereignis my_event:

```
DROP TRACE EVENT my_event;
```

DROP TRACE EVENT SESSION-Anweisung

Löscht eine Trace-Ereignissitzung.

Syntax

```
DROP TRACE EVENT SESSION [ IF EXISTS ] session-name UNCONDITIONALLY
```

Bemerkungen

Wenn UNCONDITIONALLY angegeben ist, wird durch das Löschen einer aktiven Sitzung die Sitzung automatisch gestoppt, bevor ihre Definition entfernt wird. Andernfalls wird eine Fehlermeldung zurückgegeben.

Systemprivilegien

Sie müssen das MANAGE ANY TRACE SESSION-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Ereignisprotokollierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE TEMPORARY TRACE EVENT-Anweisung“ auf Seite 753
- „CREATE TEMPORARY TRACE EVENT SESSION-Anweisung“ auf Seite 755
- „ALTER TRACE EVENT SESSION-Anweisung“ auf Seite 538
- „DROP TRACE EVENT SESSION-Anweisung“ auf Seite 836
- „NOTIFY TRACE EVENT-Anweisung“ auf Seite 963
- „sp_trace_events-Systemprozedur“ auf Seite 1407
- „sp_trace_event_fields-Systemprozedur“ auf Seite 1401
- „sp_trace_event_sessions-Systemprozedur“ auf Seite 1406
- „sp_trace_event_session_events-Systemprozedur“ auf Seite 1402
- „sp_trace_event_session_targets-Systemprozedur“ auf Seite 1405
- „sp_trace_event_session_target_options-Systemprozedur“ auf Seite 1403
- „Dienstprogramm für die Verwaltung von ETD-Dateien (dbmanageetd)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung löscht die Trace-Ereignissitzung my_session:

```
DROP TRACE EVENT SESSION my_session;
```

DROP TRIGGER-Anweisung

Entfernt einen Trigger aus der Datenbank.

Syntax

```
DROP TRIGGER [ IF EXISTS ] [ owner. ] [ table-name. ] trigger-name
```

Bemerkungen

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP-Anweisung versucht, ein Datenbankobjekt zu entfernen, das nicht existiert.

Privilegien

Wenn Sie einen Trigger für eine Tabelle löschen möchten, muss eine der folgenden Bedingungen erfüllt sein:

- Sie sind der Eigentümer der Tabelle.
- Sie haben das ALTER-Privileg für die Tabelle.
- Sie haben das ALTER ANY TABLE-Systemprivileg.

- Sie haben das ALTER ANY OBJECT-Systemprivileg.

Wenn Sie einen Trigger in einer Ansicht eines anderen Eigentümers löschen möchten, benötigen Sie entweder das ALTER ANY VIEW-Systemprivileg oder das ALTER ANY OBJECT-Systemprivileg.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Löscht den Inhalt der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** in Interactive SQL.

Siehe auch

- „CREATE TRIGGER-Anweisung“ auf Seite 762
- „ALTER TRIGGER-Anweisung“ auf Seite 539
- „ROLLBACK TRIGGER-Anweisung“ auf Seite 1017
- „Trigger löschen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** DROP TRIGGER umfasst einen Teil der optionalen SQL-Sprachenfunktion T211 (Basis-Triggerfähigkeit) des SQL/2008-Standards. Die IF EXISTS-Klausel ist eine Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird ein Trigger namens emp_upper_postal_code, der vor dem Aktualisieren der Tabelle "Employees" gewährleisten soll, dass die Postleitzahlen in Großbuchstaben vorliegen, erstellt und anschließend gelöscht. Wenn der Trigger nicht existiert, wird ein Fehler zurückgegeben.

```
CREATE TRIGGER emp_upper_postal_code
BEFORE UPDATE OF PostalCode
ON GROUPO.Employees
REFERENCING NEW AS new_emp
FOR EACH ROW
WHEN ( ISNUMERIC( new_emp.PostalCode ) = 0 )
BEGIN
    -- Ensure postal code is uppercase (employee might be
    -- in Canada where postal codes contain letters)
    SET new_emp.PostalCode = UPPER(new_emp.PostalCode)
END;
DROP TRIGGER MyTrigger;
```

DROP USER-Anweisung

Löscht einen Benutzer.

Syntax

DROP USER *user-name*

Parameter

user-name Der Name des Benutzers, den Sie löschen.

Privilegien

Sie müssen das `MANAGE ANY USER`-Systemprivileg haben.

Bemerkungen

Wenn ein Benutzer gelöscht wird, löscht das System auch alle Datenbankobjekte (wie etwa Tabellen oder Prozeduren), deren Eigentümer er ist, ebenso wie etwaige externe Logins für den Benutzer. Außerdem gilt: Wenn der Benutzer in der `USER`-Klausel für Dienste angegeben ist, werden die betreffenden Dienste ebenfalls gelöscht.

Der gelöschte Benutzer kann nicht mit der Datenbank verbunden sein, wenn die Anweisung ausgeführt wird.

Nebenwirkungen

Keine.

Siehe auch

- [„ALTER LOGIN POLICY-Anweisung“ auf Seite 471](#)
- [„ALTER USER-Anweisung“ auf Seite 540](#)
- [„COMMENT-Anweisung“ auf Seite 573](#)
- [„CREATE LOGIN POLICY-Anweisung“ auf Seite 647](#)
- [„CREATE USER-Anweisung“ auf Seite 770](#)
- [„DROP LOGIN POLICY-Anweisung“ auf Seite 812](#)
- [„Login-Richtlinien“ \[*SQL Anywhere Server - Datenbankadministration*\]](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird der Benutzer `SQLTester` erstellt und anschließend gelöscht:

```
CREATE USER SQLTester IDENTIFIED BY pass1234;  
DROP USER SQLTester;
```

DROP VARIABLE-Anweisung

Eliminiert eine SQL-Variable.

Syntax

```
DROP VARIABLE [ IF EXISTS ] identifier
```

Bemerkungen

Die `DROP VARIABLE`-Anweisung eliminiert eine SQL-Variable, die zuvor mit einer `CREATE VARIABLE`-Anweisung erstellt wurde. Variablen werden automatisch eliminiert, wenn die Datenbankverbindung freigegeben wird. Variablen werden oftmals für große Objekte verwendet. Durch das Eliminieren nach ihrem Gebrauch oder das Setzen auf `NULL` können deshalb beachtliche Ressourcen (hauptsächlich Speicherplatz) freigegeben werden.

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP-Anweisung versucht, ein Datenbankobjekt zu entfernen, das nicht existiert.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- [„CREATE VARIABLE-Anweisung“ auf Seite 771](#)
- [„SET-Anweisung“ auf Seite 1054](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

DROP VIEW-Anweisung

Entfernt eine Ansicht aus der Datenbank.

Syntax

DROP VIEW [IF EXISTS] [*owner.*]*view-name*

Bemerkungen

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP VIEW-Anweisung versucht, eine Ansicht zu entfernen, die nicht existiert.

Wenn Sie eine DROP VIEW-Anweisung ausführen, ändert sich der Status aller abhängigen regulären Ansichten auf INVALID. Um Ansichtsabhängigkeiten vor dem Löschen einer Ansicht zu ermitteln, verwenden Sie die Systemprozedur `sa_dependent_views`.

Wenn Sie eine DROP VIEW-Anweisung für eine Ansicht ausführen, die einen oder mehr INSTEAD OF-Trigger hat, wird ein Fehler zurückgegeben. Sie müssen den Trigger löschen, bevor die Ansicht gelöscht oder geändert werden kann.

Privilegien

Sie müssen entweder Eigentümer der Ansicht sein oder das DROP ANY VIEW-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Löscht den Inhalt der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** in Interactive SQL. Beim Ausführen einer DROP VIEW-Anweisung werden alle Cursor für die aktuelle Verbindung geschlossen.

Wird eine Ansicht gelöscht, werden alle Prozeduren und Trigger im Speicher entladen. Auf diese Weise berücksichtigen Prozeduren und Trigger, die die Ansicht referenzieren, dass die Ansicht nicht existiert.

Das Entladen und Laden von Prozeduren und Triggern kann sich auf die Performance auswirken, wenn Sie Ansichten häufig löschen und erstellen.

Siehe auch

- „CREATE VIEW-Anweisung“ auf Seite 773
- „ALTER VIEW-Anweisung“ auf Seite 543
- „sa_dependent_views-Systemprozedur“ auf Seite 1202

Standards und Kompatibilität

- **SQL/2008** DROP VIEW ist eine Kernfunktion des SQL/2008-Standards. Die IF EXISTS-Klausel ist eine Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird eine Ansicht namens MyView erstellt und anschließend gelöscht. Sie benötigen das SELECT-Privileg für die Tabelle "Employees", um die CREATE VIEW-Anweisung im Beispiel ausführen zu können.

```
CREATE VIEW MyView
AS SELECT * FROM GROUP0.Employees;
DROP VIEW MyView;
```

EXCEPT-Anweisung

Gibt die Mengendifferenz zwischen zwei Abfrageblöcken zurück.

Syntax

```
[ WITH temporary-views ] main-query-block
EXCEPT [ ALL | DISTINCT ] except-query-block
[ ORDER BY [ integer ] select-list-expression-name ] [ ASC | DESC ], ... ]
[ FOR XML xml-mode ]
[ OPTION( query-hint, ... ) ]
```

query-hint :

```
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| option-name = option-value
```

main-query-block: Abfrageblock. Siehe „Allgemeine Elemente der SQL-Syntax“ auf Seite 445.

except-query-block: Abfrageblock. Siehe „Allgemeine Elemente der SQL-Syntax“ auf Seite 445.

option-name : *identifier*

option-value :

```
hostvar (Bezeichner zulässig)
| string
| identifier
| number
```

Parameter

main-query-block Ein Abfrageblock mit einer SELECT-Anweisung oder einem Abfrageausdruck (möglicherweise verschachtelt).

except-query-block Ein Abfrageblock mit einer SELECT-Anweisung oder einem Abfrageausdruck (möglicherweise verschachtelt).

FOR XML-Klausel Hinweise zur Verwendung der FOR XML-Klausel finden Sie unter „[SELECT-Anweisung](#)“ auf Seite 1020.

OPTION-Klausel Verwenden Sie diese Klausel, um Hints für das Ausführen der Anweisung anzugeben. Die folgenden Hints werden unterstützt:

- MATERIALIZED VIEW OPTIMIZATION *option-value*
- FORCE OPTIMIZATION
- *option-name* = *option-value*. Die Spezifikation `OPTION(isolation_level = ...)` im Abfragetext hebt alle anderen Festlegungen der Isolationsstufe für eine Abfrage auf.

Bemerkungen

Die EXCEPT-Anweisung gibt alle Zeilen im Haupt-Abfrageblock zurück, außer denjenigen, die auch im EXCEPT-Abfrageblock erscheinen. Geben Sie EXCEPT oder EXCEPT DISTINCT an, wenn Duplikate aus dem Haupt-Abfrageblock nicht als Duplikate in den Ergebnissen erscheinen sollen. Sonst geben Sie EXCEPT ALL an. Abfrageblöcke können verschachtelt werden.

Die Verwendung von EXCEPT allein ist gleichwertig mit EXCEPT DISTINCT.

Der *main-query-block* und der *except-query-block* müssen UNION-kompatibel sein. Sie müssen dieselbe Anzahl von Elementen in ihren jeweiligen Auswahllisten aufweisen und die Art der einzelnen Ausdrücke sollte vergleichbar sein. Wenn übereinstimmende Elemente in zwei Auswahllisten verschiedene Datentypen umfassen, wählt SQL Anywhere einen Datentyp für die entsprechende Spalte im Ergebnis aus und konvertiert automatisch die Spalten in jedem *query-block*.

EXCEPT ALL implementiert eine Multimengendifferenz statt einer Mengendifferenz. Wenn beispielsweise der *main-query-block* 5 (doppelte) Zeilen mit bestimmten Werten enthält und der *except-query-block* 2 doppelte Zeilen mit identischen Werten, gibt EXCEPT ALL 3 Zeilen zurück.

Die Ergebnisse von EXCEPT sind dieselben wie die Ergebnisse von EXCEPT ALL, wenn *main-query-block* keine doppelten Zeilen enthält.

Die angezeigten Spaltennamen sind dieselben wie beim ersten *query-block* und mithilfe dieser Namen wird bestimmt, welche Ausdrucknamen mit der ORDER BY-Klausel abgeglichen werden. Eine andere Möglichkeit zum Anpassen von Spaltennamen in der Ergebnismenge ist die Verwendung eines allgemeinen Tabellenausdrucks (der WITH-Klausel).

Privilegien

Sie müssen Eigentümer der im *query-block* referenzierten Tabellen sein oder das SELECT ANY TABLE-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- [„INTERSECT-Anweisung“ auf Seite 927](#)
- [„UNION-Anweisung“ auf Seite 1096](#)
- [„SELECT-Anweisung“ auf Seite 1020](#)
- [OPTION-Klausel, SELECT-Anweisung auf Seite 1027](#)

Standards und Kompatibilität

- **SQL/2008** EXCEPT DISTINCT ist eine Kernfunktion des SQL/2008-Standards. EXCEPT ALL umfasst die optionale SQL-Sprachenfunktion F304. Das explizite Angeben des DISTINCT-Schlüsselworts mit EXCEPT ist die optionale SQL-Sprachenfunktion T551 des SQL/2008-Standards. Das Angeben einer ORDER BY-Klausel mit EXCEPT ist SQL-Sprachenfunktion F850. Ein *query-block* mit einer ORDER BY-Klausel entspricht SQL/2008-Funktion F851. Ein Abfrageblock, der eine Zeilenbegrenzungsklausel (SELECT TOP oder LIMIT) enthält, umfasst je nach Kontext die optionale SQL-Sprachenfunktion F857 oder F858. Die Klauseln FOR XML und OPTION sind Erweiterungen des Herstellers.
- **Transact-SQL** EXCEPT wird von Adaptive Server Enterprise nicht unterstützt. Sowohl EXCEPT ALL als auch EXCEPT DISTINCT können jedoch in dem von SQL Anywhere unterstützten Transact-SQL-Dialekt verwendet werden.

Beispiel

Ein Beispiel für die Verwendung von EXCEPT finden Sie unter [„Mengenoperatoren und NULL“ \[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#).

EXECUTE IMMEDIATE-Anweisung [SP]

Ermöglicht das Ausführen von dynamisch aufgebauten Anweisungen innerhalb einer Prozedur.

Syntax 1

EXECUTE IMMEDIATE [*execute-option*] *string-expression*

execute-option :

WITH QUOTES [**ON** | **OFF**]
 | **WITH ESCAPES** { **ON** | **OFF** }
 | **WITH BATCH** { **ON** | **OFF** }
 | **WITH RESULT SET** { **ON** | **OFF** }

Syntax 2 - Transact-SQL

EXECUTE (*string-expression*)

Parameter

WITH QUOTES-Klausel Wenn Sie WITH QUOTES oder WITH QUOTES ON angeben, werden Anführungszeichen im Zeichenfolgenausdruck als Begrenzungszeichen eines Bezeichners interpretiert. Wenn Sie nicht WITH QUOTES oder WITH QUOTES OFF angeben, ist die Behandlung der

Anführungszeichen in Zeichenfolgenausdrücken abhängig von der aktuellen Einstellung der Option "quoted_identifier".

WITH QUOTES ist hilfreich, wenn ein Objektname, der an die gespeicherte Prozedur übergeben wird, verwendet wird, um die auszuführende Anweisung aufzubauen, aber der Name Anführungszeichen erfordern könnte und die Prozedur aufgerufen werden könnte, wenn quoted_identifier auf OFF gesetzt ist.

WITH ESCAPES-Klausel WITH ESCAPES OFF bewirkt, dass Escapesequenzen (wie \n, \x oder \\) im Zeichenfolgenausdruck ignoriert werden. Beispiel: Zwei aufeinanderfolgende Backslashes bleiben zwei Backslashes und werden nicht in einen einzelnen Backslash konvertiert. Die Standardeinstellung entspricht WITH ESCAPES ON.

Ein Einsatzbereich von WITH ESCAPES OFF dient der einfacheren Ausführung von dynamisch aufgebauten Anweisungen, die Dateinamen mit Backslashes referenzieren.

Unter bestimmten Umständen werden Escapesequenzen im *string-expression* umgewandelt, bevor die EXECUTE IMMEDIATE-Anweisung ausgeführt wird. Beispiel: Zusammengesetzte Anweisungen werden syntaktisch analysiert, bevor sie ausgeführt werden, und Escapesequenzen werden während dieser syntaktischen Analyse unabhängig von der WITH ESCAPES-Einstellung umgewandelt. Unter diesen Umständen verhindert WITH ESCAPES OFF weitere Umwandlungen. Beispiel:

```
BEGIN
  DECLARE String1 LONG VARCHAR;
  DECLARE String2 LONG VARCHAR;
  EXECUTE IMMEDIATE
    'SET String1 = 'One backslash: \\\\' ''';
    EXECUTE IMMEDIATE WITH ESCAPES OFF
    'SET String2 = 'Two backslashes: \\\\' ''';
  SELECT String1, String2
END
```

WITH BATCH-Klausel Mit der WITH BATCH-Klausel können Sie die Ausführung von Batches in EXECUTE IMMEDIATE-Anweisungen steuern. Die Einstellung WITH BATCH OFF bietet Schutz vor unbeabsichtigter Einschleusung von SQL-Befehlen, wenn die Prozedur ausgeführt wird.

WITH BATCH ON ist die Standardeinstellung, außer für Prozeduren, deren Eigentümer dbo ist.

Wenn WITH BATCH OFF verwendet wird, muss die durch *string-expression* angegebene Anweisung eine einzige Anweisung sein.

WITH RESULT SET-Klausel Die WITH RESULT SET-Klausel ermöglicht dem Server die korrekte Definition der Prozedur, die sie enthält. Das Angeben von WITH RESULT SET ON oder WITH RESULT SET OFF wirkt sich sowohl darauf aus, was passiert, wenn die Prozedur erstellt wird, als auch darauf, was passiert, wenn die Prozedur ausgeführt wird. Die Standardoption ist WITH RESULT SET OFF.

Die EXECUTE IMMEDIATE-Anweisung gibt eine Ergebnismenge zurück, wenn Sie WITH RESULT SET ON angeben. Durch diese Klausel wird die Rückgabe einer Ergebnismenge durch die enthaltene Prozedur gekennzeichnet.

Bemerkungen

Die EXECUTE IMMEDIATE-Anweisung erweitert den Bereich der Anweisungen, die innerhalb einer Prozedur oder eines Triggers ausgeführt werden können. Mit ihrer Hilfe können dynamisch vorbereitete

Anweisungen ausgeführt werden, wie zum Beispiel Anweisungen, die unter Verwendung der an eine Prozedur übergebenen Parameter aufgebaut werden.

Hinweise zur unterstützten Syntax für benannte Parameter finden Sie unter „[Benannte Parameter](#)“ auf Seite 93.

Literalzeichenfolgen in der Anweisung müssen in Apostrophe gesetzt werden. Zeichenfolgenliterals können nicht über mehrere Zeilen laufen.

Nur globale Variablen können in einer Anweisung referenziert werden, die von EXECUTE IMMEDIATE ausgeführt wird.

Nur Syntax 2 kann innerhalb von gespeicherten Transact-SQL-Prozeduren und -Triggern verwendet werden.

Mit EXECUTE IMMEDIATE ausgeführte Anweisungen haben ihre Pläne nicht zwischengespeichert.

Privilegien

Keine.

Nebenwirkungen

Keine. Wenn es sich jedoch bei der Anweisung um eine Datendefinitionsanweisung mit Autocommit als Nebenwirkung handelt, wird das Autocommit ausgeführt.

Siehe auch

- „Benannte Parameter“ auf Seite 93
- „quoted_identifier-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „EXECUTE IMMEDIATE in Prozeduren, Triggern, benutzerdefinierten Funktionen und Batches“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „CREATE PROCEDURE-Anweisung“ auf Seite 681
- „CREATE PROCEDURE-Anweisung [externer Aufruf]“ auf Seite 661
- „CREATE PROCEDURE-Anweisung [Webdienste]“ auf Seite 670
- „BEGIN-Anweisung“ auf Seite 557
- „EXECUTE-Anweisung [ESQL]“ auf Seite 846
- „EXECUTE-Anweisung [T-SQL]“ auf Seite 848

Standards und Kompatibilität

- **SQL/2008** EXECUTE IMMEDIATE ist die optionale SQL-Sprachenfunktion B031 (Basic Dynamic SQL) des SQL/2008-Standards. Die *execute-option*-Syntax ist eine Erweiterung des Herstellers. Der SQL/2008-Standard verbietet die Verwendung von EXECUTE IMMEDIATE-Anweisungen, die eine Ergebnismenge zurückgeben.
- **Transact-SQL** Syntax 2 ist die Syntax des Transact-SQL-Dialekts für EXECUTE IMMEDIATE. Die *execute-option*-Syntax wird von Adaptive Server Enterprise nicht unterstützt.

Beispiele

Die folgende Prozedur erstellt eine Tabelle, wobei der Tabellename als ein Parameter an die Prozedur übergeben wird.

```
CREATE PROCEDURE CreateTableProc(  
    IN tablename char(30)  
)  
BEGIN  
    EXECUTE IMMEDIATE  
    'CREATE TABLE ' || tablename ||  
    ' ( column1 INT PRIMARY KEY)'  
END;
```

So rufen Sie die Prozedur auf und erstellen die Tabelle 'mytable':

```
CALL CreateTableProc( 'mytable' );
```

EXECUTE-Anweisung [ESQL]

Führt eine vorbereitete SQL-Anweisung aus.

Syntax 1

```
EXECUTE statement  
[ USING { hostvar-list | [ SQL ] DESCRIPTOR sqlda-name } ]  
[ INTO { into-hostvar-list | [ SQL ] DESCRIPTOR into-sqlda-name } ]  
[ ARRAY :row-count ]
```

row-count : *integer* | *hostvar*

statement : *identifier* | *hostvar* | *string*

sqlda-name : *identifier*

into-sqlda-name : *identifier*

Syntax 2

```
EXECUTE IMMEDIATE statement
```

statement : *string* | *hostvar*

Parameter

USING-Klausel Die Ausgabe aus einer SELECT- oder einer CALL-Anweisung wird entweder in den Variablen in der Variablenliste oder in den vom benannten SQLDA beschriebenen Programm Datenbereichen abgelegt. Die Ausgabe (Auswahlliste oder Parameter) entspricht eins zu eins der Hostvariablenliste oder dem SQLDA-Deskriptor-Feld.

INTO-Klausel Wenn EXECUTE INTO mit einer INSERT-Anweisung verwendet wird, ist die Folge, dass die eingefügte Zeile im zweiten Deskriptor zurückgegeben wird. Wenn beispielsweise automatisch inkrementierende Primärschlüssel oder BEFORE INSERT-Trigger verwendet werden, die Primärschlüsselwerte erzeugen, stellt die EXECUTE-Anweisung ein Verfahren zur Verfügung, um die Zeile sofort wieder abzurufen und den dieser Zeile zuvor zugeordneten Primärschlüsselwert zu bestimmen. Das gleiche Resultat kann mit @@identity in Verbindung mit automatisch inkrementierenden Schlüsseln erreicht werden.

ARRAY-Klausel Die optionale ARRAY-Klausel kann mit den vorbereiteten INSERT-Anweisungen verwendet werden, um weite Einfügungen von mehr als einer Zeile gleichzeitig zu ermöglichen. Dadurch

lässt sich die Performance verbessern. Der Ganzzahlwert stellt die Anzahl der einzufügenden Zeilen dar. Der SQLDA muss eine Variable für jeden Eintrag (Anzahl der Zeilen * Anzahl der Spalten) enthalten. Die erste Zeile wird in die SQLDA-Variablen von 0 bis (Spalten pro Zeile)-1 geschrieben, usw.

Bemerkungen

Die EXECUTE-Anweisung kann für jede SQL-Anweisung verwendet werden, die vorbereitet werden kann. Cursor werden für SELECT-Anweisungen oder CALL-Anweisungen verwendet, die viele Zeilen aus der Datenbank zurückgeben.

Nach der erfolgreichen Ausführung einer INSERT-, UPDATE- oder DELETE-Anweisung wird das Feld *sqlerrd[2]* des SQLCA (SQLCOUNT) mit der Anzahl der von diesem Vorgang betroffenen Zeilen ausgefüllt.

- **Syntax 1** führt die benannte dynamische Anweisung aus, die zuvor vorbereitet wurde. Wenn die dynamische Anweisung Platzhalter für Hostvariablen enthält, die Informationen für die Anforderung (Bindevariablen) liefern, muss entweder der *sqlda-name* eine C-Variable angeben, die ein Zeiger auf einen SQLDA ist, der genug Deskriptoren für alle in der Anweisung enthaltenen Bindevariablen enthält, oder die Bindevariablen müssen in der *hostvar-list*-Liste bereitgestellt werden.
- **Syntax 2** Eine Kurzform zu PREPARE und EXECUTE einer Anweisung, die weder Bindevariable noch Ausgabe enthält. Die in der Zeichenfolge oder der Hostvariablen enthaltene Zeichenfolge wird unverzüglich ausgeführt und danach gelöscht.

Privilegien

Welche Privilegien erforderlich sind, hängt davon ab, welche Anweisung ausgeführt wird.

Nebenwirkungen

Keine.

Siehe auch

- „Benannte Parameter“ auf Seite 93
- „EXECUTE IMMEDIATE-Anweisung [SP]“ auf Seite 843
- „PREPARE-Anweisung [ESQL]“ auf Seite 976
- „DECLARE CURSOR-Anweisung [ESQL] [SP]“ auf Seite 778
- „Cursor in Embedded SQL“ [*SQL Anywhere Server - Programmierung*]

Standards und Kompatibilität

- **SQL/2008** Die EXECUTE-Anweisung umfasst Teile der optionalen SQL-Sprachenfunktion B031 (Basic Dynamic SQL) des SQL/2008-Standards. Die INTO-Klausel ist Teil der optionalen Sprachenfunktion B032 (Extended Dynamic SQL) des SQL/2008-Standards. Die ARRAY-Klausel ist eine Erweiterung des Herstellers.

Die bei Embedded SQL unterstützte EXECUTE IMMEDIATE-Anweisung ist auch Teil der optionalen SQL-Sprachenfunktion B031.

Beispiel

In diesem Beispiel wird eine DELETE-Anweisung ausgeführt.

```
EXEC SQL EXECUTE IMMEDIATE  
'DELETE FROM Employees WHERE EmployeeID = 105';
```

In diesem Beispiel wird eine vorbereitete DELETE-Anweisung ausgeführt.

```
EXEC SQL PREPARE del_stmt FROM  
'DELETE FROM Employees WHERE EmployeeID = :a';  
EXEC SQL EXECUTE del_stmt USING :employee_number;
```

In diesem Beispiel wird eine vorbereitete Abfrage ausgeführt.

```
EXEC SQL PREPARE sell FROM  
'SELECT Surname FROM Employees WHERE EmployeeID = :a';  
EXEC SQL EXECUTE sell USING :employee_number INTO :surname;
```

EXECUTE-Anweisung [T-SQL]

Ruft eine Prozedur auf (Adaptive Server Enterprise-kompatible Alternative zur CALL-Anweisung) und führt eine vorbereitete SQL-Anweisung in Transact-SQL aus.

Syntax 1 - Gespeicherte Prozedur aufrufen

```
[ EXECUTE | EXEC ] [ @return_status = ] [creator.]procedure_name [ argument, ... ]
```

argument :

```
[ @parameter-name = ] expression  
| [ @parameter-name = ] @variable [ output ]
```

Syntax 2 - Dynamische Anweisungen innerhalb von gespeicherten Prozeduren und Triggern in T-SQL ausführen

```
EXECUTE ( string-expression )
```

Bemerkungen

Syntax 1 wird implementiert, um die Transact-SQL Kompatibilität zu gewährleisten. Mit EXECUTE wird eine gespeicherte Prozedur ausgeführt. Optional werden dabei Prozedurparameter geliefert und Ausgabewerte sowie Rückgabestatusinformationen abgerufen. Verwenden Sie in Watcom-SQL die Anweisungen CALL oder EXECUTE IMMEDIATE.

Mit Syntax 2 können Sie dynamische Anweisungen innerhalb gespeicherter Transact-SQL-Prozeduren und -Trigger ausführen. Die EXECUTE-Anweisung erweitert den Bereich der Anweisungen, die innerhalb einer Prozedur oder eines Triggers ausgeführt werden können. Mit ihrer Hilfe können dynamisch vorbereitete Anweisungen ausgeführt werden, wie zum Beispiel Anweisungen, die unter Verwendung der an eine Prozedur übergebenen Parameter aufgebaut werden. Literalzeichenfolgen in der Anweisung müssen in Apostrophe gesetzt werden und die Anweisung muss in einer einzigen Zeile stehen. Syntax 2 der EXECUTE-Anweisung ist für Transact-SQL-Kompatibilität implementiert, kann aber in Batches und Prozeduren sowohl in Transact-SQL als auch in Watcom-SQL verwendet werden.

Die EXECUTE-Anweisung in Transact-SQL kann nicht angeben, dass eine Ergebnismenge erwartet wird. Um anzugeben, dass eine Transact-SQL-Prozedur eine Ergebnismenge zurückgibt, fügen Sie z.B. folgenden Zusatz ein:

```
IF 1 = 0  
    SELECT 1 AS a
```

Sie können Anweisungen auch innerhalb von gespeicherten Transact-SQL-Prozeduren und -Triggern ausführen.

Hinweise zur unterstützten Syntax für benannte Parameter finden Sie unter „[Benannte Parameter](#)“ auf Seite 93.

Privilegien

Beim Aufrufen einer Prozedur müssen Sie Eigentümer der Prozedur sein oder das EXECUTE ANY PROCEDURE-Systemprivileg haben.

Wenn Sie dynamische Anweisungen innerhalb von gespeicherten Prozeduren und Triggern in T-SQL ausführen, hängen die erforderlichen Privilegien davon ab, welche Anweisung ausgeführt wird.

Nebenwirkungen

Keine.

Siehe auch

- „[Benannte Parameter](#)“ auf Seite 93
- „[CALL-Anweisung](#)“ auf Seite 564
- „[EXECUTE-Anweisung \[ESQL\]](#)“ auf Seite 846
- „[EXECUTE IMMEDIATE-Anweisung \[SP\]](#)“ auf Seite 843

Standards und Kompatibilität

- **SQL/2008** Syntax 1 ist eine Erweiterung des Herstellers. Syntax 2 bietet Funktionen, die der EXECUTE IMMEDIATE-Anweisung im SQL/2008-Standard entsprechen, nämlich der optionalen SQL-Sprachenfunktion B031 (Basic Dynamic SQL). Die Syntax von Syntax 2 unterscheidet sich jedoch von der des SQL/2008-Standards.

Beispiel

Mit der folgenden Prozedur wird die Syntax 1 veranschaulicht.

```
CREATE PROCEDURE p1( @var INTEGER = 54 )
AS
PRINT 'on input @var = %1!', @var
DECLARE @intvar integer
SELECT @intvar=123
SELECT @var=@intvar
PRINT 'on exit @var = %1!', @var;
```

Die folgende Anweisung führt die Prozedur aus, indem sie den Eingabewert 23 für den Parameter liefert. Wenn Sie Ihre Verbindung aus einer Open Client- oder JDBC-Anwendung hergestellt haben, werden die PRINT-Meldungen im Clientfenster angezeigt. Wenn die Verbindung aus einer ODBC- oder Embedded SQL-Anwendung hergestellt wurde, werden die Meldungen im Meldungsfenster des Datenbankservers angezeigt.

```
EXECUTE p1 23;
```

Im Folgenden wird eine alternative Möglichkeit für die Ausführung der Prozedur gezeigt. Diese ist vor allem nützlich, wenn mehrere Parameter vorhanden sind.

```
EXECUTE p1 @var = 23;
```

Die folgende Anweisung führt die Prozedur aus, indem sie den Standardwert für den Parameter verwendet.

```
EXECUTE p1;
```

Die folgende Anweisung führt die Prozedur aus und speichert den Rückgabewert in einer Variablen zur Überprüfung des Rückgabestatus.

```
EXECUTE @status = p1 23;
```

EXIT-Anweisung [Interactive SQL]

Beendet Interactive SQL.

Syntax

```
{ EXIT | QUIT | BYE } [ return-code ]
```

return-code : *number* | *connection-variable*

Bemerkungen

Diese Anweisung schließt das Interactive SQL-Fenster, wenn Sie Interactive SQL als Fensterprogramm ausführen, oder sie beendet Interactive SQL vollständig, wenn es im Befehlszeilenmodus (Batchmodus) ausgeführt wird. In beiden Fällen wird auch die Datenbankverbindung getrennt. Bevor es die Datenbankverbindung beendet, führt Interactive SQL automatisch eine COMMIT-Anweisung aus, falls die Option `commit_on_exit` auf ON gesetzt ist. Wenn diese Option auf OFF gesetzt ist, führt Interactive SQL ein implizites ROLLBACK durch. Standardmäßig ist die Option `commit_on_exit` auf ON gesetzt.

Der optionale Rückgabecode kann in Batchdateien aktiviert werden, um die erfolgreiche oder fehlgeschlagene Ausführung der Anweisungen in einer Interactive SQL-Befehlsdatei zu ermitteln. Standard-Rückgabecode ist "0".

Privilegien

Keine.

Nebenwirkungen

Diese Anweisung nimmt automatisch eine Festschreibung vor, wenn die Option `commit_on_exit` auf ON (Standardwert) eingestellt ist. Andernfalls wird eine implizite Zurücksetzung vorgenommen.

Unter Windows ist der optionale Rückgabecode verfügbar als `ERRORLEVEL`.

Siehe auch

- „SET OPTION-Anweisung“ auf Seite 1039
- „Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Im folgenden Beispiel setzen Sie den Interactive SQL-Rückgabewert auf "1", wenn Zeilen in Tabelle T vorhanden sind, oder auf "0", wenn T keine Zeilen enthält.

```
CREATE VARIABLE rowCount INT;
CREATE VARIABLE retcode INT;
SELECT COUNT(*) INTO rowCount FROM GROUPO.Products;
IF( rowCount > 0 ) THEN
    SET retcode = 1;
ELSE
    SET retcode = 0;
END IF;
EXIT retcode;
```

Hinweis

Sie dürfen die folgende Anweisung nicht verwenden, weil EXIT eine Interactive SQL-Anweisung (und keine SQL-Anweisung) ist und Sie keine Interactive SQL-Anweisung in andere SQL-Blockanweisungen aufnehmen dürfen.

```
CREATE VARIABLE rowCount INT;
SELECT COUNT(*) INTO rowCount FROM T;
IF( rowCount > 0 ) THEN
    EXIT 1      // <-- not allowed
ELSE
    EXIT 0      // <-- not allowed
END IF;
```

EXPLAIN-Anweisung [ESQL]

Ruft eine Textspezifikation der Optimierungsstrategie ab, die für einen bestimmten Cursor verwendet wird.

Syntax

```
EXPLAIN PLAN FOR CURSOR cursor-name
{ INTO hostvar | USING DESCRIPTOR sqlda-name }
```

cursor-name : *identifier* | *hostvar*

sqlda-name : *Bezeichner*

Bemerkungen

Die EXPLAIN-Anweisung ruft eine Textrepräsentation der Optimierungsstrategie für den benannten Cursor ab. Der Cursor muss zuvor deklariert und geöffnet worden sein.

Die *hostvar* oder die *sqlda-name*-Variable muss eine Zeichenfolge sein. Die Optimierungszeichenfolge gibt an, in welcher Reihenfolge die Tabellen durchsucht werden und auch, wenn vorhanden, welche Indizes für die Durchsuchungen verwendet werden.

Diese Zeichenfolge kann lang sein, abhängig von der Abfrage. Sie hat folgendes Format:

```
table (index), table (index), ...
```

Wenn einer Tabelle ein Korrelationsname gegeben wurde, erscheint dieser Korrelationsname an Stelle des Tabellennamens. Die Namen in der Liste erscheinen in der gleichen Reihenfolge, in welcher der Datenbankserver auf sie zugreift. Nach jeder Tabelle steht ein in Klammern gesetzter Indexname. Hierbei handelt es sich um den Index, der für den Zugriff auf die Tabelle verwendet wird. Wenn kein Index verwendet wird (die Tabelle wird sequenziell durchsucht), werden die Buchstaben "seq" an Stelle des Indexnamens angezeigt. Wenn eine bestimmte SQL SELECT-Anweisung Unterabfragen erfordert, wird jede Optimierungszeichenfolge der Unterabfrage durch einen Doppelpunkt (:) getrennt. Diese Unterabfrageabschnitte erscheinen in der Reihenfolge, in welcher der Datenbankserver die Abfragen ausführt.

Nach der erfolgreichen Ausführung der EXPLAIN-Anweisung wird in das Feld sqlerrd des SQLCA (SQLIOESTIMATE) die geschätzte Anzahl der Eingabe/Ausgabevorgänge eingetragen, die notwendig sind, um alle Zeilen der Abfrage abzurufen.

Eine Erläuterung mit zahlreichen Beispielen für Optimierungszeichenfolgen finden Sie unter „[Tools für Performanceüberwachung und Diagnose](#)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

Sie können diese Anweisung nur in Cursor ausführen, die Sie geöffnet haben.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „[DECLARE CURSOR-Anweisung \[ESQL\] \[SP\]](#)“ auf Seite 778
- „[PREPARE-Anweisung \[ESQL\]](#)“ auf Seite 976
- „[FETCH-Anweisung \[ESQL\] \[SP\]](#)“ auf Seite 853
- „[CLOSE-Anweisung \[ESQL\] \[SP\]](#)“ auf Seite 571
- „[OPEN-Anweisung \[ESQL\] \[SP\]](#)“ auf Seite 964
- „[Cursor in Embedded SQL](#)“ [[SQL Anywhere Server - Programmierung](#)]
- „[SQL-Kommunikationsbereich \(SQLCA\)](#)“ [[SQL Anywhere Server - Programmierung](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel veranschaulicht die Verwendung von EXPLAIN:

```
EXEC SQL BEGIN DECLARE SECTION;
char plan[300];
EXEC SQL END DECLARE SECTION;
EXEC SQL DECLARE employee_cursor CURSOR FOR
    SELECT EmployeeID, Surname
    FROM Employees
    WHERE Surname like :pattern;
EXEC SQL OPEN employee_cursor;
EXEC SQL EXPLAIN PLAN FOR CURSOR employee_cursor INTO :plan;
printf( "Optimization Strategy: '%s'.n", plan );
```

Die Planvariable enthält die folgende Zeichenfolge:

```
'Employees <seq>'
```

FETCH-Anweisung [ESQL] [SP]

Positioniert einen Cursor in einer bestimmten Zeile (neu) und kopiert dann Ausdruckswerte aus dieser Zeile in Variablen, auf die über die gespeicherte Prozedur oder die Anwendung zugegriffen werden kann.

Syntax 1 [SP]

```
FETCH [ cursor-position ] cursor-name
INTO variable-list [ FOR UPDATE ]
```

Syntax 2 [ESQL]

```
FETCH [ cursor-position ] cursor-name
[ INTO { hostvar-list } | USING [ SQL ] DESCRIPTOR sqlda-name ]
[ PURGE ]
[ BLOCK n ]
[ FOR UPDATE ]
[ ARRAY fetch-count ]
```

cursor-position :

```
NEXT | PRIOR | FIRST | LAST
| { ABSOLUTE | RELATIVE } row-count
```

row-count : *number* | *hostvar*

cursor-name : *identifier* | *hostvar*

hostvar-list : Kann Indikatorvariablen enthalten

variable-list : Variable von gespeicherten Prozeduren

sqlda-name : *identifier*

fetch-count : *integer* | *hostvar*

Parameter

INTO-Klausel Die INTO-Klausel ist optional. Wenn sie nicht angegeben ist, positioniert die FETCH-Anweisung nur den Cursor. Die *hostvar-list* kann nur in Embedded SQL verwendet werden.

Cursorposition Mithilfe eines optionalen Positionsparameters kann der Cursor bewegt werden, bevor die Zeile abgerufen wird. Wenn nichts angegeben ist, wird NEXT angenommen. Wenn der Abruf einen Positionierungsparameter enthält und sich die Position außerhalb der zulässigen Cursorpositionen befindet, wird die Warnung SQLE_NOTFOUND ausgegeben, und das SQLCOUNT-Feld zeigt den Ausgangspunkt von einer gültigen Position an.

Die OPEN-Anweisung positioniert den Cursor zu Beginn vor der ersten Zeile.

NEXT-Klausel NEXT ist die Standardpositionierung und bewirkt, dass der Cursor um eine Zeile vorrückt, bevor die Zeile abgerufen wird.

PRIOR-Klausel Bewirkt, dass der Cursor vor dem Abrufen um eine Zeile zurückrückt

RELATIVE-Klausel RELATIVE-Positionierung wird verwendet, um den Cursor vor dem Abruf um eine angegebene Anzahl von Zeilen in die eine oder andere Richtung zu bewegen. Eine positive Zahl zeigt eine Vorwärtsbewegung, eine negative Zahl eine Rückwärtsbewegung an. Daher ist NEXT gleichwertig mit RELATIVE 1 und PRIOR gleichwertig mit RELATIVE -1. RELATIVE 0 fragt dieselbe Zeile ab wie in der letzten FETCH-Anweisung für diesen Cursor.

ABSOLUTE-Klausel Der Positionierungsparameter ABSOLUTE wird verwendet, um zu einer bestimmten Zeile zu gehen. Eine 0 gibt die Position vor der ersten Zeile an.

Eine Eins (1) gibt die erste Zeile an, usw. Negative Zahlen werden benutzt, um eine absolute Position ab dem Ende des Cursors anzugeben. Eine negative Eins (-1) gibt die letzte Zeile des Cursors an.

FIRST-Klausel Eine Kurzform für ABSOLUTE 1

LAST-Klausel Eine Kurzform für ABSOLUTE -1

Hinweis

Einfügungen und einige Aktualisierungen zu DYNAMIC SCROLL-Cursoren können Probleme bei der Cursorpositionierung verursachen. Der Datenbankserver platziert eingefügte Zeilen an unvorhersehbaren Positionen innerhalb eines Cursors, falls die SELECT-Anweisung keine ORDER BY-Klausel hat. In einigen Fällen erscheint die eingefügte Zeile erst, wenn der Cursor geschlossen und wieder geöffnet wurde.

Dieses Verhalten tritt auf, wenn zum Öffnen eines Cursors eine temporäre Tabelle erstellt werden musste.

Die UPDATE-Anweisung kann bewirken, dass sich eine Zeile im Cursor verschiebt. Das geschieht, wenn der Cursor eine ORDER BY-Klausel besitzt, die einen vorhandenen Index verwendet (es wird keine temporäre Tabelle erstellt).

BLOCK-Klausel Eine Clientanwendung kann mehr als eine Zeile gleichzeitig abrufen. Dies wird Blockabruf, Prefetch-Vorgang oder Mehrzeilenabruf genannt. Der erste Abruf bewirkt, dass mehrere Zeilen vom Datenbankserver zurückgesendet werden. Der Client puffert diese Zeilen, und nachfolgende Abrufe werden ohne erneute Anforderung an den Datenbankserver aus diesen Puffern abgerufen.

Die BLOCK-Klausel kann nur in Embedded SQL verwendet werden. Sie gibt Client und Server einen Hinweis, wie viele Zeilen von der Anwendung abgerufen werden können. Der Spezialwert 0 bedeutet, dass die Anforderung an den Datenbankserver geschickt und eine einzelne Zeile zurückgegeben wird (keine Zeilenblockierung). Die BLOCK-Klausel verringert die Anzahl der Zeilen, die ins nächste Prefetch an den BLOCK-Wert aufgenommen werden. Um die Anzahl der Prefetch-Zeilen zu erhöhen, verwenden Sie den PrefetchRows-Verbindungsparameter.

Wenn Sie keine BLOCK-Klausel angeben, wird der für OPEN angegebene Wert verwendet.

FETCH RELATIVE 0 ruft die Zeile immer erneut ab.

Wenn Prefetch für den Cursor deaktiviert ist, wird die BLOCK-Klausel ignoriert und die Zeilen werden einzeln eingelesen. Wenn auch ARRAY angegeben ist, wird die durch ARRAY angegebene Anzahl von Zeilen eingelesen.

PURGE-Klausel Die PURGE-Klausel kann nur in Embedded SQL verwendet werden. Sie bewirkt, dass der Client seine Puffer für alle Zeilen leert und dann die Abrufanforderung an den Datenbankserver sendet. Diese Abrufanforderung gibt möglicherweise einen Zeilenblock zurück.

FOR UPDATE-Klausel Die FOR UPDATE-Klausel gibt an, dass die abgerufene Zeile anschließend mit einer UPDATE WHERE CURRENT OF CURSOR-Anweisung aktualisiert wird. Diese Klausel bewirkt, dass der Datenbankserver eine Absichtssperre auf die Zeile setzt. Die Sperre wird bis zum Ende der aktuellen Transaktion aufrechterhalten.

ARRAY-Klausel Die ARRAY-Klausel kann nur in Embedded SQL verwendet werden. Sie ermöglicht sogenannte weite Abrufe, die mehr als eine Zeile gleichzeitig abrufen und so die Performance verbessern können.

Um weite Abrufe in Embedded SQL zu verwenden, fügen Sie die Abrufanweisung wie folgt in Ihren Code ein:

```
EXEC SQL FETCH ... ARRAY nnn
```

Dabei ist ARRAY *nnn* das letzte Element der FETCH-Anweisung. Die Abrufanzahl *nnn* kann eine Hostvariable sein. SQLDA muss *nnn* * (Spalten pro Zeile)-Variablen enthalten. Die erste Zeile wird in die SQLDA-Variablen von 0 bis (Spalten pro Zeile)-1 geschrieben, usw.

Bemerkungen

Die FETCH-Anweisung ruft eine Zeile aus dem benannten Cursor ab. Der Cursor muss zuvor geöffnet worden sein.

- **Embedded SQL verwenden** Die Embedded SQL-Anweisung FETCH bietet keine Unterstützung für Arrays.

Eine DECLARE CURSOR-Anweisung muss vor der FETCH-Anweisung im C-Quellcode erscheinen, und die OPEN-Anweisung muss vor der FETCH-Anweisung ausgeführt werden. Wenn eine Hostvariable für einen Cursornamen verwendet wird, erzeugt die DECLARE-Anweisung Code und muss vor der FETCH-Anweisung ausgeführt werden.

Der Server gibt in SQLCOUNT die Anzahl der abgerufenen Datensätze zurück, wobei SQLCOUNT immer größer als 0 ist, es sei denn, es liegt ein Fehler oder eine Warnung vor.

Wenn die Warnung SQLSTATE_NOTFOUND für den Abruf zurückgegeben wird, enthält das Feld *sqlerrd[2]* des SQLCA (SQLCOUNT) die Anzahl der Zeilen, um die der Abrufversuch die zulässige Cursorposition überschritten hat. Der Wert ist 0 (Null), falls die Zeile nicht gefunden wurde, die Position aber gültig ist, z.B., wenn FETCH RELATIVE 1 ausgeführt wird, und die Position auf der letzten Zeile eines Cursors ist. Der Wert ist positiv, falls das Abrufen jenseits des Cursors versucht wurde, und negativ, falls das Abrufen vor dem Anfang des Cursors versucht wurde. Der Cursor wird auf die letzte Zeile positioniert, falls das Abrufen jenseits des Cursors versucht wurde, und auf die erste Zeile, falls das Abrufen vor dem Anfang des Cursors versucht wurde.

Nach der erfolgreichen Ausführung der FETCH-Anweisung, wird das Feld *sqlerrd[1]* des SQLCA (SQLIOCOUNT) um die Anzahl der Ein-/Ausgabevorgänge erhöht, die erforderlich sind, um den Abruf auszuführen. Dieses Feld erhöht sich bei jeder Datenbankanweisung.

- **Einzeiliges Abrufen** Eine Zeile aus dem Ergebnis der SELECT-Anweisung wird in den Variablen der Variablenliste abgelegt. Die Entsprechung von Auswahlliste zu Hostvariablenliste ist eins zu eins.
- **Mehrzeiliges Abrufen** Eine oder mehr Zeilen aus dem Ergebnis der SELECT-Anweisung werden entweder in den Variablen in der *variable-list* oder in den von *sqlda-name* beschriebenen Programm Datenbereichen abgelegt. In beiden Fällen ist eine Entsprechung von eins zu eins zwischen der SELECT-Liste und der *hostvar-list* oder dem Deskriptorfeld *sqlda-name* gegeben.

Privilegien

Der Cursor muss geöffnet sein und Sie müssen das SELECT-Privileg für die Tabellen haben, Eigentümer der in der Deklaration des Cursors referenzierten Tabellen sein oder das SELECT ANY TABLE-Systemprivileg haben.

Nebenwirkungen

Eine FETCH-Anweisung kann dazu führen, dass mehrere Zeilen vom Server auf den Client abgerufen werden, wenn Prefetch aktiviert ist.

Siehe auch

- „SELECT-Anweisung“ auf Seite 1020
- „DECLARE CURSOR-Anweisung [ESQL] [SP]“ auf Seite 778
- „FOR-Anweisung“ auf Seite 857
- „PREPARE-Anweisung [ESQL]“ auf Seite 976
- „OPEN-Anweisung [ESQL] [SP]“ auf Seite 964
- „RESUME-Anweisung“ auf Seite 1003
- „Funktionsweise von Sperren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Abruf von Daten mit Embedded SQL“ [*SQL Anywhere Server - Programmierung*]
- „Mehrzeilige Abrufe oder Array-Abrufe“ [*SQL Anywhere Server - Programmierung*]
- „Tipp: Arbeitstabellen in der Abfrageverarbeitung verwenden ("Alle Zeilen" als Optimierungsziel verwenden)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Cursor in Embedded SQL“ [*SQL Anywhere Server - Programmierung*]
- „Cursor in Prozeduren, Triggern, benutzerdefinierten Funktionen und Batches“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Mit geringfügigen Ausnahmen ist Syntax 1 der FETCH-Anweisung eine Kernfunktion des SQL/2008-Standards. Andere SCROLL-Optionen als NEXT entsprechen der optionalen SQL-Sprachenfunktion F431 (scrollfähige Cursor mit Schreibschutz). SQL Anywhere bietet keine Unterstützung für die optionale FROM-Klausel der FETCH-Anweisung, wie sie im SQL/2008-Standard dokumentiert ist.

Syntax 2 ist eine Erweiterung des Herstellers.

Die Klauseln FOR UPDATE, PURGE, ARRAY, BLOCK und USING DESCRIPTOR sind Erweiterungen des Herstellers.

Beispiel

Im Folgenden finden Sie ein Beispiel für Embedded SQL:

```
EXEC SQL DECLARE cur_employee CURSOR FOR
SELECT EmployeeID, Surname FROM Employees;
EXEC SQL OPEN cur_employee;
EXEC SQL FETCH cur_employee
INTO :emp_number, :emp_name:indicator;
```

Es folgt ein Beispiel für eine Prozedur:

```
BEGIN
  DECLARE cur_employee CURSOR FOR
    SELECT Surname
    FROM Employees;
  DECLARE name CHAR(40);
  OPEN cur_employee;
  lp: LOOP
    FETCH NEXT cur_employee into name;
    IF SQLCODE <> 0 THEN LEAVE lp END IF;
    ...
  END LOOP;
  CLOSE cur_employee;
END
```

FOR-Anweisung

Wiederholt die Ausführung einer Anweisungsliste einmal für jede Zeile in einem Cursor.

Syntax

```
[ statement-label : ]
FOR for-loop-name AS cursor-name [ cursor-type ] CURSOR
{ FOR statement [ FOR { UPDATE [ cursor-concurrency ] | READ ONLY } ]
  | USING variable-name }
DO statement-list
END FOR [ statement-label ]
```

```
cursor-type :
NO SCROLL
| DYNAMIC SCROLL
| SCROLL
| INSENSITIVE
| SENSITIVE
```

```
cursor-concurrency : BY { VALUES | TIMESTAMP | LOCK }
```

```
variable-name : identifier
```

Parameter

NO SCROLL-Klausel Ein als NO SCROLL deklarierter Cursor ist auf die Vorwärtsbewegung durch die Ergebnismenge mit den Suchvorgängen FETCH NEXT und FETCH RELATIVE 0 beschränkt.

Da er nicht mehr zu der Zeile zurückkehren kann, wenn der Cursor die Zeile verlassen hat, gibt es keine Sensitivitätseinschränkungen für den Cursor. Wenn ein NO SCROLL-Cursor angefordert wird, liefert SQL Anywhere den wirksamsten Cursor, nämlich einen asensitiven Cursor.

DYNAMIC SCROLL-Klausel DYNAMIC SCROLL ist der Standard-Cursortyp. DYNAMIC SCROLL-Cursor können alle Formate der FETCH-Anweisung verwenden.

Wenn ein DYNAMIC SCROLL-Cursor angefordert wird, liefert SQL Anywhere einen asensitiven Cursor. Wenn Sie Cursor verwenden, muss immer zwischen Effizienz und Konsistenz abgewogen werden. Asensitive Cursor bieten eine hohe Performance, allerdings bei geringerer Konsistenz.

SCROLL-Klausel Ein als SCROLL deklarierter Cursor kann alle Formate der FETCH-Anweisung verwenden. Wenn ein SCROLL-Cursor angefordert wird, liefert SQL Anywhere einen wertsensitiven Cursor.

SQL Anywhere muss wertsensitive Cursor so ausführen, dass die Zugehörigkeit zur Ergebnismenge garantiert ist. DYNAMIC SCROLL-Cursor haben einen höheren Wirkungsgrad und sollten benutzt werden, wenn nicht ein konsistentes Verhalten von SCROLL-Cursors erforderlich ist.

INSENSITIVE-Klausel Die Werte und die Zugehörigkeit eines als INSENSITIVE deklarierten Cursors werden über dessen gesamte Anwendungsdauer hinweg festgelegt. Die Ergebnismenge der SELECT-Anweisung wird beim Öffnen des Cursors materialisiert. Ein FETCH-Abruf aus einem INSENSITIVE-Cursor berücksichtigt nicht die Auswirkungen anderer INSERT-, UPDATE-, MERGE-, PUT- oder DELETE-Anweisungen aus allen Verbindungen, einschließlich der Verbindung, die den Cursor geöffnet hat.

SENSITIVE-Klausel Ein als SENSITIVE (empfindlich) deklarierter Cursor erkennt Änderungen der Zugehörigkeit oder der Ergebnismengenwerte.

FOR UPDATE-Klausel FOR UPDATE ist der Standardwert. Cursor gelten standardmäßig als FOR UPDATE bei Ein-Tabellen-Abfragen ohne eine ORDER BY-Klausel bzw. wenn die Option ansi_update_constraints auf "Off" gesetzt ist. Wenn die Option ansi_update_constraints auf "Cursors" oder "Strict" gesetzt ist, gelten Cursor über einer Abfrage, die eine ORDER BY-Klausel enthält, standardmäßig als READ ONLY. Sie können jedoch Cursor explizit als aktualisierbar markieren, indem Sie die FOR UPDATE-Klausel verwenden.

FOR READ ONLY-Klausel Ein als FOR READ ONLY deklarierter Cursor kann nicht in (positionsbasierten) UPDATE- und DELETE- sowie in PUT-Anweisungen verwendet werden. Da es kostenträchtig ist, Aktualisierungen über Cursor mit einer ORDER BY-Klausel oder einem Join zuzulassen, sind Cursor über eine Abfrage, die einen Join von zwei oder mehr Tabellen enthält, READ ONLY und können nicht aktualisierbar gemacht werden, es sei denn, die Datenbankoption ansi_update_constraints ist auf OFF gesetzt. Als Antwort auf eine Anforderung für einen FOR UPDATE-Cursor übergibt SQL Anywhere entweder einen wertsensitiven oder einen sensitiven Cursor. Inensitive und asensitive Cursor können nicht aktualisiert werden.

Siehe auch

- „Sensitive Cursor“ [[SQL Anywhere Server - Programmierung](#)]
- „Insensitive Cursor“ [[SQL Anywhere Server - Programmierung](#)]
- „Wertsensitive Cursor“ [[SQL Anywhere Server - Programmierung](#)]
- „Asensitive Cursor“ [[SQL Anywhere Server - Programmierung](#)]

Bemerkungen

Die FOR-Anweisung ist eine Steueranweisung, die es Ihnen ermöglicht, eine Liste von SQL-Anweisungen einmal für jede Zeile in einem Cursor auszuführen. Die FOR-Anweisung entspricht einer zusammengesetzten Anweisung aus einer DECLARE-Anweisung für den Cursor und einer DECLARE-Anweisung für eine Variable für jede Spalte in der Ergebnismenge des Cursors, gefolgt von einer Schleife, die eine Zeile aus dem Cursor in die lokalen Variablen abrufen und die *statement-list* einmal für jede Zeile im Cursor ausführt.

Gültige Cursortypen sind Dynamic scroll (Standardwert), Scroll, No scroll, Sensitive und Insensitive.

Name und Datentyp jeder lokalen Variablen werden aus der im Cursor verwendeten *statement* abgeleitet. Mit einer SELECT-Anweisung werden die Datentypen verwendet, die in den Ausdrücken der Auswahlliste aufgeführt sind. Die Namen sind die Aliasnamen der Auswahllistenelemente, wenn diese vorhanden sind. Andernfalls werden die Namen der Spalten verwendet. Jedes Auswahllistenelement, das keine einfache Spaltenreferenz ist, muss ein Alias haben. Bei einer CALL-Anweisung werden die Namen und Datentypen der RESULT-Klausel in der Prozedurdefinition entnommen.

Die LEAVE-Anweisung kann verwendet werden, um die Ausführung bei der ersten Anweisung nach END FOR wieder aufzunehmen. Wenn das *statement-label* am Ende angegeben ist, muss es mit dem *statement-label* am Anfang übereinstimmen.

Der durch eine FOR-Anweisung erstellte Cursor wird implizit als WITH HOLD geöffnet, sodass innerhalb der Schleife ausgeführte Anweisungen, die ein COMMIT verursachen, nicht den Cursor schließen.

Vorsicht

Wenn Sie für *cursor-name* keinen Wert angeben, wird der *cursor-type* als *cursor-name* verwendet.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „DECLARE CURSOR-Anweisung [ESQL] [SP]“ auf Seite 778
- „FETCH-Anweisung [ESQL] [SP]“ auf Seite 853
- „CONTINUE-Anweisung“ auf Seite 581
- „LOOP-Anweisung“ auf Seite 950

Standards und Kompatibilität

- **SQL/2008** Die FOR-Anweisung ist Teil der optionalen SQL/2008-Sprachfunktion P002 (Verarbeitungsvollständigkeit). Die USING-Klausel der FOR-Anweisung ist eine Erweiterung des Herstellers. Wie bei der DECLARE CURSOR-Anweisung ist die Verwendung von *cursor-concurrency* (Cursorparallelität) eine Erweiterung des Herstellers, ebenso die Kombinationen aus Optionen für Cursorempfindlichkeit und SCROLL-Fähigkeit.

Beispiel

Das folgende Fragment veranschaulicht die Verwendung einer FOR-Schleife.

```
FOR names AS curs INSENSITIVE CURSOR FOR
SELECT Surname
FROM Employees
DO
    CALL search_for_name( Surname );
END FOR;
```

Dieses Fragment veranschaulicht die Verwendung einer FOR-Schleife.

```
BEGIN
FOR names AS curs SCROLL CURSOR FOR
SELECT EmployeeID, GivenName FROM Employees where EmployeeID < 130
FOR UPDATE BY VALUES
DO
    MESSAGE 'emp: ' || GivenName;
END FOR;
END
```

Im folgenden Beispiel wird gezeigt, wie eine FOR-Schleife in einer Prozedur namens myproc verwendet wird, die die ersten 10 Mitarbeiter aus der Employees-Tabelle basierend auf der Sortierreihenfolge zurückgibt, die beim Aufruf der Prozedur angegeben wurde (asc für aufsteigend und desc für absteigend).

```
CALL sa_make_object( 'procedure', 'myproc' );
ALTER PROCEDURE myproc (
    IN @order_by VARCHAR(20) DEFAULT NULL
)
RESULT ( Surname person_name_t )
BEGIN
    DECLARE @sql LONG VARCHAR;
    DECLARE @msg LONG VARCHAR;
    DECLARE LOCAL TEMPORARY TABLE temp_names( surnames person_name_t );
    SET @sql = 'SELECT TOP(10) * FROM Employees AS t ' ;

    CASE @order_by
    WHEN 'asc' THEN
        SET @sql = @sql || 'ORDER BY t.Surname ASC';
        SET @msg = 'Sorted ascending by last name: ';
    WHEN 'desc' THEN
        SET @sql = @sql || 'ORDER BY t.Surname DESC';
        SET @msg = 'Sorted ascending by last name: ';
    END CASE;

    FOR loop_name AS curs SCROLL CURSOR USING @sql
    DO
        INSERT INTO temp_names( surnames ) VALUES( Surname );
        MESSAGE( @msg || Surname );
    END FOR;
    SELECT * FROM temp_names;
END ;
```

Der Aufruf der Prozedur myproc und die Angabe von asc (z.B. CALL myproc('asc');) gibt folgende Ergebnisse zurück:

Surname
Ahmed
Barker
Barletta
Bertrand
Bigelow
Blaikie
Braun
Breault
Bucceri
Butterfield

FORWARD TO-Anweisung

Weiterleitung von SQL-Anweisungen an einen Fremdserver.

Syntax 1

FORWARD TO *server-name sql-statement*

Syntax 2

FORWARD TO [*server-name*]

Parameter

server-name Der Name des Fremdservers.

sql-statement Ein Befehl in der eigenen SQL-Syntax des Fremdservers. Der Befehl oder die Befehlsgruppe wird in geschweifte Klammern ({}) oder in Apostrophe gesetzt.

Bemerkungen

Mit der FORWARD TO-Anweisung können Benutzer den Server angeben, zu dem eine Durchreichverbindung erforderlich ist. Die Anweisung kann auf zwei Arten verwendet werden:

- **Syntax 1** Eine einzelne Anweisung an einen Fremdserver senden
- **Syntax 2** Versetzt SQL Anywhere in den Durchreichmodus, um eine Serie von Anweisungen an einen Fremdserver zu senden. Alle darauf folgenden Anweisungen werden direkt an den Fremdserver

übergeben. Um den Durchreichmodus abzuschalten, führen Sie FORWARD TO ohne die Angabe *server-name* aus.

Wenn Sie im Durchreichmodus vom Fremdserver eine Fehlermeldung erhalten, müssen Sie trotzdem eine FORWARD TO-Anweisung absetzen, um den Durchreichmodus abzuschalten.

Wenn eine Verbindung zum Server für den Benutzer eingerichtet wird, verwendet der Datenbankserver eine der folgenden Komponenten:

- Einen Aliasnamen für das Login auf dem Fremdserver mit CREATE EXTERNLOGIN
- Wenn kein Alias für ein Login auf dem Fremdserver erstellt ist, werden der Name und das Kennwort verwendet, die für die Kommunikation mit SQL Anywhere benutzt werden.

Wenn die Verbindung nicht mit dem angegebenen Server hergestellt werden kann, ist die Ursache dafür einer Meldung zu entnehmen, die an den Benutzer zurückgegeben wird.

Nachdem Anweisungen an den angeforderten Server geschickt wurden, werden alle Ergebnisse in eine Form verwandelt, die vom Clientprogramm erkannt wird.

Hinweis

Die FORWARD TO-Anweisung ist eine Serverdirektive und kann nicht in gespeicherten Prozeduren, Triggern, Ereignissen oder Batches verwendet werden.

Privilegien

Keine

Nebenwirkungen

Die Verbindung mit dem Fremdserver ist für die Dauer der FORWARD TO-Sitzung auf AUTOCOMMIT (unverkettet) gesetzt. Alle Verarbeitungsaufgaben, die vor der FORWARD TO-Anweisung noch ausstanden, werden automatisch festgeschrieben.

Siehe auch

- „PASSTHROUGH-Anweisung [SQL Remote]“ auf Seite 975

Beispiel

Das folgende Beispiel sendet eine SQL-Anweisung an den Fremdserver 'RemoteASE':

```
FORWARD TO RemoteASE { SELECT * FROM titles };
```

Das folgende Beispiel zeigt eine Durchreichsitzung mit dem Fremdserver 'aseprod':

```
FORWARD TO aseprod;  
  SELECT * FROM titles;  
  SELECT * FROM authors;  
FORWARD TO;
```

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

FROM-Klausel

Gibt die Datenbanktabellen oder Ansichten an, die in einer DELETE-, SELECT- oder UPDATE-Anweisung verwendet werden. Wenn sie in einer SELECT-Anweisung verwendet wird, kann die FROM-Klausel auch in einer MERGE- oder INSERT-Anweisung verwendet werden.

Syntax

FROM *table-expression*, ...

table-expression :

table-name

| *view-name*

| *procedure-name*

| *derived-table*

| *lateral-derived-table*

| *join-expression*

| (*table-expression*, ...)

| *openstring-expression*

| *apply-expression*

| *contains-expression*

| *dml-derived-table*

table-name :

[*userid*.] *table-name*

[[**AS**] *correlation-name*]

[**WITH** (*hint* [...])]

[**FORCE INDEX** (*index-name*)]

view-name :

[*userid*.] *view-name* [[**AS**] *correlation-name*]

[**WITH** (*table-hint*)]

procedure-name :

[*owner*.] *procedure-name* ([*parameter*, ...])

[**WITH** (*column-name data-type*, ...)]

[[**AS**] *correlation-name*]

derived-table :

(*select-statement*)

[**AS**] *correlation-name* [(*column-name*, ...)]

lateral-derived-table :

LATERAL (*select-statement* | *table-expression*)

[**AS**] *correlation-name* [(*column-name*, ...)]

join-expression :

table-expression *join-operator* *table-expression*

[**ON** *join-condition*]

join-operator :

[**KEY** | **NATURAL**] [*join-type*] **JOIN**

| **CROSS JOIN**

join-type :

INNER

| **LEFT** [**OUTER**]
| **RIGHT** [**OUTER**]
| **FULL** [**OUTER**]

hint :

table-hint | *index-hint*

table-hint :

READPAST
| **UPDLOCK**
| **XLOCK**
| **FASTFIRSTROW**
| **HOLDLOCK**
| **NOLOCK**
| **READCOMMITTED**
| **READUNCOMMITTED**
| **REPEATABLEREAD**
| **SERIALIZABLE**

index-hint :

NO INDEX
| **INDEX** ([**PRIMARY KEY** | **FOREIGN KEY**] *index-name* [, ...]) [**INDEX ONLY** { **ON** | **OFF** }]
| **CLUSTERED INDEX** [**INDEX ONLY** { **ON** | **OFF** }]

openstring-expression :

OPENSTRING ({ **FILE** | **VALUE** } *string-expression*)
WITH (*rowset-schema*)
[**OPTION** (*scan-option* ...)]
[**AS**] *correlation-name*

apply-expression :

table-expression { **CROSS** | **OUTER** } **APPLY** *table-expression*

contains-expression :

{ *table-name* | *view-name* } **CONTAINS** (*column-name* [,...], *contains-query*) [[**AS**] *score-correlation-name*]

rowset-schema :

column-schema-list
| **TABLE** [*owner*.]*table-name* [(*column-list*)]

column-schema-list :

{ *column-name* *user-or-base-type* | **filler**() } [, ...]

column-list :

{ *column-name* | **filler**() } [, ...]

scan-option :

BYTE ORDER MARK { **ON** | **OFF** }
| **COMMENTS INTRODUCED BY** *comment-prefix*
| { **COMPRESSED** | **AUTO** | **NOT COMPRESSED** }
| **DELIMITED BY** *string*
| **ENCODING** *encoding*
| { **ENCRYPTED KEY** *key-expression* | **NOT ENCRYPTED** }
| **ESCAPE CHARACTER** *character*
| **ESCAPES** { **ON** | **OFF** }

```

| FORMAT { TEXT | BCP }
| HEXADECIMAL { ON | OFF }
| QUOTE string
| QUOTES { ON | OFF }
| ROW DELIMITED BY string
| SKIP integer
| STRIP { ON | OFF | LTRIM | RTRIM | BOTH }

```

key-expression : *string* | *variable*

contains-query : *string*

dml-derived-table :

(*dml-statement*) **REFERENCING** ([*table-version-names* | **NONE**])

dml-statement :

insert-statement

delete-statement

update-statement

merge-statement

table-version-names :

OLD [**AS**] *correlation-name* [**FINAL** [**AS**] *correlation-name*]

| **FINAL** [**AS**] *correlation-name*

Parameter

- **table-name** Eine Basistabelle oder temporäre Tabelle. Tabellen, deren Eigentümer ein anderer Benutzer ist, können durch Angeben der Benutzer-ID qualifiziert werden. Tabellen, deren Eigentümer eine benutzerdefinierte Rolle ist, zu deren Berechtigungsempfängern der Benutzer gehört, werden standardmäßig auch ohne Angabe der Benutzer-ID gefunden.
- **view-name** Gibt eine in die Abfrage einzubeziehende Ansicht an. Genau wie Tabellen können auch Ansichten, die einem anderen Benutzer gehören, durch die Angabe der Benutzer-ID qualifiziert werden. Ansichten, deren Eigentümer Gruppen sind, zu denen der aktuelle Benutzer gehört, werden standardmäßig auch ohne Angabe der Benutzer-ID gefunden. Obwohl die Syntax Tabellen-Hints zulässt, haben solche Hints keine Auswirkung.
- **procedure-name** Eine gespeicherte Prozedur, die eine Ergebnismenge zurückgibt. Diese Klausel gilt nur für die FROM-Klausel von SELECT-Anweisungen. Die Klammern nach dem Prozedurnamen sind erforderlich, auch wenn die Prozedur keine Parameter erhält. DEFAULT kann anstelle eines optionalen Parameters angegeben werden.

Die Argumentliste kann nach Position oder durch die Verwendung von Schlüsselwörtern angegeben werden. Die Argumente stimmen mit dem entsprechenden Parameter in der Parameterliste für die Prozedur hinsichtlich Position überein (DEFAULT kann für optionale Parameter verwendet werden). Bei Angabe nach Schlüsselwörtern werden den Argumenten die benannten Parameter zugeordnet. Hinweise zur unterstützten Syntax für benannte Parameter finden Sie unter „Benannte Parameter“ auf Seite 93.

Wenn die gespeicherte Prozedur mehrere Ergebnismengen zurückgibt, wird nur die erste verwendet.

Die WITH-Klausel bietet eine Möglichkeit, Spaltennamenaliase für die Ergebnismenge der Prozedur anzugeben. Wenn eine WITH-Klausel angegeben ist, muss die Anzahl der Spalten der Anzahl von

Spalten in der Ergebnismenge der Prozedur entsprechen, und die Datentypen müssen mit denen in der Ergebnismenge der Prozedur übereinstimmen. Wenn keine WITH-Klausel angegeben ist, werden die Spaltennamen und Typen verwendet, die von der Prozedurdefinition definiert wurden. Die folgende Abfrage veranschaulicht die Verwendung der WITH-Klausel:

```
SELECT sp.ident, sp.quantity, Products.name
FROM GROUPO.ShowCustomerProducts( 149 )
  WITH ( ident INT, description CHAR(20), quantity INT ) sp
JOIN GROUPO.Products
ON sp.ident = Products.ID;
```

Für Embedded SQL-Anwendungen. Wenn Sie eine Prozedur ohne RESULT-Klausel erstellen und die Prozedur eine variable Ergebnismenge zurückgibt, kann ein DESCRIBE der SELECT-Anweisung mit Verweis auf die Prozedur fehlschlagen. Um ein Fehlschlagen des DESCRIBE-Vorgangs zu verhindern, ist es empfehlenswert, eine WITH-Klausel einzubeziehen, die das erwartete Schema der Ergebnismenge beschreibt.

Beim Auswählen aus einer Prozedur wird eine temporäre Tabelle generiert. Die folgende Abfrage erstellt beispielsweise eine lokale temporäre Tabelle:

```
BEGIN
...
  my: LOOP
    BEGIN
      SELECT TOP 1 NUMBER INTO conn_id FROM sa_conn_info( ) ORDER BY
NUMBER DESC;
      END;
...
  END LOOP my;
END
```

- **derived-table** Sie können eine SELECT-Anweisung anstelle eines Tabellen- oder Ansichtsnamens in der FROM-Klausel angeben. Eine auf diese Art verwendete SELECT-Anweisung wird als abgeleitete Tabelle bezeichnet und muss einen Alias erhalten. Beispiel: Die nachstehende Anweisung enthält eine abgeleitete Tabelle namens MyDerivedTable, die Produkte in der Products-Tabelle nach UnitPrice ordnet.

```
SELECT TOP 3 *
  FROM ( SELECT Description,
               Quantity,
               UnitPrice,
               RANK() OVER ( ORDER BY UnitPrice ASC )
               AS Rank
        FROM GROUPO.Products ) AS MyDerivedTable
ORDER BY Rank;
```

- **lateral-derived-table** Eine abgeleitete Tabelle, gespeicherte Prozedur oder verknüpfte Tabelle, die Referenzen auf Objekte in der übergeordneten Anweisung (oder Outer-Referenzen) enthalten kann. Sie müssen eine abgeleitete Lateral-Tabelle verwenden, wenn Sie in der FROM-Klausel eine äußere Referenz verwenden möchten.

Sie können äußere Referenzen nur für Tabellen verwenden, die der lateral abgeleiteten Tabelle in der FROM-Klausel vorangehen. Beispielsweise können Sie keine äußere Referenz für ein Element in der Auswahlliste verwenden.

Die Tabelle und die äußere Referenz müssen durch ein Komma getrennt sein. Beispielsweise sind folgende Abfragen gültig:

```
SELECT *
  FROM A, LATERAL( B LEFT OUTER JOIN C ON ( A.x = B.x ) ) myLateralDT;

SELECT *
  FROM A, LATERAL( SELECT * FROM B WHERE A.x = B.x ) myLateralDT;

SELECT *
  FROM A, LATERAL( procedure-name( A.x ) ) myLateralDT;
```

Die Angabe von LATERAL (*table-expression*) ist zur Angabe von LATERAL (SELECT * FROM *table-expression*) äquivalent.

- **openstring-expression** Verwenden Sie eine OPENSTRING-Klausel, um eine Abfrage innerhalb einer Datei oder einem Blob durchzuführen, wobei der Inhalt dieser Quellen als Zeilenmenge behandelt wird. Dabei geben Sie auch Informationen über das Schema der Datei oder des Blobs für die Erzeugung der Ergebnismenge an, da Sie keine definierte Struktur wie eine Tabelle oder Ansicht abfragen. Diese Klausel gilt nur für die FROM-Klausel einer SELECT-Anweisung. Wird für UPDATE- oder DELETE-Anweisungen nicht unterstützt.

Die ROWID-Funktion wird über die Ergebnismenge einer Tabelle unterstützt, die von einem OPENSTRING-Ausdruck erstellt wird.

Die folgenden Unterklauseln und Parameter der OPENSTRING-Klausel werden verwendet, um Daten in Dateien und Blobs zu definieren und abzufragen:

FILE- und VALUE-Klauseln Verwenden Sie die FILE-Klausel, um die abzufragende Datei anzugeben. Verwenden Sie die VALUE-Klausel, um den abzufragenden Blob-Ausdruck anzugeben. Der Datentyp für den Blob-Ausdruck wird als LONG BINARY angenommen. Sie können die READ_CLIENT_FILE-Funktion als Wert für die VALUE-Klausel angeben.

Hinweis

Wenn Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Hauptdatenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Wenn weder das FILE- noch das VALUE-Schlüsselwort angegeben ist, wird VALUE angenommen.

Bei der Verwendung von FORMAT SHAPEFILE wird nur FILE angenommen.

WITH-Klausel Verwenden Sie diese Klausel, um das Rowset-Schema (Spaltennamen und Datentypen) der abzufragenden Daten anzugeben. Sie können die Spalten direkt angeben (z.B. WITH (Surname CHAR(30), GivenName CHAR(30))). Sie können auch die TABLE-Unterklausel verwenden, um eine Tabelle für den Abruf von Schemainformationen zu referenzieren (z.B. WITH (TABLE dba.Employees (Surname, GivenName))). Sie müssen Eigentümer der angegebenen Tabelle sein oder SELECT-Privilegien dafür haben.

Bei der Angabe von Spalten können Sie filler() für Spalten verwenden, die sie bei den Eingabedaten auslassen möchten (z.B. WITH (filler(), Surname CHAR(30), GivenName CHAR(30))).

OPTION-Klausel Verwenden Sie die OPTION-Klausel, um Optionen für die syntaktische Analyse der Eingabedatei anzugeben, wie Escapezeichen, Begrenzer, Kodierung, usw. Unterstützte Optionen enthalten jene Optionen aus der LOAD TABLE-Anweisung, die die syntaktische Analyse einer Eingabedatei steuern.

- **scan-option** Hinweise zu den einzelnen Suchoptionen finden Sie unter den Ladeoptionen, die im „LOAD TABLE-Anweisung“ auf Seite 931 beschrieben werden.
- **apply-expression** Verwenden Sie diese Klausel, um eine Join-Bedingung anzugeben, in der der rechte *table-expression* für jede Zeile im linken *table-expression* ausgewertet wird. Sie können zum Beispiel einen Apply-Ausdruck verwenden, um eine Funktion, eine Prozedur oder eine abgeleitete Tabelle für jede Zeile in einem Tabellenausdruck auszuwerten.
- **contains-expression** Verwenden Sie die CONTAINS-Klausel nach einem Tabellennamen, um die Tabelle zu filtern und nur jene Zeilen zurückzugeben, die zur Volltextabfrage passen, die Sie in *contains-expression* eingegeben haben. Jede übereinstimmende Zeile der Tabelle wird gemeinsam mit einer Score-Spalte zurückgegeben, die mit *score-correlation-name* referenziert werden kann, wenn dieser angegeben ist. Wenn *score-correlation-name* nicht angegeben ist, kann die Score-Spalte mit dem Standardkorrelationsnamen "contains" referenziert werden.

Mit Ausnahme des optionalen Korrelationsname-Arguments übernimmt die CONTAINS-Klausel dieselben Argumente wie die CONTAINS-Suchbedingung.

Es muss ein Textindex für die Spalten vorhanden sein, die in der CONTAINS-Klausel aufgelistet sind.

Die *contains-query* darf nicht NULL oder eine leere Zeichenfolge sein. Wenn die Textkonfigurationseinstellungen das Löschen aller Begriffe in der *contains-query* bewirken, werden Zeilen aus der vom *contains-expression* referenzierten Basistabelle nicht zurückgegeben.

- **correlation-name** Verwenden Sie *correlation-name*, um einen Ersatznamen für eine Tabelle oder Ansicht in der FROM-Klausel anzugeben. Der Ersatzname kann dann von anderen Elementen in der Anweisung referenziert werden. Beispiel: emp und dep sind Korrelationsnamen für die Tabellen Employees und Departments:

```
SELECT Surname, GivenName, DepartmentName
FROM GROUPO.Employees emp, GROUPO.Departments dep,
WHERE emp.DepartmentID=dep.DepartmentID;
```

Korrelationsnamen können verwendet werden, um zwischen verschiedenen Instanzen derselben Tabelle zu unterscheiden. Die folgende Abfrage verknüpft beispielsweise die Tabelle "Employee" über den Korrelationsnamen "Mgr" mit sich selbst, um den Nachnamen des Managers für die einzelnen Mitarbeiter in das Ergebnis einzubeziehen:

```
SELECT Emp.EmployeeID, Emp.Surname, Dept.DepartmentName, Mgr.Surname AS
ManagerName
FROM GROUPO.Employees AS Emp, GROUPO.Departments AS Dept,
GROUPO.Employees AS Mgr
WHERE Emp.DepartmentID = Dept.DepartmentID AND Emp.ManagerID =
Mgr.EmployeeID;
```

- **dml-statement** Verwenden Sie *dml-statement* zum Festlegen der DML-Anweisung (INSERT, DELETE, UPDATE oder MERGE), aus der Sie Zeilen auswählen möchten. Während der Ausführung

wird zuerst die als *dml-derived-table* angegebene DML-Anweisung ausgeführt und die davon betroffenen Zeilen werden in eine temporäre Tabelle materialisiert, deren Spalten durch die REFERENCING-Klausel beschrieben werden. Die temporäre Tabelle repräsentiert die Ergebnismenge von *dml-derived-table*.

Verwenden Sie REFERENCING () oder REFERENCING (NONE), wenn die Ergebnisse nicht in eine temporäre Tabelle materialisiert werden müssen, weil sie nicht in der Abfrage referenziert werden.

Wenn Sie REFERENCING () oder REFERENCING (NONE) angeben, werden die aktualisierten Zeilen nicht in eine temporäre Tabelle materialisiert, welche die Ergebnismenge von *dml-derived-table* darstellt, weil sie nicht in der Abfrage referenziert werden. Die temporäre Tabelle ist in diesem Fall eine leere Tabelle. Diese Funktion können Sie verwenden, wenn die *dml-statement* vor der Hauptanweisung ausgeführt werden soll.

In den Ergebnissen enthalten OLD-Spalten die Werte gemäß dem Scan, bei dem die zu aktualisierenden Zeilen gesucht wurden. FINAL-Spalten enthalten die Werte nach der Prüfung der referenziellen Integrität, der Berechnung und der Aktualisierung der Standardspalten und nachdem alle Trigger ausgelöst haben (ausgenommen AFTER-Trigger vom Typ FOR STATEMENT).

Anweisung	Unterstützte Tabellenversionen
INSERT	FINAL
DELETE	OLD
UPDATE	FINAL und/oder OLD
MERGE	FINAL und/oder OLD

Wenn Sie einen OLD-Namen und einen FINAL-Namen angeben, werden zwei Korrelationsnamen verwendet. Es handelt sich jedoch nicht um echte Korrelationen, weil sich beide auf die dieselbe Ergebnismenge beziehen. Wenn Sie REFERENCING (OLD AS O FINAL AS F) angeben, gibt es ein implizites Join-Prädikat: *O.rowid* = *F.rowid*.

Die INSERT-Anweisung unterstützt nur FINAL. Daher erscheinen die Werte von aktualisierten Zeilen, die durch eine INSERT ON EXISTING UPDATE-Anweisung geändert werden, nicht in der Ergebnismenge der abgeleiteten Tabelle. Verwenden Sie stattdessen die MERGE-Anweisung, um den insert-else-update-Vorgang zu verarbeiten.

Die DML-Anweisung kann nur eine einzelne aktualisierbare Tabelle referenzieren. Aktualisierungen mehrerer Tabellen geben einen Fehler zurück. Darüber hinaus ist eine Auswahl aus *dml-statement* nicht zulässig, wenn die DML-Anweisung in einer korrelierten Unterabfrage oder einem allgemeinen Tabellenausdruck erscheint, weil die Semantik dieser Konstrukte unklar sein kann.

- **WITH *table-hint*-Klausel** Mit der Klausel WITH *table-hint* können Sie das Verhalten festlegen, das nur für diese Tabelle und nur für diese Anweisung angewendet wird. Sie können diese Klausel verwenden, um das Verhalten zu ändern, ohne die Isolationsstufe zu ändern oder eine Datenbank- bzw. Verbindungsoption festlegen zu müssen. Tabellen-Hints können für Basistabellen, temporäre Tabellen und materialisierte Ansichten benutzt werden.

Vorsicht

Bei der Klausel *WITH table-hint* handelt es sich um eine erweiterte Funktion, die nur dann verwendet werden sollte, wenn sie wirklich benötigt wird, und auch dann nur von erfahrenen Datenbankadministratoren. Zusätzlich kann es sein, dass die Einstellungen nicht in allen Situationen berücksichtigt werden.

- **Mit der Isolationsstufe zusammenhängende Tabellen-Hints** Die Tabellen-Hints für Isolationsstufen werden verwendet, um das Isolationsstufenverhalten bei der Abfrage von Tabellen anzugeben. Sie geben eine Sperrmethode an, die nur bei den angegebenen Tabellen und nur für die aktuelle Abfrage gilt. Snapshot-Isolationsstufen können Sie nicht als Tabellen-Hints angeben.

Es folgt eine Liste der mit der Isolationsstufe zusammenhängenden Tabellen-Hints, die unterstützt werden:

Tabellen-Hint	Beschreibung
HOLDLOCK	Setzt das Verhalten auf ein Äquivalent der Isolationsstufe 3. Dieser Tabellen-Hint ist synonym mit <code>SERIALIZABLE</code> .
NOLOCK	Setzt das Verhalten auf ein Äquivalent der Isolationsstufe 0. Dieser Tabellen-Hint ist synonym mit <code>READUNCOMMITTED</code> .
READCOMMITTED	Setzt das Verhalten auf ein Äquivalent der Isolationsstufe 1.
READPAST	Weist den Datenbankserver an, schreibgeschützte Zeilen zu ignorieren anstatt bei ihnen anzuhalten. Dieser Tabellen-Hint kann nur mit Isolationsstufe 1 verwendet werden. Der <code>READPAST</code> -Hint wird nur respektiert, wenn der Korrelationsname in der <code>FROM</code> -Klausel eine Basistabelle oder eine global gemeinsam genutzte temporäre Tabelle referenziert. Unter anderen Umständen (Ansichten, Proxytabellen und Tabellenfunktionen) wird der <code>READPAST</code> -Hint ignoriert. Abfragen mit Ansichten können <code>READPAST</code> verwenden, wenn der Hint für einen Korrelationsnamen angegeben ist, der eine Basistabelle bezeichnet. Die Verwendung des <code>READPAST</code> -Tabellen-Hints kann aufgrund der Interaktion von Sperren und Prädikatauswertungen im Server zu abnormalen Situationen führen. Außerdem können Sie den <code>READPAST</code> -Hint nicht mit Tabellen verwenden, die Ziel einer <code>DELETE</code> -, <code>INSERT</code> - oder <code>UPDATE</code> -Anweisung sind.
READUNCOMMITTED	Setzt das Verhalten auf ein Äquivalent der Isolationsstufe 0. Dieser Tabellen-Hint ist synonym mit <code>NOLOCK</code> .
REPEATABLEREAD	Setzt das Verhalten auf ein Äquivalent der Isolationsstufe 2.

Tabellen-Hint	Beschreibung
SERIALI-ZABLE	Setzt das Verhalten auf ein Äquivalent der Isolationsstufe 3. Dieser Tabellen-Hint ist synonym mit HOLDLOCK.
UPDLOCK	Zeigt an, dass die von der Anweisung verarbeiteten Zeilen aus der Hint-Tabelle mit Absichtssperren gesperrt werden. Die betroffenen Zeilen bleiben bis zum Abschluss der Transaktion gesperrt. UPDLOCK gilt für alle Isolationsstufen und verwendet Absichtssperren.
XLOCK	Zeigt an, dass die von der Anweisung verarbeiteten Zeilen aus der Hint-Tabelle exklusiv gesperrt werden müssen. Die betroffenen Zeilen bleiben bis zum Abschluss der Transaktion gesperrt. XLOCK gilt für alle Isolationsstufen und verwendet Schreibsperren.

Hinweis

Wenn Sie Abfragen für Datenbanken schreiben, die an einer MobiLink-Synchronisation teilnehmen, wird empfohlen, dass Sie den READPAST-Tabellen-Hint nicht in Ihren Synchronisationsskripts verwenden.

Weitere Hinweise finden Sie unter:

- [„download_cursor \(Tabellenereignis\)“](#) [*MobiLink - Serveradministration*]
- [„download_delete_cursor \(Tabellenereignis\)“](#) [*MobiLink - Serveradministration*]
- [„upload_fetch \(Tabellenereignis\)“](#) [*MobiLink - Serveradministration*]

Wenn Sie eine Verwendung von READPAST in Erwägung ziehen, weil Ihre Anwendung viele Aktualisierungen durchführt, die sich auf die Download-Performance auswirken, wäre die Verwendung der Snapshot-Isolation eine alternative Lösung.

- **Optimierungsziel-Tabellen-Hint (FASTFIRSTROW)** Der Tabellen-Hint FASTFIRSTROW ermöglicht Ihnen die Einstellung des Optimierungsziels für die Abfrage, ohne die optimization_goal-Option auf First-row setzen zu müssen. Wenn Sie FASTFIRSTROW benutzen, wählt SQL Anywhere einen Zugriffsplan, der die Zeit für den Abruf der ersten Zeile in den Abfrageergebnissen reduzieren soll.
- **WITH (index-hint)-Klausel** Mit der Klausel WITH (*index-hint*) können Sie Index-Hints angeben, mit denen die Algorithmen für die Auswahl des Abfrageoptimiererplans außer Kraft gesetzt werden, und den Optimierer anweisen, wie über Indizes auf die Tabelle zugegriffen werden soll. Index-Hints können für Basistabellen, temporäre Tabellen und materialisierte Ansichten benutzt werden.
- **NO INDEX** Verwenden Sie diese Klausel, um ein sequenzielles Durchsuchen der Tabelle zu erzwingen (Indizes werden nicht benutzt). Sequenzielles Durchsuchen kann sehr kostenaufwändig sein.

- **INDEX ([PRIMARY KEY | FOREIGN KEY] *index-name* [...])** Verwenden Sie diese Klausel, um bis zu vier Indizes anzugeben, die der Optimierer benutzen muss, um die Abfrage durchzuführen.

Wenn einer der angegebenen Indizes nicht verwendet werden kann, wird ein Fehler zurückgegeben.

Sie können PRIMARY KEY oder FOREIGN KEY angeben, um eine Mehrdeutigkeit in den Fällen auszuschließen, in denen der PRIMARY KEY-Index und der FOREIGN KEY-Index für eine Tabelle denselben Namen haben.

Wenn Sie im Index-Hint einen Indexnamen angeben, ohne den PRIMARY- oder FOREIGN-Schlüssel anzugeben, und mehrere Indizes mit demselben Namen für eine Tabelle existieren, wählt der Optimierer den normalen Index. Wenn ein normaler Index nicht vorhanden ist, wählt der Optimierer den Primärschlüsselindex. Wenn ein Primärschlüsselindex nicht existiert, wird anstelle dessen der Fremdschlüsselindex verwendet.

index-name kann durch die Angabe der Benutzer-ID und des Tabellennamens für den Index qualifiziert werden.

Die in der INDEX-Klausel angegebenen Indizes müssen für diese Tabelle definiert sein. Andernfalls wird ein Fehler zurückgegeben. `FROM Products WITH(INDEX (Products.xx))` gibt beispielsweise einen Fehler zurück, wenn der Index `xx` für die Tabelle `Products` nicht definiert ist. Und `FROM Products WITH(INDEX (sales_order_items.sales_order_items))` gibt einen Fehler zurück, weil der Index `sales_order_items.sales_order_items` zwar vorhanden, aber nicht für die Tabelle `Products` definiert ist.

- **INDEX ONLY { ON | OFF }** Verwenden Sie diese Klausel, um zu steuern, ob ein reiner Indexabruf von Daten durchgeführt wird. Wenn die Klausel `INDEX (index-name...)` mit `INDEX ONLY ON` angegeben ist, versucht der Datenbankserver einen reinen Indexabruf mit den angegebenen Indizes. Wenn einer der angegebenen Indizes für eine reine Indexabfrage nicht benutzt werden kann, wird eine Fehlermeldung zurückgegeben (z.B. wenn keine Indizes vorhanden sind oder die bestehenden für die Abfrage nicht geeignet sind).

Geben Sie `INDEX ONLY OFF` an, um einen reinen Indexabruf zu verhindern.

- **FORCE INDEX (*index-name*)** Die Syntax `FORCE INDEX (index-name)` wird aus Kompatibilitätsgründen bereitgestellt und kann nicht mehr als eine Indexbezeichnung aufnehmen. Diese Klausel ist gleichwertig mit `WITH (INDEX (index-name))`.
- **CLUSTERED INDEX** Verwenden Sie diese Klausel, um anzugeben, dass der Optimierer einen Clustered-Index benutzen muss, falls einer vorhanden ist. Der Indexname wird nicht angegeben, weil nur ein Clustered-Index für eine Basistabelle vorhanden sein kann. Wenn ein Clustered-Index nicht existiert oder nicht verwendet werden kann, wird ein Fehler zurückgegeben.

Bemerkungen

Unterabfragen und Unterabfrage-Bedingungen werden als Argumente für gespeicherte Prozeduren und Tabellenfunktionen in der FROM-Klausel unterstützt. Die folgende FROM-Klausel ist beispielsweise gültig:

```
SELECT *, ( SELECT 12 x ) D
FROM sa_rowgenerator( 1,( SELECT 12 x ) );
```

Die Anweisungen SELECT, UPDATE und DELETE erfordern eine Tabellenliste, um anzugeben, welche Tabellen von der Anweisung verwendet werden.

Hinweis

Obwohl sich die Beschreibung der FROM-Klausel auf Tabellen bezieht, gilt sie, sofern nicht anders angegeben, auch für Ansichten und abgeleitete Tabellen.

Die FROM-Klausel erstellt eine Ergebnismenge, die aus allen Spalten aller angegebenen Tabellen besteht. Anfänglich sind alle Zeilenkombinationen in den Komponententabellen in der Ergebnismenge enthalten, und die Anzahl der Kombinationen wird im Allgemeinen durch JOIN-Bedingungen bzw. WHERE-Klauseln verringert.

Es ist nicht möglich, eine ON-Klausel mit CROSS JOIN zu verwenden.

Privilegien

Die FILE-Klausel im *openstring-expression* erfordert das READ FILE-Privileg.

Die TABLE-Klausel im *openstring-expression* erfordert, dass der Benutzer Eigentümer der referenzierten Tabellen ist oder das SELECT ANY TABLE-Privileg hat.

Nebenwirkungen

Keine.

Siehe auch

- „CONTAINS-Suchbedingung“ auf Seite 59
- „Konzepte und Referenz zu Textindizes“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „optimization_goal-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- Schreibsperrern [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- Absichtssperren [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Beispiel-Textkonfigurationsobjekte“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „SELECT über eine DML-Anweisung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Datenmanipulationsanweisungen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Isolationsstufen und Konsistenz“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „MobiLink-Isolationsstufen“ [*MobiLink - Serveradministration*]
- „LOAD TABLE-Anweisung“ auf Seite 931
- „DELETE-Anweisung“ auf Seite 791
- „SELECT-Anweisung“ auf Seite 1020
- „UPDATE-Anweisung“ auf Seite 1109
- „INSERT-Anweisung“ auf Seite 917
- „MERGE-Anweisung“ auf Seite 952
- Abgeleitete Tabellen abfragen [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Joins: Daten aus mehreren Tabellen abrufen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

SQL/2008 Die FROM-Klausel ist ein grundlegender Bestandteil des SQL/2008-Standards. Aufgrund der Komplexität der FROM-Klausel sollten Sie einzelne Komponenten einer FROM-Klausel anhand der entsprechenden Teile des Standards überprüfen. Im Folgenden finden Sie eine nicht vollständige Liste der optionalen SQL/2008-Sprachenfunktionen, die in SQL Anywhere unterstützt werden:

- CROSS JOIN, FULL OUTER JOIN und NATURAL JOIN bilden die optionale SQL/2008-Funktion F401.
- INTERSECT und INTERSECT ALL bilden die optionale SQL/2008-Funktion F302.
- EXCEPT ALL ist die optionale Sprachenfunktion F304.
- Abgeleitete Tabellen sind SQL/2008-Funktion F591.
- Prozeduren in der FROM-Klausel (Tabellenfunktionen) sind Funktion T326. Der SQL/2008-Standard verlangt das TABLE-Schlüsselwort, um die Ausgabe einer Prozedur als Tabellenausdruck zu kennzeichnen, während in SQL Anywhere das TABLE-Schlüsselwort nicht erforderlich ist.
- Allgemeine Tabellenausdrücke sind die optionale SQL/2008-Sprachenfunktion T121. Die Verwendung eines allgemeinen Tabellenausdrucks in einer abgeleiteten, in einem anderen allgemeinen Tabellenausdruck verschachtelten Tabelle ist Sprachenfunktion T122.
- Rekursive Tabellenausdrücke sind Funktion T131. Die Verwendung eines rekursiven Tabellenausdrucks in einer abgeleiteten, in einem allgemeinen Tabellenausdruck verschachtelten Tabelle ist die optionale SQL/2008-Sprachenfunktion T132.

Die folgenden Komponenten der FROM-Klausel sind Erweiterungen des Herstellers:

- KEY JOIN.
- CROSS APPLY- und OUTER APPLY-Joins.
- OPENSTRING.
- *table-expression* mit CONTAINS (Volltextsuche).
- Angabe einer *dml-statement* als abgeleitete Tabelle.
- Alle Tabellen-Hints, einschließlich der Verwendung von WITH, FORCE INDEX und READPAST sowie Isolationsstufen-Hints.
- LATERAL (*table-expression*) ist eine Erweiterung des Herstellers. LATERAL (*select-statement*) ist im SQL-Standard als optionale Sprachenfunktion T491 enthalten.

Beispiel

Die folgenden Klauseln sind gültige FROM-Klauseln:

```
...
FROM GROUPO.Employees
...

...
FROM GROUPO.Employees NATURAL JOIN GROUPO.Departments
...

...
FROM GROUPO.Customers
KEY JOIN GROUPO.SalesOrders
KEY JOIN GROUPO.SalesOrderItems
KEY JOIN GROUPO.Products
...

...
FROM GROUPO.Employees CONTAINS ( Street, ' Way ' )
...
```

Die folgende Abfrage veranschaulicht die Verwendung von abgeleiteten Tabellen in einer Abfrage:

```
SELECT Surname, GivenName, number_of_orders
FROM GROUPO.Customers JOIN
  ( SELECT CustomerID, COUNT(*)
    FROM GROUPO.SalesOrders
    GROUP BY CustomerID )
  AS sales_order_counts( CustomerID,
                        number_of_orders )
ON ( Customers.ID = sales_order_counts.CustomerID )
WHERE number_of_orders > 3;
```

Die folgende Abfrage veranschaulicht, wie Zeilen aus Ergebnismengen von gespeicherten Prozeduren ausgewählt werden.

```
SELECT t.ID, t.QuantityOrdered AS q, p.name
FROM GROUPO.ShowCustomerProducts( 149 ) t JOIN GROUPO.Products p
ON t.ID = p.ID;
```

Das folgende Beispiel zeigt, wie eine Abfrage mit der OPENSTRING-Klausel ausgeführt wird, um eine Datei abzufragen. Die CREATE TABLE-Anweisung erstellt eine Tabelle namens testtable mit den zwei Spalten column1 und column2. Die UNLOAD-Anweisung erstellt eine Datei namens *testfile.dat* durch Entladen von Zeilen aus der Tabelle RowGenerator. Die SELECT-Anweisung verwendet die OPENSTRING-Klausel in einer FROM-Klausel, um eine Abfrage aus *testfile.dat* mit den Schemainformationen aus den Tabellen testtable und RowGenerator durchzuführen. Die Abfrage gibt eine Zeile mit dem Wert 49 zurück.

```
CREATE TABLE testtable( column1 CHAR(10), column2 INT );
UNLOAD SELECT * FROM RowGenerator TO 'testfile.dat';
SELECT A.column2
  FROM OPENSTRING( FILE 'testfile.dat' )
  WITH ( TABLE testtable( column2 ) ) A, RowGenerator B
 WHERE A.column2 = B.row_num
  AND A.column2 < 50
  AND B.row_num > 48;
```

Das folgende Beispiel zeigt, wie eine Abfrage mit der OPENSTRING-Klausel ausgeführt wird, um einen Zeichenfolgenwert abzufragen. Die SELECT-Anweisung verwendet die OPENSTRING-Klausel in einer FROM-Klausel, um einen Zeichenfolgenwert unter Verwendung der Schemainformationen abzurufen, die in der WITH-Klausel übergeben wurden. Die Abfrage gibt zwei Spalten mit drei Zeilen zurück.

```
SELECT *
  FROM OPENSTRING( VALUE '1,"First"$2,"Second"$3,"Third"' )
  WITH (c1 INT, c2 VARCHAR(30))
  OPTION ( DELIMITED BY ',' ROW DELIMITED BY '$' )
  AS VALS
```

Das folgende Beispiel zeigt die Durchführung einer Abfrage für die Auswahl der Zeilen, die durch eine Datenmanipulationsanweisung geändert wurden. In diesem Beispiel wird eine Warnung ausgegeben, wenn der Lagerbestand an blauen Artikeln um mehr als die Hälfte fällt.

```
SELECT old_products.name, old_products.quantity, final_products.quantity
FROM
  ( UPDATE GROUPO.Products SET quantity = quantity - 10 WHERE color = 'Blue' )
REFERENCING ( OLD AS old_products FINAL AS final_products )
WHERE final_products.quantity < 0.5 * old_products.quantity;
```

GET DATA-Anweisung [ESQL]

Ruft Zeichenfolgen- oder Binärdaten für eine Spalte der aktuellen Zeile eines Cursors ab.

Syntax

```
GET DATA cursor-name
COLUMN column-num
OFFSET start-offset
[ WITH TEXTPTR ]
{ USING DESCRIPTOR sqlda-name | INTO hostvar, ... }
```

cursor-name : *identifier* | *hostvar*

column-num : *integer* | *hostvar*

start-offset : *integer* | *hostvar*

sqlda-name : *identifizier*

Parameter

COLUMN-Klausel Der Wert *column-num* beginnt bei eins und gibt an, aus welcher Spalte die Daten abgerufen werden sollen. Die Spalte muss ein Zeichenfolgen- oder Binärtyp sein.

OFFSET-Klausel Der *start-offset* gibt die im Feldwert zu überspringende Byte-Anzahl an. Im Allgemeinen ist das die Anzahl der Bytes, die zuvor abgerufen wurde. Die Anzahl der Bytes, die von dieser GET DATA-Anweisung abgerufen wird, ist durch die Länge der Zielhostvariablen bestimmt.

WITH TEXTPTR-Klausel Wenn die WITH TEXTPTR-Klausel angegeben ist, wird ein Textzeiger in eine zweite Hostvariable oder in das zweite Feld im SQLDA abgerufen. Dieser Textzeiger kann mit den Transact-SQL-Anweisungen READ TEXT und WRITE TEXT verwendet werden. Der Textzeiger ist ein 16-Bit Binärwert und kann folgendermaßen deklariert werden:

```
DECL_BINARY( 16 ) textptr_var;
```

Die WITH TEXTPTR-Klausel kann nur in Long-Datentypen (LONG BINARY, LONG VARCHAR, TEXT, IMAGE) verwendet werden. Wenn Sie versuchen, ihn mit einem anderen Datentyp zu verwenden, wird der Fehler INVALID_TEXTPTR_VALUE zurückgegeben.

Die Gesamtlänge der Daten wird im Feld SQLCOUNT der SQLCA-Struktur zurückgegeben.

USING DESCRIPTOR-Klausel Der *sqlda-name* gibt den SQLDA-Bereich (SQL Descriptor Area) an, der die abgerufenen Daten aufnimmt. Die USING DESCRIPTOR-Klausel bietet eine dynamische Methode für die Angabe von Hostvariablen für die Aufnahme der abgerufenen Daten.

INTO-Klausel Verwenden Sie die INTO-Klausel zur Angabe der Hostvariablen, die die abgerufenen Daten aufnimmt. Der Indikatorwert für die Zielhostvariable ist vom Typ *a_sql_len*, der derzeit ein 16-Bit Wert ist, sodass er nicht immer groß genug bemessen sein wird, um die Anzahl von gekürzten Byte aufzunehmen. Stattdessen enthält er einen negativen Wert, wenn das Feld NULL enthält, einen positiven Wert (NICHT unbedingt die Anzahl der abgeschnittenen Bytes), wenn der Wert abgeschnitten ist, und NULL, wenn ein Nicht-NULL-Wert nicht abgeschnitten ist.

Dementsprechend gilt: Wenn eine LONG VARCHAR-, LONG NVARCHAR- oder eine LONG BINARY-Hostvariable mit einem Offset verwendet wird, der größer als Null ist, gibt das Feld *untrunc_len* nicht die genaue Größe vor der Kürzung an.

Bemerkungen

Mit dieser Anweisung können Sie einen Teil von einem Spaltenwert aus der Zeile an der aktuellen Cursorposition abrufen. Der Cursor muss geöffnet und mithilfe von FETCH auf einer Zeile positioniert sein.

GET DATA wird normalerweise verwendet, um Felder des Typs LONG BINARY oder LONG VARCHAR abzurufen.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „FETCH-Anweisung [ESQL] [SP]“ auf Seite 853
- „READTEXT-Anweisung [T-SQL]“ auf Seite 986
- „SET-Anweisung“ auf Seite 1054
- INTO-Klausel auf Seite 877

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird GET DATA verwendet, um ein binäres großes Objekt (oftmals auch Binary Large Object oder kurz BLOB genannt) abzurufen.

```
EXEC SQL BEGIN DECLARE SECTION;
DECL_BINARY(1000) piece;
short ind;

EXEC SQL END DECLARE SECTION;
int size;
/* Open a cursor on a long varchar field */
EXEC SQL DECLARE big_cursor CURSOR FOR
SELECT long_data FROM some_table
WHERE key_id = 2;
EXEC SQL OPEN big_cursor;
EXEC SQL FETCH big_cursor INTO :piece;
for( offset = 0; ; offset += piece.len ) {
    EXEC SQL GET DATA big_cursor COLUMN 1
    OFFSET :offset INTO :piece:ind;
    /* Done if the NULL value */
    if( ind < 0 ) break;
    write_out_piece( piece );
    /* Done when the piece was not truncated */
    if( ind == 0 ) break;
}
EXEC SQL CLOSE big_cursor;
```

GET DESCRIPTOR-Anweisung [ESQL]

Ruft Informationen über eine Variable innerhalb eines Deskriptorbereichs oder deren Wert ab.

Syntax

```
GET DESCRIPTOR descriptor-name
{ hostvar = COUNT | VALUE { integer | hostvar } assignment, ... }

assignment :
hostvar =
TYPE
| LENGTH
| PRECISION
| SCALE
```

DATA
INDICATOR
NAME
NULLABLE
RETURNED_LENGTH

descriptor-name : *identifier*

Bemerkungen

Die GET DESCRIPTOR-Anweisung wird verwendet, um Informationen über eine Variable innerhalb eines Deskriptorbereichs abzurufen oder um ihren Inhalt zu ermitteln.

Mit dem Wert von { *integer* | *hostvar* } wird die Variable innerhalb des Deskriptorbereichs bestimmt, über den die Daten abgerufen werden sollen. Bei GET...DATA findet eine Typüberprüfung statt, um sicherzustellen, dass die Hostvariable und die Deskriptorvariable den gleichen Datentyp haben. LONG VARCHAR und LONG BINARY werden von GET DESCRIPTOR...DATA nicht unterstützt.

Wenn ein Fehler auftritt, wird er in dem SQLCA zurückgegeben.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „ALLOCATE DESCRIPTOR-Anweisung [ESQL]“ auf Seite 449
- „DEALLOCATE DESCRIPTOR-Anweisung [ESQL]“ auf Seite 777
- „SET DESCRIPTOR-Anweisung [ESQL]“ auf Seite 1033
- „Der SQL-Deskriptor-Bereich (SQLDA)“ [*SQL Anywhere Server - Programmierung*]

Standards und Kompatibilität

- **SQL/2008** GET DESCRIPTOR ist Teil der optionalen SQL/2008-Sprachenfunktion B031 (Basic Dynamic SQL).

Beispiel

Mit dem folgenden Beispiel wird der Typ einer Spalte mit der Position "Spaltennummer" in SQLDA zurückgegeben.

```
int get_type( SQLDA *sqlda, int col_num )
{
    EXEC SQL BEGIN DECLARE SECTION;
    int ret_type;
    int col = col_num;
    EXEC SQL END DECLARE SECTION;
    EXEC SQL GET DESCRIPTOR sqlda VALUE :col :ret_type = TYPE;
    return( ret_type );
}
```

GET OPTION-Anweisung [ESQL]

Ruft die aktuelle Einstellung einer Option ab. Es wird jedoch empfohlen, stattdessen die CONNECTION_PROPERTY-Funktion zu verwenden.

Syntax

```
GET OPTION [ userid.]option-name  
{ INTO hostvar | USING DESCRIPTOR sqlda-name }
```

userid : *identifier*, *string* | *hostvar*

option-name :
identifier
| *string*
| *hostvar*

hostvar : Indikatorvariable zulässig

sqlda-name : *identifier*

Bemerkungen

Die GET OPTION-Anweisung wird für die Kompatibilität mit früheren Versionen der Software bereitgestellt. Das empfohlene Verfahren zum Abrufen der Optionswerte jedoch ist die Verwendung der Systemfunktion CONNECTION_PROPERTY.

Die GET OPTION-Anweisung liefert die Optionseinstellung der Option *option-name* für den Benutzer *userid* oder für den verbundenen Benutzer, wenn *userid* nicht angegeben ist. Hierbei handelt es sich entweder um die persönliche Einstellung des Benutzers oder um die PUBLIC-Einstellung, wenn keine Einstellung für den verbundenen Benutzer vorliegt. Wenn es sich bei der angegebenen Option um eine Datenbankoption handelt und der Benutzer eine temporäre Einstellung für diese Option hat, dann wird die temporäre Einstellung abgerufen.

Wenn *option-name* nicht existiert, gibt die Anweisung GET OPTION die Warnung SQLE_NOTFOUND zurück.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- [„SET OPTION-Anweisung“ auf Seite 1039](#)
- [„Alphabetische Liste der Systemprozeduren“ auf Seite 1168](#)
- [„CONNECTION_PROPERTY-Funktion \[System\]“ auf Seite 199](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung veranschaulicht die Verwendung von GET OPTION.

```
EXEC SQL GET OPTION 'date_format' INTO :datefmt;
```

GOTO-Anweisung [T-SQL]

Erstellt eine Verzweigung zu einer mit einem Label versehenen Anweisung.

Syntax

label : **GOTO** *label*

Bemerkungen

Jede Anweisung in einer Prozedur, einem Trigger oder einem Batch in Transact-SQL kann mit einem Label versehen sein. Der Labelname ist ein gültiger Bezeichner, auf den ein Doppelpunkt folgt. In der GOTO-Anweisung wird der Doppelpunkt nicht verwendet.

Wenn Sie zusammengesetzte Anweisungen verschachteln, können Sie nur zu Labels in der aktuellen zusammengesetzten Anweisung und allen übergeordneten zusammengesetzten Anweisungen gehen (GOTO). Es ist nicht möglich, zu Labels in anderen zusammengesetzten Anweisungen zu gehen, die in den übergeordneten Anweisungen verschachtelt sind.

Privilegien

Keine.

Nebenwirkungen

Keine.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Der folgende Transact-SQL-Batch gibt vier Mal eine Ja-Meldung im Meldungsfenster des Datenbankservers aus:

```
DECLARE @count SMALLINT
SELECT @count = 1
restart:
PRINT 'yes'
SELECT @count = @count + 1
WHILE @count <=4
    GOTO restart
```

GRANT-Anweisung

Erteilen Sie Rollen und Privilegien an Benutzer und Rollen.

Hinweis

Die in Versionen der Software vor 16.0 verwendete GRANT-Syntax für Berechtigungen und Gruppen wird weiterhin unterstützt, jedoch nicht mehr empfohlen. Weitere Hinweise zu dieser Syntax finden Sie unter „GRANT-Anweisung (Berechtigungen und Gruppen) (nicht mehr empfohlen)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Zum Vergleich finden Sie Informationen zur GRANT-Anweisungssyntax in SQL Anywhere 12.0.1 unter <http://dcx.sybase.com/index.html#1201/en/dbreference/grant-statement.html>.

Syntax: Systemrollen erteilen

```
GRANT ROLE system-role  
TO grantee [ , ... ]
```

system-role :

dbo

DIAGNOSTICS

PUBLIC

rs_systabgroup

SA_DEBUG

SYS

SYS_REPLICATION_ADMIN_ROLE

SYS_RUN_REPLICATION_ROLE

SYS_SAMONITOR_ROLE

SYS_SPATIAL_ADMIN_ROLE

Syntax: Benutzerdefinierte Rollen erteilen

```
GRANT ROLE user-defined-role [...]  
TO grantee [ , ... ]  
[ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
```

Syntax: Kompatibilitätsrollen erteilen

```
GRANT compatibility-role-name [...]  
TO grantee [ , ... ]  
[ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]  
[ WITH NO SYSTEM PRIVILEGE INHERITANCE ]
```

compatibility-role-name :

SYS_AUTH_BACKUP_ROLE

SYS_AUTH_DBA_ROLE

SYS_AUTH_PROFILE_ROLE

SYS_AUTH_READCLIENTFILE_ROLE

SYS_AUTH_READFILE_ROLE

SYS_AUTH_RESOURCE_ROLE

SYS_AUTH_SA_ROLE

SYS_AUTH_SSO_ROLE

SYS_AUTH_VALIDATE_ROLE

SYS_AUTH_WRITECLIENTFILE_ROLE

SYS_AUTH_WRITEFILE_ROLE

Syntax: Systemprivilegien erteilen

```
GRANT system-privilege [,...]
TO grantee [ , ... ]
[ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
```

Syntax: Privilegien auf Objektebene erteilen

```
GRANT object-level-privilege, ...
ON [ owner.]object-name
TO to-userid, ...
[ WITH GRANT OPTION ]
[ FROM from-userid ]

object-level-privilege :
ALL [ PRIVILEGES ]
| ALTER
| DELETE
| INSERT
| REFERENCES [ ( column-name, ... ) ]
| SELECT [ ( column-name, ... ) ]
| UPDATE [ ( column-name, ... ) ]
```

Syntax: SET USER-Systemprivileg erteilen

```
GRANT SET USER [ ( user-list | ANY [ WITH ROLES role-list ] ) ]
TO grantee [,...]
[ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
```

Syntax: CHANGE PASSWORD-Systemprivileg erteilen

```
GRANT CHANGE PASSWORD [ ( user-list | ANY [ WITH ROLES role-list ] ) ]
TO grantee [,...]
[ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
```

Parameter

- ***role-name*** Der Name einer Systemrolle, Kompatibilitätsrolle, benutzererweiterten Rolle oder benutzerdefinierten Rolle.
- ***grantee*** Die Benutzer-ID eines Benutzers oder der Name einer Rolle. Sie können Kompatibilitätsrollen keine Privilegien und Rollen erteilen. Sie können jeder Systemrolle Rollen und Privilegien erteilen, jedoch unterstützen nur die folgenden Systemrollen Logins: PUBLIC, dbo, diagnostics, rs_systabgroup und SA_DEBUG.
- **WITH [NO] ADMIN OPTION-Klausel** Diese Klausel ist nur beim Erteilen von Systemprivilegien und Nicht-Systemrollen anwendbar. Sie können keine Administrationsrechte für Systemrollen erteilen. Nur Benutzer mit MANAGE ROLES-Systemprivileg können Systemrollen verwalten (erteilen und entziehen).

Der Standardwert ist WITH NO ADMIN OPTION, was bedeutet, dass der *grantee* die Rollen bzw. Privilegien erhält, aber nicht die Möglichkeit, sie zu verwalten.

Wenn WITH ADMIN OPTION angegeben ist, erhält jeder *grantee* Administrationsrechte für jede *role-granted* und jedes *system-privilege-granted*.

Wenn WITH ADMIN ONLY OPTION angegeben ist, erhält der *grantee* nur Administrationsrechte für die Rollen, aber nicht die Rollen und Privilegien selbst. Sie können nie die WITH NO SYSTEM PRIVILEGE INHERITANCE-Klausel zusammen mit der WITH ADMIN ONLY OPTION-Klausel verwenden.

Die WITH ADMIN OPTION-Klausel können Sie nur beim Erteilen von SYS_AUTH_DBA_ROLE zusammen mit der WITH NO SYSTEM PRIVILEGE INHERITANCE-Klausel verwenden.

- **WITH NO SYSTEM PRIVILEGE INHERITANCE** Diese Klausel verhindert, dass die Berechtigungsempfänger einer Rolle die Systemprivilegien der Rolle erben. Normalerweise gilt: Wenn Sie einem Benutzer oder einer Rolle eine Kompatibilitätsrolle erteilen, sind die Systemprivilegien der Kompatibilitätsrolle sowohl für die Rolle als auch für ihre Berechtigungsempfänger verfügbar. Wenn Sie die Vererbung der Systemprivilegien für eine Kompatibilitätsrolle deaktivieren, sind die Systemprivilegien nur für die Rolle verfügbar, aber nicht für ihre Berechtigungsempfänger.

Die WITH NO SYSTEM PRIVILEGE INHERITANCE-Klausel wird aus Gründen der Abwärtskompatibilität bereitgestellt. Das Deaktivieren der Vererbung von Systemprivilegien für eine Kompatibilitätsrolle entspricht dem Verhalten der nicht vererbten Berechtigungen in Datenbanken bis Version 12. Das Aktivieren der Vererbung von Systemprivilegien für eine Kompatibilitätsrolle entspricht dem Verhalten aller Systemrollen und benutzerdefinierten Rollen.

Sie können die Vererbung der Systemprivilegien deaktivieren, wenn Sie eine der folgenden Rollen Benutzern, benutzererweiterten Rollen oder Systemrollen erteilen:

- SYS_AUTH_DBA_ROLE-Rolle
- SYS_AUTH_RESOURCE_ROLE
- SYS_AUTH_BACKUP_ROLE
- SYS_AUTH_VALIDATE_ROLE
- SYS_RUN_REPLICATION_ROLE

Außerdem können Sie die WITH ADMIN OPTION-Klausel nur beim Erteilen von SYS_AUTH_DBA_ROLE zusammen mit der WITH NO SYSTEM PRIVILEGE INHERITANCE-Klausel verwenden. Sie können nie die WITH NO SYSTEM PRIVILEGE INHERITANCE-Klausel zusammen mit der WITH ADMIN ONLY OPTION-Klausel verwenden.

Das Deaktivieren der Vererbung von Systemprivilegien für einen Benutzer ist nur dann sinnvoll, wenn Sie vorhaben, den Benutzer in eine benutzererweiterte Rolle zu konvertieren.

Siehe „[Änderungen des Vererbungsverhaltens für einige Berechtigungen, die zu Rollen geworden sind](#)“ [[SQL Anywhere Server - Datenbankadministration](#)]

- **object-level-privilege**
 - **ALL-Privileg** Dieses Privileg erteilt die Privilegien ALTER, DELETE, INSERT, REFERENCES, SELECT und UPDATE für Tabellen. Dieses Privileg erteilt die Privilegien DELETE, INSERT und UPDATE für Ansichten.
 - **ALTER-Privileg** Dieses Privileg ermöglicht es dem Benutzer, die benannte Tabelle mit der ALTER TABLE-Anweisung zu ändern. Für Ansichten ist dieses Privileg nicht zulässig.

- **DELETE-Privileg** Dieses Privileg ermöglicht es dem Benutzer, Zeilen aus der benannten Tabelle oder Ansicht zu löschen.
 - **INSERT-Privileg** Dieses Privileg ermöglicht es dem Benutzer, Zeilen in die benannte Tabelle oder Ansicht einzufügen.
 - **LOAD-Privileg** Dieses Privileg ermöglicht es dem Benutzer, die benannte Tabelle zu laden.
 - **REFERENCES-Privileg** Dieses Privileg ermöglicht es dem Benutzer, Indizes für die benannte Tabelle zu erstellen und für die Fremdschlüssel, die die benannten Tabellen referenzieren. Wenn Spaltennamen angegeben sind, kann der Benutzer nur diese Spalten referenzieren. REFERENCES-Privilegien für Spalten können nicht für Ansichten, sondern nur für Tabellen erteilt werden. INDEX ist ein Synonym für REFERENCES.
 - **SELECT-Privileg** Dieses Privileg ermöglicht es dem Benutzer, Informationen in der betreffenden Ansicht oder Tabelle anzuzeigen. Wenn Spaltennamen angegeben sind, können die Benutzer lediglich diese Spalten ansehen. SELECT-Privilegien für Spalten können nicht für Ansichten, sondern nur für Tabellen erteilt werden.
 - **TRUNCATE-Privileg** Dieses Privileg ermöglicht es dem Benutzer, das benannte Objekt zu kürzen.
 - **UPDATE-Privileg** Dieses Privileg ermöglicht es dem Benutzer, Zeilen in der betreffenden Ansicht oder Tabelle zu aktualisieren. Wenn Spaltennamen angegeben sind, kann der Benutzer nur diese Spalten aktualisieren.
 - **WITH GRANT OPTION** Wenn WITH GRANT OPTION angegeben wird, erhält die angegebene Benutzer-ID auch die Berechtigung, dasselbe Privileg anderen Benutzer-IDs zu erteilen. Mitglieder von Gruppen erben WITH GRANT OPTION nicht, wenn dies einer Gruppe erteilt wird.
- **GRANT SET USER**
 - **user-list** Geben Sie die kommagetrennte Liste aller Benutzer-IDs (Zielbenutzer) ein, die *grantee-list* impersonieren kann. Beispiel: GRANT SET USER (u1, u2, u3) ...
 - **ANY [WITH ROLES *target_role_list*]-Klausel** Legen Sie fest, wen der *grantee* impersonieren kann, ohne bestimmte Benutzer-IDs anzugeben.

Wenn nur ANY angegeben wird, kann der Benutzer jeden anderen Benutzer impersonieren. Dies ist die Standardeinstellung.

Wenn ANY WITH ROLES *role-list* angegeben wird, können Benutzer in *grantee-list* alle Benutzer impersonieren, die mindestens eine der in *role-list* aufgeführten Rollen haben, wobei *role-list* eine kommagetrennte Liste von Rollen ist.
 - **WITH ADMIN [ONLY] OPTION-Option** Die Klauseln WITH ADMIN OPTION und WITH ADMIN ONLY OPTION können nur zusammen mit der ANY-Klausel angegeben werden.
 - **GRANT CHANGE PASSWORD**

- **user-list** Geben Sie eine kommagetrennte Liste von Benutzern an, für die der *grantee* Kennwörter ändern kann.
- **ANY [WITH ROLES role-list]** Legen Sie fest, für wen der *grantee* das Kennwort ändern kann, ohne bestimmte Benutzer-IDs anzugeben.

Wenn nur ANY angegeben wird, kann der Benutzer das Kennwort für jeden Benutzer ändern. Dies ist die Standardeinstellung.

Wenn ANY WITH ROLES *role-list* angegeben wird, kann der *grantee* das Kennwort für alle Benutzer ändern, die mindestens eine der in *role-list* aufgeführten Rollen haben, wobei *role-list* eine kommagetrennte Liste von Rollen ist.

- **WITH ADMIN [ONLY] OPTION-Option** Die Klauseln WITH ADMIN OPTION und WITH ADMIN ONLY OPTION können nur zusammen mit der ANY-Klausel angegeben werden.

Bemerkungen

Sie können Privilegien für deaktivierte Objekte erteilen. Privilegien für deaktivierte Objekte werden in der Datenbank gespeichert und treten in Kraft, wenn das betreffende Objekt aktiviert wird.

Mit Ausnahme der SYS-Rolle können Sie einer Systemrolle zusätzliche Privilegien und Rollen erteilen und entziehen, sofern Sie Administrationsrechte für die zu erteilenden bzw. zu entziehenden Privilegien und Rollen haben.

Wenn Sie SET USER einem Benutzer mehrmals erteilen und dabei unterschiedliche Benutzer-IDs angeben, die der Benutzer impersonieren kann, werden zusätzliche Benutzer zur Liste derjenigen hinzugefügt, die der Benutzer impersonieren kann. (Die zuvor erteilten Privilegien werden also nicht überschrieben.)

Das Erteilen von Impersonierungsrechten (GRANT SET USER) sagt nichts darüber aus, ob ein Benutzer einen anderen Benutzer erfolgreich impersonieren kann. Die Auswertung, ob ein Benutzer einen anderen Benutzer impersonieren kann, erfolgt, wenn die betreffende Benutzer-ID durch Ausführen einer SETUSER-Anweisung versucht, die Impersonierung eines anderen Benutzers zu starten. Der impersonierende Benutzer muss das SET USER-Systemprivileg haben und außerdem die erforderlichen Mindestkriterien für die Impersonierung erfüllen.

Privilegien

Sie benötigen Administrationsrechte für alle zu erteilenden Privilegien und Rollen.

Wenn Sie Privilegien auf Objektebene erteilen möchten, benötigen Sie außerdem die MANAGE ANY OBJECT PRIVILEGE-Systemprivileg mit Administrationsrechten.

Nebenwirkungen

Keine

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Privilegien an Rollen erteilen Wenn Sie der Rolle RoleA das CREATE ANY OBJECT-Systemprivileg erteilen möchten, ohne ihr Administrationsrechte zu geben, führen Sie die folgende Anweisung aus:

```
GRANT CREATE ANY OBJECT TO RoleA;
```

Wenn Sie der Rolle RoleA das CREATE ANY OBJECT-Systemprivileg erteilen möchten, zusammen mit der Möglichkeit, Benutzern und anderen Rollen das Systemprivileg zu erteilen oder zu entziehen, führen Sie die folgende Anweisung aus:

```
GRANT CREATE ANY OBJECT TO RoleA WITH ADMIN OPTION;
```

Wenn Sie der Rolle RoleA Administrationsrechte für das BACKUP DATABASE-Systemprivileg erteilen möchten, aber nicht die Möglichkeit, das BACKUP DATABASE-Privileg zu nutzen, führen Sie die folgende Anweisung aus:

```
GRANT BACKUP DATABASE TO RoleA WITH ADMIN ONLY OPTION;
```

Benutzern Rollen erteilen Wenn Sie dem Benutzer Bob die Rolle SecurityRole ohne Administrationsrechte erteilen möchten, führen Sie die folgende Anweisung aus:

```
GRANT ROLE SecurityRole TO Bob;
```

Wenn Sie der Rolle RoleB alle RoleA zugeordneten Privilegien erteilen möchten, aber keine Administrationsrechte für RoleA, führen Sie die folgende Anweisung aus:

```
GRANT ROLE RoleA TO RoleB;
```

Wenn Sie der Benutzerin Jane die Rolle RoleB einschließlich der dazugehörigen Administrationsrechte erteilen möchten, führen Sie die folgende Anweisung aus:

```
GRANT ROLE RoleB TO Jane WITH ADMIN OPTION;
```

Wenn Sie dem Benutzer John die Administrationsrechte für RoleB erteilen möchten, jedoch ohne die Möglichkeit, diese Rolle zu nutzen, führen Sie die folgende Anweisung aus:

```
GRANT ROLE RoleB TO John WITH ADMIN ONLY OPTION;
```

Benutzern SET USER erteilen Im folgenden Beispiel wird festgelegt, dass User4 und User5 die User1, User2 und User3 impersonieren können.

```
GRANT SET USER ( User1, User2, User3 ) TO User4, User5;
```

Im folgenden Beispiel wird festgelegt, dass User1 alle Benutzer in der Datenbank impersonieren kann. Außerdem kann User1 das SET USER-Systemprivileg anderen Benutzern erteilen.

```
GRANT SET USER (ANY) TO User1 WITH ADMIN OPTION;
```

Im folgenden Beispiel wird festgelegt, dass User1 alle Benutzer impersonieren kann, denen die SYS_AUTH_BACKUP_ROLE-Kompatibilitätsrolle erteilt wurde.

```
GRANT SET USER (ANY WITH ROLES SYS_AUTH_BACKUP_ROLE) TO User1;
```

Siehe auch

- „Systemprivilegien“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Rollen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Kompatibilitätsrollen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Benutzerdefinierte Rollen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Impersonierung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „SETUSER-Anweisung“ auf Seite 1059
- „GRANT-Anweisung (Berechtigungen und Gruppen) (nicht mehr empfohlen)“ [[SQL Anywhere Server - Datenbankadministration](#)]

GRANT CONNECT-Anweisung

Erstellt einen neuen Benutzer und kann auch von einem Benutzer verwendet werden, um sein eigenes Kennwort zu ändern.

Hinweis

Es wird empfohlen, zum Erstellen von Benutzern die CREATE USER-Anweisung zu verwenden statt der GRANT CONNECT-Anweisung.

Syntax - Verbindungsprivilegien erteilen

```
GRANT CONNECT TO userid, ...  
[ AT starting-id ]  
[ IDENTIFIED BY password, ... ]
```

Parameter

- **CONNECT TO-Klausel** Erstellt einen neuen Benutzer. GRANT CONNECT kann außerdem jeder Benutzer verwenden, um sein eigenes Kennwort zu ändern. Um einen Benutzer mit einer leeren Zeichenfolge als Kennwort zu erstellen, verwenden Sie folgende Angaben:

```
GRANT CONNECT TO userid IDENTIFIED BY "";
```

Um einen Benutzer ohne Kennwort zu erstellen, verwenden Sie folgende Angaben:

```
GRANT CONNECT TO userid;
```

Ein Benutzer ohne Kennwort kann keine Verbindung zur Datenbank herstellen. Das ist nützlich, wenn Sie eine Gruppe erstellen und nicht möchten, dass jemand mit der Benutzer-ID der Gruppe eine Verbindung zur Datenbank herstellt.

Mit der Option `verify_password_function` können Sie eine Funktion zum Implementieren von Kennwortregeln angeben (z.B., dass Kennwörter mindestens eine Ziffer enthalten müssen). Wenn eine Funktion zur Kennwortüberprüfung verwendet wird, können Sie nicht mehr als eine Benutzer-ID mit Kennwort in der GRANT CONNECT-Anweisung angeben.

- **AT-Klausel** Diese Klausel ist für die interne Verwendung durch das Dienstprogramm zum Entladen vorgesehen.

Bemerkungen

Keine

Privilegien

Sie müssen entweder Ihr eigenes Kennwort mit GRANT CONNECT ändern oder das MANAGE ANY USER-Privileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE USER-Anweisung“ auf Seite 770
- „verify_password_function-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird ein neuer Datenbankbenutzer namens SQLTester mit dem Kennwort "welcome" erstellt.

```
GRANT CONNECT TO SQLTester  
IDENTIFIED BY welcome
```

GRANT CONSOLIDATE-Anweisung [SQL Remote]

Identifiziert die Datenbank unmittelbar über der aktuellen Datenbank in einer SQL Remote-Hierarchie, die Nachrichten von der aktuellen Datenbank empfangen wird.

Syntax

```
GRANT CONSOLIDATE  
TO userid  
TYPE message-system, ...  
ADDRESS address-string, ...  
[ SEND { EVERY | AT } hh:mm:ss ]
```

message-system :
FILE | FTP | SMTP

address : *string*

Parameter

userid Die Benutzer-ID des Benutzers, dem die Privilegien erteilt werden sollen.

message-system Eines der Nachrichtensysteme, die von SQL Remote unterstützt werden.

address Die Adresse für das angegebene Nachrichtensystem

Bemerkungen

In einer SQL Remote-Installation muss der Datenbank unmittelbar über der aktuellen Datenbank in einer SQL Remote-Hierarchie das CONSOLIDATE-Privileg erteilt werden. GRANT CONSOLIDATE wird in der entfernten Datenbank ausgeführt, um ihre konsolidierte Datenbank zu kennzeichnen. Jede Datenbank kann nur eine Benutzer-ID mit CONSOLIDATE-Privilegien haben: Sie können keine Datenbank einrichten, die für mehr als eine konsolidierte Datenbank als entfernte Datenbank fungiert.

Der konsolidierte Benutzer wird anhand eines Nachrichtensystems identifiziert, indem die Methode gekennzeichnet wird, mit der Nachrichten an den konsolidierten Benutzer gesendet und von ihm empfangen werden. Der Adressenname muss eine gültige Adresse für das Nachrichtensystem sein, die von Apostrophen umschlossen ist. Pro entfernter Datenbank kann nur ein konsolidierter Benutzer angemeldet sein.

Beim FILE-Nachrichtentyp ist die Adresse ein Unterverzeichnis des Verzeichnisses, auf das die SQLREMOTE-Umgebungsvariable zeigt.

Die GRANT CONSOLIDATE-Anweisung ist für die konsolidierte Datenbank erforderlich, um Nachrichten zu empfangen; sie selbst subskribiert aber keine Daten für die konsolidierte Datenbank. Um die Daten zu subskribieren, muss eine Subskription für die konsolidierte Benutzer-ID für eine der Publikationen in der aktuellen Datenbank erstellt werden. Durch die Ausführung des Extraktionsdienstprogramms in einer konsolidierten Datenbank wird eine entfernte Datenbank erstellt, bei der die korrekte GRANT CONSOLIDATE-Anweisung bereits ausgeführt ist.

Die optionalen SEND EVERY- und SEND AT-Klauseln legen die Häufigkeit fest, mit der Nachrichten versendet werden. Die Zeichenfolge enthält eine Zeitangabe, welche die Dauer zwischen den Nachrichten (bei SEND EVERY) oder eine Uhrzeit (bei SEND AT) darstellt, zu denen Nachrichten versendet werden. Mit SEND AT werden Nachrichten einmal pro Tag versendet.

Wenn einem Benutzer Privilegien als entfernter Benutzer ohne SEND EVERY- oder SEND AT-Klausel erteilt wurden, verarbeitet der Nachrichtenagent Nachrichten und wird anschließend gestoppt. Damit der Nachrichtenagent kontinuierlich läuft, müssen Sie sicherstellen, dass für jeden Benutzer mit REMOTE-Privileg entweder eine SEND AT- oder eine SEND EVERY-Frequenz festgelegt ist.

Man kann davon ausgehen, dass der Nachrichtenagent in vielen entfernten Datenbanken periodisch ausgeführt wird, und dass in der konsolidierten Datenbank keine SEND-Klausel angegeben ist.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „GRANT PUBLISH-Anweisung [SQL Remote]“ auf Seite 894
- „GRANT REMOTE-Anweisung [SQLRemote]“ auf Seite 900
- „REVOKE CONSOLIDATE-Anweisung [SQL Remote]“ auf Seite 1006
- „ALTER REMOTE MESSAGE TYPE-Anweisung [SQL Remote]“ auf Seite 487
- „CREATE REMOTE [MESSAGE] TYPE-Anweisung [SQL Remote]“ auf Seite 694
- „CREATE SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 730
- „CONSOLIDATE-Privileg erteilen“ [*SQL Remote*]
- „SQL Remote-Nachrichtensysteme“ [*SQL Remote*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung erteilt der Benutzer-ID "Sam_Singer" den konsolidierten Status:

```
GRANT CONSOLIDATE TO Sam_Singer  
TYPE SMTP  
ADDRESS 'Singer, Samuel';
```

GRANT CREATE-Anweisung

Ermöglicht es einem Benutzer, Datenbankobjekte im angegebenen DBSpace zu erstellen.

Syntax

```
GRANT CREATE ON dbspace-name  
TO userid, ...
```

Bemerkungen

Wenn Sie eine Datenbank initialisieren, enthält sie eine Datenbankdatei. Die erste Datenbankdatei wird als Hauptdatei bezeichnet. Standardmäßig werden alle Datenbankobjekte und alle Daten in der Hauptdatei gespeichert. Ein DBSpace ist eine zusätzliche Datenbankdatei, die zusätzlichen Speicherplatz für Daten bereitstellt. Eine Datenbank kann in bis zu 13 verschiedenen Dateien gespeichert sein (einer Hauptdatei und 12 DBSpaces). Jede Tabelle muss zusammen mit ihren Indizes in einer Datenbankdatei enthalten sein. Der SQL-Befehl CREATE DBSPACE fügt eine neue Datei zur Datenbank hinzu.

Privilegien

Sie müssen das MANAGE ANY USER-Privileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Datenbankdateitypen“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

GRANT EXECUTE-Anweisung

Erteilt einem Benutzer das Privileg zum Ausführen einer Prozedur oder einer benutzerdefinierten Funktion.

Syntax

```
GRANT EXECUTE ON [ owner.]{ procedure-name | user-defined-function }  
TO userid, ...
```

Bemerkungen

Keine.

Privilegien

Sie müssen Eigentümer der Prozedur sein oder das MANAGE ANY OBJECT PRIVILEGE-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Systemprozeduren“ auf Seite 1161

Standards und Kompatibilität

- **SQL/2008** Kernfunktion des SQL/2008-Standards zum Erteilen von EXECUTE-Privilegien für gespeicherte Prozeduren.

Beispiel

Im folgenden Beispiel wird es dem Benutzer SQLTester ermöglicht, die Prozedur "Calculate_Report" auszuführen.

```
GRANT EXECUTE ON Calculate_Report  
TO SQLTester;
```

Die folgende SQL-Anweisung erteilt M_Haneef das erforderliche Privileg zum Ausführen einer Prozedur namens my_procedure.

```
GRANT EXECUTE  
ON my_procedure  
TO M_Haneef;
```

GRANT INTEGRATED LOGIN-Anweisung

Erteilt einem Benutzer das Privileg zum Ausführen einer Prozedur oder einer benutzerdefinierten Funktion.

Syntax

```
GRANT INTEGRATED LOGIN TO user-profile-name, ...  
AS USER userid
```

Bemerkungen

Die GRANT INTEGRATED LOGIN-Anweisung erstellt eine explizite Zuordnung des integrierten Logins zwischen einem oder mehreren Windows-Benutzer- oder -Gruppenprofilen und einer vorhandenen Datenbankbenutzer-ID. Diese ermöglicht es Benutzern, die sich am lokalen Computer angemeldet haben, eine Verbindung mit einer Datenbank herzustellen, ohne eine Benutzer-ID oder ein Kennwort angeben zu müssen. Der *user-profile-name* kann die Form *domain\user-name* haben. Die Datenbankbenutzer-ID, der das integrierte Login zugeordnet ist, muss ein Kennwort haben.

Privilegien

Sie müssen das MANAGE ANY USER-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Integrierte Windows-Logins“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

GRANT KERBEROS LOGIN-Anweisung

Erstellt eine Kerberos-authentifizierte Login-Zuordnung von einem oder mehreren Kerberos-Prinzipalen zu einer vorhandenen Datenbankbenutzer-ID.

Syntax

```
GRANT KERBEROS LOGIN TO client-Kerberos-principal, ...  
AS USER userid
```

Bemerkungen

Die GRANT KERBEROS LOGIN-Anweisung erstellt eine Kerberos-authentifizierte Login-Zuordnung von einem oder mehreren Kerberos-Prinzipalen zu einer vorhandenen Datenbankbenutzer-ID. Diese Login-Zuordnung ermöglicht es Benutzern, die sich bei Kerberos angemeldet haben (Benutzern, die in Kerberos über ein gültiges Ticket-erteilendes Ticket verfügen), eine Verbindung mit einer Datenbank herstellen, ohne eine Benutzer-ID oder ein Kennwort angeben zu müssen.

So verwenden Sie die GRANT KERBEROS LOGIN-Anweisung:

- Sie müssen Kerberos bereits für die Verwendung von SQL Anywhere konfiguriert haben.
- Sie müssen Ihre SQL Anywhere-Datenbank bereits für die Verwendung von Kerberos konfiguriert haben.

- Der Datenbankbenutzer und der Kerberos-Prinzipal müssen bereits vorhanden sein.
- Die Datenbankbenutzer-ID, der das Kerberos-Login zugeordnet ist, muss ein Kennwort haben.
- Der *client-Kerberos-principal* muss das Format *Benutzer/Instanz@REALM* haben, wobei */Instanz* optional ist. Der vollständige Prinzipal muss einschließlich des Realms angegeben werden, und Prinzipale, die sich nur in Instanz oder Realm unterscheiden, werden unterschiedlich behandelt.
- Bei Prinzipalen wird die Groß-/Kleinschreibung berücksichtigt, d.h., sie müssen in der korrekten Schreibweise angegeben werden. Zuordnungen für mehrere Prinzipale, die sich nur in der Groß- und Kleinschreibung unterscheiden, werden nicht unterstützt (z.B. können Sie keine Zuordnungen für jjordan@MYREALM.COM und JJordan@MYREALM.COM erhalten).
- Wenn es keine explizite Zuordnung für einen Kerberos-Prinzipal gibt und die Guest-Datenbankbenutzer-ID existiert und ein Kennwort hat, dann stellt der Kerberos-Prinzipal die Verbindung her, indem er die Guest-Datenbankbenutzer-ID verwendet (dieselbe Guest-Datenbankbenutzer-ID wie bei integrierten Logins).

Privilegien

Sie müssen das MANAGE ANY USER-Privileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Kerberos-Authentifizierung“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende SQL-Anweisung erteilt einem fiktiven Kerberos-Benutzer namens pchin Datenbank-Login-Privilegien.

```
GRANT KERBEROS LOGIN TO "pchin@MYREALM.COM"  
AS USER "kerberos-user";
```

GRANT PUBLISH-Anweisung [SQL Remote]

Erteilt einer Benutzer-ID Publikationseigentümerrechte. Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben, um Publikationseigentümerrechte erteilen zu können.

Syntax

```
GRANT PUBLISH TO userid
```

Bemerkungen

Jede Datenbank in einer SQL Remote-Installation wird in ausgehenden Nachrichten durch eine Benutzer-ID gekennzeichnet, die Publikationseigentümer genannt wird. Die GRANT PUBLISH-Anweisung legt fest, welche Publikationseigentümer-ID diesen ausgehenden Nachrichten zugeordnet wird. Wenn Sie den Publikationseigentümer ändern möchten, müssen Sie die Publikationseigentümerrechte dem aktuellen Publikationseigentümer entziehen (REVOKE PUBLISH) und anschließend dem neuen Publikationseigentümer erteilen.

Sie können den Publikationseigentümer einer Datenbank ermitteln, indem Sie folgendermaßen den CURRENT PUBLISHER-Spezialwert abfragen:

```
SELECT CURRENT PUBLISHER;
```

Wenn es keinen Publikationseigentümer gibt, ist der CURRENT PUBLISHER-Wert NULL.

Der Spezialwert CURRENT PUBLISHER kann als Standardeinstellung für Spalten verwendet werden. Es ist häufig nützlich, eine CURRENT PUBLISHER-Spalte als Teil des Primärschlüssels zum Replizieren von Tabellen einzurichten, weil dadurch Primärschlüsselkonflikte aufgrund von Aktualisierungen an mehr als einem Standort verhindert werden können.

Um den Publikationseigentümer zu wechseln, müssen Sie zuerst den aktuellen Publikationseigentümer mit der REVOKE PUBLISH-Anweisung löschen, und dann einen neuen Publikationseigentümer mit der GRANT PUBLISH-Anweisung erstellen.

Durch Ausführen dieser Anweisung wird der Wert der PUBLIC.db_publisher-Datenbankoption geändert.

Privilegien

Sie müssen das SET ANY SYSTEM OPTION-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „GRANT ROLE SYS_REPLICATION_ADMIN_ROLE-Anweisung [MobiLink] [SQL Remote]“ auf Seite 896
- „db_publisher-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „GRANT CONSOLIDATE-Anweisung [SQL Remote]“ auf Seite 889
- „REVOKE PUBLISH-Anweisung [SQL Remote]“ auf Seite 1007
- „CREATE SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 730
- „CURRENT PUBLISHER-Spezialwert“ auf Seite 71
- „CREATE REMOTE [MESSAGE] TYPE-Anweisung [SQL Remote]“ auf Seite 694
- „GRANT REMOTE-Anweisung [SQL Remote]“ auf Seite 900
- „Erstellen eines Publikationseigentümers“ [*SQL Remote*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung setzt den Publikationseigentümer der Datenbank auf die Benutzer-ID "JohnS".

```
GRANT PUBLISH TO JohnS;
```

Die folgenden Anweisungen entziehen JohnS die Publikationseigentümerrechte für die Datenbank und erteilen sie IrisM.

```
REVOKE PUBLISH FROM JohnS;  
GRANT PUBLISH TO IrisM;
```

GRANT ROLE SYS_REPLICATION_ADMIN_ROLE- Anweisung [MobiLink] [SQL Remote]

Erteilt die SYS_REPLICATION_ADMIN_ROLE-Systemrolle, die zum Ausführen von Administrationsaufgaben im Zusammenhang mit der Replikation, z.B. Erteilen von Replikationsrollen, Verwalten von Publikationen, Subskriptionen, Synchronisationsbenutzern und Profilen, Verwalten von Nachrichtentypen, Festlegen von Optionen für die Replikation, erforderlich ist.

Syntax

```
GRANT ROLE SYS_REPLICATION_ADMIN_ROLE  
TO grantee [ , ... ]  
| [ WITH NO ADMIN OPTION ]
```

Parameter

- ***grantee*** Die Benutzer-ID eines Benutzers oder der Name einer Rolle. Die Rolle kann entweder eine benutzererweiterte Rolle oder eine benutzerdefinierte Rolle sein. Systemrollen oder Kompatibilitätsrollen können Sie keine Privilegien erteilen.
- **WITH NO ADMIN OPTION-Klausel** Der *grantee* erhält die Rollen bzw. Privilegien, aber nicht die Möglichkeit, sie zu verwalten.

Bemerkungen

SYS_REPLICATION_ADMIN_ROLE stellt die folgenden Systemprivilegien bereit:

- MANAGE REPLICATION
- SET ANY SYSTEM OPTION
- SET ANY PUBLIC OPTION
- SET ANY USER DEFINED OPTION
- SELECT ANY TABLE
- CREATE ANY PROCEDURE
- DROP ANY PROCEDURE
- MANAGE ANY WEB SERVICE
- CREATE ANY TABLE
- DROP ANY TABLE
- SERVER OPERATOR
- MANAGE ANY USER
- MANAGE ROLES
- MANAGE ANY OBJECT PRIVILEGE

Außerdem können Benutzer die folgenden Anweisungen ausführen:

- SYNCHRONIZE PROFILE
- REMOTE RESET
- LOCK FEATURE

Privilegien

Sie müssen das MANAGE ROLES-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Systemrollen im Zusammenhang mit der Replikation“ [[SQL Anywhere Server - Datenbankadministration](#)]
- SQL Remote: „REVOKE ROLE SYS_REPLICATION_ADMIN_ROLE-Anweisung [[MobiLink](#)] [[SQL Remote](#)]“ auf Seite 1013
- SQL Remote: „SYS_RUN_REPLICATION_ROLE-Systemrolle“ [[SQL Remote](#)]
- SQL Remote: „SYS_REPLICATION_ADMIN_ROLE-Systemrolle erteilen“ [[SQL Remote](#)]
- SQL Remote: „SYS_REPLICATION_ADMIN_ROLE-Systemrolle entziehen“ [[SQL Remote](#)]
- SQL Remote: „Modi des SQL Remote-Nachrichtenagenten (dbremote)“ [[SQL Remote](#)]
- SQL Remote: „Sicherheit“ [[SQL Remote](#)]
- MobiLink: „Initiieren einer Synchronisation“ [[MobiLink - Clientadministration](#)]
- MobiLink: Privilegien für dbmlsync [[MobiLink - Clientadministration](#)]
- „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [[MobiLink](#)] [[SQL Remote](#)]“ auf Seite 898

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Im folgenden Beispiel wird die SYS_REPLICATION_ADMIN_ROLE-Rolle dem Berechtigungsempfänger Sam_Singer erteilt:

```
GRANT ROLE SYS_REPLICATION_ADMIN_ROLE  
TO Sam_Singer;
```

GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]

Erteilt die SYS_RUN_REPLICATION_ROLE-Systemrolle, die erforderlich ist, um die Dienstprogramm dbremote für die Replikation bzw. das Dienstprogramm dbmlsync für die Synchronisation ausführen zu können.

Syntax

```
GRANT ROLE SYS_RUN_REPLICATION_ROLE  
TO grantee  
[WITH NO SYSTEM PRIVILEGE INHERITANCE ]
```

Parameter

- **grantee** Die Benutzer-ID eines Benutzers oder der Name einer Rolle. Die Rolle kann entweder eine benutzererweiterte Rolle oder eine benutzerdefinierte Rolle sein. Systemrollen oder Kompatibilitätsrollen können Sie keine Privilegien erteilen.
- **WITH NO SYSTEM PRIVILEGE INHERITANCE** Diese Klausel verhindert, dass die Berechtigungsempfänger der SYS_RUN_REPLICATION_ROLE-Rolle die Systemprivilegien der Rolle erben. Normalerweise gilt: Wenn Sie einem Benutzer oder einer Rolle eine Systemrolle erteilen, sind die Systemprivilegien der Rolle sowohl für die Rolle als auch für ihre Berechtigungsempfänger verfügbar. Wenn Sie die Vererbung der Systemprivilegien für die SYS_RUN_REPLICATION_ROLE-Rolle deaktivieren, sind die Systemprivilegien nur für die Rolle verfügbar, jedoch nicht für ihre Berechtigungsempfänger.

Die WITH NO SYSTEM PRIVILEGE INHERITANCE-Klausel wird aus Gründen der Abwärtskompatibilität für die SYS_RUN_REPLICATION_ROLE-Systemrolle bereitgestellt. Das Deaktivieren der Vererbung von Systemprivilegien für diese Rolle entspricht dem Verhalten der nicht vererbten Berechtigungen in Datenbanken bis Version 12. Das Aktivieren der Vererbung von Systemprivilegien für diese Rolle entspricht dem Verhalten aller Systemrollen und benutzerdefinierten Rollen.

Das Deaktivieren der Vererbung von Systemprivilegien für einen Benutzer ist nur dann sinnvoll, wenn Sie vorhaben, den Benutzer in eine benutzererweiterte Rolle zu konvertieren.

Siehe „[Änderungen des Vererbungsverhaltens für einige Berechtigungen, die zu Rollen geworden sind](#)“ [*SQL Anywhere Server - Datenbankadministration*]

Bemerkungen

SYS_RUN_REPLICATION_ROLE stellt die folgenden Systemprivilegien bereit:

- MONITOR
- BACKUP DATABASE
- DROP CONNECTION
- SELECT ANY TABLE
- SET ANY SYSTEM OPTION
- SET ANY USER DEFINED OPTION

Außerdem können Benutzer die folgende Anweisung ausführen:

- SYNCHRONIZE PROFILE

Wenn in MobiLink die Verbindung über den SQL Anywhere-Synchronisationsclient (das Dienstprogramm dbmlsync) hergestellt wird, gewährt die SYS_RUN_REPLICATION_ROLE-Systemrolle dbmlsync vollen Zugriff auf die Datenbank. Andere Verbindungen, die die gleiche Benutzer-ID verwenden, erhalten keine besonderen Berechtigungen.

Wenn in SQL Remote die Verbindung über den Nachrichtenagenten hergestellt wird, gewährt die SYS_RUN_REPLICATION_ROLE-Systemrolle dem Nachrichtenagenten vollen Zugriff auf die Datenbank, um in den Nachrichten enthaltene Änderungen vorzunehmen. Andere Verbindungen, die die gleiche Benutzer-ID verwenden, erhalten keine besonderen Berechtigungen.

Wenn eine Verbindung mit der Datenbank auf andere Weise hergestellt wird, kann der Benutzer keine dieser Rolle zugeordneten Privilegien ausüben. Der Benutzer kann beispielsweise nicht das SELECT ANY TABLE-Systemprivileg nutzen.

Mit der SYS_RUN_REPLICATION_ROLE-Systemrolle muss der betreffenden Benutzer-ID keine volle DBA-Berechtigung erteilt werden. Dadurch werden Sicherheitsprobleme im Zusammenhang mit dem Verteilen von DBA-Benutzer-IDs und -Kennwörtern vermieden.

Eine SQL Remote-Benutzer-ID mit SYS_RUN_REPLICATION_ROLE-Systemrolle hat beispielsweise keine zusätzlichen Privilegien für eine Verbindung, abgesehen vom Nachrichtenagenten. Selbst wenn die Benutzer-ID und das Kennwort für diesen Benutzer frei verteilt werden, entsteht dadurch kein Sicherheitsproblem. Solange der Benutzer-ID keine Berechtigungen außer CONNECT für die Datenbank zugeordnet sind, kann niemand diese Benutzer-ID verwenden, um auf Daten in der Datenbank zuzugreifen.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Systemrollen im Zusammenhang mit der Replikation“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „REVOKE ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [[MobiLink](#)] [[SQL Remote](#)]“ auf Seite 1014
- MobiLink: „Initiieren einer Synchronisation“ [[MobiLink - Clientadministration](#)]
- SQL Remote: „SYS_RUN_REPLICATION_ROLE-Systemrolle“ [[SQL Remote](#)]
- SQL Remote: „SYS_RUN_REPLICATION_ROLE-Systemrolle erteilen“ [[SQL Remote](#)]
- SQL Remote: „SYS_RUN_REPLICATION_ROLE-Systemrolle entziehen“ [[SQL Remote](#)]
- SQL Remote: „Modi des SQL Remote-Nachrichtenagenten (dbremote)“ [[SQL Remote](#)]
- SQL Remote: „Sicherheit“ [[SQL Remote](#)]
- Privilegien für dbmlsync [[MobiLink - Clientadministration](#)]
- „GRANT ROLE SYS_REPLICATION_ADMIN_ROLE-Anweisung [[MobiLink](#)] [[SQL Remote](#)]“ auf Seite 896

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Im folgenden Beispiel wird die SYS_RUN_REPLICATION_ROLE-Systemrolle dem Berechtigungsempfänger Sam_Singer erteilt:

```
GRANT ROLE SYS_RUN_REPLICATION_ROLE  
TO Sam_Singer;
```

GRANT REMOTE-Anweisung [SQLRemote]

Identifiziert eine Datenbank unmittelbar unter der aktuellen Datenbank in einer SQL Remote-Hierarchie, die Nachrichten von der aktuellen Datenbank empfangen wird. Hierbei handelt es sich um entfernte Benutzer.

Syntax

```
GRANT REMOTE TO userid, ...  
TYPE message-system, ...  
ADDRESS address-string, ...  
[ SEND { EVERY | AT } send-time ]
```

Parameter

userid Die Benutzer-ID des Benutzers, dem das Privileg erteilt werden soll.

message-system Eines der Nachrichtensysteme, die von SQL Remote unterstützt werden. Hierfür muss einer der folgenden Werte verwendet werden:

- FILE
- FTP
- SMTP

address-string Eine Zeichenfolge, die eine gültige Adresse für das angegebene Nachrichtensystem enthält.

send-time Eine Zeichenfolge, die Uhrzeit-Spezifikationen im Format *hh:mm:ss* enthält.

Bemerkungen

In einer SQL Remote-Installation muss das REMOTE-Privileg jeder Datenbank erteilt werden, die Nachrichten von der aktuellen Datenbank empfängt.

Die einzige Ausnahme ist die Datenbank unmittelbar über der aktuellen Datenbank in einer SQL Remote-Hierarchie, der das CONSOLIDATE-Privileg erteilt werden muss.

Der entfernte Benutzer wird anhand eines Nachrichtensystems identifiziert, indem die Methode gekennzeichnet wird, mit der Nachrichten an den konsolidierten Benutzer versendet und von ihm empfangen werden. Der Adressenname muss eine gültige Adresse für das Nachrichtensystem sein, die von Apostrophen umschlossen ist.

Beim FILE-Nachrichtentyp ist die Adresse ein Unterverzeichnis des Verzeichnisses, auf das die SQLREMOTE-Umgebungsvariable zeigt.

Die GRANT REMOTE-Anweisung ist für die entfernte Datenbank erforderlich, um Nachrichten zu empfangen; sie selbst subskribiert aber keine Daten für den entfernten Benutzer. Um die Daten zu subskribieren, muss eine Subskription für die Benutzer-ID für eine der Publikationen in der aktuellen Datenbank erstellt werden, indem das Extraktionsdienstprogramm für Datenbanken oder die CREATE SUBSCRIPTION-Anweisung verwendet werden.

Die optionalen SEND EVERY- und SEND AT-Klauseln legen die Häufigkeit fest, mit der Nachrichten versendet werden. Die Zeichenfolge enthält eine Zeitangabe, welche die Dauer zwischen den Nachrichten (bei SEND EVERY) oder eine Uhrzeit (bei SEND AT) darstellt, zu denen Nachrichten versendet werden. Mit SEND AT werden Nachrichten einmal pro Tag versendet.

Wenn einem Benutzer das REMOTE-Privileg ohne SEND EVERY- oder SEND AT-Klausel erteilt wurde, verarbeitet der Nachrichtenagent Nachrichten und wird anschließend gestoppt. Damit der Nachrichtenagent kontinuierlich läuft, müssen Sie sicherstellen, dass für jeden Benutzer mit REMOTE-Privileg entweder eine SEND AT- oder eine SEND EVERY-Frequenz festgelegt ist.

Man kann davon ausgehen, dass der Nachrichtenagent in vielen entfernten Datenbanken kontinuierlich ausgeführt wird, sodass für alle entfernten Datenbanken eine SEND-Klausel angegeben ist. Eine typische Systemeinrichtung könnte so aussehen, dass Nachrichten an Laptop-Benutzer täglich (SEND AT) und an entfernte Server alle ein oder zwei Stunden (SEND EVERY) versendet werden. Sie sollten aus Gründen der Effizienz wenig verschiedene Zeitangaben wie möglich verwenden.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „REMOTE-Privileg“ [[SQL Remote](#)]
- „REMOTE-Privilegien erteilen“ [[SQL Remote](#)]
- „REVOKE REMOTE-Anweisung [SQL Remote]“ auf Seite 1008
- „Subskriptionen“ [[SQL Remote](#)]
- „Sendefrequenz“ [[SQL Remote](#)]
- „SQL Remote-Nachrichtensysteme“ [[SQL Remote](#)]
- „GRANT PUBLISH-Anweisung [SQL Remote]“ auf Seite 894
- „GRANT CONSOLIDATE-Anweisung [SQL Remote]“ auf Seite 889
- „CREATE SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 730
- „CREATE REMOTE [MESSAGE] TYPE-Anweisung [SQL Remote]“ auf Seite 694

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung erteilt dem Benutzer Sam_Singer das REMOTE-Privileg, mit einem SMTP-E-Mail-System, das alle zwei Stunden Nachrichten an die Adresse "Singer, Samuel" sendet:

```
GRANT REMOTE TO Sam_Singer
TYPE SMTP
ADDRESS 'Singer, Samuel'
SEND EVERY '02:00';
```

GRANT USAGE ON SEQUENCE-Anweisung

Erteilt das Privileg zum Verwenden einer angegebenen Sequenz.

Syntax

```
GRANT USAGE ON SEQUENCE sequence-name
TO userid, ...
```

Bemerkungen

Keine.

Privilegien

Sie müssen das MANAGE ANY OBJECT PRIVILEGE-Privileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Verwendung einer Sequenz zum Generieren von eindeutigen Werten“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Teil der optionalen SQL/2008-Sprachenfunktion T176.

GROUP BY-Klausel

Gruppieren Spalten, Aliasnamen und Funktionen als Teil der SELECT-Anweisung.

Syntax

```
GROUP BY
| group-by-term, ...
| simple-group-by-term, ... WITH ROLLUP
| simple-group-by-term, ... WITH CUBE
| GROUPING SETS ( group-by-term, ... )
```

```
group-by-term :
simple-group-by-term
| ( simple-group-by-term, ... )
| ROLLUP ( simple-group-by-term, ... )
| CUBE ( simple-group-by-term, ... )
```

```
simple-group-by-term :
expression
| ( expression )
| ( )
```

Parameter

GROUPING SETS-Klausel Mit der GROUPING SETS-Klausel können Sie Aggregatvorgänge für mehrere Gruppierungen von einer einzigen Abfragenspezifikation aus durchführen. Jede Menge, die in einer GROUPING SET-Klausel angegeben ist, entspricht einer GROUP BY-Klausel.

Die folgenden beiden Abfragen sind beispielsweise gleichwertig:

```
SELECT a, b, SUM( c ) FROM t
GROUP BY GROUPING SETS ( ( a, b ), ( a ), ( b ), ( ) );
```

```
SELECT a, b, SUM( c ) FROM t
  GROUP BY a, b
UNION ALL
SELECT a, NULL, SUM( c ) FROM t
  GROUP BY a
UNION ALL
SELECT NULL, b, SUM( c ) FROM t
  GROUP BY b
UNION ALL
SELECT NULL, NULL, SUM( c ) FROM t;
```

Ein GROUPING-Ausdruck kann in der Ergebnismenge als NULL dargestellt werden, abhängig von der Gruppierung, zu der die Zeile gehört. Das kann zu Unklarheiten darüber führen, ob NULL das Ergebnis einer anderen Gruppierung ist, oder ob NULL das Ergebnis eines tatsächlichen NULL-Werts in den zugrunde liegenden Daten ist. Um zwischen NULL in den Eingabedaten und NULL-Werten zu unterscheiden, die durch den GROUPING-Operator eingefügt wurden, benutzen Sie die GROUPING-Funktion.

Durch die Angabe eines leeren Klammersatzes () in der GROUPING SETS-Klausel wird eine einzelne Zeile zurückgegeben, die das Gesamttaggregat enthält.

ROLLUP-Klausel Die ROLLUP-Klausel ähnelt der GROUPING SETS-Klausel insofern, als dass sie verwendet werden kann, um mehrere Gruppierungsangaben innerhalb einer einzigen Abfragenspezifikation anzugeben. Eine ROLLUP-Klausel mit *n simple-group-by-term* generiert *n+1* Gruppierungskombinationen, die formuliert werden, indem mit den leeren Klammern begonnen wird und dann *group-by-term* aufeinanderfolgend von links nach rechts angehängt wird.

Die folgenden Anweisungen sind beispielsweise gleichwertig:

```
SELECT a, b, SUM( c ) FROM t
GROUP BY ROLLUP ( a, b );
```

```
SELECT a, b, SUM( c ) FROM t
GROUP BY GROUPING SETS ( ( a, b ), a, ( ) );
```

Sie können eine ROLLUP-Klausel innerhalb einer GROUPING SETS-Klausel verwenden.

CUBE-Klausel Die CUBE-Klausel ähnelt den ROLLUP- und GROUPING SETS-Klauseln insofern, als dass sie verwendet werden kann, um mehrere Gruppierungsangaben innerhalb einer einzigen Abfragenspezifikation anzugeben. Die CUBE-Klausel wird verwendet, um alle möglichen Kombinationen darzustellen, die aus den in der CUBE-Klausel enthaltenen Ausdrücken gemacht werden können.

Die folgenden Anweisungen sind beispielsweise gleichwertig:

```
SELECT a, b, SUM( c ) FROM t
GROUP BY CUBE ( a, b, c );
```

```
SELECT a, b, SUM( c ) FROM t
GROUP BY GROUPING SETS ( ( a, b, c ), ( a, b ), ( a, c ),
( b, c ), a, b, c, ( ) );
```

Sie können eine CUBE-Klausel innerhalb einer GROUPING SETS-Klausel verwenden.

WITH ROLLUP-Klausel Dies ist eine alternative Syntax zur ROLLUP-Klausel und wird für Kompatibilität mit Transact-SQL verfügbar gemacht.

WITH CUBE-Klausel Dies ist eine alternative Syntax zur CUBE-Klausel und wird für Kompatibilität mit Transact-SQL verfügbar gemacht.

Bemerkungen

Wenn Sie die GROUP BY-Klausel verwenden, können Sie nach Ausdrücken (mit ein paar Einschränkungen), Spalten, Aliasnamen oder Funktionen gruppieren. Das Ergebnis der Abfrage enthält eine Zeile für alle eindeutigen Werte (oder Wertmengen) der einzelnen Gruppierungskombinationen.

Die leere GROUP BY-Liste, (), bedeutet, dass die gesamte Eingabe als eine einzige Gruppe behandelt wird. Die folgenden Anweisungen sind beispielsweise gleichwertig:

```
SELECT COUNT(), SUM(Salary) FROM GROUPO.Employees;
```

```
SELECT COUNT(), SUM(Salary) FROM GROUPO.Employees GROUP BY ( );
```

Privilegien

Keine.

Siehe auch

- „Abfrageergebnisse zusammenfassen, gruppieren und sortieren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „SELECT-Anweisung“ auf Seite 1020
- „GROUP BY-Klauselerweiterungen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „GROUPING-Funktion [Aggregat]“ auf Seite 271
- „GROUP BY und der Standard SQL/2008“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- Leere Gruppierungsspezifikation angeben [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „CUBE-Klausel“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „ROLLUP-Klausel“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** GROUP BY ist eine Kernfunktion des SQL/2008-Standards. GROUPING SETS, GROUP BY (), ROLLUP und CUBE sind Teile der optionalen SQL/2008-Sprachenfunktion T431. SQL Anywhere bietet keine Unterstützung für die optionale SQL/2008-Sprachenfunktion T432, "Nested and concatenated GROUPING SETS".

Zu den Erweiterungen des Herstellers für GROUP BY-Klausel gehören folgende:

- WITH ROLLUP
- WITH CUBE
- die Möglichkeit zum Angeben beliebiger Ausdrücke, z.B. von GROUP BY-Begriffen. Im SQL/2008-Standard muss jeder GROUP BY-Begriff eine Spaltenreferenz aus einer Basistabelle in der FROM-Klausel der Abfrage sein.

Beispiele

Das nachfolgende Beispiel gibt eine Ergebnismenge zurück, die die Gesamtanzahl der Bestellungen anzeigt und dann Zwischensummen für die Anzahl der Bestellungen der jeweiligen Jahre (2000 und 2001) liefert.

```
SELECT year ( OrderDate ) Year, Quarter ( OrderDate ) Quarter, count(*)
Orders
FROM GROUPO.SalesOrders
GROUP BY ROLLUP ( Year, Quarter )
ORDER BY Year, Quarter;
```

Wie im vorhergehenden Beispiel für einen ROLLUP-Vorgang gibt das folgende Beispiel einer CUBE-Abfrage eine Ergebnismenge zurück, die die Gesamtanzahl der Bestellungen anzeigt und Zwischensummen für die Anzahl der Bestellungen der jeweiligen Jahre (2000 und 2001) liefert. Anders als bei ROLLUP zeigt diese Abfrage auch Zwischensummen für die Anzahl der Bestellungen in jedem Quartal an (1, 2, 3 und 4).

```
SELECT year (OrderDate) Year, Quarter ( OrderDate ) Quarter, count(*) Orders
FROM GROUPO.SalesOrders
GROUP BY CUBE ( Year, Quarter )
ORDER BY Year, Quarter;
```

Das nachfolgende Beispiel gibt eine Ergebnismenge zurück, die Zwischensummen für die Anzahl der Bestellungen in den Jahren 2000 und 2001 liefert. Der GROUPING SETS-Vorgang ermöglicht Ihnen die Auswahl der Spalten, aus denen die Zwischensummen ermittelt werden sollen, und nicht die Wiedergabe aller Kombinationen von Zwischensummen wie im CUBE-Vorgang.

```
SELECT year (OrderDate) Year, Quarter ( OrderDate ) Quarter, count(*) Orders
FROM GROUPO.SalesOrders
GROUP BY GROUPING SETS ( ( Year, Quarter ), ( Year ) )
ORDER BY Year, Quarter;
```

HELP-Anweisung [Interactive SQL]

Ruft die Hilfe in der Interactive SQL-Umgebung auf.

Syntax

HELP ['*topic*']

Bemerkungen

Die HELP-Anweisung wird verwendet, um auf die Dokumentation von SQL Anywhere zuzugreifen.

Das Hilfethema *topic* kann wahlweise angegeben werden. Sie müssen *topic* in Apostrophe setzen. In einigen Hilfeformaten kann das Thema nicht angegeben werden. In diesem Fall wird ein Link zur allgemeinen Hilfeseite von Interactive SQL angezeigt.

Sie können folgende *topic*-Werte festlegen:

- SQL Anywhere-Fehlercodes
- SQL-Anweisungsschlüsselwörter (wie INSERT, UPDATE, SELECT, CREATE DATABASE)

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

IF-Anweisung

Steuert die bedingte Ausführung von SQL-Anweisungen.

Syntax

```

IF search-condition THEN statement-list
[ ELSEIF { search-condition | operation-type } THEN statement-list ] ...
[ ELSE statement-list ]
{ END IF | ENDIF }

```

Bemerkungen

Die IF-Anweisung ist eine Steueranweisung, mit der Sie bedingt die erste Liste von SQL-Anweisungen ausführen können, deren *search-condition* mit TRUE bewertet wird. Wenn keine *search-condition* mit TRUE bewertet wird, und eine ELSE-Klausel vorhanden ist, wird die *statement-list* in der ELSE-Klausel ausgeführt.

Die Ausführung wird bei der ersten Anweisung nach END IF wieder aufgenommen.

Hinweis

Verwechseln Sie die Syntax des IF-Ausdrucks nicht mit der Syntax der IF-Anweisung.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „BEGIN-Anweisung“ auf Seite 557
- „IF-Ausdrücke“ auf Seite 24
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Suchbedingungen“ auf Seite 42

Standards und Kompatibilität

- **SQL/2008** Die IF-Anweisung ist Teil der optionalen SQL/2008-Sprachenfunktion P002 (Verarbeitungsvollständigkeit). Das ENDIF-Schlüsselwort ist eine Erweiterung des Herstellers.

Beispiel

Die folgende Prozedur veranschaulicht die Verwendung der IF-Anweisung:

```

CREATE PROCEDURE TopCustomer2 (OUT TopCompany CHAR(35), OUT TopValue INT)
BEGIN
    DECLARE err_notfound EXCEPTION
    FOR SQLSTATE '02000';
    DECLARE curThisCust CURSOR FOR
    SELECT CompanyName, CAST(      sum(SalesOrderItems.Quantity *
    Products.UnitPrice) AS INTEGER) VALUE
    FROM Customers
    LEFT OUTER JOIN SalesOrders
    LEFT OUTER JOIN SalesOrderItems
    LEFT OUTER JOIN Products
    GROUP BY CompanyName;
    DECLARE ThisValue INT;

```

```
DECLARE ThisCompany CHAR(35);
SET TopValue = 0;
OPEN curThisCust;
CustomerLoop:
LOOP
    FETCH NEXT curThisCust
    INTO ThisCompany, ThisValue;
    IF SQLSTATE = err_notfound THEN
        LEAVE CustomerLoop;
    END IF;
    IF ThisValue > TopValue THEN
        SET TopValue = ThisValue;
        SET TopCompany = ThisCompany;
    END IF;
END LOOP CustomerLoop;
CLOSE curThisCust;
END;
```

IF-Anweisung [T-SQL]

Steuert die bedingte Ausführung einer SQL-Anweisung, als Alternative zur IF-Anweisung von Watcom-SQL.

Syntax

```
IF expression statement
[ ELSE [ IF expression ] statement ]
```

Bemerkungen

Die Transact-SQL-Bedingungen IF und ELSE steuern beide die Ausführung einer einzelnen SQL-Anweisung oder einer zusammengesetzten Anweisung (zwischen den Schlüsselwörtern BEGIN und END).

Im Vergleich zur IF-Anweisung in Watcom-SQL gibt es in der IF-Anweisung in Transact-SQL kein THEN. In der Transact-SQL-Version gibt es auch die Schlüsselwörter ELSEIF oder END IF nicht.

Privilegien

Keine.

Nebenwirkungen

Keine.

Standards und Kompatibilität

- **SQL/2008** Transact-SQL-Erweiterung.

Beispiel

Das folgende Beispiel veranschaulicht die Verwendung der Transact-SQL-Anweisung IF:

```
IF (SELECT max(ID) FROM sysobjects) < 100
    RETURN
ELSE
    BEGIN
        PRINT 'These are the user-created objects'
```

```

        SELECT name, type, ID
        FROM sysobjects
        WHERE ID < 100
    END

```

Die folgenden zwei Anweisungsblöcke veranschaulichen die Kompatibilität zwischen Transact-SQL und Watcom-SQL:

```

/* Transact-SQL IF statement */
IF @v1 = 0
    PRINT '0'
ELSE IF @v1 = 1
    PRINT '1'
ELSE
    PRINT 'other'
/* Watcom SQL IF statement */
IF v1 = 0 THEN
    PRINT '0'
ELSEIF v1 = 1 THEN
    PRINT '1'
ELSE
    PRINT 'other'
END IF

```

INCLUDE-Anweisung [ESQL]

Schließt eine Datei in ein Quellprogramm mit ein, die vom SQL-Präprozessor erfasst werden soll.

Syntax

INCLUDE *filename*

filename : **SQLDA** | **SQLCA** | *string*

Bemerkungen

Die INCLUDE-Anweisung ähnelt sehr stark der #include-Direktive des C-Präprozessors. Der SQL-Präprozessor liest eine Embedded SQL-Quelldatei und ersetzt alle Embedded SQL-Anweisungen mit Quellcode in C. Wenn eine Datei Informationen enthält, die der SQL-Präprozessor benötigt, beziehen Sie sie mit der Embedded SQL INCLUDE-Anweisung ein.

Zwei Dateinamen werden besonders erkannt: SQLCA und SQLDA. Die folgende Anweisung muss vor allen Embedded SQL-Anweisungen in allen Embedded SQL-Quelldateien erscheinen.

```
EXEC SQL INCLUDE SQLCA;
```

Diese Anweisung muss an einer Position im C-Programm erscheinen, wo statische Variablendeklarationen zulässig sind. Viele Embedded SQL-Anweisungen erfordern Variablen (für den Programmierer nicht sichtbar), die vom SQL-Präprozessor an der Position der SQLCA-Anweisung INCLUDE deklariert werden. Die SQLDA-Datei muss einbezogen werden, wenn SQLDAs verwendet werden.

Privilegien

Keine.

Nebenwirkungen

Keine.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

INPUT-Anweisung [Interactive SQL]

Mit diesem Befehl werden Daten aus einer externen Datei, einer ODBC-Datenquelle, einer Formdatei oder über Tastatureingaben in eine Datenbanktabelle importiert.

Syntax 1 - Aus einer externen Datei oder über Tastatureingaben importieren

INPUT INTO [*owner*.]*table-name* *input-options*

input-options :
[(*column-name*, ...)]
[**BYTE ORDER MARK** { **ON** | **OFF** }]
[**COLUMN WIDTHS** (*integer*, ...)]
[**DELIMITED BY** *string*]
[**ENCODING** *encoding*]
[**ESCAPE CHARACTER** *character*]
[**ESCAPES** { **ON** | **OFF** }]
[**FORMAT** *input-format*]
[**FROM** *filename* | **PROMPT**]
[**NOSTRIP**]
[**SKIP** *integer*]

input-format : **TEXT** | **FIXED**

encoding : *identifier* | *string*

Syntax 2 - Von einer ODBC-Datenquelle importieren

INPUT USING *connection-string*
FROM *source-table-name*
INTO *destination-table-name*
[**CREATE TABLE** { **ON** | **OFF** }]

connection-string :
{ **DRIVER**=*odbc-driver-name*
| **DSN**=*odbc-data-source* } [; { *connection-parameter* = *value* }]

Syntax 3 - Formdateien laden

INPUT INTO [*owner*.]*table-name*
FROM *filename*
FORMAT SHAPEFILE
[**SRID** *srid-number*]
[**ENCODING** *encoding*]

encoding : *identifier* | *string*

Parameter

column-name Listen Sie die Namen der Spalten in der Tabelle in der Reihenfolge auf, die der Reihenfolge der Werte in der Eingabedatei entspricht. Verwenden Sie diesen Parameter, wenn die Reihenfolge der Spalten der Tabelle nicht mit der Reihenfolge der Werte in der Eingabedatei übereinstimmt oder wenn die Eingabedatei weniger Spalten enthält als die Tabelle.

BYTE ORDER MARK-Klausel Verwenden Sie diese Klausel, um anzugeben, ob eine BOM (Byte Order Mark - Bytereihenfolgemarkierung) in der Eingabedatei verarbeitet werden soll.

Die BYTE ORDER MARK-Klausel ist nur relevant, wenn das Einlesen aus Dateien im Textformat erfolgt. Wenn Sie versuchen, die BYTE ORDER MARK-Klausel mit anderen FORMAT-Klauseln als TEXT zu verbinden, wird ein Fehler zurückgegeben.

Die BYTE ORDER MARK-Klausel wird nur verwendet, wenn Dateien gelesen oder geschrieben werden, die mit UTF-8 oder UTF-16 (und ihren Varianten) kodiert sind. Wenn Sie versuchen, die BYTE ORDER MARK-Klausel mit anderen Kodierungen zu verbinden, wird ein Fehler zurückgegeben.

Wenn die ENCODING-Klausel angegeben ist:

- Wenn die BYTE ORDER MARK-Option ON ist und Sie eine UTF-16-Kodierung mit Endian wie UTF-16BE oder UTF-16LE angeben, sucht Interactive SQL nach einer BOM am Beginn der Daten. Wenn eine BOM vorhanden ist, wird sie benutzt, um den Endian der Daten zu prüfen. Wenn Sie den falschen Endian angeben, wird ein Fehler zurückgegeben.
- Wenn die BYTE ORDER MARK-Option ON ist und Sie eine UTF-16-Kodierung ohne expliziten Endian angegeben haben, sucht Interactive SQL nach einer BOM am Beginn der Daten. Wenn eine BOM vorhanden ist, wird sie benutzt, um den Endian der Daten zu bestimmen. Sonst wird der Endian des Betriebssystems angenommen.
- Wenn die BYTE ORDER MARK-Option ON ist und Sie eine UTF-8-Kodierung angegeben haben, sucht Interactive SQL nach einer BOM am Beginn der Daten. Wenn eine BOM vorhanden ist, wird sie ignoriert.

Wenn die ENCODING-Klausel nicht angegeben ist:

- Wenn Sie keine ENCODING-Klausel angeben und die BYTE ORDER MARK-Option ON ist, sucht Interactive SQL nach einer BOM am Beginn der Eingabedaten. Wenn es eine BOM findet, wird die Quellkodierung basierend auf der Kodierung der BOM (UTF-16BE, UTF-16LE oder UTF-8) automatisch gewählt und die BOM wird nicht als Teil der zu ladenden Daten angesehen.
- Wenn Sie keine ENCODING-Klausel angeben und die BYTE ORDER MARK-Option OFF ist oder keine BOM am Beginn der Eingabedaten gefunden wird, verwendet Interactive SQL die Kodierung auf dem Clientcomputer.

COLUMN WIDTHS-Klausel Die COLUMN WIDTHS-Klausel legt die Breite der Spalten in der Eingabedatei fest und kann nur im FIXED-Format angegeben werden. Wenn COLUMN WIDTHS nicht angegeben ist, wird die Breite von den Datenbank-Spaltentypen bestimmt.

CREATE TABLE-Klausel Verwenden Sie die CREATE TABLE-Klausel, um anzugeben, ob die Zieltabelle erstellt werden soll, wenn sie nicht existiert. Der Standardwert ist ON.

DELIMITED BY-Klausel Mit der DELIMITED BY-Klausel können Sie eine Zeichenfolge angeben, die als Begrenzer zwischen Spaltenwerten im TEXT-Eingabeformat verwendet werden soll. Das Standardtrennzeichen ist ein Komma.

ENCODING-Klausel Das Argument *encoding* gibt die Kodierung an, mit der die Datei gelesen werden soll. Die ENCODING-Klausel kann nur in den Formaten TEXT und SHAPEFILE verwendet werden.

Wenn Sie Interactive SQL ausführen, wird die beim Importieren der Daten verwendete Kodierung in der folgenden Reihenfolge ermittelt:

- Die durch die ENCODING-Klausel angegebene Kodierung (sofern diese Klausel angegeben ist).
- Die durch die Option "default_isql_encoding" angegebene Kodierung (sofern diese Option festgelegt wurde).
- Wenn die Datei hat eine BOM (Byte-Order Mark) hat, wird die entsprechenden Unicodekodierung verwendet.
- Die Standardkodierung für den Computer, den Sie verwenden. Auf Computern mit englischem Windows ist die Standardkodierung 1252 oder 850.

Wenn die Eingabedatei mit der OUTPUT-Anweisung erstellt und eine Kodierung angegeben wurde, muss dieselbe ENCODING-Klausel in der INPUT-Anweisung angegeben werden.

ESCAPE CHARACTER-Klausel Das standardmäßige Escape-Zeichen für hexadezimale Codes ist ein Backslash (\). Beispielsweise ist \x0A das Zeilenvorschubzeichen.

Die Zeilenendmarke kann als \n eingegeben werden. Andere Zeichen können mit hexadezimalen ASCII-Codes angegeben werden, wie etwa \x09 für das Tabulatorzeichen. Eine Sequenz von zwei Backslashes (\\) wird als ein einzelner Backslash interpretiert. Ein Backslash gefolgt von einem beliebigen Zeichen außer n, x, X oder \ wird als zwei separate Zeichen interpretiert. Zum Beispiel wird \q als Backslash mit dem Buchstaben q interpretiert.

Das Escape-Zeichen kann mit der ESCAPE CHARACTER-Klausel verändert werden. Wenn Sie beispielsweise das Ausrufezeichen als Escapezeichen verwenden möchten, geben Sie Folgendes ein:

```
... ESCAPE CHARACTER '!'
```

ESCAPES-Klausel Wenn ESCAPES aktiviert ist (Standardwert), werden die Zeichen, die auf das Escapezeichen folgen, vom Datenbankserver als Sonderzeichen interpretiert. Wenn ESCAPES deaktiviert ist, werden die Zeichen genauso gelesen, wie sie in der Quelle erscheinen.

FORMAT-Klausel Verwenden Sie die FORMAT-Klausel, um das Format der Eingabedatei anzugeben.

Wenn die FORMAT-Klausel nicht angegeben ist, legt die input_format-Option das Format fest. Die Standardeinstellung ist TEXT. Sie können auch das Eingabeformat in Interactive SQL angeben, indem Sie **Extras » Optionen » Import/Export** auswählen und im Feld **Standard-Importformat** entweder **Text** oder **Feste Breite** angeben.

Die unterstützten Formate sind folgende:

- **TEXT** Eingabezeilen werden als Zeichen vorausgesetzt, eine Zeile pro Ausgabezeile, wobei die Spaltenwerte durch Trennzeichen voneinander getrennt werden. Alphabetische Zeichenfolgen können in Apostrophe oder Anführungszeichen eingeschlossen werden. Zeichenfolgen, die Trennzeichen enthalten, müssen in Apostrophe oder Anführungszeichen gesetzt werden. Wenn die Zeichenfolge selbst Apostrophe oder Anführungszeichen enthält, müssen Sie die Anführungszeichen oder Apostrophe verdoppeln, um sie innerhalb der Zeichenfolge zu verwenden. Sie können auch die DELIMITED BY-Klausel verwenden, um ein anderes Trennzeichen als das standardmäßige Komma anzugeben.

Drei weitere spezielle Sequenzen werden ebenfalls erkannt. Die beiden Zeichen \n stellen eine Zeilenendmarke dar, \\ stellt einen einzelnen \ und die Sequenz \xDD stellt das Zeichen mit dem Hexadezimalcode DD dar.

Ausgelassene Werte werden als NULL behandelt. Wenn der Wert in dieser Position nicht NULL sein kann, wird eine 0 in numerische Spalten und eine leere Zeichenfolge in Zeichenspalten eingefügt.

- **FIXED** Eingabezeilen sind im festen Format. Die Spaltenbreite kann mit der COLUMN WIDTHS-Klausel angegeben werden. Wenn sie nicht angegeben werden, müssen die Spaltenbreiten in der Datei der maximalen Anzahl der Zeichen entsprechen, die für die entsprechende Datenbankspalte zulässig sind.

Das FIXED-Format kann nicht für Binärspalten verwendet werden, die eingebettete Sequenzen von Zeilenendmarken und Datei-Ende-Zeichen enthalten.

- **SHAPEFILE** Die Eingabe erfolgt in Form einer ESRI-Formdatei. Anders als die LOAD-Anweisung muss sich die Formdatei beim Laden von Formdateien mit der INPUT-Anweisung auf dem Clientcomputer befinden. Die zugeordneten .shx- und .dbf-Dateien müssen sich in demselben Verzeichnis befinden wie die .shp-Datei und denselben Basisdateinamen aufweisen.

Wenn beim Laden einer Formdatei keine Kodierung angegeben ist, lautet der Standardwert ISO-8859-1.

Verwenden Sie die SRID-Klausel, um eine SRID anzugeben, die den Geometrien zugeordnet werden soll. Wenn Sie keine SRID angeben, wird standardmäßig SRID 0 verwendet. Idealerweise sollten Sie dieselbe SRID wie diejenige angeben, die in der Projektdatei (.prj) für die Formdatei angegeben ist. Wenn diese SRID nicht verfügbar ist, verwenden Sie eine gleichwertige. SQL Anywhere bietet Tausende SRIDs, die Sie mithilfe der sa_install_ feature-Systemprozedur der Datenbank hinzufügen können.

Wenn Sie andere Formate verwenden möchten, z.B. DBASE II, DBASE III, FoxPro, Lotus 123 oder Excel 97, müssen Sie INPUT USING verwenden.

FROM filename-Klausel Der *filename* kann mit oder ohne Apostrophe angegeben werden. Wenn die Zeichenfolge in Apostroph steht, unterliegt sie denselben Formatierungsanforderungen wie andere SQL-Zeichenfolgen.

Beim Angeben von Verzeichnispfaden muss das Backslashzeichen (\) durch zwei Backslashes dargestellt werden.

Der Speicherort eines relativen *filename* wird wie folgt ermittelt:

- Wenn die INPUT-Anweisung direkt in Interactive SQL ausgeführt wird, berechnet sich der Pfad von *filename* relativ zum Verzeichnis, in dem Interactive SQL läuft. Beispiel: Angenommen, Sie öffnen Interactive SQL aus dem Verzeichnis *c:\work* und führen folgende Anweisung aus:

```
INPUT INTO GROUPO.Employees  
FROM 'inputs\inputfile.dat';
```

Interactive SQL sucht nach *c:\work\inputs\inputfile.dat*.

- Wenn die INPUT-Anweisung in einer *.sql* liegt, versucht Interactive SQL erst, den Pfad zu *filename* relativ zum Speicherort der Datei aufzulösen. Wenn dies nicht gelingt, sucht Interactive SQL nach *filename* in einem Pfad, der relativ zu dem Verzeichnis liegt, in dem Interactive SQL läuft.

Beispiel: Angenommen, Sie haben eine Datei *c:\homework\inputs.sql*, die folgende Anweisung enthält:

```
INPUT INTO GROUPO.Employees  
FROM 'inputs\inputfile.dat';
```

Interactive SQL sucht erst nach *inputfile.dat* in *c:\homework\inputs*. Wenn Interactive SQL *inputfile.dat* an diesem Speicherort nicht findet, sucht Interactive SQL in dem Verzeichnis, in dem Interactive SQL läuft.

FROM source-table-name-Klausel Der *source-table-name*-Parameter ist eine Zeichenfolge in Anführungszeichen, die den Namen der Tabelle in der Quelldatenbank enthält. Der Name kann im Format *database-name.owner.table-name*, *owner.table-name* oder *table-name* angegeben werden. Verwenden Sie einen Punkt, um die Komponenten voneinander zu trennen, auch wenn dies nicht das systemeigene Trennzeichen in der Quelldatenbank ist. Wenn die Quelldatenbank einen Datenbanknamen, aber keinen Eigentüternamen verlangt, ist das Format von *source-table-name* *database..table* (in diesem Fall ist der Eigentümername leer). Setzen Sie die Namen im Parameter nicht in Anführungszeichen. Beispiel: Verwenden Sie nicht 'dba."my-table"', sondern 'dba.my-table'.

INTO-Klausel Der Name der Tabelle, in der die Daten einzugeben sind.

PROMPT-Klausel Die PROMPT-Klausel ermöglicht dem Benutzer, Werte für jede Spalte in eine Zeile einzugeben. Wenn sie im Befehlsfenstermodus ausgeführt wird, erscheint ein Fenster, in das der Benutzer die Werte für eine neue Zeile eingeben kann. Wenn Sie Interactive SQL von der Befehlszeile ausführen, fordert Interactive SQL Sie auf, den Wert für jede Spalte in die Befehlszeile einzugeben.

NOSTRIP-Klausel Normalerweise werden für das TEXT-Eingabeformat nachgestellte Leerzeichen aus nicht mit Anführungszeichen versehenen Zeichenfolgen entfernt, bevor der Wert eingefügt wird. NOSTRIP kann verwendet werden, um die Entfernung der nachgestellten Leerzeichen zu unterdrücken. Nachgestellte Leerzeichen werden nicht aus mit Anführungszeichen versehenen Zeichenfolgen entfernt, unabhängig davon, ob die Option verwendet wird oder nicht. Führende Leerzeichen werden unabhängig von der Einstellung der Option NOSTRIP aus nicht mit Anführungszeichen versehenen Zeichenfolgen entfernt.

SKIP-Klausel Beim Einfügen von Zeilen aus einer TEXT-Datei lässt die SKIP-Klausel die angegebene Anzahl an Zeilen weg, beginnend am Anfang der Datei. Die angegebene Zahl muss eine nicht-negative Ganzzahl sein. Die SKIP-Klausel QUOTE gilt nur für das TEXT-Format.

Wenn die angegebene Zeilenanzahl die Anzahl der Zeilen in der Datei überschreitet, fügt die INPUT-Anweisung keine Daten ein und es wird kein Fehler zurückgegeben.

USING-Klausel Die USING-Klausel gibt Daten von einer ODBC-Datenquelle ein. Sie können entweder den ODBC-Datenquellennamen mit der DSN-Option oder den ODBC-Treibernamen und den Verbindungsparameter mit der DRIVER-Option angeben. Der Parameter *connection-parameter* ist eine Liste von datenbankspezifischen Verbindungsparametern.

Der Parameter *odbc-data-source* ist der Name eines Benutzers oder der ODBC-Datenquellennamen. Beispiel: *odbc-data-source* ist für die SQL Anywhere-Beispieldatenbank "SQL Anywhere 16 Demo".

Der Parameter *odbc-driver-name* ist der ODBC-Treibername. Für eine SQL Anywhere-Datenbank ist *odbc-driver-name* SQL Anywhere 16. Für eine UltraLite-Datenbank ist *odbc-driver-name* UltraLite 16.

Bemerkungen

Mit der INPUT-Anweisung können effiziente Einfügungen großer Datenmengen in die genannte Datenbanktabelle vorgenommen werden. Eingabezeilen werden entweder vom Benutzer über das Eingabefenster eingelesen (wenn PROMPT angegeben ist), oder von einer Datei (wenn ein FROM *filename* angegeben ist). Wenn keine der beiden Angaben festgelegt ist, wird die Eingabe aus der SQL-Skriptdatei gelesen, die die INPUT-Anweisung enthält. In Interactive SQL kann dies sogar direkt im Fensterausschnitt **SQL-Anweisungen** erfolgen. In diesem Fall wird die Eingabe mit einer Zeile beendet, die lediglich die Zeichenfolge END enthält.

Wenn die Eingabe direkt aus dem Fensterausschnitt **SQL-Anweisungen** gelesen wird, müssen Sie ein Semikolon vor den Werten angeben, damit die Datensätze am Ende der INPUT-Anweisung eingefügt werden. Beispiel:

```
INPUT INTO Owner.TableName;  
value1, value2, value3  
value1, value2, value3  
value1, value2, value3  
value1, value2, value3  
END
```

Das END-Schlüsselwort (nicht ein Semikolon) schließt Daten bei INPUT-Anweisungen ab, die keine Datei benennen und nicht das PROMPT-Schlüsselwort enthalten.

Wenn eine Spaltenliste angegeben wird, ist die Folge, dass die Daten in die angegebenen Spalten der benannten Tabelle eingefügt werden. Standardmäßig geht die INPUT-Anweisung davon aus, dass Spaltenwerte in der Eingabedatei in derselben Reihenfolge erscheinen wie in der Datenbanktabellendefinition. Wenn die Spaltenreihenfolge in der Tabelle anders ist, müssen Sie den Parameter *column-name* verwenden, um die Tabellenspalten in derselben Reihenfolge aufzulisten wie die Spaltenwerte in der Eingabedatei.

Standardmäßig stoppt die INPUT-Anweisung, wenn sie versucht, eine Zeile einzufügen, die einen Fehler verursacht. Es gibt verschiedene Möglichkeiten, Fehler zu behandeln, indem die Optionen *on_error* und *conversion_error* eingestellt werden.

Interactive SQL gibt eine Warnung auf der Registerkarte **Meldungen** aus, wenn Zeichenfolgenwerte bei der Eingabe mit INPUT gekürzt werden. Fehlende Werte für NOT NULL-Spalten werden für numerische Typen auf Null gesetzt und für nicht-numerische Typen auf die leere Zeichenfolge. Wenn INPUT versucht, einen NULL-Zeile einzufügen, enthält die Eingabedatei eine leere Zeile.

Da die INPUT-Anweisung ein Interactive SQL-Befehl ist, kann sie nicht in Anweisungen verwendet werden, die vom Server ausgeführt werden.

Privilegien

Sie müssen Eigentümer der Tabelle sein, das INSERT ANY TABLE-Systemprivileg haben oder das INSERT-Privileg für die Tabelle haben. Außerdem müssen Sie das SELECT ANY TABLE-Systemprivileg oder das SELECT-Privileg für die Tabelle haben.

Nebenwirkungen

Keine.

Siehe auch

- „Neue Zeilen in Interactive SQL-Ergebnismengen einfügen“ [*SQL Anywhere Server - Datenbankadministration*]
- „Daten mit der INPUT-Anweisung importieren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Datenimport“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Importieren von ASCII-Daten in eine neue UltraLite-Datenbank“ [*UltraLite - Datenbankverwaltung*]
- „OUTPUT-Anweisung [Interactive SQL]“ auf Seite 968
- „INSERT-Anweisung“ auf Seite 917
- „SET OPTION-Anweisung [Interactive SQL]“ auf Seite 1043
- „LOAD TABLE-Anweisung“ auf Seite 931
- „Interactive SQL“ [*SQL Anywhere Server - Datenbankadministration*]
- „Dienstprogramm zum Entladen (dbunload)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Unterstützung von ESRI-Formdateien“ [*SQL Anywhere Server - Unterstützung für räumliche Daten*]
- „Unterstützte Zeichensätze“ [*SQL Anywhere Server - Datenbankadministration*]
- „Zulässige Anweisungen in Prozeduren, Triggern, Ereignissen und Batches“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Praktische Einführung: Mit den räumlichen Funktionen experimentieren“ [*SQL Anywhere Server - Unterstützung für räumliche Daten*]
- „input_format-Option [Interactive SQL]“ [*SQL Anywhere Server - Datenbankadministration*]
- „sa_install_feature-Systemprozedur“ auf Seite 1245
- „default_isql_encoding-Option [Interactive SQL]“ [*SQL Anywhere Server - Datenbankadministration*]
- „Neue Zeilen in Interactive SQL-Ergebnismengen einfügen“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel werden Daten aus einer fiktiven TEXT-Datei, *new_emp.inp*, in die Tabelle "Employees" importiert:

```
INPUT INTO GROUP0.Employees
FROM new_emp.inp
FORMAT TEXT;
```

In diesem Beispiel wird die Tabelle "ulTest" in eine Tabelle namens saTest kopiert. Die ulTest-Tabelle befindet sich in einer UltraLite-Datenbank in der Datei *C:\test\myULDatabase.udb* und die saTest-Tabelle wird in *demo.db* erstellt:

```
INPUT USING 'driver=UltraLite 16;dbf=C:\\test\\myULDatabase.udb'
FROM "ulTest" INTO "saTest";
```

In diesem Beispiel wird die Formdatei *myshapefile.shp* in die Tabelle "myTable" geladen und den Geometrien in der Formdatei wird die SRID 4269 zugeordnet.

```
INPUT INTO myTable
FROM 'myshapefile.shp'
FORMAT SHAPEFILE SRID 4269
```

In diesem Beispiel wird eine neue Zeile zur Tabelle "Products" hinzugefügt und der Benutzer wird aufgefordert, die Werte für die einzelnen Spalten einzugeben.

```
INPUT INTO GROUP0.Products PROMPT;
```

In diesem Beispiel werden Daten aus der Datei *c:\temp\input.dat* in die Tabelle "Employees" geladen. Beachten Sie die verdoppelten Backslashes.

```
INPUT INTO GROUP0.Employees
FROM 'c:\\temp\\input.dat';
```

Im folgenden Beispiel wird eine Tabelle namens myInventory erstellt und Daten werden aus der Datei *stock.txt* importiert, die die Daten enthält, jedoch in anderer Spaltenreihenfolge als die Tabellendefinition. Um die nicht übereinstimmende Reihenfolge zu korrigieren, wird die richtige Spaltenreihenfolge für den Import durch den Parameter *column-name* angegeben, der in der INPUT-Anweisung auf den Tabellennamen folgt. Mit **(item, Quantity)** wird Interactive SQL also angewiesen, den ersten Spaltenwert aus der Eingabedatei in die zweite Spalte der Tabelle einzufügen und anschließend den zweiten Spaltenwert aus der Eingabedatei in die erste Spalte der Tabelle einzufügen.

```
CREATE TABLE myInventory (
Quantity INTEGER,
item VARCHAR(60)
);
INPUT INTO myInventory (item, Quantity)
FROM stock.txt;
```

INSERT-Anweisung

Fügt eine einzelne Zeile (Syntax 1) oder eine Reihe von Zeilen von einer anderen Stelle in der Datenbank (Syntax 2) in eine Tabelle ein.

Syntax 1

```
INSERT [ INTO ] [ owner.]table-name [ ( column-name, ... ) ]
[ ON EXISTING {
ERROR
| SKIP
| UPDATE [ DEFAULTS { ON | OFF } ]
} ]
{ DEFAULT VALUES
```

```
| VALUES row-value-constructor }  
[ OPTION( query-hint [, ... ] ) ]
```

Syntax 2

```
INSERT [ INTO ] [ owner.]table-name [ ( [ column-name [, ... ] ] ) ]  
[ ON EXISTING {  
    ERROR  
    | SKIP  
    | UPDATE [ DEFAULTS { ON | OFF } ]  
} ]  
[ WITH AUTO NAME ]  
select-statement  
[ OPTION( query-hint [, ... ] ) ]
```

query-hint :

```
MATERIALIZED VIEW OPTIMIZATION option-value  
| FORCE OPTIMIZATION  
| FORCE NO OPTIMIZATION  
| option-name = option-value
```

option-name :
identifier

option-value :
hostvar (Bezeichner zulässig)
| *string*
| *identifier*
| *number*

insert-expression :
expression | **DEFAULT**

row-value-constructor :
([*insert-expression* [, ...]]) [, [*insert-expression* [, ...]]) ...]

Parameter

VALUES-Klausel Verwenden Sie die VALUES-Klausel, um die einzufügenden Werte anzugeben. Wenn Sie die für die Spalten definierten Standardwerte einfügen möchten, geben Sie DEFAULT VALUES an. Sie können auch VALUES () angeben. Dies ist DEFAULT VALUES gleichwertig. Die VALUES-Klausel unterstützt auch Zeilenwertkonstruktoren, damit Sie mehrere Zeilen von Werten mit einer einzigen Anweisung einfügen können. Anzahl und Reihenfolge der *insert-expression*-Werte in jedem *row-value-constructor* müssen der in der INTO-Klausel angegebenen Spaltenliste entsprechen. Wenn keine Spaltenliste angegeben ist, wird angenommen, dass es sich um die vollständige sortierte Spaltenliste für die Tabelle handelt. Wenn Sie eine leere Spaltenliste () angeben, muss jede der Spalten in der Tabelle einen Standardwert haben. Wenn Sie mehrere Zeilenwertkonstruktoren angeben, müssen sie durch Kommata getrennt werden.

Wenn beim Einfügen von einer der Zeilen ein Fehler auftritt, werden alle Änderungen zurückgesetzt.

WITH AUTO NAME-Klausel WITH AUTO NAME gilt nur für Syntax 2. Wenn Sie WITH AUTO NAME angeben, bestimmen die Namen der Elemente im Abfrageblock, in welche Spalte die Daten gehören. Die Elemente des Abfrageblocks sollten entweder Spaltenreferenzen oder Aliasausdrücke sein.

Zielspalten, die im Abfrageblock nicht definiert sind, erhalten ihren Standardwert. Dies ist sinnvoll, wenn die Anzahl von Spalten in der Zieltabelle sehr groß ist.

Die INSERT-Anweisung gibt einen Fehler zurück, wenn die WITH AUTO NAME-Klausel angegeben ist und der Abfrageblock Spalten enthält, die nicht mit Spalten in der Zieltabelle übereinstimmen. Das Ausführen der folgenden Anweisung gibt beispielsweise einen Fehler zurück, wonach die operation-Spalte aus dem SELECT-Abfrageblock nicht in der Tabelle gefunden wurde:

```
CREATE TABLE MyTable5(
    pk INT PRIMARY KEY DEFAULT AUTOINCREMENT,
    TableName CHAR(128),
    TableNameLen INT );
INSERT
INTO MyTable5 WITH AUTO NAME
SELECT length(t.table_name) AS TableNameLen,
       t.table_name AS TableName, 1 as operation
FROM SYS.SYSTAB t
WHERE table_id <= 10;
```

ON EXISTING-Klausel Die ON EXISTING-Klausel der INSERT-Anweisung gilt für beide Syntaxen. Bestehende Zeilen werden in einer Tabelle basierend auf einer Primärschlüsselschleife mit neuen Spaltenwerten aktualisiert. Diese Klausel kann nur bei Tabellen angewendet werden, die einen Primärschlüssel besitzen. Der Versuch, diese Klausel bei Tabellen ohne Primärschlüssel anzuwenden, generiert einen Syntaxfehler. Sie können keine Werte in eine Proxytabelle mit der ON EXISTING-Klausel einlesen.

Hinweis

Wenn es voraussichtlich viele Zeilen gibt, die der ON EXISTING-Bedingung entsprechen, kann es sinnvoll sein, stattdessen die MERGE-Anweisung zu verwenden. Die MERGE-Anweisung bietet mehr Kontrolle über die Aktionen, die Sie bei übereinstimmenden Zeilen durchführen können. Sie stellt auch eine ausgefeiltere Syntax zur Verfügung um festzulegen, was eine Übereinstimmung ausmacht.

Wenn Sie die ON EXISTING-Klausel angeben, führt der Datenbankserver eine Primärschlüsselsuche für jede Eingabezeile durch. Wenn die entsprechende Zeile in der Tabelle nicht bereits existiert, wird die neue Zeile eingefügt. Für Zeilen, die in der Tabelle bereits existieren, können Sie sich dafür entscheiden, die Eingabezeile stillschweigend zu ignorieren (SKIP), eine Fehlermeldung aufgrund doppelter Schlüsselwerte auszugeben (ERROR) oder die alten Werte mit den Werte aus der Eingabezeile zu aktualisieren (UPDATE). Wenn Sie nicht ON EXISTING definieren, führt die Einfügung von Zeilen in eine Tabelle, in der diese Zeilen bereits existieren, standardmäßig zur Ausgabe der Fehlermeldung über doppelte Schlüsselwerte, was der Verwendung der ON EXISTING ERROR-Klausel entspricht. Übersprungene Zeilen werden in die @@rowcount-Variable einbezogen.

Wenn Sie die ON EXISTING UPDATE-Klausel verwenden, wird die Eingabezeile mit der gespeicherten Zeile verglichen. Spaltenwerte, die in der Eingabezeile explizit angeführt sind, ersetzen die entsprechenden Spaltenwerte in der gespeicherten Zeile. Spaltenwerte, die nicht explizit in der Eingabezeile angegeben sind, führen hingegen zu keiner Änderung der entsprechenden Werte in der gespeicherten Zeile. Ausgenommen davon sind Spalten mit Standardwerten. Wenn Sie die ON EXISTING UPDATE-Klausel mit Spalten verwenden, die Standardwerte umfassen (einschließlich DEFAULT AUTOINCREMENT-Spalten), können Sie zusätzlich angeben, ob die Spaltenwerte mit den Standardwerten aktualisiert werden sollen, indem Sie ON EXISTING UPDATE DEFAULTS ON angeben, oder ob der Spaltenwert so wie er ist belassen werden soll, indem Sie ON EXISTING UPDATE

DEFAULTS OFF angeben. Wenn keine Angaben gemacht werden, ist das Standardverhalten ON EXISTING UPDATE DEFAULTS OFF.

Hinweis

DEFAULTS ON- und DEFAULTS OFF-Parameter wirken sich nicht auf Werte in DEFAULT TIMESTAMP, DEFAULT UTC TIMESTAMP oder DEFAULT LAST USER aus. Bei diesen Spalten wird der Wert in der gespeicherten Zeile immer während des UPDATE aktualisiert.

Wenn Sie die Klauseln ON EXISTING SKIP und ON EXISTING ERROR verwenden und die Tabelle DEFAULT-Spalten enthält, berechnet der Server die Standardwerte auch für Zeilen, die bereits existieren. Standardwerte wie AUTOINCREMENT bewirken daher sogar für übersprungene Zeilen Nebeneffekte. Im Fall von AUTOINCREMENT führt dies zu übersprungenen Werten in der AUTOINCREMENT-Sequenz. Mit dem folgenden Beispiel wird dies veranschaulicht:

```
CREATE TABLE t1( c1 INT PRIMARY KEY, c2 INT DEFAULT AUTOINCREMENT );
INSERT INTO t1( c1 ) ON EXISTING SKIP VALUES( 20 );
INSERT INTO t1( c1 ) ON EXISTING SKIP VALUES( 20 );
INSERT INTO t1( c1 ) ON EXISTING SKIP VALUES( 30 );
```

Die in der ersten INSERT-Anweisung definierte Zeile wird eingefügt und c2 wird auf 1 gesetzt. Die in der zweiten INSERT-Anweisung definierte Zeile wird übersprungen, weil sie mit der bestehenden Zeile übereinstimmt. Der autoincrement-Zähler allerdings zählt auf 2 hoch (ohne Auswirkung auf die existierende Zeile). Die in der dritten INSERT-Anweisung definierte Zeile wird eingefügt und der Wert von c2 wird auf 3 gesetzt. Daher sind die im vorigen Beispiel eingefügten Werte die folgenden:

```
20,1
30,3
```

Vorsicht

Wenn Sie SQL Remote verwenden, dürfen Sie DEFAULT LAST USER-Spalten nicht replizieren. Wenn die Spalte repliziert wird, wird der Spaltenwert auf den SQL Remote-Benutzer, und nicht auf den replizierten Wert gesetzt.

OPTION-Klausel Verwenden Sie diese Klausel, um Hints für das Ausführen der Anweisung anzugeben. Die folgenden Hints werden unterstützt:

- MATERIALIZED VIEW OPTIMIZATION *option-value*
- FORCE OPTIMIZATION
- FORCE NO OPTIMIZATION
- *option-name* = *option-value*. Die Spezifikation OPTION(*isolation_level* = ...) im Abfragetext hebt alle anderen Festlegungen der Isolationsstufe für eine Abfrage auf.

Bemerkungen

Die INSERT-Anweisung wird verwendet, um Zeilen in eine Datenbanktabelle einzufügen.

Da Textindizes und materialisierte Ansichten von Änderungen an den darunterliegenden Tabellendaten betroffen sind, ist es sinnvoll, abhängige Textindizes oder materialisierte Ansichten vor dem Massenimport von Daten (LOAD TABLE, INSERT, MERGE) in die Basistabellen zu kürzen.

- **Syntax 1** Fügt eine einzelne Zeile oder mehrere Zeilen mit den angegebenen Ausdruckswerten für die Spalte ein. Falls mehrere Zeilen angegeben sind, werden sie durch zusätzliche Klammern getrennt. Das Schlüsselwort DEFAULT kann verwendet werden, um den Standardwert für die Spalte einzufügen. Wenn die optionale Liste der Spaltennamen angegeben ist, werden die Werte einzeln in die angegebenen Spalten eingefügt. Wird keine Liste für die Spaltennamen definiert, werden die Werte in der Reihenfolge in die Tabelle eingefügt, in der sie generiert werden (dieselbe Reihenfolge wie mit SELECT *). Die Zeile wird an einer beliebigen Stelle in die Tabelle eingefügt. (In relationalen Datenbanken werden Tabellen nicht sortiert.)
- **Syntax 2** Führt eine Masseneinfügung in eine Tabelle mit den Ergebnismengen einer vollständigen allgemeinen SELECT-Anweisung aus. Einfügungen werden in einer beliebigen Reihenfolge vorgenommen, es sei denn, die SELECT-Anweisung enthält eine ORDER BY-Klausel.

Die Spalten aus der Auswahlliste werden ordinal den in der Spaltenliste angegebenen Spalten oder sequentiell in der Reihenfolge, in der die Spalten erstellt wurden, zugeordnet.

INSERT-Anweisungen können in Ansichten ausgeführt werden, wenn die Abfragespezifikation, die die Ansicht definiert, aktualisierbar ist.

In Tabellen eingefügte Zeichenfolgen werden immer in der gleichen Schreibung (groß oder klein) gespeichert, in der sie eingegeben wurden, unabhängig davon, ob die Datenbank die Groß- und Kleinschreibung berücksichtigt. Dementsprechend wird die Zeichenfolge "Value", die in eine Tabelle eingefügt wurde, in der Datenbank immer mit einem großen V und dem Rest in Kleinbuchstaben gespeichert. SELECT-Anweisungen geben die Zeichenfolge als Value zurück. Wenn die Datenbank die Groß-/Kleinschreibung jedoch nicht berücksichtigt, ist Value dasselbe wie value, VALUE usw. Wenn außerdem ein Primärschlüssel für eine Spalte einen Eintrag mit Value enthält, wird ein INSERT von value zurückgewiesen, da der Primärschlüssel dann nicht eindeutig wäre.

Das Einfügen großer Datenmengen mit der INSERT-Anweisung aktualisiert auch die Spaltenstatistiken.

Hinweis

Wenn Sie mehrere Zeilen in eine Tabelle einfügen möchten, ist es effektiver, einen Cursor zu deklarieren, wo dies möglich ist, als viele getrennte INSERT-Anweisungen auszuführen. Bevor Sie Daten einfügen, können Sie den Prozentsatz des Speicherplatzes festlegen, der für spätere Updates frei bleiben soll.

Privilegien

Sie müssen Eigentümer der Tabelle sein, das INSERT ANY TABLE-Privileg haben oder das INSERT-Privileg für die Tabelle haben. Wenn die ON EXISTING UPDATE-Klausel angegeben wird, müssen Sie außerdem das UPDATE ANY TABLE-Systemprivileg oder das UPDATE-Privileg für die Tabelle haben.

Nebenwirkungen

Keine.

Siehe auch

- Globale Variable @@rowcount auf Seite 90
- „Datenimport“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Reguläre Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „ALTER TABLE-Anweisung“ auf Seite 516
- „MERGE-Anweisung“ auf Seite 952
- „INPUT-Anweisung [Interactive SQL]“ auf Seite 910
- „LOAD TABLE-Anweisung“ auf Seite 931
- „UPDATE-Anweisung“ auf Seite 1109
- „SELECT-Anweisung“ auf Seite 1020
- „TRUNCATE-Anweisung“ auf Seite 1089
- „TRUNCATE TEXT INDEX-Anweisung“ auf Seite 1091
- „DELETE-Anweisung“ auf Seite 791
- „PUT-Anweisung [ESQL]“ auf Seite 981
- „Zugriff auf Daten auf Clientcomputern“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Daten mit INSERT hinzufügen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- OPTION-Klausel, SELECT-Anweisung auf Seite 1027

Standards und Kompatibilität

- **SQL/2008** Die INSERT-Anweisung ist eine Kernfunktion des SQL/2008-Standards. Die DEFAULT VALUES-Klausel ist die optionale SQL-Sprachenfunktion F222, (INSERT-Anweisung: DEFAULT VALUES-Klausel). Die Unterstützung für Zeilenwert-Konstruktoren in einer INSERT-Anweisung ist Teil der optionalen SQL Sprachenfunktion F641, "Zeile und Tabelle Konstruktoren". Das VALUES-Schlüsselwort ist eine Erweiterung des Herstellers und in SQL Anywhere obligatorisch, um die Liste der einzufügenden Ausdrücke anzugeben. VALUES ist jedoch nicht Teil von SQL/2008.

Einige optionale Konstruktionen sind Erweiterungen des Herstellers. Es handelt sich dabei um die folgenden:

- Die INSERT...ON EXISTING-Klausel ist eine Erweiterung des Herstellers. Eine SQL/2008-kompatible Entsprechung ist in vielen Fällen die MERGE-Anweisung.
- Die OPTION-Klausel.
- Die WITH AUTO NAME-Klausel.

Beispiele

Fügen Sie "Eastern Sales department" zur Datenbank hinzu:

```
INSERT INTO GROUP0.Departments ( DepartmentID, DepartmentName )
VALUES ( 230, 'Eastern Sales' );
```

Erstellen Sie die Tabelle 'DepartmentHead' mit den Namen der Abteilungsleiter (department heads) und ihren Abteilungen (departments) unter Verwendung der Syntax WITH AUTO NAME:

```
CREATE TABLE DepartmentHead(
    pk INT PRIMARY KEY DEFAULT AUTOINCREMENT,
    DepartmentName VARCHAR(128),
    ManagerName VARCHAR(128) );
```

```

INSERT
INTO DepartmentHead WITH AUTO NAME
SELECT GivenName || ' ' || Surname AS ManagerName,
       DepartmentName
FROM GROUPO.Employees JOIN GROUPO.Departments
ON EmployeeID = DepartmentHeadID;

```

Erstellen Sie die Tabelle 'MyTable6' und fügen Sie mithilfe der WITH AUTO NAME-Syntax Daten ein.

```

CREATE TABLE MyTable6(
    pk INT PRIMARY KEY DEFAULT AUTOINCREMENT,
    TableName CHAR(128),
    TableNameLen INT );
INSERT INTO MyTable6 WITH AUTO NAME
SELECT
    length(t.table_name) AS TableNameLen,
    t.table_name AS TableName
FROM SYS.SYSTAB t
WHERE table_id <= 10;

```

Fügen Sie eine neue Abteilung ein, wobei Sie die Anweisung auf Isolationsstufe 3 ausführen und nicht die aktuelle Isolationsstufeneinstellung der Datenbank verwenden.

```

INSERT INTO GROUPO.Departments
    (DepartmentID, DepartmentName, DepartmentHeadID)
VALUES(600, 'Foreign Sales', 129)
OPTION( isolation_level = 3 );

```

Im folgenden Beispiel werden drei Zeilen in die fiktive Tabelle "T" eingefügt:

```

INSERT INTO T (c1,c2,c3)
VALUES (1,10,100), (2,20,200), (3,30,300);

```

In diesem Beispiel werden drei Zeilen in die fiktive Tabelle "T" eingefügt, die aus vier Spalten besteht, für die jeweils ein Standardwert definiert ist:

```

INSERT INTO T ()
VALUES (), (), ();

```

INSTALL EXTERNAL OBJECT-Anweisung

Installiert ein Objekt, das in einer externen Umgebung ausgeführt werden kann.

Syntax

```

INSTALL EXTERNAL OBJECT object-name
[ update-mode ]
FROM { FILE file-path | VALUE expression }
ENVIRONMENT environment-name
[ AS USER user-name ]

```

environment-name :

```

PERL
| PHP

```

update-mode :

```

NEW
| UPDATE

```

Parameter

- object-name** Der Name, mit dem das installierte Objekt in der Datenbank gekennzeichnet wird.
- update-mode** Der Aktualisierungsmodus für das Objekt. Wenn der Aktualisierungsmodus nicht angegeben ist, wird NEW angenommen.
- file-path** Der Speicherort auf dem Servercomputer, auf dem das Objekt installiert wird.
- environment-name** Der Name der externen Umgebung, in der das externe Objekt ausgeführt wird.
- AS USER-Klausel** Gibt den Eigentümer des Objekts an.

Bemerkungen

Weitere Hinweise zu externen Umgebungen finden Sie unter „Unterstützung für externe Umgebungen in SQL Anywhere“ [[SQL Anywhere Server - Programmierung](#)].

Privilegien

Sie müssen das MANAGE ANY EXTERNAL OBJECT-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Unterstützung für externe Umgebungen in SQL Anywhere“ [[SQL Anywhere Server - Programmierung](#)]
- „ALTER EXTERNAL ENVIRONMENT-Anweisung“ auf Seite 463
- „REMOVE EXTERNAL OBJECT-Anweisung“ auf Seite 996
- „START EXTERNAL ENVIRONMENT-Anweisung“ auf Seite 1066
- „STOP EXTERNAL ENVIRONMENT-Anweisung“ auf Seite 1075
- „SYSEXTERNENV-Systemansicht“ auf Seite 1448
- „SYSEXTERNENVOBJECT-Systemansicht“ auf Seite 1450

Standards und Kompatibilität

- SQL/2008** Erweiterung des Herstellers.

Beispiele

In diesem Beispiel installieren Sie ein Perl-Skript, das sich in einer Datei in der Datenbank befindet.

```
INSTALL EXTERNAL OBJECT 'PerlScript'  
NEW  
FROM FILE 'perlfile.pl'  
ENVIRONMENT PERL;
```

Perl-Code kann wie folgt aus einem Ausdruck extrahiert und installiert werden:

```
INSTALL EXTERNAL OBJECT 'PerlConsoleExample'  
NEW  
FROM VALUE 'sub WriteToServerConsole { print $sa_output_handle $_[0]; }'  
ENVIRONMENT PERL;
```

Perl-Code kann wie folgt aus einer Variablen extrahiert und installiert werden:

```
CREATE VARIABLE PerlVariable LONG VARCHAR;
SET PerlVariable =
  'sub WriteToServerConsole { print $sa_output_handle $_[0]; }';

INSTALL EXTERNAL OBJECT 'PerlConsoleExample'
NEW
FROM VALUE PerlVariable
ENVIRONMENT PERL;
```

INSTALL JAVA-Anweisung

Macht Java-Klassen in einer Datenbank verfügbar.

Syntax

```
INSTALL JAVA
[ NEW | UPDATE ]
[ JAR jar-name ]
FROM { FILE filename | expression }
[ AS USER user-name ]
```

Parameter

Schlüsselwort-Klauseln NEW und UPDATE Wenn Sie NEW angeben, muss die referenzierte Java-Klasse oder JAR-Datei neue Klassen enthalten statt Aktualisierungen für bereits installierte Klassen bzw. JAR-Dateien. Ein Fehler tritt auf, wenn eine Klasse oder JAR-Datei mit demselben Namen in der Datenbank bereits vorhanden ist und NEW verwendet wird.

Wenn Sie UPDATE angeben, darf die referenzierte Datei Ersetzungen für Java-Klassen oder JAR-Dateien enthalten, die in der angegebenen Datenbank bereits installiert sind.

Wenn nichts angegeben wird, ist der Standardwert NEW.

JAR-Klausel Wenn dies angegeben wird, muss *filename* eine JAR-Datei bezeichnen. JAR-Dateien haben in der Regel die Erweiterungen *.jar* oder *.zip*.

Installierte JAR- und ZIP-Dateien können komprimiert oder unkomprimiert sein. Aufgrund von Unterschieden in den Komprimierungsschemata wird empfohlen, JAR-Dateien, die Textressourcen enthalten, mit deaktivierter Komprimierung zu erstellen.

Wenn die JAR-Klausel angegeben ist, wird die JAR-Datei als JAR-Datei gespeichert, nachdem die darin enthaltenen Klassen installiert wurden. Diese JAR-Datei ist für alle diese Klassen die zugeordnete JAR-Datei. Die in einer Datenbank mit der JAR-Klausel installierten JAR-Dateien werden als gespeicherte JAR-Dateien der Datenbank bezeichnet.

Der *jar-name* ist ein Zeichenfolgenwert mit einer Länge von bis zu 255 Byte. Der *jar-name* wird verwendet, um die gespeicherte JAR-Datei in nachfolgenden Anweisungen (INSTALL JAVA UPDATE und REMOVE JAVA) zu identifizieren.

FROM FILE-Klausel Gibt den Speicherort der zu installierenden Java-Klasse oder JAR-Datei an.

Zu den für *filename* unterstützten Formaten gehören voll qualifizierte Dateinamen wie *c:\\libs\\jarname.jar* und */usr/u/libs/jarname.jar* sowie relative Dateinamen (relativ zum aktuellen Arbeitsverzeichnis des Datenbankservers). Wenn der CLASSPATH des Datenbankservers den Pfad zu der betreffenden Klasse oder JAR-Datei enthält, muss der Pfad nicht in den Dateinamen einbezogen werden.

Der *filename* muss eine Klasse, eine JAR-Datei oder eine ZIP-Datei identifizieren.

FROM-Klausel Ausdrücke müssen in einen Binärtyp ausgewertet werden, dessen Wert eine gültige Klasse oder JAR-Datei enthält.

AS USER-Klausel Gibt den Eigentümer des Objekts an.

Bemerkungen

Die Klassendefinition für jede Klasse wird von der VM jeder Verbindung geladen, wenn diese Klasse zum ersten Mal benutzt wird. Wenn Sie eine Klasse installieren, wird die VM auf Ihrer Verbindung implizit neu gestartet. Deshalb haben Sie sofort Zugriff auf die neue Klasse. Da die VM neu gestartet wird, gehen alle Werte in statischen Java-Variablen verloren und SQL-Variablen mit Java-Klassentypen werden gelöscht.

Für andere Verbindungen wird die neue Klasse das nächste Mal geladen, wenn eine VM zum ersten Mal auf die Klasse zugreift. Wenn die Klasse bereits von einer VM geladen wurde, erkennt die Verbindung die neue Klasse erst, wenn die VM für diese Verbindung neu gestartet wird.

Alle installierten Klassen können auf jede Art von jedem Benutzer referenziert werden.

Diese Anweisung wird unter Windows Mobile nicht unterstützt.

Privilegien

Sie müssen das **MANAGE ANY EXTERNAL OBJECT**-Systemprivileg haben.

Siehe auch

- „JAR-Dateien installieren“ [[SQL Anywhere Server - Programmierung](#)]
- „Klassendateien installieren“ [[SQL Anywhere Server - Programmierung](#)]
- „REMOVE JAVA-Anweisung“ auf Seite 997
- „SYSJAR-Systemansicht“ auf Seite 1458
- „SYSJARCOMPONENT-Systemansicht“ auf Seite 1459
- „SYSJAVACLASS-Systemansicht“ auf Seite 1460

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird eine fiktive Java-Klasse namens Demo installiert, indem Dateiname und Speicherort der Klasse angegeben werden.

```
INSTALL JAVA NEW  
FROM FILE 'D:\\JavaClass\\Demo.class';
```

In diesem Beispiel werden alle in einer fiktiven ZIP-Datei enthaltenen Klassen installiert und innerhalb der Datenbank dem JAR-Dateinamen "Widgets" zugeordnet. Nach der Installation wird der Speicherort der ZIP-Datei nicht gespeichert und Klassen müssen mit dem voll qualifizierten Klassennamen (Paket- und Klassenname) referenziert werden.

```
INSTALL JAVA
JAR 'Widgets'
FROM FILE 'C:\\Jars\\Widget.zip';
```

INTERSECT-Anweisung

Berechnet den Querschnitt zwischen den Ergebnismengen von zwei oder mehr Abfragen.

Syntax

```
[ WITH temporary-views ] query-block
INTERSECT [ ALL | DISTINCT ] query-block
[ ORDER BY [ integer | select-list-expression-name ] [ ASC | DESC ], ... ]
[ FOR XML xml-mode ]
[ OPTION( query-hint, ... ) ]
```

query-block : See „Allgemeine Elemente der SQL-Syntax“ auf Seite 445.

query-hint :

```
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| option-name = option-value
```

option-name : *identifier*

option-value :

```
hostvar (Bezeichner zulässig)
| string
| identifier
| number
```

Parameter

FOR XML-Klausel Hinweise zur Verwendung der FOR XML-Klausel finden Sie unter „SELECT-Anweisung“ auf Seite 1020.

OPTION-Klausel Verwenden Sie diese Klausel, um Hints für das Ausführen der Anweisung anzugeben. Die folgenden Hints werden unterstützt:

- MATERIALIZED VIEW OPTIMIZATION *option-value*
- FORCE OPTIMIZATION
- *option-name* = *option-value*. Die Spezifikation `OPTION(isolation_level = ...)` im Abfragetext hebt alle anderen Festlegungen der Isolationsstufe für eine Abfrage auf.

Bemerkungen

INTERSECT berechnet die Schnittmenge zwischen den Ergebnismengen von zwei Abfrageblöcken. Abfrageblöcke können verschachtelt werden und ihrerseits aus verschachtelten SELECT-Anweisungen

oder aus den Mengenoperatoren (UNION, EXCEPT oder INTERSECT) bestehen. Die Verwendung von INTERSECT allein ist gleichwertig mit INTERSECT DISTINCT.

INTERSECT ALL implementiert einen Multimengendurchschnitt statt einer Schnittmenge. Wenn beispielsweise der erste *query-block* 5 (doppelte) Zeilen mit bestimmten Werten enthält und der zweite *query-block* 3 doppelte Zeilen mit zum ersten identischen Werten, gibt INTERSECT ALL 3 Zeilen zurück.

Die Ergebnisse von INTERSECT sind dieselben wie die Ergebnisse von INTERSECT ALL, wenn weder der eine noch der andere *query-block* doppelte Zeilen enthält.

Die beiden *query-block*-Ergebnismengen müssen UNION-kompatibel sein. Sie müssen dieselbe Anzahl von Elementen in ihren jeweiligen Auswahllisten aufweisen und die Art der einzelnen Ausdrücke sollte vergleichbar sein. Wenn übereinstimmende Elemente in zwei Auswahllisten verschiedene Datentypen umfassen, wählt SQL Anywhere einen Datentyp für die entsprechende Spalte im Ergebnis aus und konvertiert automatisch die Spalten in jedem *query-block*.

Die angezeigten Spaltennamen sind dieselben wie beim ersten *query-block* und mithilfe dieser Namen wird bestimmt, welche Ausdrucknamen mit der ORDER BY-Klausel abgeglichen werden. Eine andere Möglichkeit zum Anpassen von Spaltennamen in der Ergebnismenge ist die Verwendung eines allgemeinen Tabellenausdrucks (der WITH-Klausel).

Privilegien

Sie müssen das SELECT ANY TABLE-Systemprivileg haben, Eigentümer der in *query-block* angegebenen Objekte sein oder SELECT-Privilegien für die einzelnen Abfrageblöcke haben.

Nebenwirkungen

Keine.

Siehe auch

- „EXCEPT-Anweisung“ auf Seite 841
- „UNION-Anweisung“ auf Seite 1096
- „SELECT-Anweisung“ auf Seite 1020
- OPTION-Klausel, SELECT-Anweisung auf Seite 1027

Standards und Kompatibilität

- **SQL/2008** INTERSECT ist die optionale SQL-Sprachenfunktion F302 des SQL/2008-Standards. Das explizite Angeben des DISTINCT-Schlüsselworts mit INTERSECT ist die optionale SQL-Sprachenfunktion T551 des SQL-Standards. Das Angeben einer ORDER BY-Klausel mit INTERSECT ist SQL-Sprachenfunktion F850. Ein *query-block* mit einer ORDER BY-Klausel entspricht SQL/2008-Funktion F851. Ein Abfrageblock, der eine Zeilenbegrenzungsklausel (SELECT TOP oder LIMIT) enthält, umfasst je nach Kontext die optionale SQL-Sprachenfunktion F857 oder F858. Die Klauseln FOR XML und OPTION sind Erweiterungen des Herstellers.
- **Transact-SQL** INTERSECT wird von Adaptive Server Enterprise nicht unterstützt. Sowohl INTERSECT ALL als auch INTERSECT DISTINCT können jedoch in dem von SQL Anywhere unterstützten Transact-SQL-Dialekt verwendet werden.

Beispiel

Ein Beispiel für die Verwendung von INTERSECT finden Sie unter [„Mengenoperatoren und NULL“ \[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#).

LEAVE-Anweisung

Verlässt eine zusammengesetzte Anweisung oder eine Schleife.

Syntax

LEAVE *statement-label*

Bemerkungen

Die LEAVE-Anweisung ist eine Steueranweisung, mit der Sie eine benannte zusammengesetzte Anweisung oder eine benannte Schleife verlassen können. Die Ausführung wird bei der ersten Anweisung nach der zusammengesetzten Anweisung oder der Schleife wieder aufgenommen.

Die zusammengesetzte Anweisung, die den Hauptteil einer Prozedur oder eines Triggers darstellt, hat ein implizites Label, das mit dem Namen der Prozedur oder des Triggers übereinstimmt.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „LOOP-Anweisung“ auf Seite 950
- „FOR-Anweisung“ auf Seite 857
- „BEGIN-Anweisung“ auf Seite 557
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [\[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#)

Standards und Kompatibilität

- **SQL/2008** Die LEAVE-Anweisung ist Teil der optionalen SQL/2008-Sprachenfunktion P002 (Verarbeitungsvollständigkeit).

Beispiel

Das folgende Fragment zeigt, wie die LEAVE-Anweisung verwendet wird, um eine Schleife zu verlassen

```
SET i = 1;
lbl:
LOOP
    INSERT
    INTO Counters ( number )
    VALUES ( i );
    IF i >= 10 THEN
        LEAVE lbl;
    END IF;
```

```
        SET i = i + 1
    END LOOP lbl
```

Im folgenden Fragment wird LEAVE in einer verschachtelten Schleife verwendet.

```
outer_loop:
LOOP
    SET i = 1;
    inner_loop:
    LOOP
        ...
        SET i = i + 1;
        IF i >= 10 THEN
            LEAVE outer_loop
        END IF
    END LOOP inner_loop
END LOOP outer_loop
```

LOAD STATISTICS-Anweisung

Diese Anweisung ist für die interne Verwendung vorgesehen und wird vom Dienstprogramm dbunload verwendet, um Spaltenstatistiken aus der alten Datenbank in die ISYSCOLSTAT-Systemtabelle zu entladen.

Syntax

LOAD STATISTICS [[*owner*.]*table-name*.]*column-name*
format-id, *density*, *max-steps*, *actual-steps*, *step-values*, *frequencies*

Parameter

format-id Internes Feld für die Festlegung des Formats der restlichen Zeile in der Systemtabelle ISYSCOLSTAT.

density Eine Schätzung der gewichteten mittleren Selektivität eines einzelnen Wertes für die Spalte. Dabei wird die Selektivität großer Einzelwertselektivitäten, die in der Zeile gespeichert sind, nicht berücksichtigt.

max-steps Die maximale Anzahl der im Histogramm zulässigen Schritte

actual-steps Die Anzahl der tatsächlich verwendeten Schritte

step-values Grenzwerte der Histogrammschritte

frequencies Selektivitäten von Histogrammschritten

Privilegien

Sie müssen das MANAGE ANY STATISTICS-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „SYSCOLSTAT-Systemansicht“ auf Seite 1441
- „Dienstprogramm zum Entladen (dbunload)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

LOAD TABLE-Anweisung

Importiert Massendaten aus einer externen Datei in eine Datenbanktabelle.

Syntax

```
LOAD [ INTO ] TABLE [ owner.]table-name
[ ( column-name, ... ) ]
load-source
[ load-option ... ]
[ statistics-limitation-option ]
```

load-source :

```
{ FROM filename-expression
  | USING FILE filename-expression
  | USING CLIENT FILE client-filename-expression
  | USING VALUE value-expression
  | USING COLUMN column-expression }
```

filename-expression : *string* | *variable*

client-filename-expression : *string* | *variable*

value-expression : *expression*

column-expression :

```
column-name
FROM table-name
ORDER BY column-list
```

load-option :

```
ALLOW { integer | ALL | NO } ERRORS ]
| BYTE ORDER MARK { ON | OFF }
| CHECK CONSTRAINTS { ON | OFF }
| { COMPRESSED | AUTO COMPRESSED | NOT COMPRESSED }
| { ENCRYPTED KEY 'key' | NOT ENCRYPTED }
| COMMENTS INTRODUCED BY comment-prefix
| COMPUTES { ON | OFF }
| DEFAULTS { ON | OFF }
| DELIMITED BY string
| ENCODING encoding
| ESCAPE CHARACTER character
| ESCAPES { ON | OFF }
| FORMAT {
  | TEXT
  | BCP
```

```
| XML row-xpath ( column-xpath,... ) [ NAMESPACES namespace ] }
| SHAPEFILE
| HEXADECIMAL { ON | OFF }
| MESSAGE LOG log-target
| ORDER { ON | OFF }
| PCTFREE percent-free-space
| QUOTE string
| QUOTES { ON | OFF }
| ROW DELIMITED BY string
| ROW LOG log-target
| SKIP integer
| STRIP { OFF | LTRIM | RTRIM | BOTH }
| WITH CHECKPOINT { ON | OFF }
| WITH { FILE NAME | ROW | CONTENT } LOGGING

statistics-limitation-option :
STATISTICS {
  ON [ ALL COLUMNS ]
  | ON KEY COLUMNS
  | ON ( column-list )
  | OFF
}

comment-prefix : string

encoding : string

log-target : {
  FILE server-filename
  | CLIENT FILE client-filename
  | VARIABLE variable-name
}
```

Parameter

column-name Mit dieser Klausel geben Sie eine oder mehrere Spalten an, in die Daten geladen werden sollen. Alle nicht in der Spaltenliste vorhandenen Spalten werden NULL, wenn DEFAULTS auf OFF gesetzt ist. Wenn DEFAULTS auf ON gesetzt ist und die Spalte einen Standardwert hat, wird dieser Wert benutzt. Wenn DEFAULTS auf OFF gesetzt ist und eine nicht nullwertfähige Spalte aus der Spaltenliste ausgelassen wird, versucht der Datenbankserver, die leere Zeichenfolge in den Datentyp der Spalte zu konvertieren.

Wenn eine Spaltenliste angegeben ist, werden die in der Datei erwarteten Spalten sowie die Reihenfolge, in der sie erwartungsgemäß erscheinen werden, aufgelistet. Spaltennamen können nicht wiederholt werden. Spalten, deren Namen nicht in der Liste erscheinen, werden auf NULL/null/leer oder DEFAULT gesetzt (abhängig von der Nullwertfähigkeit der Spalte, vom Datentyp und der DEFAULTS-Einstellung). Spalten, die in der Eingabedatei existieren und von LOAD TABLE ignoriert werden sollen, können mit dem Spaltennamen **filler()** angegeben werden.

load-source Verwenden Sie diese Klausel, um die Datenquelle anzugeben, aus der die Daten geladen werden sollen. Es gibt mehrere Datenquellen, aus denen Daten geladen werden können. Die folgende Liste enthält die unterstützten Ladequellen:

FROM-Klausel Verwenden Sie diese Klausel, um eine Datei anzugeben. Der *filename-expression* wird an die Datenbank als Zeichenfolge übergeben. Die Zeichenfolge unterliegt daher denselben Formatierungsanforderungen für die Datenbank wie auch andere SQL-Zeichenfolgen. Speziell sind folgende Punkte zu beachten:

- Bei Angabe des Verzeichnissuchpfads muss das Backslashzeichen (\) durch zwei Backslashes dargestellt werden.
- Der Pfadname ist relativ zum Datenbankserver, nicht aber zur Clientanwendung.
- Sie können eine UNC-Pfadangabe verwenden, um Daten aus Dateien von anderen Computern als dem Datenbankserver zu laden.

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „Sandboxing“ [[SQL Anywhere Server - Datenbankadministration](#)].

USING FILE-Klausel Verwenden Sie diese Klausel, um Daten aus einer Datei zu laden. Dies ist synonym mit Angabe der FROM *filename*-Klausel.

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „Sandboxing“ [[SQL Anywhere Server - Datenbankadministration](#)].

Wenn die LOAD TABLE-Anweisung mit der USING FILE-Klausel verwendet wird, können Sie Meldungen zum Verarbeitungsfortschritt anfordern.

Sie können auch mithilfe der Verbindungseigenschaft "Progress" feststellen, wie viel von der Anweisung ausgeführt wurde.

USING CLIENT FILE-Klausel Verwenden Sie diese Klausel, um Daten aus einer Datei auf dem Clientsystem zu laden. Wenn der Datenbankserver Daten aus *client-filename-expression* bezieht, werden die Daten im Speicher des Servers nicht materialisiert. Daher gilt die Begrenzung des Datenbankservers für die Größe von Blob-Ausdrücken nicht für die Datei. Die Clientdatei kann daher jede beliebige Größe haben.

Dateinamenprotokollierung ist nicht zulässig, wenn die Tabelle aus einer Clientdatei geladen wird. Wenn der Protokollierungstyp nicht angegeben ist, wird WITH CONTENT LOGGING verwendet.

Wenn die LOAD TABLE-Anweisung mit der USING CLIENT FILE-Klausel verwendet wird, können Sie Meldungen zum Verarbeitungsfortschritt anfordern.

Sie können auch mithilfe der Verbindungseigenschaft "Progress" feststellen, wie viel von der Anweisung ausgeführt wurde.

USING VALUE-Klausel Verwenden Sie diese Klausel, um Daten aus einem Ausdruck des Typs CHAR, NCHAR, BINARY oder LONG BINARY bzw. einer BLOB-Zeichenfolge zu laden. Nachstehend finden Sie Beispiele für die Einsatzmöglichkeiten dieser Klausel:

- Die folgende Syntax benutzt die xp_read_file-Systemprozedur, um die zu ladenden Werte aus der Zielfeile zu holen:

```
... USING VALUE xp_read_file( 'filename' )...
```

- Die folgende Syntax gibt den Wert direkt an, indem zwei Zeilen mit den Werten 4 und 5 eingefügt werden.

```
... USING VALUE '4\n5'...
```

- Die folgende Syntax benutzt die Ergebnisse der READ_CLIENT_FILE-Funktion als Wert:

```
... USING VALUE READ_CLIENT_FILE( client-filename-expression )
```

In diesem Fall können Sie auch USING CLIENT FILE client-filename-expression angeben, da dies semantisch gleichwertig ist.

Wenn die ENCODING-Klausel in der LOAD TABLE-Anweisung nicht angegeben ist, erfolgt das Kodieren von *value-expression* im Zeichensatz der Datenbank (db_charset), wenn *value-expression* vom Typ CHAR oder BINARY ist, und im NCHAR-Datenbank-Zeichensatz (nchar_charset), wenn *value-expression* vom Typ NCHAR ist.

USING COLUMN-Klausel Verwenden Sie diese Klausel, um Daten aus einer einzelnen Spalte in eine andere Tabelle zu laden. Diese Klausel wird vom Datenbankserver benutzt, wenn er das Transaktionslog während der Wiederherstellung abarbeitet, indem die LOAD TABLE...WITH CONTENT LOGGING-Anweisungen neu ausgeführt werden. Die Transaktionslogeinträge für LOAD TABLE...WITH CONTENT LOGGING-Anweisungen enthalten Abschnitte der ursprünglichen Eingabedatei. Wenn ein Datenbankserver bei der Wiederherstellung im Transaktionslog auf diese Abschnitte trifft, lädt er die Teile in eine temporäre Tabelle und danach alle Daten aus dem ursprünglichen Ladevorgang.

Die folgenden Klauseln sind in der USING COLUMN-Klausel erforderlich:

- **Tabellenname** Der Name der Basistabelle oder temporären Tabelle, die die Spalte enthält, aus der die Daten geladen werden sollen. Wenn diese Tabelle vom Datenbankserver während der Wiederherstellung aus dem Transaktionslog benutzt wird, ist dies die Tabelle, die die Abschnitte der Zeilen enthält, die syntaktisch analysiert und geladen werden sollen.
- **column-name** Der Name der Spalte in *column-name*, die die zu ladenden Abschnitte der Zeilen enthält.
- **column-list** Eine oder mehr Spalten in der Zieltabelle, die benutzt werden, um die Zeilen vor dem Laden der Daten zu sortieren. *column-list* muss eine nachweisbar eindeutige Wertemenge sein, wie z. B. ein Primärschlüssel oder ein eindeutiger Index für nicht-nullwertfähige Spalten, die in der Spaltenliste enthalten sind.

load-option-Klausel Es gibt mehrere Ladeoptionen, mit denen Sie steuern können, wie Daten geladen werden sollen. Die folgende Liste enthält die unterstützten Ladeoptionen:

- **ALLOW (Ganzzahl | ALL | NO) ERRORS-Klausel** Diese Klausel kann für die Anweisung nur einmal angegeben werden. Der Standardwert für diese Klausel ist 0, was bedeutet, dass bei Verletzung ein Fehler generiert und die Anweisung zurückgesetzt wird. Wenn Sie eine Ganzzahl *n* angeben, setzt der Datenbankserver die Anweisung bei Fehler *n*+1 zurück. Die Werte ALLOW NO ERRORS und ALLOW 0 ERRORS sind äquivalent. Diese Klausel ermöglicht es dem Datenbankserver, problematische Daten beiseite zu lassen und mit dem Ladevorgang fortzufahren.

Der Datenbankserver meldet dem Benutzer den letzten aufgetretenen Fehler und dieser Fehler wird auch im MESSAGE-Log protokolliert. In das ROW-Log geschriebene Zeilen können geändert und als Eingabe für eine nachfolgende LOAD TABLE-Anweisung verwendet werden.

Beim Schreiben eines ROW LOG-Objekts in eine Datenbankserver- oder Clientdatei wird für den Inhalt derselbe Zeichensatz verwendet wie für die ursprüngliche Eingabedatei. Beim Schreiben eines MESSAGE LOG-Objekts in eine Server- oder Clientdatei werden für den Inhalt die Sprache des Clients und der CHAR-Zeichensatz der Clientverbindung verwendet. Wenn ein ROW LOG- oder MESSAGE LOG-Objekt in eine CHAR- oder NCHAR-Variable geschrieben wird, geschieht dies im CHAR- bzw. NCHAR-Zeichensatz.

- **BYTE ORDER MARK-Klausel** Verwenden Sie diese Klausel, um anzugeben, ob der Server eine Byte Order Mark (BOM) am Beginn der Daten suchen und interpretieren soll. Standardmäßig ist diese Option aktiviert. Wenn BYTE ORDER MARK OFF ist, sucht der Server nicht nach einer BOM.

Wenn die ENCODING-Klausel angegeben ist:

- Wenn die BYTE ORDER MARK-Option ON ist und Sie eine UTF-16-Kodierung mit Endian wie UTF-16BE oder UTF-16LE angeben, sucht der Datenbankserver nach einer BOM am Beginn der Daten. Wenn eine BOM vorhanden ist, wird sie benutzt, um den Endian der Daten zu prüfen. Wenn Sie den falschen Endian angeben, wird ein Fehler zurückgegeben.
- Wenn die BYTE ORDER MARK-Option ON ist und Sie eine UTF-16-Kodierung ohne expliziten Endian angegeben haben, sucht der Datenbankserver nach einer BOM am Beginn der Daten. Wenn eine BOM vorhanden ist, wird sie benutzt, um den Endian der Daten zu bestimmen. Sonst wird der Endian des Betriebssystems angenommen.
- Wenn die BYTE ORDER MARK-Option ON ist und Sie eine UTF-8-Kodierung angegeben haben, sucht der Datenbankserver nach einer BOM am Beginn der Daten. Wenn eine BOM vorhanden ist, wird sie ignoriert.

Wenn die ENCODING-Klausel nicht angegeben ist:

- Wenn Sie keine ENCODING-Klausel angeben und die BYTE ORDER MARK-Option ON ist, sucht der Server nach einer BOM am Beginn der Eingabedaten. Wenn es eine BOM findet, wird die Quellkodierung basierend auf der Kodierung der BOM (UTF-16BE, UTF-16LE oder UTF-8) automatisch gewählt und die BOM wird nicht als Teil der zu ladenden Daten angesehen. Wenn keine BOM gefunden wird, verwendet Interactive SQL die CHAR-Kodierung.
 - Wenn Sie keine ENCODING-Klausel angeben und die BYTE ORDER MARK-Option deaktiviert ist, wird der CHAR-Zeichensatz der Datenbank verwendet und der Datenbankserver sucht oder interpretiert keine BOM am Beginn der Daten.
- **CHECK CONSTRAINTS-Klausel** Verwenden Sie diese Klausel, um zu steuern, ob Integritätsregeln während des Ladens geprüft werden. CHECK CONSTRAINTS ist standardmäßig auf ON gesetzt, das Dienstprogramm Unload (dbunload) schreibt aber LOAD TABLE-Anweisungen mit der Option CHECK CONSTRAINTS auf OFF. Wenn Sie CHECK CONSTRAINTS auf OFF setzen, deaktivieren Sie Prüf-Integritätsregeln, was z.B. während des Datenbank-Neuaufbaus nützlich sein kann. Wenn einer Tabelle Prüf-Integritätsregeln zugeordnet sind, die benutzerdefinierte Funktionen aufrufen, welche noch nicht erstellt wurden, schlägt der Neuaufbau fehl. Dies ist allerdings nicht der Fall, wenn CHECK CONTRAINTS auf OFF gesetzt ist.

- **COMMENTS INTRODUCED BY-Klausel** Mit dieser Klausel können Sie die Zeichenfolge angeben, die in der Datendatei verwendet wird, um einen Kommentar einzuleiten. Wird diese Option verwendet, ignoriert LOAD TABLE jede Zeile, die mit der Zeichenfolge *comment-prefix* beginnt.

Kommentare sind nur am Anfang einer neuen Zeile zulässig.

Wenn die COMMENTS INTRODUCED BY-Klausel weggelassen wird, darf die Datendatei keine Kommentare enthalten, weil diese als Daten interpretiert werden.

- **COMPRESSED-Klausel** Geben Sie COMPRESSED an, wenn die zu ladenden Daten in der Eingabedatei komprimiert sind. Der Datenbankserver dekomprimiert die Daten, bevor sie geladen werden. Wenn Sie COMPRESSED angeben und die Daten nicht komprimiert sind, schlägt LOAD fehl und gibt einen Fehler zurück.

Geben Sie AUTO COMPRESSED an, damit der Datenbankserver ermitteln kann, ob die Daten in der Eingabedatei komprimiert sind oder nicht. Wenn dies der Fall ist, dekomprimiert der Datenbankserver die Daten, bevor sie geladen werden.

Geben Sie NOT COMPRESSED an, um anzuzeigen, dass die Daten in der Eingabedatei nicht komprimiert werden. Sie können auch NOT COMPRESSED angeben, wenn die Daten komprimiert sind, Sie aber nicht wollen, dass der Datenbankserver sie entkomprimiert. In diesem Fall bleiben die Daten in der Datenbank komprimiert. Wenn hingegen eine Datei sowohl verschlüsselt als auch komprimiert ist, können Sie NOT ENCRYPTED nicht ohne NOT COMPRESSED verwenden.

- **COMPUTES-Klausel** Standardmäßig ist diese Option ON, damit eine Neuberechnung berechneter Spalten möglich ist. Mit COMPUTES auf OFF werden Neuberechnungen von berechneten Spalten deaktiviert. COMPUTES OFF ist beispielsweise sinnvoll, wenn Sie eine Datenbank neu aufbauen und eine Tabelle eine berechnete Spalte umfasst, die eine noch nicht erstellte benutzerdefinierte Funktion aufruft. Der Neuaufbau würde fehlschlagen, wenn diese Option nicht auf OFF gesetzt ist.

Das Dienstprogramm Unload (dbunload) schreibt LOAD TABLE-Anweisungen mit COMPUTES auf OFF.

- **DEFAULTS-Klausel** Standardmäßig hat DEFAULTS die Einstellung OFF. Wenn DEFAULTS auf OFF gesetzt ist, wird allen Spalten, die in der Spaltenliste nicht vorhanden sind, NULL zugeordnet. Wenn DEFAULTS auf OFF gesetzt ist und eine nicht nullwertfähige Spalte aus der Liste ausgelassen wird, versucht der Datenbankserver, die leere Zeichenfolge in den Datentyp der Spalte zu konvertieren. Wenn DEFAULTS auf ON gesetzt ist und die Spalte einen Standardwert hat, wird dieser Wert benutzt.
- **DELIMITED BY-Klausel** Verwenden Sie diese Klausel, um die Spaltentrennzeichenfolge anzugeben. Das Standardtrennzeichen für Spalten ist ein Komma. Sie können jedoch auch eine beliebige Zeichenfolge mit einer Länge von bis zu 255 Byte verwenden (z.B. . . . DELIMITED BY '###' . . .). Der von Ihnen angegebene Begrenzer ist eine Zeichenfolge und muss in Anführungszeichen stehen. Wenn Sie durch Tabulatoren getrennte Werte angeben möchten, könnten Sie z.B. die hexadezimale Escapesequenz für das Tabulatorzeichen (9) verwenden, . . .
DELIMITED BY '\x09' . . .

- **ENCODING-Klausel** Verwenden Sie diese Klausel, um die Zeichenkodierung anzugeben, die für die in die Datenbank eingelesenen Daten verwendet wird. Die ENCODING-Klausel kann nicht im BCP-Format verwendet werden.

Wenn während eines Ladevorgangs ein Konvertierungsfehler auftritt, wird er entsprechend der Einstellung der `on_charset_conversion_failure`-Option gemeldet.

Geben Sie die `BYTE ORDER`-Klausel an, um eine BOM (Byte Order Mark) in den Daten zu interpretieren.

Wenn die `ENCODING`-Klausel angegeben ist:

- Wenn die `BYTE ORDER MARK`-Option `ON` ist und Sie eine UTF-16-Kodierung mit Endian wie UTF-16BE oder UTF-16LE angeben, sucht der Datenbankserver nach einer BOM am Beginn der Daten. Wenn eine BOM vorhanden ist, wird sie benutzt, um den Endian der Daten zu prüfen. Wenn Sie den falschen Endian angeben, wird ein Fehler zurückgegeben.
- Wenn die `BYTE ORDER MARK`-Option `ON` ist und Sie eine UTF-16-Kodierung ohne expliziten Endian angegeben haben, sucht der Datenbankserver nach einer BOM am Beginn der Daten. Wenn eine BOM vorhanden ist, wird sie benutzt, um den Endian der Daten zu bestimmen. Sonst wird der Endian des Betriebssystems angenommen.
- Wenn die `BYTE ORDER MARK`-Option `ON` ist und Sie eine UTF-8-Kodierung angegeben haben, sucht der Datenbankserver nach einer BOM am Beginn der Daten. Wenn eine BOM vorhanden ist, wird sie ignoriert.

Wenn die `ENCODING`-Klausel nicht angegeben ist:

- Wenn Sie keine `ENCODING`-Klausel angeben und die `BYTE ORDER MARK`-Option `ON` ist, sucht der Server nach einer BOM am Beginn der Eingabedaten. Wenn es eine BOM findet, wird die Quellkodierung basierend auf der Kodierung der BOM (UTF-16BE, UTF-16LE oder UTF-8) automatisch gewählt und die BOM wird nicht als Teil der zu ladenden Daten angesehen.
 - Wenn Sie keine `ENCODING`-Klausel angeben und die `BYTE ORDER MARK`-Option deaktiviert ist, wird der `CHAR`-Zeichensatz der Datenbank verwendet und der Datenbankserver sucht oder interpretiert keine BOM am Beginn der Daten.
- **ENCRYPTED-Klausel** Verwenden Sie diese Klausel, um Verschlüsselungseinstellungen anzugeben. Wenn Sie verschlüsselte Daten laden, geben Sie `ENCRYPTED KEY` gefolgt vom Schlüssel an, der bei der Verschlüsselung der Daten in der Eingabedatei verwendet wurde. Der Schlüssel kann entweder eine Zeichenfolge oder ein Variablenname sein.

Geben Sie `NOT ENCRYPTED` an, um anzuzeigen, dass die Daten in der Eingabedatei nicht verschlüsselt sind. Sie können auch `NOT ENCRYPTED` angeben, wenn die Daten verschlüsselt sind, Sie aber nicht wollen, dass der Datenbankserver sie entschlüsselt. In diesem Fall bleiben die Daten in der Datenbank verschlüsselt. Wenn hingegen eine Datei sowohl verschlüsselt als auch komprimiert ist, können Sie `NOT ENCRYPTED` nicht ohne `NOT COMPRESSED` verwenden.

- **ESCAPE CHARACTER-Klausel** Verwenden Sie diese Klausel, um das Escapezeichen anzugeben, das in den Daten verwendet wird. Das Standard-Escapezeichen für Zeichen, die als hexadezimale Codes und Symbole gespeichert werden, ist ein Backslash (`\`), so ist beispielsweise `\x0A` die Zeilenendmarke. Mit der `ESCAPE CHARACTER`-Klausel kann das geändert werden. Wenn Sie beispielsweise das Ausrufezeichen als Escape-Zeichen verwenden möchten, geben Sie Folgendes ein:

ESCAPE CHARACTER '!''

Es wird empfohlen, dass die Zeichenfolge, die Sie für das Escape-Zeichen angeben, nicht länger als ein Mehrbyte-Zeichen ist.

- **ESCAPES-Klausel** Verwenden Sie diese Klausel, um zu steuern, ob Escapezeichen erkannt werden. Wenn ESCAPES auf ON gesetzt ist (Standardwert), werden die Zeichen nach dem Escape-Zeichen (standardmäßig \) vom Datenbankserver als Sonderzeichen erkannt und interpretiert. Zeilenumbruch-Zeichen können als Kombination \n eingefügt werden, und andere Zeichen können als hexadezimale ASCII-Codes in die Daten eingefügt werden, wie zum Beispiel als \x09 für das Tabulatorzeichen. Eine Sequenz von zwei Backslashes (\\) wird als ein einzelner Backslash interpretiert. Ein Backslash gefolgt von einem beliebigen Zeichen außer n, x, X oder \ wird als zwei separate Zeichen interpretiert. Zum Beispiel werden mit "\q" ein Backslash und der Buchstabe q eingefügt. Es wird empfohlen, dass die Zeichenfolge, die Sie für das Escape-Zeichen angeben, nicht länger als ein Mehrbyte-Zeichen ist.
- **FORMAT TEXT** Wenn Sie FORMAT TEXT (Standardwert) angeben, werden Eingabezeilen als Zeichen vorausgesetzt (wie von der ENCODING-Option festgelegt), eine Zeile pro Ausgabezeile, wobei die Werte durch die Spalten-Trennzeichenfolge voneinander getrennt werden.
- **FORMAT BCP** Geben Sie FORMAT BCP an, um mit Adaptive Server Enterprise generierte BCP-Ausgabedateien zu laden.
- **FORMAT SHAPEFILE** Geben Sie FORMAT SHAPEFILE an, um ESRI-Formdateien zu laden. Die Formdatei muss sich auf dem Datenbankserversystem befinden und mit FROM *filename-expression* oder USING FILE *filename-expression* geladen werden, wobei sich *filename-expression* auf eine ESRI-Formdatei mit der Erweiterung *.shp* bezieht. Die zugeordneten *.shx*- und *.dbf*-Dateien müssen sich in demselben Verzeichnis befinden wie die *.shp*-Datei und denselben Basisdateinamen aufweisen.

Für FORMAT SHAPEFILE wird standardmäßig die Kodierung ISO-8859-1 verwendet, wenn die ENCODING-Klausel nicht angegeben ist.

Wenn Sie FORMAT SHAPEFILE angeben, sind die folgenden Ladeoptionen zulässig:

- CHECK CONSTRAINTS
- COMPUTES
- DEFAULTS
- ENCODING
- ORDER
- PCTFREE
- WITH CHECKPOINT
- WITH LOGGING

Die LOAD TABLE-Anweisung ruft die SRID aus dem zweiten Spaltentyp ab, in den Sie laden. Wenn Sie z.B. die zweite Spalte mit dem ST_Geometry-Typ erstellen und für SRID 4326 angegeben ist, werden die Geometrien mit SRID 4326 geladen. Wenn die zweite Spalte den ST_Geometry-Typ ohne explizite SRID aufweist, werden die Geometrien mit SRID 0 geladen.

- **FORMAT XML** Wenn Sie FORMAT XML angeben, sind nur die folgenden Ladeoptionen zulässig:

- BYTE ORDER MARK
- CHECK CONSTRAINTS
- COMPRESSED
- COMPUTES
- DEFAULTS
- ENCODING
- ENCRYPTED
- ORDER
- PCTFREE
- WITH CHECKPOINT
- WITH...LOGGING

Wenn Sie FORMAT XML verwenden, wird die Eingabedatei genauso syntaktisch analysiert wie eine Abfrage mit dem OPENXML-Operator. Die Argumente der SQL-Anweisung entsprechen den Parametern der Systemprozedur, wie im Folgenden dargestellt:

Klausel der LOAD TABLE-Anweisung	OPENXML-Operatorargument	Details
<i>row-xpath</i>	<i>xpath</i>	
—	<i>flags</i>	Es gibt keine Möglichkeit, einen Wert mit FORMAT XML anzugeben, der dem <i>flags</i> -Argument von OPENXML entspricht.
NAMESPACES	<i>namespaces</i>	

Die FORMAT XML-Klausel verwendet die folgenden Parameter:

- **row-xpath** Eine Zeichenfolge oder Variable, die eine XPath-Abfrage enthält. Mit XPath können Sie Muster angeben, die die Struktur des abzufragenden XML-Dokumentes beschreiben. Das in diesem Argument enthaltene XPath-Muster wählt die Knoten aus dem XML-Dokument. Jeder Knoten, der mit der XPath-Abfrage im *row-xpath*-Argument übereinstimmt, erzeugt eine Zeile in der Tabelle.

Meta-Eigenschaften können nur in *row-xpath*-Argumenten der FORMAT XML-Klausel angegeben werden. Auf eine Meta-Eigenschaft wird innerhalb einer XPath-Abfrage so zugegriffen, als handele es sich um ein Attribut. Wenn *namespaces* nicht angegeben wird, bindet das System standardmäßig das Präfix "mp" an den Uniform Resource Identifier (URI) urn:ianywhere-com:sa-xpath-metaprop. Wenn *namespaces* angegeben wird, muss dieser URI an "mp" oder an ein anderes Präfix gebunden werden, damit auf die Meta-Eigenschaften der Abfrage zugegriffen werden kann. Meta-Eigenschaften berücksichtigen die Groß- und Kleinschreibung. Folgende Meta-Eigenschaften werden unterstützt:

- **@mp:id** Gibt eine ID für einen Knoten zurück, der innerhalb des XML-Dokumentes eindeutig ist. Die ID für einen bestimmten Knoten in einem bestimmten Dokument kann sich

ändern, falls der Datenbankserver neu gestartet wird. Der Wert dieser Meta-Eigenschaft steigt mit der Dokumentordnung.

- **@mp:localname** Gibt den lokalen Teil des Knotennamens zurück oder NULL, falls der Knoten keinen Namen trägt.
 - **@mp:prefix** Gibt den Präfixteil des Knotennamens zurück oder NULL, falls der Knoten keinen Namen trägt oder falls der Name kein Präfix hat.
 - **@mp:namespaceuri** Gibt den URI des Namespaces zurück, dem der Knoten angehört, bzw. NULL, falls der Knoten sich in keinem Namespace befindet.
 - **@mp:xmltext** Gibt eine Unterstruktur des XML-Dokuments in XML-Form zurück. Wenn Sie z.B. einen internen Knoten suchen, können Sie diese Meta-Eigenschaft dafür benutzen, eine XML-Zeichenfolge zurückzugeben, und nicht verkettete Werte der untergeordneten Textknoten.
 - **column-xpath** Eine Zeichenfolge oder Variable, die das Schema der Ergebnismenge angibt und festlegt, wie der Wert für die einzelnen Spalten in der Ergebnismenge gefunden wird. Wenn ein FORMAT XML-Klauselausdruck mehr als einen Knoten findet, wird nur der erste Knoten in der Dokumentordnung verwendet. Falls es sich bei dem Knoten nicht um einen Textknoten handelt, wird das Ergebnis gefunden, indem alle untergeordneten Elemente des Textknotens angehängt werden. Wenn ein FORMAT XML-Klauselausdruck mit keinem der Knoten übereinstimmt, ist die Spalte für die betreffende Zeile NULL.
 - **namespace** Eine Zeichenfolge oder Variable, die ein XML-Dokument enthält. Die bekannten Namespaces ("in-scope namespaces") für die Abfrage werden aus dem Wurzelement des Dokumentes entnommen.
 - **HEXADECIMAL-Klausel** Verwenden Sie diese Klausel, um anzugeben, ob Binärwerte als hexadezimale Werte gelesen werden sollen. Standardmäßig ist HEXADECIMAL auf ON gesetzt. Wenn HEXADECIMAL ON ist, werden binäre Spaltenwerte als **0xnnnnnn...** gelesen, wobei 0x eine Null, gefolgt vom Buchstaben x ist, und jedes *n* ein hexadezimaless Zeichen ist. Es ist wichtig, HEXADECIMAL ON zu verwenden, wenn Sie mit Mehrbyte-Zeichensätzen arbeiten.
- Die HEXADECIMAL-Klausel kann nur mit der FORMAT TEXT-Klausel verwendet werden.
- **MESSAGE LOG-Klausel** Diese Klausel kann für die Anweisung nur einmal angegeben werden. Wenn während des Einfügens oder der syntaktischen Analyse einer Zeile ein Fehler auftritt, schreibt der Datenbankserver den Fehler an den angegebenen Speicherort.
 - **ORDER-Klausel** Verwenden Sie diese Klausel, um anzugeben, ob die Daten beim Laden sortiert werden. Die Standardeinstellung für ORDER ist ON. Wenn ORDER auf ON gesetzt ist und ein Clustered-Index deklariert wurde, sortiert LOAD TABLE die Eingabedaten nach dem Clustered-Index und fügt Spalten in derselben Reihenfolge ein. Wenn die zu ladenden Daten bereits sortiert sind, müssen Sie ORDER auf OFF setzen.
 - **PCTFREE-Klausel** Verwenden Sie diese Klausel, um den Prozentsatz des freien Speicherplatzes anzugeben, den Sie für jede Tabellenseite reservieren möchten. Diese Einstellung überschreibt jede permanente Einstellung für die Tabelle, jedoch nur für die Dauer des Ladevorgangs und nur bei den

Daten, die geladen werden. Der Wert für *percent-free-space* ist ein Ganzzahlwert zwischen 0 und 100. Der Wert 0 gibt an, dass kein freier Speicherplatz auf den einzelnen Seiten zur Verfügung stehen darf. Jede Seite muss komplett vollgeschrieben werden. Ein hoher Wert führt dazu, dass jede Zeile auf eine eigene Seite geschrieben wird.

- **QUOTE-Klausel** Die QUOTE-Klausel ist nur für TEXT-Daten bestimmt. Die *string* wird vor und nach den Zeichenfolgenwerten gesetzt. Der Standardwert ist ein Apostroph.
- **QUOTES-Klausel** Verwenden Sie diese Klausel, um anzugeben, ob Zeichenfolgen in Anführungszeichen gesetzt werden. Wenn QUOTES aktiviert ist (der Standardwert), wird von LOAD TABLE erwartet, dass alle Zeichenfolgen von Anführungszeichen oder Apostrophen umschlossen sind. Wenn die QUOTES-Klausel nicht angegeben ist, ist das Hervorhebungszeichen entweder ein Apostroph oder ein Anführungszeichen, und das erste Zeichen dieser Art, das in einer Zeichenfolge gefunden wird, wird als Hervorhebungszeichen für die Zeichenfolge verwendet. Die Zeichenfolgen müssen durch ein jeweils passendes Anführungszeichen abgeschlossen werden.

Wenn QUOTES ON ist, können Spaltenbegrenzungszeichenfolgen in Spaltenwerte einbezogen werden. Es wird auch vorausgesetzt, dass Apostrophe oder Anführungszeichen nicht Teil des Wertes sind. Daher wird die folgende Zeile wie zwei Werte behandelt und nicht wie drei, obwohl es ein Komma in der Adresse gibt. Außerdem werden die Anführungszeichen um die Adresse nicht in die Datenbank eingefügt.

```
'123 High Street, Anytown',(715)398-2354
```

Wenn Sie bei QUOTES ON ein Apostroph oder ein Anführungszeichen in einen Wert einbeziehen möchten, müssen Sie zwei Apostrophe oder Anführungszeichen verwenden. Die folgende Zeile enthält in der dritten Spalte einen Wert, der ein Apostrophzeichen ist:

```
'123 High Street, Anytown','(715)398-2354','''
```

- **ROW DELIMITED BY-Klausel** Verwenden Sie diese Klausel, um die Zeichenfolge anzugeben, die das Ende eines Eingabedatensatzes anzeigt. Die Standardtrennzeichenfolge ist eine Zeilenendmarke (`\n`). Sie können jedoch auch eine beliebige Zeichenfolge mit einer Länge von bis zu 255 Byte verwenden (z.B. `ROW DELIMITED BY '###'`). Wenn Sie durch Tabulatoren getrennte Werte angeben möchten, könnten Sie z.B. die hexadezimale Escapesequenz für das Tabulatorzeichen (9) verwenden, `ROW DELIMITED BY '\x09'`. Wenn Ihre Trennzeichenfolge ein `\n` enthält, entspricht es entweder `\r\n` oder `\n`.
- **ROW LOG-Klausel** Diese Klausel kann für die Anweisung nur einmal angegeben werden. Wenn während des Einfügens oder der syntaktischen Analyse einer Zeile ein Fehler auftritt, schreibt der Datenbankserver ein Bild der Eingabezeile an den angegebenen Speicherort und meldet zusätzlich die Zeile an den Benutzer.
- **SKIP-Klausel** Mit dieser Klausel geben Sie an, ob Zeilen am Anfang einer Datei ignoriert werden sollen. Das Argument *integer* gibt die Anzahl der zu überspringenden Zeilen an. Sie können diese Klausel verwenden, um beispielsweise Zeilen mit Spaltenüberschriften zu überspringen. Wenn als Zeilentrennzeichen nicht der Standardwert (Zeilenendmarke) eingestellt ist, kann es sein, dass ein Überspringen nicht korrekt funktioniert, wenn innerhalb der in Anführungszeichen stehenden Daten solche Zeilentrennzeichen enthalten sind.

- **STRIP-Klausel** Verwenden Sie diese Klausel, um anzugeben, ob bei Werten ohne Anführungszeichen führende oder nachgestellte Leerzeichen entfernt werden sollen, bevor sie eingefügt werden. Die Option STRIP kann mit den folgenden Optionen verwendet werden:
 - **STRIP OFF** Voran- bzw. nachgestellte Leerzeichen werden nicht entfernt.
 - **STRIP LTRIM** Führende Leerzeichen werden entfernt.
 - **STRIP RTRIM** Nachgestellte Leerzeichen werden entfernt.
 - **STRIP BOTH** Sowohl führende als auch nachgestellte Leerzeichen werden entfernt.

Das Verhalten von STRIP ist mit der QUOTES-Klausel verknüpft. Wenn Sie QUOTES OFF angeben, funktionieren STRIP OFF, STRIP LTRIM, STRIP RTRIM und STRIP BOTH genauso, wie es der jeweilige Wortlaut andeutet. Wenn Sie keine QUOTES-Klausel oder QUOTES ON angeben, werden Zeichenfolgen ohne Apostrophe oder Anführungszeichen stets links und rechts abgeschnitten. Sie können allerdings STRIP OFF oder STRIP LTRIM angeben, wenn Sie nicht wollen, dass die Zeichenfolgen auch rechts abgeschnitten werden.

- **WITH CHECKPOINT-Klausel** Mit dieser Klausel geben Sie an, ob ein Checkpoint gesetzt werden soll. Die Standardeinstellung ist OFF. Wenn diese Klausel auf ON gesetzt ist, wird ein Checkpoint gesetzt, nachdem die Anweisung erfolgreich abgeschlossen und im Log verzeichnet ist. Wenn diese Klausel auf ON eingestellt ist und die Datenbank eine automatische Wiederherstellung erfordert, bevor ein Checkpoint gesetzt wurde, muss die für das Laden der Tabelle verwendete Datendatei vorhanden sein, damit die Wiederherstellung abgeschlossen werden kann, wenn Sie FILE NAME LOGGING verwenden. Wenn WITH CHECKPOINT ON angegeben wurde und anschließend eine Wiederherstellung erforderlich ist, beginnt die Wiederherstellung nach dem Checkpoint und die Datendatei wird nicht benötigt.

Die Datendateien sind unabhängig von den Angaben für diese Klausel erforderlich, falls die Datenbank beschädigt wird und Sie auf eine Sicherung zurückgreifen und die aktuelle Logdatei übernehmen müssen, sofern Sie FILE NAME LOGGING verwenden.

- **WITH { FILE NAME | ROW | CONTENT } LOGGING** Verwenden Sie diese Klausel, um zu steuern, wie detailliert während eines Ladevorgangs im Transaktionslog protokolliert wird. Folgende Protokollierungsebenen sind möglich:
 - **WITH FILE NAME LOGGING-Klausel** Die WITH FILE NAME LOGGING-Klausel bewirkt, dass nur die LOAD TABLE-Anweisung ins Transaktionslog geschrieben wird. Um konsistente Ergebnisse zu garantieren, wenn das Transaktionslog während der Wiederherstellung verwendet wird, muss die Datei, die für den ursprünglichen Ladevorgang verwendet wurde, an ihrem ursprünglichen Speicherort vorhanden sein und die ursprünglichen Daten enthalten. Diese Stufe der Protokollierung hat die beste Performance. Allerdings sollten Sie sie nicht verwenden, wenn Ihre Datenbank in eine Spiegel- oder Synchronisationsumgebung integriert ist. Außerdem kann diese Stufe nicht beim Laden aus einem Ausdruck oder einer Clientdatei verwendet werden.

Wenn Sie in der LOAD TABLE-Anweisung keine Protokollierungsstufe angeben, ist WITH ROW LOGGING die Standardstufe bei folgender Eingabe:

- FROM *filename-expression*
- USING FILE *filename-expression*
- **WITH ROW LOGGING-Klausel** Die WITH ROW LOGGING-Klausel bewirkt, dass jede geladene Zeile im Transaktionslog als INSERT-Anweisung protokolliert wird. Diese Protokollierungsstufe wird für Datenbanken in einer Synchronisationsumgebung empfohlen und ist die Standardeinstellung für Datenbankspiegelung bei der Verwendung von FROM *filename-expression* oder USING FILE *filename-expression*. Wenn Sie jedoch große Datenmengen laden, kann dieser Protokollierungstyp die Performance beeinträchtigen und erzeugt ein wesentlich längeres Transaktionslog.

Wenn keine nicht-deterministischen Werte vorhanden sind, führt WITH CONTENT LOGGING wahrscheinlich zu einer besseren Performance.

Diese Stufe ist auch für Datenbanken ideal, in denen die Tabelle, in die geladen wird, nicht-deterministische Werte enthält, wie etwa berechnete Spalten oder CURRENT TIMESTAMP-Standardwerte.

- **WITH CONTENT LOGGING-Klausel** Die WITH CONTENT LOGGING-Klausel bewirkt, dass der Datenbankserver die Eingabedatei in Abschnitten in das Transaktionslog kopiert. Diese Abschnitte können später in einer Kopie der Eingabedatei wiederhergestellt werden, zum Beispiel während der Wiederherstellung aus dem Transaktionslog. Beim Laden großer Datenmengen hat diese Protokollierungsart wenig Auswirkungen auf die Performance und bietet mehr Datenschutz. Allerdings entsteht dabei ein sehr großes Transaktionslog. Diese Protokollierungsstufe wird für Datenbanken in einer Spiegelumgebung empfohlen, oder wenn die Originaldateien nicht für eine spätere Wiederherstellung aufbewahrt werden, sofern es keine nicht-deterministischen Werte gibt.

Die Klausel WITH CONTENT LOGGING kann nicht verwendet werden, wenn die Datenbank in einer Synchronisationsumgebung integriert ist. Die WITH CONTENT LOGGING-Klausel ist erforderlich, wenn die Tabelle aus einer Clientdatei geladen wird.

Wenn Sie in der LOAD TABLE-Anweisung keine Protokollierungsstufe angeben, ist WITH CONTENT LOGGING die Standardstufe bei folgender Eingabe:

- USING CLIENT FILE *client-filename-expression*
- USING VALUE *value-expression*
- USING COLUMN *column-expression*

Ebenso ist WITH CONTENT LOGGING die Standardeinstellung, wenn Sie eine LOAD TABLE-Anweisung auf einem Primär- oder Stammserver ausführen.

statistics-limitation-option Mit diesen Optionen können Sie die Anzahl der Spalten begrenzen, für die Statistiken während der Ausführung von LOAD TABLE generiert werden. Andernfalls werden für alle Spalten Statistiken generiert. Sie sollten diese Klausel nur verwenden, wenn Sie sicher sind, dass Statistiken für manche Spalten nicht verwendet werden. Sie können ON ALL COLUMNS

(Standardeinstellung), OFF, ON KEY COLUMNS oder eine Liste von Spalten, für die Statistiken generiert werden sollen, angeben.

Bemerkungen

LOAD TABLE ermöglicht das effiziente Einfügen großer Datenmengen aus einer Datei in die Datenbanktabelle. LOAD TABLE ist effektiver als die Interactive SQL-Anweisung INPUT.

LOAD TABLE verwendet eine exklusive Schemasperrung. Für Basistabellen sowie globale und lokale temporäre Tabellen wird ein Festschreiben durchgeführt.

Wenn Sie versuchen, LOAD TABLE mit einer Tabelle zu verwenden, für die ein Soforttextindex aufgebaut wird oder die von einer Sofortansicht referenziert wird, schlägt das Laden fehl. Dies kommt bei Nicht-Soforttextindizes oder materialisierten Ansichten nicht vor. Es wird aber empfohlen, die Daten in abhängigen Indizes und materialisierten Ansichten zu kürzen, bevor die LOADTABLE-Anweisung für eine Tabelle ausgeführt wird, und dann die Indizes und materialisierten Ansichten zu aktualisieren.

Verwenden Sie die LOAD TABLE-Anweisung nicht bei einer temporären Tabelle, für die ON COMMIT DELETE ROWS während der Erstellung entweder explizit oder standardmäßig angegeben wurde. Allerdings *können* Sie LOAD TABLE verwenden, wenn ON COMMIT PRESERVE ROWS oder NOT TRANSACTIONAL angegeben wurde.

Bei FORMAT TEXT wird ein NULL-Wert angezeigt, indem gar kein Wert angegeben wird. Beispiel: Wenn drei Werte erwartet werden und die Datei 1, , 'Fred' , enthält, dann sind die eingefügten Werte 1, NULL und Fred. Wenn die Datei 1, 2, , enthält, werden die Werte 1, 2 und NULL eingefügt. Werte, die nur aus Leerstellen bestehen, werden ebenfalls als NULL angesehen. Beispiel: Wenn die Datei 1, , , 'Fred' , enthält, dann werden die Werte 1, NULL und Fred eingefügt. Alle anderen Werte werden als Nicht-NULL angesehen. Beispiel: " (Apostroph gefolgt von Apostroph) ist eine leere Zeichenfolge. "NULL" ist eine Zeichenfolge, die vier Buchstaben enthält.

Wenn eine Spalte, die mit LOAD TABLE geladen wird, NULL nicht zulässt und der Dateiwert NULL ist, erhalten numerische Werte den Wert "0" (Null) und Zeichenspalten eine leere Zeichenfolge ("") zugeordnet. Wenn eine Spalte, die mit LOAD TABLE geladen wird, NULL zulässt und der Dateiwert NULL ist, dann ist der Spaltenwert NULL (bei allen Typen).

Wenn die Tabelle die Spalten a, b und c enthält und die Eingabedaten a, b und c enthalten, aber die LOAD-Anweisung nur a und b als Spalten angibt, in die Daten geladen werden sollen, werden die folgenden Werte in die Spalte c eingefügt:

- Wenn DEFAULT ON angegeben ist und Spalte c einen Standardwert hat, wird der Standardwert verwendet.
- Wenn die Spalte c keinen Standardwert definiert hat und NULL-Werte zulässt, wird NULL benutzt.
- Wenn Spalte c keinen Standardwert hat und keine NULL-Werte zulässt, wird entweder eine Null (0) oder eine leere Zeichenfolge (") verwendet bzw. ein Fehler zurückgegeben, abhängig vom Datentyp der Spalte.
- **LOAD TABLE und Datenbankspiegelung**

- **LOAD TABLE und Spaltenstatistiken** Um Histogramme zu Tabellenspalten zu erstellen, erfasst LOAD TABLE statistische Daten über Spalten, wenn Daten geladen werden. Die Histogramme werden vom Optimierer verwendet.

Es folgen zusätzliche Tipps zum Ladevorgang und zu Spaltenstatistiken:

- LOAD TABLE speichert Statistiken für Basistabellen für die zukünftige Verwendung. Die Anweisung speichert keine Statistiken für globale temporäre Tabellen.
- Wenn Sie in eine leere Tabelle laden, die früher Daten enthalten haben könnte, lohnt es sich möglicherweise, zunächst die statistischen Daten für die Spalte zu löschen, bevor Sie die LOAD TABLE-Anweisung ausführen.
- Wenn beim Durchführen einer LOAD TABLE-Anweisung für eine Spalte Spaltenstatistiken existieren, werden die Statistiken für die Spalte *nicht* neu berechnet. Stattdessen werden die Statistiken für die neuen Daten in die vorhandenen Statistiken eingefügt. Wenn die vorhandenen Spaltenstatistiken veraltet sind, ist dies nach dem Laden von neuen Daten in die Spalte weiterhin der Fall. Wenn Sie vermuten, dass die Spaltenstatistiken veraltet sind, sollten Sie sie entweder vor oder nach dem Ausführen der LOAD TABLE-Anweisung aktualisieren.
- LOAD TABLE fügt Statistiken nur hinzu, wenn die Tabelle fünf oder mehr Zeilen umfasst. Wenn die Tabelle mindestens fünf Zeilen enthält, werden Histogramme folgendermaßen geändert:

Daten bereits in der Tabelle?	Histogramm vorhanden?	Maßnahme
Ja	Ja	Änderungen in die bestehenden Histogramme integrieren
Ja	Nein	Keine Histogramme erstellen
Nein	Ja	Änderungen in die bestehenden Histogramme integrieren
Nein	Nein	Neue Histogramme aufbauen

- LOAD TABLE generiert keine Statistiken für Spalten, die NULL für mehr als 90% der zu ladenden Zeilen enthalten.
- **Dynamisch aufgebaute Dateinamen verwenden** Sie können eine LOAD TABLE-Anweisung mit einem dynamisch aufgebauten Dateinamen ausführen, indem Sie den Dateinamen einer Variablen zuordnen und den Variablennamen in der LOAD TABLE-Anweisung verwenden.

Die LOAD TABLE-Anweisung erfordert eine exklusive Schemasperre.

Privilegien

Welches Privileg erforderlich ist, hängt von der Serveroption -gl ab.

Wenn -gl auf ALL gesetzt ist, muss eine der folgenden Bedingungen erfüllt sein:

- Sie sind der Eigentümer der Tabelle
- Sie haben das LOAD-Privileg für die Tabelle
- Sie haben das LOAD ANY TABLE-Systemprivileg
- Sie haben das ALTER ANY TABLE-Systemprivileg

Wenn die Option -gl auf DBA gesetzt ist, müssen Sie das LOAD ANY TABLE-Systemprivileg oder das ALTER ANY TABLE-Systemprivileg haben.

Wenn -gl auf NONE gesetzt ist, wird LOAD TABLE nicht zugelassen.

Beim Laden aus einer Datei von einem Clientcomputer gilt Folgendes:

- Das READ CLIENT FILE-Privileg ist ebenfalls erforderlich.
- Leseprivilegien sind in dem Verzeichnis erforderlich, aus dem gelesen werden soll.
- Die Datenbankoption allow_read_client_file muss aktiviert sein.
- Die gesicherte Funktion read_client_file muss aktiviert sein.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Einfügungen werden nicht in der Transaktionslogdatei aufgezeichnet, wenn die WITH ROW LOGGING-Klausel nicht angegeben ist, sodass die eingefügten Zeilen je nach Protokollierungstyp möglicherweise nicht wiederhergestellt werden, wenn ein Fehler auftritt. Die ursprüngliche Datei ist erforderlich, wenn Sie die Zeilen wiederherstellen müssen und WITH FILE NAME LOGGING verwendet wird. Außerdem sollte die LOAD TABLE-Anweisung ohne WITH ROW LOGGING-Klausel nicht in Datenbanken verwendet werden, die als MobiLink-Clients fungieren, oder in einer Datenbank, die in ein SQL Remote-Replikationssystem eingebunden ist, weil diese Technologien Änderungen über die Analyse der Logdatei replizieren.

Die LOAD TABLE-Anweisung löst keine Trigger aus, die der Tabelle zugeordnet sind.

Ein Checkpoint wird zu Beginn des Vorgangs ausgeführt. Ein zweiter Checkpoint wird am Ende durchgeführt, wenn WITH CHECKPOINT ON angegeben ist.

Spaltenstatistiken werden aktualisiert, wenn eine signifikante Menge von Daten geladen wird.

Siehe auch

- „Datenimport und -export“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Datenimport mit der LOAD TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Unterstützte Zeichensätze“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „UNLOAD-Anweisung“ auf Seite 1098
- „progress_messages-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption -sf“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „allow_read_client_file-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption -gl“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „on_charset_conversion_failure-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Unterstützung von ESRI-Formdateien“ [[SQL Anywhere Server - Unterstützung für räumliche Daten](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Angenommen, Sie erstellen folgendermaßen eine Tabelle namens myTable:

```
CREATE TABLE myTable( a CHAR(100), let_me_default INT DEFAULT 1, c
CHAR(100) );
```

Anschließend erstellen Sie eine Eingabedatei namens `c:\temp\input.txt` und platzieren darin die folgenden Daten:

```
ignore_me, this_is_for_column_c, this_is_for_column_a
```

Nun laden Sie die Daten aus `c:\temp\input.txt` folgendermaßen in myTable:

```
LOAD TABLE myTable ( filler(), c, a ) FROM 'c:\\temp\\input.txt' FORMAT TEXT
DEFAULTS ON;
```

Der Befehl `SELECT * FROM myTable` ergibt folgende Ergebnismenge:

a	let_me_default	c
this_is_for_column_a	1	this_is_for_column_c

Im folgenden Beispiel wird die LOAD TABLE-Anweisung über die Anweisung IMMEDIATE mit einem dynamisch aufgebauten Dateinamen ausgeführt:

```
CREATE PROCEDURE LoadData( IN from_file LONG VARCHAR )
BEGIN
    DECLARE path LONG VARCHAR;
    SET path = 'd:\\data\\' || from_file;
    LOAD TABLE MyTable FROM path;
END;
```

Im folgenden Beispiel werden UTF-8-kodierte Tabellendaten aus einer fiktiven Datei in mytable geladen:

```
LOAD TABLE mytable FROM 'c:\\temp\\mytable_data_in_utf8.dat' ENCODING
'UTF-8';
```

In diesem Beispiel werden Zeilen in der fiktiven Datei `c:\temp\input2.dat` ignoriert, wenn sie mit `//` anfangen.

```
LOAD TABLE GROUP0.Employees FROM 'c:\\temp\\input2.dat' COMMENTS INTRODUCED  
BY '//'
```

LOCK FEATURE-Anweisung

Verhindert die gemeinsame Verwendung einer Datenbankserverfunktion durch gleichzeitig ausgeführte Verbindungen.

Syntax

LOCK FEATURE *feature-name* { **ON** | **OFF** }

feature-name :
'synchronization schema'
| 'all'

Parameter

feature-name Der Name der Funktion, die gesperrt bzw. entsperrt werden soll. Geben Sie "all" an, um alle von einer Verbindung gesperrten Funktionen zu entsperren.

ON | OFF Geben Sie ON an, um zu verhindern, dass andere Verbindungen die Funktion verwenden. Geben Sie OFF an, um die Verwendung dieser Funktion durch Verbindungen zuzulassen.

Bemerkungen

Sie können keine Funktion mehr als einmal für die dieselbe Verbindung sperren. Wenn Sie versuchen, eine Funktion zu entsperren, die nicht durch die aktuelle Verbindung gesperrt ist, und als Funktionsnamen nicht "all" angeben, wird ein Fehler zurückgegeben. Wenn eine Funktion durch zwei oder mehr Verbindungen gesperrt ist, muss sie von allen Verbindungen entsperrt werden, bevor sie von anderen Verbindungen verwendet werden kann. Von einer Verbindung erstellte Funktionssperren werden entfernt, wenn die Verbindung gelöscht wird. Funktionssperren werden entfernt, wenn der Datenbankserver heruntergefahren wird.

Wenn die Synchronisationsschema-Funktion gesperrt ist, können die folgenden Anweisungen nicht von anderen Verbindungen ausgeführt werden:

- START SYNCHRONIZATION SCHEMA CHANGE
- CREATE SYNCHRONIZATION SUBSCRIPTION
- DROP SYNCHRONIZATION SUBSCRIPTION
- ALTER SYNCHRONIZATION SUBSCRIPTION
- ALTER PUBLICATION

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Keine

Siehe auch

- „LOCK TABLE-Anweisung“ auf Seite 949

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung verhindert, dass andere Verbindungen die Synchronisationsschema-Funktion verwenden:

```
LOCK FEATURE 'synchronization schema' ON;
```

LOCK TABLE-Anweisung

Verhindert, dass andere gleichzeitige Transaktionen auf eine Tabelle zugreifen oder sie ändern.

Syntax

```
LOCK TABLE table-name
[ WITH HOLD ]
IN { SHARE | EXCLUSIVE } MODE
```

Parameter

table-name Der Name der Tabelle. Die Tabelle muss eine Basistabelle und keine Ansicht sein. Da Daten in temporären Tabellen zur aktuellen Verbindung lokal sind, hat das Sperren von globalen oder lokalen temporären Tabellen keine Auswirkungen.

WITH HOLD-Klausel Geben Sie diese Klausel an, um die Tabelle bis zum Ende der Verbindung zu sperren. Wenn die Klausel nicht angegeben ist, wird die Sperre aufgehoben, wenn die aktuelle Transaktion festgeschrieben oder zurückgesetzt ist.

IN SHARE MODE-Klausel Geben Sie die Klausel an, um eine gemeinsame Tabellensperre für die Tabelle zu setzen, damit andere Transaktionen nicht die Tabelle ändern, wohl aber darin lesen können. Wenn eine Transaktion eine gemeinsame Tabellensperre für eine Tabelle setzt, kann sie Daten in der Tabelle ändern, sofern keine andere Transaktion eine Sperre für die zu ändernden Zeilen gesetzt hat. Lesesperren für einzelne Zeilen werden nicht gesetzt, wenn die IN SHARE MODE-Klausel ausgewählt ist.

IN EXCLUSIVE MODE-Klausel Geben Sie diese Klausel an, um eine exklusive Tabellensperre für die Tabelle zu setzen, sodass andere Transaktionen nicht auf die Tabelle zugreifen können. Keine andere Transaktion kann Abfragen, Aktualisierungen oder eine andere Aktion für die Tabelle ausführen. Wenn eine Tabelle mit einer Anweisung wie z.B. LOCK TABLE . . . IN EXCLUSIVE MODE exklusiv gesperrt ist, ist es das Standardverhalten, keine Zeilensperren für die Tabelle zu setzen. Dieses Verhalten kann deaktiviert werden, indem Sie die subsume_row_locks-Option auf OFF setzen.

Bemerkungen

Die LOCK TABLE-Anweisung gestattet eine direkte Steuerung des gleichzeitigen Zugriffs auf eine Tabelle, und zwar unabhängig von der aktuellen Isolationsstufe.

Während die Isolationsstufe einer Transaktion im Allgemeinen die Art der Sperren bestimmt, die gesetzt werden, wenn die aktuelle Transaktion eine Anforderung ausführt, ermöglicht die LOCK TABLE-Anweisung eine explizitere Steuerung der Zeilenspernung in einer Tabelle.

Sie können die LOCK TABLE-Anweisung nicht mit einer Ansicht ausführen. Wenn Sie aber die LOCK TABLE-Anweisung mit einer Basistabelle ausführen, wird eine gemeinsame Schemasperre erstellt, die abhängige Ansichten sperrt. Eine gemeinsame Schemasperre bleibt, bis die Transaktion festgeschrieben oder zurückgesetzt ist.

Privilegien

Wenn Sie eine Tabelle im SHARE-Modus sperren möchten, müssen Sie Eigentümer der Tabelle sein, das SELECT-Privileg für die Tabelle haben oder das SELECT ANY TABLE-Systemprivileg haben.

Wenn Sie eine Tabelle im EXCLUSIVE-Modus sperren möchten, muss eine der folgenden Bedingungen erfüllt sein:

- Sie müssen Eigentümer sein.
- Sie müssen für die Tabelle die Privilegien DELETE, UPDATE, INSERT, ALTER, LOAD, und TRUNCATE haben.
- Sie müssen das DELETE ANY TABLE-Systemprivileg, das UPDATE ANY TABLE-Systemprivileg, das INSERT ANY TABLE-Systemprivileg, das ALTER ANY TABLE-Systemprivileg oder das LOAD ANY TABLE-Systemprivileg und das TRUNCATE ANY TABLE-Systemprivileg haben.

Nebenwirkungen

Andere Transaktionen, die auf die gesperrte Tabelle zugreifen müssen, können verzögert oder blockiert werden.

Siehe auch

- „Tabellensperren“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „SELECT-Anweisung“ auf Seite 1020
- „sa_locks-Systemprozedur“ auf Seite 1250
- „Funktionsweise von Sperren“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung hält andere Transaktionen davon ab, die Tabelle 'Customers' während der aktuellen Transaktionen zu ändern:

```
LOCK TABLE GROUPO.Customers  
IN SHARE MODE;
```

LOOP-Anweisung

Wiederholt die Ausführung einer Anweisungsliste.

Syntax

```
[ statement-label : ]
[ WHILE search-condition ] LOOP
  statement-list
END LOOP [ statement-label ]
```

Bemerkungen

Die WHILE- und LOOP-Anweisungen sind Steueranweisungen, mit denen Sie eine Liste von SQL-Anweisungen wiederholt ausführen können, solange eine *search-condition* TRUE ist. Die LEAVE-Anweisung kann verwendet werden, um die Ausführung bei der ersten Anweisung nach END LOOP wieder aufzunehmen.

Wenn das *statement-label* am Ende angegeben ist, muss es mit dem *statement-label* am Anfang übereinstimmen.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „FOR-Anweisung“ auf Seite 857
- „CONTINUE-Anweisung“ auf Seite 581
- „WHILE-Anweisung [T-SQL]“ auf Seite 1123

Standards und Kompatibilität

- **SQL/2008** Die LOOP/END LOOP-Anweisung ist Teil der optionalen SQL/2008-Sprachenfunktion P002 (Verarbeitungsvollständigkeit). In SQL/2008 ist die WHILE DO/END WHILE-Anweisung eine separate Anweisung, die ebenfalls Teil der Sprachenfunktion P002 ist. Die Unterstützung der Syntaxkombination WHILE *search-condition* LOOP in SQL Anywhere ist eine Erweiterung des Herstellers.
- **Transact-SQL** Im Transact-SQL-Dialekt wird LOOP nicht unterstützt. Der Schleifendurchlauf innerhalb von gespeicherten Transact-SQL-Prozeduren erfolgen mit der WHILE-Anweisung für Transact-SQL.

Beispiel

Das folgende Beispielfragment zeigt eine WHILE-Schleife in einer Prozedur.

```
...
SET i = 1;
WHILE i <= 10 LOOP
  INSERT INTO Counters( number ) VALUES ( i );
  SET i = i + 1;
END LOOP;
...
```

Das folgende Beispielfragment zeigt ein benanntes LOOP-Objekt in einer Prozedur.

```
SET i = 1;
lbl:
LOOP
  INSERT
  INTO Counters( number )
  VALUES ( i );
  IF i >= 10 THEN
    LEAVE lbl;
  END IF;
  SET i = i + 1;
END LOOP lbl
```

MERGE-Anweisung

Führt Tabellen, Ansichten und Prozedurergebnisse in einer Tabelle oder Ansicht zusammen.

Syntax

```
MERGE
INTO target-object [ into-column-list ]
USING [ WITH AUTO NAME ] source-object
ON merge-search-condition
merge-operation [...]
[ OPTION ( query-hint, ... ) ]

target-object :
[ userid.]target-table-name [ [ AS ] target-correlation-name ]
| [ userid.]target-view-name [ [ AS ] target-correlation-name ]
| ( select-statement ) [ AS ] target-correlation-name

source-object :
[ userid.]source-table-name [ [ AS ] source-correlation-name ] [ WITH ( table-hints ) ]
| [ userid.]source-view-name [ [ AS ] source-correlation-name ]
| [ userid.]source-mat-view-name [ [ AS ] source-correlation-name ]
| ( select-statement ) [ AS ] source-correlation-name [ using-column-list ]
| procedure

procedure :
[ owner.]procedure-name ( procedure-syntax )
  [ WITH ( column-name data-type, ... ) ]
  [ [ AS ] source-correlation-name ]

merge-search-condition :
search-condition
| PRIMARY KEY

merge-operation :
WHEN MATCHED [ AND search-condition ] THEN match-action
| WHEN NOT MATCHED [ AND search-condition ] THEN not-match-action

match-action :
DELETE
| RAISERROR [ error-number ]
| SKIP
| UPDATE SET set-item, ...
| UPDATE [ DEFAULTS { ON | OFF } ]
```

not-match-action :

```
INSERT
| INSERT [ insert-column-list ] VALUES ( value, ... )
| RAISERROR [ error-number ]
| SKIP
```

set-item :

```
[target-correlation-name.]column-name = { expression | DEFAULT }
| [ owner-name.]target-table-name.column-name = { expression | DEFAULT }
```

insert-column-list :

```
( column-name, ... )
```

query-hint :

```
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| option-name = option-value
```

into-column-list :

```
( column-name, ... )
```

using-column-list :

```
( column-name, ... )
```

error-number: positive Ganzzahl oder Variable über 17000

option-name : Bezeichner

option-value :

```
hostvar (Bezeichner zulässig)
| string
| identifier
| number
```

table-hints: Siehe „[FROM-Klausel](#)“ auf Seite 863.

search-condition : Siehe „[Suchbedingungen](#)“ auf Seite 42.

set-clause-list: Siehe „[SET-Anweisung](#)“ auf Seite 1054.

Parameter

INTO-Klausel Verwenden Sie diese Klausel, um das Zielobjekt für die MERGE-Anweisung zu definieren. *target-object* kann der Name einer Basistabelle, einer regulären Ansicht oder einer abgeleiteten Tabelle sein, jedoch keine materialisierte Ansicht. Die abgeleitete Tabelle oder Ansicht muss einen aktualisierbaren Abfrageblock darstellen. Beispiel: Wenn die Definition der Ansicht oder abgeleiteten Tabelle UNION, INTERSECT, EXCEPT oder GROUP BY enthält, kann sie nicht als Zielobjekt für die MERGE-Anweisung benutzt werden.

Wenn *target-object* eine abgeleitete Tabelle ist, kann die optionale *into-column-list* verwendet werden, um andere Namen für die Spalte der abgeleiteten Tabellen anzugeben. Wenn sie auf diese Art verwendet wird, muss die *into-column-list* eine Größe haben, die zur Spaltenliste der abgeleiteten Tabelle passt. Außerdem muss die Sortierung der beiden Tabellen gleich sein.

Wenn *target-object* eine Basistabelle oder Ansicht ist, kann *into-column-list* verwendet werden, um eine Teilmenge der Spalten der Tabelle oder Ansicht anzugeben, die für den Rest der MERGE-Anweisung relevant sind.

Der Datenbankserver benutzt *into-column-list* für folgende Auflösungen:

- UPDATE ohne SET-Klausel in WHEN MATCHED-Klausel.
- INSERT ohne VALUES-Klausel in einer WHEN NOT MATCHED-Klausel
- PRIMARY KEY-Suchbedingung in der ON-Klausel.
- WITH AUTO NAME-Klausel in der USING-Klausel

Wenn Sie die *into-column-list* nicht angeben, wird angenommen, dass die *into-column-list* alle Spalten des *target-object* enthält.

USING-Klausel Verwenden Sie diese Klausel, um die Quelle der Daten für die Zusammenführung festzulegen. *source-object* kann eine Basistabelle (einschließlich Tabellen-Hints), eine Ansicht, eine materialisierte Ansicht, eine abgeleitete Tabelle oder eine Prozedur sein. Wenn *source-object* eine abgeleitete Tabelle ist, können Sie eine *using-column-list* angeben. Alle Spalten des *source-object* werden verwendet, wenn Sie nicht eine *using-column-list* angeben.

WITH AUTO NAME-Klausel Verwenden Sie diese Klausel, damit der Server automatisch Spaltennamen für die Übereinstimmung der Spalten in der *into-column-list* mit den Spalten im *target-object* für den Zusammenführungsvorgang benutzt. Die folgenden Beispiele sind gleichwertig und zeigen, dass die Reihenfolge der Spalten in der *into-column-list* geändert wird, damit eine Übereinstimmung mit den Namen der Spalten im *source-object* erzielt wird, wenn WITH AUTO NAME angegeben ist:

```
... INTO T ( Name, ID, Description )
    USING WITH AUTO NAME ( SELECT Description, Name, ID FROM PRODUCTS WHERE
Description LIKE '%cap%' )
... INTO T ( Description, Name, ID )
    USING ( SELECT Description, Name, ID FROM PRODUCTS WHERE Description LIKE
'%cap%' )
```

ON-Klausel Verwenden Sie diese Klausel, um die Bedingung für die Übereinstimmung einer Zeile im *source-object* mit Zeilen im *target-object* anzugeben.

Sie können ON PRIMARY KEY angeben, um eine Übereinstimmung von *source-object*-Zeilen basierend auf der Primärschlüsseldefinition des *target-object* zu erzielen. *source-object* benötigt keinen Primärschlüssel. *target-object* muss hingegen einen Primärschlüssel haben. Wenn Sie ON PRIMARY KEY angeben:

- Ein Fehler wird zurückgegeben, wenn *target-object* keine Basistabelle ist oder keinen Primärschlüssel hat.
- Ein Fehler wird zurückgegeben, wenn mindestens ein Primärschlüssel in der *into-column-list* nicht enthalten ist.
- Die Anzahl der Spalten in *into-column-list* und *using-column-list* kann verschieden sein, wenn jede Primärschlüsselspalte in *into-column-list* eine passende Spalte in *using-column-list* hat. Beispiel: Wenn

into-column-list (I1, I2, I3) enthält, *using-column-list* (U1, U2) enthält und die Primärschlüsselspalten (I2, I3) sind, wird ein Fehler zurückgegeben, weil Spalte (I3) des *target-object*-Primärschlüssels keine Übereinstimmung in *using-column-list* hat.

- Unbeschadet der Definition des Primärschlüssels basiert die Übereinstimmung der Primärschlüsselspalten in *into-column-list* mit Ausdrücken in *using-column-list* auf der Position der Primärschlüsselspalten in *into-column-list*. Beispiel: Angenommen, der Primärschlüssel für *target-object* ist definiert als (B, C) und *into-column-list* ist (E, C, F, A, D, B). Wenn ON PRIMARY KEY angegeben ist, wird *target-object* Spalte B mit dem sechsten Element anhand von *using-column-list* verglichen, weil Spalte B an der sechsten Stelle von *into-column-list* steht. Ebenso gilt: Spalte C von *target-object* wird mit dem zweiten Element von *using-column-list* verglichen.

ON PRIMARY KEY ist eine syntaktische Kurzform für eine entsprechende ON-Bedingung. Beispiel: Angenommen, *into-column-list* ist (I1, I2, .. In) und die entsprechend übereinstimmende Spaltenliste *using-column-list* ist (U1, U2, .. Um). Außerdem wird angenommen, dass die Primärschlüsselspalten von *target-object* I1, I2, I3 sind und alle Primärschlüsselspalten in *into-column-list* enthalten sind. In diesem Fall wird *merge-search-condition* als "I1=U1 AND I2=U2 AND I3=U3"-Konjunkt definiert.

WHEN MATCHED- und WHEN NOT MATCHED-Klauseln Verwenden Sie die WHEN MATCHED- und WHEN NOT MATCHED-Klauseln, um eine Aktion zu definieren, wenn eine Zeile aus *source-object* mit einer Zeile in *target-object* übereinstimmt oder nicht übereinstimmt. Die Aktion wird nach dem THEN-Schlüsselwort definiert. Sie können die Aktionen für Teilmengen von übereinstimmenden oder nicht übereinstimmenden Zeilen steuern, indem Sie eine zusätzliche AND-Klausel eingeben.

Die ON-Klausel legt fest, wie Zeilen aus *source-object* in übereinstimmenden und nicht übereinstimmenden Zeilen aufgeteilt werden. Eine Zeile in *source-object* wird als übereinstimmende Zeile angesehen, wenn die ON-Klausel für mindestens eine Zeile in *target-object* TRUE ist. Eine Zeile in *source-object* wird als nicht übereinstimmende Zeile angesehen, wenn die ON-Klausel für eine Zeile in *target-object* nicht TRUE ist. Verwenden Sie WHEN MATCHED- und WHEN NOT MATCHED-Klauseln, um übereinstimmende und nicht übereinstimmende Zeilen in getrennte Teilmengen zu zerlegen. Jede Teilmenge wird über eine WHEN-Klausel verarbeitet. WHEN MATCHED- und WHEN NOT MATCHED-Klauseln werden in der Reihenfolge abgearbeitet, in der sie in der MERGE-Anweisung erscheinen.

Die in der AND-Klausel einer WHEN MATCHED- oder WHEN NOT MATCHED-Klausel angegebene Suchbedingung legt fest, ob eine in Frage kommende Zeile durch eine spezielle Klausel verarbeitet wird. Wenn Sie eine WHEN MATCHED- oder WHEN NOT MATCHED-Klausel ohne die AND-Klausel angeben, wird die Suchbedingung in der AND-Klausel als TRUE angenommen. Wenn eine Zeile der AND-Bedingung für mehr als eine Klausel entspricht, wird die Zeile von der Klausel verarbeitet, die in der MERGE-Anweisung zuerst kommt.

Ein Fehler wird zurückgegeben, wenn eine der WHEN MATCHED-Klauseln dieselbe *target-object*-Zeile mehr als einmal verarbeitet. Eine *target-object*-Zeile kann mehr als einmal zu derselben Teilmenge derselben WHEN MATCHED-Klausel gehören, wenn sie mit zwei unterschiedlichen Eingabezeilen aus *source-object* übereinstimmt.

Im folgenden Beispiel wird ein Fehler zurückgegeben, weil die Zeile mit ID 300 aus *target-object* Products mit 111 Zeilen in *source-object* von SalesOrderItems übereinstimmt. Alle Übereinstimmungen gehören entsprechend der WHEN MATCHED THEN UPDATE-Klausel zu derselben Teilmenge.

```
MERGE INTO GROUPO.Products
  USING GROUPO.SalesOrderItems S
  ON S.ProductID = Products.ID
  WHEN MATCHED THEN UPDATE SET Products.Quantity = 20;
```

WHEN MATCHED: Bei einer übereinstimmenden Zeile können Sie eine der folgenden Aktionen für *match-action* festlegen:

- **DELETE** Geben Sie DELETE an, um die Zeile aus *target-object* zu löschen.
- **RAISERROR** Geben Sie RAISERROR an, um den Zusammenführungsvorgang zu beenden, Änderungen zurückzusetzen und eine Fehlermeldung zurückzugeben. Standardmäßig gilt: Wenn Sie RAISERROR angeben, gibt der Datenbankserver SQLSTATE 23510 und SQLCODE -1254 zurück.

Optional können Sie den zurückgegebenen SQLCODE anpassen, indem Sie den *error-number*-Parameter nach dem RAISERROR-Schlüsselwort angeben. Der benutzerdefinierte SQLCODE muss eine positive Ganzzahl größer als 17.000 sein und kann entweder als Zahl oder als Variable angegeben werden. Wenn Sie einen benutzerdefinierten SQLCODE angeben, ist die zurückgegebene Nummer negativ.

Beispiel: Wenn Sie WHEN MATCHED AND search-condition THEN RAISERROR 17001 angeben und eine Zeile gefunden wird, die den Bedingungen der WHEN-Klausel entspricht, schlägt der MERGE-Vorgang fehl, die Änderungen werden zurückgesetzt und der zurückgegebene Fehler hat die Nummern SQLSTATE 23510 und SQLCODE -17001.

- **SKIP** Geben Sie SKIP an, um die Zeile zu überspringen. Es findet keine Aktion statt.
- **UPDATE** Geben Sie UPDATE SET an, um die Zeile mit den Werten in *set-item* zu aktualisieren. *set-item* ist ein einfacher Zuordnungsausdruck, in dem eine Spalte auf den Wert von *expression* gesetzt wird. Es gibt keine Einschränkungen für *expression*. Sie können auch DEFAULT angeben, um die Spalte auf den für sie definierten Standardwert zu setzen.

Beispiel: UPDATE SET target-column1=DEFAULT, target-column2=source-column2 setzt target-column1 auf ihren Standardwert und target-column2 auf denselben Wert wie die Änderungszeile aus source-column2 in *source-object*.

Wenn Sie die SET-Klausel nicht angeben, wird die SET-Klausel durch *into-column-list* und *using-column-list* definiert. Beispiel: Wenn die *into-column-list* (I1, I2, .. In) und die *using-column-list* (U1, U2, .. Un) ist, wird die SET-Klausel als "SET I1=U1 , I2=U2 , .. In=Un" angenommen.

WHEN NOT MATCHED: Bei einer nicht übereinstimmenden Zeile können Sie eine der folgenden Aktionen für *non-match-action* festlegen:

- **INSERT** Geben Sie INSERT...VALUES an, um die Zeile mit den angegebenen Werten einzufügen. Wenn Sie die INSERT-Klausel ohne VALUES-Klausel angeben, wird die VALUES-Klausel durch *into-column-list* und *using-column-list* definiert. Beispiel: Wenn die *into-column-list* (I1, I2, .. In) und die *using-column-list* (U1, U2, .. Un) ist, entspricht INSERT ohne eine VALUES-Klausel INSERT (I1, I2, .. In) VALUES (U1, U2, .. Un).
- **RAISERROR** Geben Sie RAISERROR an, um den Zusammenführungsvorgang zu beenden, Änderungen zurückzusetzen und eine Fehlermeldung zurückzugeben. Wenn Sie RAISERROR

angeben, gibt der Datenbankserver standardmäßig SQLSTATE 23510 und SQLCODE -1254 zurück. Optional können Sie den zurückgegebenen SQLCODE anpassen, indem Sie den *error-number*-Parameter nach dem RAISERROR-Schlüsselwort angeben. Der benutzerdefinierte SQLCODE muss eine positive Ganzzahl größer als 17.000 sein und kann entweder als Zahl oder als Variable angegeben werden. Wenn Sie einen benutzerdefinierten SQLCODE angeben, ist die zurückgegebene Nummer negativ.

Beispiel: Wenn Sie WHEN NOT MATCHED AND search-condition THEN RAISERROR 17001 angeben und eine Zeile gefunden wird, die den Bedingungen der WHEN-Klausel entspricht, schlägt der MERGE-Vorgang fehl, die Änderungen werden zurückgesetzt und der zurückgegebene Fehler hat die Nummern SQLSTATE 23510 und SQLCODE -17001.

- **SKIP** Geben Sie SKIP an, um die Zeile zu überspringen. Es findet keine Aktion statt.

OPTION-Klausel Verwenden Sie diese Klausel, um Hints für das Ausführen der Anweisung anzugeben. Die folgenden Hints werden unterstützt:

- MATERIALIZED VIEW OPTIMIZATION *option-value*
- FORCE OPTIMIZATION
- *option-name* = *option-value*. Die Spezifikation OPTION(*isolation_level* = ...) im Abfragetext hebt alle anderen Festlegungen der Isolationsstufe für eine Abfrage auf.

Bemerkungen

Zeilen im *source-object* werden mit Zeilen im *target-object* verglichen und als übereinstimmend oder nicht übereinstimmend befunden, je nachdem ob sie den Bedingungen in der ON-Klausel entsprechen oder nicht. Zeilen in *source-object* werden als übereinstimmend angesehen, wenn es zumindest eine Zeile in der Zieltabelle *target-table* gibt, sodass die Suchbedingung für die Zusammenführung (*merge-search-condition*) als TRUE ausgewertet wird. Übereinstimmende Zeilen und nicht übereinstimmende Zeilen werden dann nach den Aktionen gruppiert, die für sie in den Klauseln WHEN MATCHED und WHEN NOT MATCHED gemäß den durch die AND-Klauseln festgelegten Suchbedingungen festgelegt wurden. Das Gruppieren nach Aktionen für übereinstimmende und nicht übereinstimmende Zeilen wird als **Verzweigen** bezeichnet und die einzelnen Gruppen werden **Verzweigung** genannt.

Nachdem das Verzweigen abgeschlossen ist, beginnt die Datenbank mit dem Ausführen der Aktion, die für die Zeilen der Verzweigung definiert wurden. Die Verzweigungen werden in der Reihenfolge verarbeitet, in der sie auftreten, und dies entspricht der Reihenfolge, in der die WHEN-Klauseln in der Anweisung enthalten sind. Wenn während des Verzweigens mehr als eine Zeile im *source-object* eine Aktion hat, die für dieselbe Zeile im *target-object* definiert wurde, schlägt der MERGE-Vorgang fehl und eine Fehlermeldung wird zurückgegeben. Damit wird verhindert, dass der MERGE-Vorgang mehr als eine Aktion für eine Zeile im *target-object* ausführt.

Während der Verarbeitung der Verzweigungen werden die Einfüge-, Aktualisierungs- und Löschaktionen im Transaktionslog wie die entsprechenden INSERT-, UPDATE- und DELETE-Anweisungen gespeichert.

Privilegien

Welche Privilegien erforderlich sind, wird zur Ausführungszeit ermittelt und hängt ab von den angegebenen Objekten sowie den Vorgängen, die die Zusammenführung zur Folge hat:

- **auswählen *source-object* oder *target-object* auswählen** Sie müssen Eigentümer der Objekte sein, das SELECT ANY TABLE-Systemprivileg haben oder das SELECT-Privileg für das Zielobjekt haben.
- **Zeilen in *target-object* einfügen** Sie müssen Eigentümer von *target-object* sein, das INSERT ANY TABLE-Systemprivileg haben oder das INSERT-Privileg für das Zielobjekt haben.
- **Zeilen in *target-object* aktualisieren** Sie müssen Eigentümer von *target-object* sein, das UPDATE ANY TABLE-Systemprivileg haben oder das UPDATE-Privileg für das Zielobjekt haben.
- **Zeilen aus *target-object* löschen** Sie müssen Eigentümer von *target-object* sein, das DELETE ANY TABLE-Systemprivileg haben oder das DELETE-Privileg für das Zielobjekt haben.

Das EXECUTE-Privileg ist für jede Prozedur erforderlich, die in der MERGE-Anweisung referenziert wird.

Nebenwirkungen

Trigger, die für das *target-object* definiert wurden, werden ausgelöst.

Siehe auch

- „Importieren von Daten mit der MERGE-Anweisung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „UPDATE-Anweisung“ auf Seite 1109
- „INSERT-Anweisung“ auf Seite 917
- „DELETE-Anweisung“ auf Seite 791
- „SELECT-Anweisung“ auf Seite 1020
- „Suchbedingungen“ auf Seite 42
- „MERGE-Anweisung für die Tabelle '%1' ist aufgrund einer RAISERROR-Spezifikation in der Anweisung fehlgeschlagen“ [[Fehlermeldungen](#)]
- Die RAISERROR-Aktion verwenden [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Die MERGE-Anweisung umfasst die Funktionen F312 und F313 des SQL/2008-Standards. Die MERGE-Anweisung in SQL Anywhere entspricht der Spezifikation für die MERGE-Anweisung im SQL/2008-Standard und enthält Erweiterungen. Zu den SQL Anywhere-spezifischen Erweiterungen für die MERGE-Anweisung gehören:
 - DELETE in einer WHEN MATCHED-Klausel.
 - RAISERROR in einer WHEN [NOT] MATCHED-Klausel.
 - SKIP in einer WHEN [NOT] MATCHED-Klausel
 - OPTION-Klausel
 - PRIMARY KEY-Klausel
 - DEFAULTS-Klausel

- INSERT-Klausel ohne VALUES-Klausel.
- WITH AUTO NAME-Klausel
- UPDATE-Klausel ohne SET-Klausel.

Beispiele

Im folgenden Beispiel wird eine Zeile aus einer abgeleiteten Tabelle in die Products-Tabelle zusammengeführt, wobei ein neues T-Shirt mit denselben Attributen wie ein vorhandenes T-Shirt, aber mit neuer Farbe, Menge und Produktbezeichner hinzugefügt wird. In diesem Beispiel gilt: Wenn das Produkt mit der Identifizierungsnummer 304 in der Products-Tabelle bereits existiert, wird die Zeile nicht eingefügt:

```
MERGE INTO Products ( ID, Name, Description, Size, Color, Quantity,
UnitPrice, Photo )
  USING WITH AUTO NAME (
    SELECT 304 AS ID,
           'Purple' AS Color,
           100 AS Quantity,
           Name,
           Description,
           Size,
           UnitPrice,
           Photo
    FROM Products WHERE Products.ID = 300 ) AS DT
  ON PRIMARY KEY
  WHEN NOT MATCHED THEN INSERT;
```

MESSAGE-Anweisung

Zeigt eine Meldung an.

Syntax

```
MESSAGE expression
[ TYPE { INFO | ACTION | WARNING | STATUS } ]
[ TO { CONSOLE
  | CLIENT [ FOR { CONNECTION conn-id-number [ IMMEDIATE ] | ALL } ]
  | [ EVENT | SYSTEM ] LOG }
  [ DEBUG ONLY ] ]
```

connection-ID : integer

Parameter

TYPE-Klausel Diese Klausel gibt den Meldungstyp an. Die Clientanwendung muss entscheiden, wie die Meldung zu verarbeiten ist. Wenn Sie zum Beispiel TO CLIENT angeben, zeigt Interactive SQL Meldungen an folgenden Stellen an:

- **INFO** Die Registerkarte **Meldungen**. INFO ist der Standardtyp.
- **ACTION** Ein Fenster mit einer **OK**-Schaltfläche.
- **WARNING** Ein Fenster mit einer **OK**-Schaltfläche.

- **STATUS** Die Registerkarte **Meldungen**.

TO-Klausel Diese Klausel gibt das Ziel einer Meldung an:

- **CONSOLE** Meldungen werden an das Meldungsfenster des Datenbankservers und in die Meldungslogdatei des Datenbankservers geschickt, wenn eine definiert wurde. CONSOLE ist der Standardwert.
- **CLIENT** Sendet Meldungen an die Clientanwendung. Ihre Anwendung muss entscheiden, wie die Meldung zu verarbeiten ist. Sie können TYPE als Information verwenden, auf deren Basis diese Entscheidung getroffen werden kann.
- **LOG** Sendet Meldungen an die Datenbankserver-Meldungslogdatei, die durch die Option -o angegeben wird. Wenn EVENT oder SYSTEM angegeben ist, wird die Meldung auch in das Meldungsfenster des Datenbankservers geschrieben sowie unter der Ereignisquelle **SQLANY 16.0 Admin** in das Windows-Ereignisprotokoll und unter dem Namen **SQLANY 16.0 Admin** (*servername*) in das Unix-Syslog auf 32-Bit-Datenbankservern. Auf 64-Bit-Datenbankservern ist **SQLANY64 16.0 Admin** die Ereignisquelle. Meldungen im Datenbankserver-Meldungslog werden folgendermaßen gekennzeichnet:
 - **i** Meldungen vom Typ INFO oder STATUS
 - **w** Meldungen vom Typ WARNING
 - **e** Meldungen vom Typ ACTION

FOR-Klausel Diese Klausel bestimmt bei TO CLIENT-Nachrichten, welche Verbindungen eine Benachrichtigung zu der Nachricht erhalten. Standardmäßig empfängt die Verbindung die Nachricht, wenn das nächste Mal eine SQL-Anweisung oder eine WAITFOR DELAY-Anweisung ausgeführt wird.

- **CONNECTION *conn-id-number*** Gibt die Verbindungs-ID-Nummer des Empfängers an. Wenn IMMEDIATE angegeben ist, erhält die Verbindung die Nachricht innerhalb weniger Sekunden, gleichgültig wann die SQL-Anweisung ausgeführt wird.
- **ALL** Gibt an, dass alle offenen Verbindungen die Nachricht erhalten.

DEBUG ONLY Mit dieser Klausel können Sie kontrollieren, ob Meldungen, die während der Fehlersuche erzeugt werden, in gespeicherten Prozeduren und Triggern aktiviert oder deaktiviert werden. Hierfür ändern Sie die Einstellung für die Option debug_messages. Wenn DEBUG ONLY angegeben ist, wird die MESSAGE-Anweisung nur dann ausgeführt, wenn die Option debug_messages auf ON gesetzt ist.

Hinweis

DEBUG ONLY-Meldungen sind kostengünstig, wenn die Option debug_messages auf "Off" gesetzt wurde, sodass diese Anweisungen auf einem Produktionssystem normalerweise in gespeicherten Prozeduren belassen werden können. Dort, wo sie häufig ausgeführt werden würden, sollten Sie jedoch mit Vorsicht eingesetzt werden, weil sonst mit Performance-Einbußen gerechnet werden muss.

Bemerkungen

Die MESSAGE-Anweisung zeigt eine Meldung an, die jeder beliebige Ausdruck sein kann. Mit Klauseln können der Nachrichtentyp und der Ort, wo die Meldung angezeigt wird, festgelegt werden.

Wenn die Größe von *expression* die Seitengröße der Datenbank überschreitet, wird *expression* auf die Seitengröße der Datenbank gekürzt. Um die für die Datenbank geltende Seitengröße zu überprüfen, können Sie die Datenbankeigenschaft PageSize abfragen (SELECT DB_PROPERTY('PageSize');).

Die Prozedur, die eine MESSAGE...TO CLIENT-Anweisung ausführt, muss einer Verbindung zugeordnet sein.

Beispielsweise wird das Meldungsfenster im folgenden Beispiel nicht angezeigt, da das Ereignis außerhalb einer Verbindung auftritt.

```
CREATE EVENT CheckIdleTime
TYPE ServerIdle
WHERE event_condition( 'IdleTime' ) > 100
HANDLER
BEGIN
    MESSAGE 'Idle server' TYPE WARNING TO CLIENT;
END;
```

Im folgenden Beispiel hingegen wird die Meldung im Meldungsfenster des Datenbankservers ausgegeben.

```
CREATE EVENT CheckIdleTime2
TYPE ServerIdle
WHERE event_condition( 'IdleTime' ) > 100
HANDLER
BEGIN
    MESSAGE 'Idle server' TYPE WARNING TO CONSOLE;
END;
```

Gültige Ausdrücke können eine Zeichenfolge in Anführungszeichen, eine Konstante, Variable oder Funktion enthalten.

Die FOR-Klausel kann verwendet werden, um eine andere Anwendung über ein Ereignis zu benachrichtigen, das auf dem Datenbankserver entdeckt wurde, ohne dass die Anwendung explizit nach dem Ereignis suchen muss. Wenn die FOR-Klausel verwendet wird, erhalten Empfänger die Nachricht, wenn sie das nächste Mal eine SQL-Anweisung ausführen. Wenn der Empfänger derzeit eine SQL-Anweisung ausführt, erhält er die Nachricht nach Ausführung der Anweisung. Wenn die auszuführende Anweisung ein Aufruf einer gespeicherten Prozedur ist, erhält der Empfänger die Nachricht vor Beendigung des Aufrufs.

Wenn eine Anwendung eine Benachrichtigung innerhalb kurzer Frist nach dem Absenden der Nachricht benötigt und die Verbindung keine SQL-Anweisungen ausführt, verwenden Sie die Klausel IMMEDIATE, um die Clientbenachrichtigung zu implementieren, und nicht mehrere, gleichzeitige WAITFOR DELAY-Anweisungen.

In der Regel werden Nachrichten, die mit der IMMEDIATE-Klausel verwendet werden, in weniger als fünf Sekunden versendet, auch wenn die Zielverbindung keine Datenbankserveranfragen durchführt. Die Nachrichtenzustellung kann verzögert werden, wenn die Clientverbindung mehrere Anfragen pro

Sekunde durchführt, sehr umfangreiche BLOB-Daten erhält oder der Nachrichten-Callback des Clients länger als eine Sekunde ausgeführt wird. Außerdem kann das Senden von mehr als einer IMMEDIATE-Nachricht an eine einzige Verbindung alle zwei Sekunden die Nachrichtenzustellung verzögern und eine Fehlermeldung bewirken. Wenn die Clientverbindung getrennt ist, kann eine erfolgreiche MESSAGE...IMMEDIATE-Anweisung eventuell nicht zugestellt werden.

Ohne IMMEDIATE-Klausel versendete Nachrichten werden nur zugestellt, wenn der Client eine spezifische Anforderung oder eine WAITFOR DELAY-Anweisung ausführt. Daher ist die Zustellungszeit von Nachrichten unbegrenzt.

Die IMMEDIATE-Klausel erfordert einen DBLib, ODBC, oder SQL Anywhere JDBC-Treiber der Version SQL Anywhere 11 oder später. Die IMMEDIATE-Klausel wird von Unix-Clientbibliotheken ohne Thread nicht unterstützt. Eine Fehlermeldung wird generiert, wenn eine Nachricht an eine Zielverbindung gesendet wird, die die IMMEDIATE-Klausel nicht unterstützt. Eine Fehlermeldung wird generiert, wenn eine IMMEDIATE-Nachricht an die dieselbe Verbindung gerichtet ist, die die MESSAGE-Anweisung ausführt.

```
MESSAGE 'Please disconnect' TYPE WARNING TO CLIENT
FOR CONNECTION 16 IMMEDIATE;
```

Ein MESSAGE...TO CLIENT-Ausdruck kann auf 2048 Byte gekürzt werden. Für Meldungen, die mit der IMMEDIATE-Klausel gesendet werden, kann der Nachrichtenausdruck auf die Paketgröße der Verbindung oder 2048 Byte gekürzt werden, je nachdem, was kleiner ist.

Embedded SQL- und ODBC-Clients erhalten Nachrichten über Nachrichten-Callbackfunktionen. Diese Funktionen müssen in jedem Fall registriert werden. In Embedded SQL wird das Nachrichten-Callback mit db_register_a_callback mithilfe des DB_CALLBACK_MESSAGE-Parameters registriert. In ODBC wird das Nachrichten-Callback mit SQLSetConnectAttr mithilfe des SA_REGISTER_MESSAGE_CALLBACK-Parameters registriert.

Privilegien

Wenn Sie eine MESSAGE-Anweisung ausführen möchten, die eine FOR-Klausel, eine TO EVENT LOG-Klausel oder eine TO SYSTEM LOG-Klausel enthält, benötigen Sie das SERVER OPERATOR-Systemprivileg. Andernfalls sind für diese Anweisung keine Privilegien erforderlich.

Nebenwirkungen

Keine.

Siehe auch

- „sa_conn_info-Systemprozedur“ auf Seite 1183
- „CREATE PROCEDURE-Anweisung“ auf Seite 681
- „debug_messages-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „db_register_a_callback-Funktion“ [*SQL Anywhere Server - Programmierung*]
- „WAITFOR-Anweisung“ auf Seite 1120

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Prozedur zeigt eine Nachricht im Meldungsfenster des Datenbankservers an:

```
CREATE PROCEDURE message_text( )
BEGIN
MESSAGE 'The current date and time: ', Now( );
END;
```

Die folgende Anweisung zeigt die Zeichenfolge The current date and time, gefolgt vom aktuellen Datum und der aktuellen Uhrzeit im Meldungsfenster des Datenbankservers an.

```
CALL message_text( );
```

NOTIFY TRACE EVENT-Anweisung

Protokolliert ein benutzerdefiniertes Trace-Ereignis in einer Trace-Sitzung.

Syntax

```
NOTIFY TRACE EVENT trace-event-name ( [ param1 [ ,... ] ] )
```

Parameter

trace-event-name Der Trace-Ereignisname muss der Name eines benutzerdefinierten Trace-Ereignisses sein. Es darf sich nicht um ein systemdefiniertes Trace-Ereignis handeln.

param1 Die Werte der Trace-Ereignisfelder.

Bemerkungen

Diese Anweisung dient zum Benachrichtigen aller Sitzungen, die das angegebene Trace-Ereignis enthalten. Wenn ein Trace-Ereignis nicht von einer Sitzung verfolgt wird, hat diese Anweisung keine Auswirkungen und die Parameter werden nicht ausgewertet (z.B. durch einen Aufruf an eine benutzerdefinierte Funktion).

Systemprivilegien

Sie müssen das NOTIFY TRACE EVENT-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Ereignisprotokollierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE TEMPORARY TRACE EVENT-Anweisung“ auf Seite 753
- „CREATE TEMPORARY TRACE EVENT SESSION-Anweisung“ auf Seite 755
- „ALTER TRACE EVENT SESSION-Anweisung“ auf Seite 538
- „DROP TRACE EVENT-Anweisung“ auf Seite 835
- „DROP TRACE EVENT SESSION-Anweisung“ auf Seite 836
- „sp_trace_events-Systemprozedur“ auf Seite 1407
- „sp_trace_event_fields-Systemprozedur“ auf Seite 1401
- „sp_trace_event_sessions-Systemprozedur“ auf Seite 1406
- „sp_trace_event_session_events-Systemprozedur“ auf Seite 1402
- „sp_trace_event_session_target_options-Systemprozedur“ auf Seite 1403
- „Dienstprogramm für die Verwaltung von ETD-Dateien (dbmanagetd)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgenden Anweisungen protokollieren Ereignisse in der aktuellen (fiktiven) Ereignis-Trace-Sitzung my_event.

```
NOTIFY TRACE EVENT my_event( 1, 'Hello world' ); -- trigger user-defined
trace events
NOTIFY TRACE EVENT my_event( 2, 'Hello world 2' );
NOTIFY TRACE EVENT my_event( 3, 'Hello world 3' );
```

OPEN-Anweisung [ESQL] [SP]

Öffnet einen zuvor deklarierten Cursor, um auf Informationen aus der Datenbank zuzugreifen.

Syntax 1 [ESQL]

```
OPEN cursor-name
[ USING { DESCRIPTOR sqlda-name | hostvar, ... } ]
[ WITH HOLD ]
[ ISOLATION LEVEL isolation-level ]
[ BLOCK n ]
```

Syntax 2 [SP]

```
OPEN cursor-name
[ WITH HOLD ]
[ ISOLATION LEVEL isolation-level ]
```

cursor-name : *identifier* | *hostvar*

sqlda-name : *identifier*

isolation-level : 0 | 1 | 2 | 3 | **SNAPSHOT** | **STATEMENT SNAPSHOT** | **READONLY STATEMENT SNAPSHOT**

Parameter

USING DESCRIPTOR-Klausel Die Klausel USING DESCRIPTOR gilt nur für Embedded SQL. Sie gibt die Hostvariablen an, die an die Platzhalter-Indikatorvariablen in der SELECT-Anweisung gebunden sind, für die der Cursor deklariert wurde.

OPEN...USING kann nicht in einer gespeicherten Prozedur benutzt werden.

WITH HOLD-Klausel Standardmäßig werden alle Cursor automatisch bei Abschluss der aktuellen Transaktion (COMMIT oder ROLLBACK) geschlossen. Wenn die optionale WITH HOLD-Klausel angegeben wurde, dann wird der Cursor für nachfolgende Transaktionen offen gehalten. Der Cursor bleibt bis zum Ende der aktuellen Verbindung oder bis zu einer expliziten CLOSE-Anweisung geöffnet. Cursor werden automatisch geschlossen, wenn die Verbindung zur Datenbank beendet wird.

Bei COMMIT oder ROLLBACK werden alle von der Verbindung gehaltenen langfristigen Zeilensperren freigegeben, einschließlich der Zeilen, die die Ergebnismenge eines WITH HOLD-Cursors darstellen. Dagegen bleiben Cursor-Stabilitätssperren, die auf den Isolationsstufen 1, 2, und 3 gesetzt wurden, über die gesamte Lebensdauer des Cursors erhalten und werden erst freigegeben, wenn der Cursor geschlossen oder die Verbindung beendet wird.

Nach Abschluss einer ROLLBACK-Anweisung sind Inhalt und dessen Positionierung innerhalb eines WITH HOLD-Cursor nicht vorhersehbar und können nicht garantiert werden. Mithilfe der Option `ansi_close_cursors_on_rollback` können sie steuern, ob ein WITH HOLD-Cursor durch eine ROLLBACK-Anweisung automatisch geschlossen wird.

ISOLATION LEVEL-Klausel Mit der ISOLATION LEVEL-Klausel kann dieser Cursor auf einer anderen Isolationsstufe geöffnet werden als der, die durch die Option `isolation_level` zurzeit festgelegt ist. Alle Vorgänge dieses Cursors werden ungeachtet der Optionseinstellung auf der angegebenen Isolationsstufe ausgeführt. Wenn diese Klausel nicht angegeben ist, dann entspricht die Isolationsstufe des Cursors während der gesamten Zeit, die der Cursor geöffnet ist, dem Wert der Option `isolation_level`, der eingestellt war, als der Cursor geöffnet wurde.

Folgende Werte werden unterstützt:

- 0
- 1
- 2
- 3
- SNAPSHOT
- STATEMENT SNAPSHOT
- READONLY STATEMENT SNAPSHOT

Der Cursor wird vor der ersten Zeile positioniert.

BLOCK-Klausel Diese Klausel kann nur in Embedded SQL verwendet werden. Eine Clientanwendung kann mehr als eine Zeile gleichzeitig abrufen. Dies wird Blockabruf, Prefetch-Vorgang oder Mehrzeilenabruf genannt. Die BLOCK-Klausel kann die Anzahl von Prefetch-Zeilen verringern. Die

Angabe der BLOCK-Klausel in OPEN ist dasselbe wie ein Angeben der BLOCK-Klausel in jedem FETCH.

Bemerkungen

Die OPEN-Anweisung öffnet den Cursor mit dem angegebenen Namen. Der Cursor muss zuvor deklariert worden sein.

Die OPEN-Anweisung kann eine SQL-Warnung zurückgeben, wenn der Cursortyp nicht zu den Eigenschaften der dem Cursor zugrunde liegenden Anweisung passt.

Ist der Cursor für eine CALL-Anweisung geöffnet, dann führt OPEN dazu, dass die Prozedur solange abgearbeitet wird, bis die erste Ergebnismenge angetroffen wird (bei einer SELECT-Anweisung ohne INTO-Klausel). Wenn die Prozedur abgearbeitet ist und keine Ergebnismenge ermittelt wurde, wird die Warnung `SQLSTATE_PROCEDURE_COMPLETE` gesetzt.

- **Embedded SQL-Verwendung** Nach der erfolgreichen Ausführung der OPEN-Anweisung wird das Feld `sqlerrd[3]` des SQLCA (SQLIOESTIMATE) mit einer geschätzten Anzahl der Eingabe/Ausgabevorgänge gefüllt, die notwendig sind, um alle Zeilen der Abfrage abzurufen. Außerdem wird in das Feld `sqlerrd[2]` des SQLCA (SQLCOUNT) die tatsächliche Anzahl der Zeilen im Cursor (ein Wert größer oder gleich 0) oder eine Schätzung hiervon (eine negative Zahl, deren absoluter Wert eine Schätzung ist) eingetragen. Es wird sich um die tatsächliche Anzahl der Zeilen handeln, wenn der Datenbankserver sie errechnen kann, ohne die Zeilen zu zählen. Die Datenbank kann ebenfalls so konfiguriert werden, dass sie immer die tatsächliche Anzahl der Zeilen liefert, was jedoch sehr kostenträchtig sein kann.

Wenn der *cursor-name* von einem Bezeichner oder einer Zeichenfolge angegeben wird, muss die entsprechende DECLARE CURSOR-Anweisung im C-Programm vor der OPEN-Anweisung stehen. Wenn der *cursor-name* von einer Hostvariablen angegeben wird, muss die DECLARE CURSOR-Anweisung vor der OPEN-Anweisung ausgeführt werden.

Privilegien

Wenn der Cursor für eine SELECT-Anweisung gilt, müssen Sie entweder Eigentümer des im Cursor referenzierten Objekts sein, das SELECT-Privileg für das Objekt haben oder das entsprechende SELECT-Systemprivileg haben (z.B. SELECT ANY TABLE).

Wenn der Cursor für eine CALL-Anweisung gilt, müssen Sie Eigentümer der Prozedur sein, das EXECUTE-Privileg für die Prozedur haben oder das EXECUTE ANY PROCEDURE-Systemprivileg haben.

Nebenwirkungen

OPEN führt zu einer vollständigen Materialisierung der Ergebnismenge eines INSENSITIVE-Cursors.

Wenn das Caching des Zugriffsplans aktiviert ist, werden einige SQL-Warnungen, die an die Anwendung während der OPEN-Zeit normalerweise zurückgegeben werden, unterdrückt. Die unterdrückten Warnmeldungen sind Warnungen, die anzeigen, dass der Cursortyp geändert wurde, dass die zugrunde liegende Abfrage nicht deterministisch ist oder dass eine Kürzung der Zeichenfolge aufgetreten ist, wobei eine oder mehrere Literalkonstanten in der Anweisung eingebettet sind.

Siehe auch

- „DECLARE CURSOR-Anweisung [ESQL] [SP]“ auf Seite 778
- „RESUME-Anweisung“ auf Seite 1003
- „PREPARE-Anweisung [ESQL]“ auf Seite 976
- „FETCH-Anweisung [ESQL] [SP]“ auf Seite 853
- „DECLARE CURSOR-Anweisung [ESQL] [SP]“ auf Seite 778
- „RESUME-Anweisung“ auf Seite 1003
- „CLOSE-Anweisung [ESQL] [SP]“ auf Seite 571
- „FOR-Anweisung“ auf Seite 857
- „ansi_close_cursors_on_rollback-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Plan-Caching“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Insensitive Cursor“ [*SQL Anywhere Server - Programmierung*]
- „row_counts-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „close_on_endtrans-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Cursor in Embedded SQL“ [*SQL Anywhere Server - Programmierung*]
- „Cursor in Prozeduren, Triggern, benutzerdefinierten Funktionen und Batches“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Funktionsweise von Sperren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Sperrendauer“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Die Verwendung der OPEN-Anweisung in Embedded SQL ist Teil der optionalen SQL-Sprachenfunktion B031 (Basic Dynamic SQL). Die Verwendung der OPEN-Anweisung in einer gespeicherten Prozedur ist eine Kernfunktion von SQL/2008. Die Klauseln ISOLATION LEVEL und BLOCK sind Erweiterungen des Herstellers, ebenso die Möglichkeit, einen Cursor über eine CALL-Anweisung zu öffnen. Im SQL/2008-Standard wird WITH HOLD als Teil der DECLARE CURSOR-Anweisung angegeben und nicht bei OPEN.

Das Setzen bestimmter Werte im SQLCA ist eine Erweiterung des Herstellers.

- **Transact-SQL** Die OPEN-Anweisung wird von Adaptive Server Enterprise unterstützt. Adaptive Server Enterprise unterstützt nicht die Klauseln ISOLATION LEVEL, BLOCK und WITH HOLD.

Beispiel

Die folgenden Beispiele zeigen, wie OPEN in Embedded SQL verwendet wird.

```
EXEC SQL OPEN employee_cursor;

EXEC SQL PREPARE emp_stat FROM
'SELECT empnum, empname FROM GROUPO.Employees WHERE name like ?';
EXEC SQL DECLARE employee_cursor CURSOR FOR emp_stat;
EXEC SQL OPEN employee_cursor USING :pattern;
```

Dieses Beispielfragment zeigt eine OPEN-Anweisung in einer Prozedur oder einem Trigger.

```
BEGIN
  DECLARE cur_employee CURSOR FOR
  SELECT Surname
  FROM GROUPO.Employees;
  DECLARE name CHAR(40);
  OPEN cur_employee;
```

```
LP: LOOP
    FETCH NEXT cur_employee INTO name;
    IF SQLCODE <> 0 THEN LEAVE LP END IF;
    ...
END LOOP
CLOSE cur_employee;
END
```

OUTPUT-Anweisung [Interactive SQL]

Gibt die aktuellen Abfrageergebnisse in eine Datei oder ODBC-Datenquelle aus.

Syntax 1 - Schreibt das Protokoll in eine Datei

```
OUTPUT TO filename
[ APPEND ]
[ BYTE ORDER MARK { ON | OFF } ]
[ COLUMN WIDTHS ( integer, ... ) ]
[ DELIMITED BY string ]
[ ENCODING encoding ]
[ ESCAPE CHARACTER character ]
[ ESCAPES { ON | OFF } ]
[ FORMAT output-format ]
[ HEXADECIMAL { ON | OFF | ASIS } ]
[ QUOTE string [ ALL ] ]
[ VERBOSE ]
[ WITH COLUMN NAMES ]
```

output-format :

```
TEXT
| FIXED
| HTML
| SQL
| XML
```

encoding : *string* | *identifier*

Syntax 2 - Ausgabe in eine ODBC- Datenquelle

```
OUTPUT
USING connection-string
INTO destination-table-name
[ CREATE TABLE { ON | OFF } ]
```

connection-string :

```
{ DSN = odbc-data-source
| DRIVER = odbc-driver-name [; connection-parameter = value [; ... ] ] }
```

Parameter

APPEND-Klausel Dieses optionale Schlüsselwort wird verwendet, um die Ergebnisse der Abfrage am Ende der vorhandenen Ausgabedatei anzufügen, ohne den Inhalt der Datei zu überschreiben. Wenn die APPEND-Klausel nicht verwendet wird, überschreibt die Anweisung OUTPUT standardmäßig den Inhalt der Ausgabedatei. Das Schlüsselwort APPEND ist gültig, wenn das Ausgabeformat TEXT, FIXED oder SQL ist.

BYTE ORDER MARK-Klausel Verwenden Sie diese Klausel, um anzugeben, ob eine Byte Order Mark (BOM) an den Beginn einer Unicode-Datei gesetzt werden soll. Standardmäßig ist diese Option ON, wodurch Interactive SQL angewiesen wird, eine Byte Order Mark (BOM) an den Beginn der Datei zu schreiben. Wenn BYTE ORDER MARK OFF ist, schreibt DBISQL keine BOM.

Die BYTE ORDER MARK-Klausel ist nur relevant, wenn das Schreiben in Dateien im Textformat erfolgt. Wenn Sie versuchen, die BYTE ORDER MARK-Klausel mit anderen FORMAT-Klauseln als TEXT zu verbinden, wird ein Fehler zurückgegeben.

Die BYTE ORDER MARK-Klausel wird nur verwendet, wenn Dateien gelesen oder geschrieben werden, die mit UTF-8 oder UTF-16 (und ihren Varianten) kodiert sind. Wenn Sie versuchen, die BYTE ORDER MARK-Klausel mit anderen Kodierungen zu verbinden, wird ein Fehler zurückgegeben.

COLUMN WIDTHS-Klausel Die COLUMN WIDTHS-Klausel wird verwendet, um die Spaltenbreite für die Ausgabe im FIXED-Format anzugeben.

CREATE TABLE-Klausel Verwenden Sie die CREATE TABLE-Klausel, um anzugeben, ob die Zieltabelle erstellt werden soll, wenn sie nicht existiert. Der Standardwert ist ON.

DELIMITED BY-Klausel Die Klausel DELIMITED BY gilt nur für das Ausgabeformat TEXT. Die Trennzeichenfolge wird zwischen Spalten platziert. Der Standardwert ist das Komma.

ENCODING-Klausel Mit der ENCODING-Klausel können Sie die Kodierung angeben, mit der die Datei geschrieben werden soll. Die ENCODING-Klausel kann nur im TEXT-Format verwendet werden.

Die ENCODING-Klausel ist nützlich, wenn Sie über Daten verfügen, die im Zeichensatz des Betriebssystems nicht angezeigt werden können. Wenn Sie in einem solchen Fall die ENCODING-Klausel nicht verwenden, gehen Zeichen, die im Standardzeichensatz nicht angezeigt werden können, verloren (und eine verlustreiche Konvertierung wird durchgeführt).

Wenn die Eingabedatei mit der OUTPUT-Anweisung erstellt und eine Kodierung angegeben wurde, muss dieselbe ENCODING-Klausel in der INPUT-Anweisung angegeben werden.

Wenn Sie Interactive SQL ausführen, wird die beim Exportieren der Daten verwendete Kodierung in der folgenden Reihenfolge ermittelt:

- Die durch die ENCODING-Klausel angegebene Kodierung (sofern diese Klausel angegeben ist).
- Die durch die Option default_isql_encoding angegebene Kodierung (sofern diese Option festgelegt wurde).
- Die Standardkodierung für die Plattform, die Sie verwenden. Auf Computern mit englischem Windows ist die Standardkodierung 1252.

ESCAPE CHARACTER-Klausel Das Standard-Escapezeichen für als hexadezimale Codes und Symbole gespeicherte Zeichen ist ein Backslash (\). Beispielsweise ist \x0A das Zeilenvorschubzeichen.

Diese Einstellung kann mit der ESCAPE CHARACTER-Klausel geändert werden. Wenn Sie beispielsweise das Ausrufezeichen als Escapezeichen verwenden möchten, geben Sie Folgendes ein:

```
... ESCAPE CHARACTER '!'
```

Die Zeilenendmarke kann als "\n" angegeben werden. Andere Zeichen können mit hexadezimalen ASCII-Codes wie etwa \x09 für das Tabulatorzeichen angegeben werden. Eine Sequenz von zwei Backslashes (\\) wird als ein einzelner Backslash interpretiert. Ein Backslash gefolgt von einem beliebigen Zeichen außer n, x, X oder \ wird als zwei separate Zeichen interpretiert. Zum Beispiel wird \q als Backslash mit dem Buchstaben q interpretiert.

ESCAPES-Klausel Wenn ESCAPES aktiviert ist (Standardwert), werden die Zeichen, die auf das Backslashzeichen folgen, vom Datenbankserver als Sonderzeichen erkannt und interpretiert. Wenn ESCAPES deaktiviert ist, werden die Zeichen genauso geschrieben, wie sie in den Quelldaten erscheinen.

FORMAT-Klausel Mit der FORMAT-Klausel können Sie das Dateiformat für die Ausgabe angeben. Wenn Sie die FORMAT-Klausel nicht angeben, wird das von der Option output_format festgelegte Format verwendet. Wenn Sie die FORMAT-Klausel festlegen, wird die Einstellung der output_format-Option ignoriert. Das Standard-Ausgabeformat ist TEXT. Folgende Ausgabeformate sind zulässig:

- **TEXT** Die Ausgabe ist eine Datei im TEXT-Format mit einer Tabellenzeile pro Ausgabezeile in der Datei. Alle Werte sind durch Kommas getrennt, und Zeichenfolgen werden in Apostrophe eingeschlossen. Die Trennzeichen und Apostrophe oder Anführungszeichen können mit den Klauseln DELIMITED BY und QUOTE geändert werden. Wenn ALL in der QUOTE-Klausel angegeben wird, werden alle Werte (nicht nur Zeichenfolgen) in Apostrophe oder Anführungszeichen gesetzt. TEXT ist der Standard-Ausgabedatentyp.

Drei weitere spezielle Sequenzen werden ebenfalls verwendet. Die beiden Zeichen \n stellen eine Zeilenendmarke dar, \\ stellt einen einzelnen \ und die Sequenz \xDD stellt das Zeichen mit dem Hexadezimalcode DD dar.

Wenn Sie TEXT verwenden möchten, ohne Anführungszeichen oder Zeilenumbruchzeichen in Ihre Ausgabe einzubeziehen, deaktivieren Sie Anführungszeichen und Escapes folgendermaßen: QUOTE ' ' ESCAPES OFF.

- **FIXED** Die Ausgabe ist festes Format, wobei jede Spalte eine feste Breite hat. Die Breite der einzelnen Spalten kann mit der COLUMN WIDTHS-Klausel angegeben werden. In diesem Format werden keine Spaltentitel ausgegeben.

Wenn die COLUMN WIDTHS-Klausel weggelassen wird, errechnet das System die Breite für jede einzelne Spalte aus dem Datentyp für die Spalte und ist breit genug für Werte dieses Datentyps. Eine Ausnahme bilden die Daten LONG VARCHAR und LONG BINARY, deren Standardwert 32 kB beträgt.

- **HTML** Die Ausgabe ist im Hyper Text Markup Language-Format.
- **SQL** Die Ausgabe ist eine Interactive SQL INPUT-Anweisung, die erforderlich ist, um die Daten in der Tabelle neu zu erstellen, in Form einer .sql-Datei.
- **XML** Die Ausgabe ist eine XML-Datei, kodiert in UTF-8 mit einer eingebetteten DTD. Binärwerte sind in CDATA-Blöcken kodiert, wobei die Binärdaten in Form von Zeichenfolgen mit jeweils zwei hexadezimalen Zeichen dargestellt werden.

HEXADECIMAL-Klausel Die HEXADECIMAL-Klausel gibt an, wie Binärwerte für das TEXT-Format ausgegeben werden. Zulässige Werte sind:

- **ON** Wenn auf ON eingestellt, werden Binärwerte mit einem Ox-Präfix, gefolgt durch eine Serie von hexadezimalen Paaren, geschrieben: z.B. 0xabcd.
- **OFF** Bei der Einstellung OFF wird nicht druckbaren Zeichenwerten das Escapezeichen vorangestellt, z.B. der Backslash, gefolgt von x und dem hexadezimalen Paar für das Byte. Druckbare Zeichen werden normal ausgegeben.
- **ASIS** Wenn auf ASIS eingestellt, werden Werte in ihrer Istform geschrieben, ohne Escapezeichen, auch wenn die Zeichen Steuerzeichen enthalten. Die Option ASIS ist für Text von Nutzen, der Formatierungsinformationen wie Tabulatoren oder Zeilenumbrüche enthält.

QUOTE-Klausel Die Klausel QUOTE gilt nur für das Ausgabeformat TEXT. Die Zeichenfolge für die Anführungszeichen wird um die Zeichenfolgenwerte herum platziert. Der Standardwert ist ein Apostroph ('). Wenn ALL in der QUOTE-Klausel angegeben ist, werden alle Werte und nicht nur die Zeichenfolgen von Apostrophen oder Anführungszeichen umschlossen. Um Anführungszeichen zu unterdrücken, müssen sie leere Apostrophe eingeben. Beispiel: QUOTE ' '.

USING-Klausel Die USING-Klausel exportiert Daten in eine ODBC-Datenquelle. Sie können entweder den ODBC-Datenquellennamen mit der DSN-Option oder den ODBC-Treibernamen und den Verbindungsparameter mit der DRIVER-Option angeben. *Connection-parameter*: Eine optionale Liste von datenbankspezifischen Verbindungsparametern.

odbc-data-source ist der Name eines Benutzers oder der ODBC-Datenquellennamen. Die *odbc-data-source* für die SQL Anywhere-Beispieldatenbank ist beispielsweise SQL Anywhere 16 Demo.

odbc-driver-name ist der ODBC-Treibername. Für eine SQL Anywhere-Datenbank ist der *Odbc-driver-name* SQL Anywhere , für eine UltraLite-Datenbank ist der *Odbc-driver-name* UltraLite 16.

VERBOSE-Klausel Wenn das optionale VERBOSE-Schlüsselwort einbezogen ist, werden Fehlermeldungen über die Abfrage, die SQL-Anweisung, die zur Auswahl der Daten verwendet wurde, und die Daten in die Ausgabedatei geschrieben. Zeilen, die keine Daten enthalten, haben zwei Bindestriche als Präfix. Wenn VERBOSE ausgelassen wird (Standardwert), werden nur die Daten in die Datei geschrieben. Das Schlüsselwort VERBOSE ist gültig, wenn das Ausgabeformat TEXT, FIXED oder SQL ist.

WITH COLUMN NAMES-Klausel Die WITH COLUMN NAMES-Klausel fügt die Spaltennamen in der ersten Zeile der Textdatei ein. Die WITH COLUMN NAMES-Klausel gilt nur für das TEXT-Format.

Bemerkungen

Die OUTPUT-Anweisung gibt Daten in eine Datei oder eine Datenbank aus. Die OUTPUT-Anweisung wird direkt nach einer Anweisung verwendet, die die auszugebenden Daten abrufen.

Um mehrere Ergebnismengen zu exportieren, verwenden Sie Syntax 1 und setzen die Option `isql_show_multiple_result_sets` auf ON. Interactive SQL erstellt eine Datei für jede Ergebnismenge. Die Dateien erhalten den Namen *filename-x*, wobei *x* ein mit 1 beginnender Zähler ist. Zum Beispiel werden durch die Angabe von OUTPUT TO für eine Datei mit dem Namen *data.txt* Dateien mit dem Namen *data-1.txt*, *data-2.txt* usw. ausgegeben.

Sie können Syntax 2 nicht zum Exportieren mehrerer Ergebnismengen verwenden.

Das Ausgabeformat kann mit der optionalen FORMAT-Klausel angegeben werden. Das Standardformat ist TEXT. Wenn keine FORMAT-Klausel angegeben ist, wird die Interactive SQL-Optionseinstellung output_format verwendet.

Da die OUTPUT-Anweisung ein Interactive SQL-Befehl ist, kann sie nicht in zusammengesetzten Anweisungen (wie IF) oder in gespeicherten Prozeduren verwendet werden.

Beim Exportieren von Spalten mit BINARY- oder LONG BINARY-Daten in eine Microsoft Excel-Arbeitsmappe müssen Sie die Daten in eine Zeichenfolge oder Zahl konvertieren. Außerdem gilt: Wenn Daten in eine Microsoft Excel Arbeitsmappe exportiert werden, sind die Daten schreibgeschützt, es sei denn, der ReadOnly-Parameter wird beim Auswählen der DSN-Option auf Null gesetzt oder deaktiviert.

Privilegien

Keine.

Nebenwirkungen

In Interactive SQL enthält die Registerkarte **Ergebnisse** die Ergebnisse der aktuellen Abfrage.

Siehe auch

- „SELECT-Anweisung“ auf Seite 1020
- „INPUT-Anweisung [Interactive SQL]“ auf Seite 910
- „UNLOAD-Anweisung“ auf Seite 1098
- „output_format-Option [Interactive SQL]“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenimport und -export“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „default_isql_encoding-Option [Interactive SQL]“ [*SQL Anywhere Server - Datenbankadministration*]
- „isql_show_multiple_result_sets-Option [Interactive SQL]“ [*SQL Anywhere Server - Datenbankadministration*]
- „Interactive SQL“ [*SQL Anywhere Server - Datenbankadministration*]
- „Tipps zum Exportieren von Daten mit der OUTPUT-Anweisung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Zulässige Anweisungen in Prozeduren, Triggern, Ereignissen und Batches“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Rückgabe von mehreren Ergebnismengen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „output_format-Option [Interactive SQL]“ [*SQL Anywhere Server - Datenbankadministration*]
- „Unterstützte Zeichensätze“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Inhalt der Employees-Tabelle in eine Textdatei schreiben:

```
SELECT * FROM GROUP0.Employees;  
OUTPUT TO 'c:\\temp\\Employees.txt'  
FORMAT TEXT;
```

Inhalte der Tabelle 'Employees' ans Ende einer bestehenden Datei schreiben und Meldungen über die Abfrage in diese Datei einfügen:

```
SELECT * FROM GROUPO.Employees;
OUTPUT TO 'c:\\temp\\Employees.txt'
APPEND VERBOSE;
```

Nehmen wir an, dass Sie einen Wert exportieren müssen, der ein eingebettetes Zeilenvorschubzeichen enthält. Ein Zeilenvorschubzeichen hat den numerischen Wert 10, den Sie als die Zeichenfolge '\x0a' in einer SQL-Anweisung angeben können. Führen Sie beispielsweise die folgende Anweisung mit der Option HEXADEDECIMAL auf ON aus:

```
SELECT CAST ('line1\x0aline2' AS VARBINARY);
OUTPUT TO 'c:\\temp\\file.txt' HEXADEDECIMAL ON;
```

Sie erhalten eine Datei, die eine Zeile mit folgendem Text enthält: 0x6c696e65310a6c696e6532

Wenn Sie jedoch dieselbe Anweisung ausführen und HEXADEDECIMAL auf OFF setzen, erhalten Sie Folgendes: 'line1\x0Aline2'

Schließlich setzen Sie HEXADEDECIMAL auf ASIS und erhalten eine Datei mit den folgenden zwei Zeilen:

```
'line1
line2'
```

Wenn Sie ASIS verwenden, werden zwei Zeilen ausgegeben, da das eingebettete Zeilenvorschubzeichen ohne Umwandlung in eine zweistellige hexadezimale Entsprechung und ohne ein vorangestelltes Escapezeichen exportiert wurde.

Beim folgenden Beispiel werden die Daten aus der Customers-Tabelle in eine neue Tabelle, Customers2, ausgegeben:

```
SELECT * FROM Customers;
OUTPUT USING 'DSN=SQL Anywhere 16 Demo'
INTO "Customers2";
```

Im folgenden Beispiel wird die Tabelle "Customers" aus der Beispieldatenbank in eine fiktive Datenbank namens *mydatabase.db* kopiert, wobei die DRIVER-Option verwendet wird.

```
SELECT * FROM Customers;
OUTPUT USING 'DRIVER=SQL Anywhere 16;UID=DBA;PWD=sql;DBF=c:\\test\\
\\mydatabase.db'
INTO "Customers";
```

Im folgenden Beispiel wird die Customers-Tabelle aus der SQL Anywhere-Beispieldatenbank in eine Tabelle namens Customers in einer fiktiven UltraLite-Datenbank namens *myULDatabase.db* kopiert, wobei die DRIVER-Option verwendet wird.

```
SELECT * FROM Customers;
OUTPUT USING 'DRIVER=UltraLite 16;DBF=c:\\test\\myULDatabase.udb'
INTO "Customers";
```

Im folgenden Beispiel wird die Customers-Tabelle in eine fiktive MySQL-Datenbank *mydatabase* mit der DRIVER-Option exportiert.

```
SELECT * FROM GROUPO.Customers;
OUTPUT USING 'DRIVER=MySQL ODBC 5.1
```

```
Driver;DATABASE=mydatabase;SERVER=mySQLHost;UID=me;PWD=secret '  
INTO "Customers";
```

Der folgende Befehl gibt eine Datei aus, die 'one\x0Atwo\x0Athree' enthält:

```
SELECT 'one\ntwo\nthree';  
OUTPUT TO 'c:\\temp\\test.txt' HEXADECIMAL OFF;
```

PARAMETERS-Anweisung [Interactive SQL]

Gibt die Parameter für eine Interactive SQL-Skriptdatei an.

Syntax

PARAMETERS *parameter1, parameter2, ...*

Bemerkungen

Die PARAMETERS-Anweisung benennt die Parameter für eine Skriptdatei, sodass sie später in der Skriptdatei referenziert werden können.

Parameter werden referenziert, indem Sie {parameter1} in die Datei eingeben, in der Sie den benannten Parameter ersetzen möchten. Zwischen den geschweiften Klammern und dem Parameternamen darf es keine Leerstellen geben.

Wenn eine Skriptdatei mit weniger als der erforderlichen Anzahl von Parametern aufgerufen wird, fordert Interactive SQL alle fehlenden Parameter an.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „[READ-Anweisung \[Interactive SQL\]](#)“ auf Seite 984
- „[Interactive SQL](#)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Interactive SQL-Skriptdatei benötigt zwei Parameter.

```
PARAMETERS department_id, file;  
SELECT Surname  
FROM GROUPO.Employees  
WHERE DepartmentID = {department_id}  
>#{file}.dat;
```

Wenn Sie dieses Skript in einer fiktiven Datei namens `c:\temp\test.sql` speichern, können Sie es über Interactive SQL mit dem folgenden Befehl ausführen:

```
READ 'c:\\temp\\test.sql' [100] [data]
```

PASSTHROUGH-Anweisung [SQL Remote]

Startet oder stoppt den Durchreichmodus bei der SQL Remote-Verwaltung. Syntax 1 und 2 starten den Durchreichmodus, während Syntax 3 ihn stoppt.

Syntax 1

PASSTHROUGH [ONLY] FOR *userid*, ...

Syntax 2

**PASSTHROUGH [ONLY] FOR SUBSCRIPTION
TO [*owner.*]*publication-name* [(*constant*)]**

Syntax 3

PASSTHROUGH STOP

Bemerkungen

Im Durchreichmodus werden alle SQL-Anweisungen vom Datenbankserver ausgeführt und in das Transaktionslog geschrieben, um in Nachrichten an die Subskribenten versendet zu werden. Wenn das ONLY-Schlüsselwort verwendet wird, um den Durchreichmodus zu starten, werden die Anweisungen nicht auf dem Server ausgeführt, sondern nur an die Empfänger versendet. Wenn eine Passthrough-Sitzung Aufrufe an gespeicherte Prozeduren enthält, müssen die Prozeduren auf dem Server vorhanden sein, der die Passthrough-Befehle ausgibt, auch wenn sie auf diesem Server nicht lokal ausgeführt werden. Die Empfänger der Durchreich-SQL-Anweisung sind entweder eine Liste von Benutzer-IDs (Syntax 1) oder alle Subskribenten (Format 2) der angegebenen Publikation. Der Durchreichmodus kann verwendet werden, um Änderungen an einer entfernten Datenbank von der konsolidierten Datenbank aus anzuwenden, oder um Anweisungen von einer entfernten Datenbank an die konsolidierte Datenbank zu versenden.

Format 2 versendet Anweisungen an entfernte Datenbanken, deren Subskriptionen gestartet sind. Es werden keine Anweisungen an entfernte Datenbanken versendet, deren Subskriptionen erstellt, aber nicht gestartet sind.

Syntax 3 stoppt den Durchreichmodus für die aktuelle Verbindung. Sie müssen die PASSTHROUGH STOP-Anweisung auf derselben Verbindung ausführen, die den Durchreichmodus initiiert hat. Wenn Sie Syntax 1 oder 2 verwenden, um den Durchreichmodus auf einer Verbindung zu starten, und Verbindung vor dem Ausführen einer PASSTHROUGH STOP-Anweisung zu getrennt wird, bedeutet die Unterbrechung eine implizite PASSTHROUGH STOP-Anweisung.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Keine.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Siehe auch

- „FORWARD TO-Anweisung“ auf Seite 861
- „UNLOAD-Anweisung“ auf Seite 1098
- „Systemanbindungstests“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „SQL Remote-Passthrough-Modus“ [*SQL Remote*]
- „Passthrough-Modus starten und stoppen“ [*SQL Remote*]
- „Einschränkungen beim Passthrough-Modus“ [*SQL Remote*]
- „In einem laufenden System zu vermeidende Änderungen“ [*SQL Remote*]
- „Upgrades und Resynchronisation“ [*SQL Remote*]
- „Resynchronisation von Subskriptionen“ [*SQL Remote*]

Beispiel

```
PASSTHROUGH FOR Sam_Singer ;  
...  
( SQL statements to be executed at the remote database )  
...  
PASSTHROUGH STOP ;
```

PREPARE-Anweisung [ESQL]

Bereitet eine Anweisung vor, die zu einem späteren Zeitpunkt ausgeführt wird, oder definiert einen Cursor.

Syntax

```
PREPARE statement-name  
  FROM statement [ FOR { UPDATE [ cursor-concurrency ] | READ ONLY } ]  
  [ DESCRIBE describe-type INTO [ [ SQL ] DESCRIPTOR ] descriptor ]  
  [ WITH EXECUTE ]
```

statement-name : *identifier* | *hostvar*

statement : *string* | *hostvar*

describe-type :
 [**ALL** | **BIND VARIABLES** | **INPUT** | **OUTPUT** | **SELECT LIST**]
 [**LONG NAMES** [[**OWNER.**] **TABLE.**] **COLUMN**]
 | **WITH VARIABLE RESULT**]

cursor-concurrency :
BY { **VALUES** | **TIMESTAMP** | **LOCK** }

Parameter

statement-name Der Anweisungsname kann ein Identifizierer oder eine Hostvariable sein. Sie dürfen jedoch keinen Bezeichner verwenden, wenn Sie mehrere SQLCA einsetzen. Wenn Sie es dennoch tun, könnten zwei vorbereiteten Anweisungen die gleiche Anweisungsnummer zugeordnet werden, was dazu führen kann, dass die falsche Anweisung ausgeführt oder geöffnet wird. Auch ist ein Bezeichner für einen Anweisungsnamen bei Anwendungen mit mehreren Threads nicht empfehlenswert, bei denen der Anweisungsname gleichzeitig von mehreren Threads referenziert werden kann.

DESCRIBE-Klausel Unter Verwendung der Klausel DESCRIBE INTO DESCRIPTOR wird die vorbereitete Anweisung im angegebenen Deskriptor beschrieben. Der DESCRIBE-Typ kann jeder der zulässigen DESCRIBE-Typen sein.

FOR UPDATE | FOR READ ONLY Definiert die Cursoraktualisierbarkeit, wenn die Anweisung von einem Cursor verwendet wird. Ein FOR READ ONLY-Cursor kann weder in einem (positionsbasierten) UPDATE- noch in einem (positionsbasierten) DELETE-Vorgang verwendet werden. FOR READ ONLY ist der Standardwert.

Als Antwort auf eine Anforderung für einen FOR UPDATE-Cursor übergibt SQL Anywhere entweder einen wertsensitiven oder einen sensitiven Cursor. Insensitive und asensitive Cursor können nicht aktualisiert werden.

BY VALUES | BY TIMESTAMP Der Datenbankserver verwendet einen durch Keyset gesteuerten Cursor, damit die Anwendung benachrichtigt werden kann, wenn Zeilen geändert oder gelöscht werden, während die Ergebnismenge durchlaufen wird.

BY LOCK-Klausel Der Datenbankserver erwirbt Absichtszeilensperren für abgerufene Zeilen der Ergebnismenge. Dies sind langfristige Sperren, die gehalten werden, bis die Transaktion festgeschrieben oder zurückgesetzt wird.

- **WITH EXECUTE-Klausel** Unter Verwendung der WITH EXECUTE-Klausel wird die Anweisung nur dann ausgeführt, wenn es sich nicht um eine CALL- oder SELECT-Anweisung handelt und sie keine Hostvariablen besitzt. Die Anweisung wird sofort nach der erfolgreichen Ausführung gelöscht. Wenn PREPARE und DESCRIBE (falls es sie geben sollte) erfolgreich sind, aber die Anweisung nicht ausgeführt werden kann, wird die Warnung SQLCODE 111, SQLSTATE 01W08 ausgegeben, und die Anweisung wird gelöscht.

Die Klauseln DESCRIBE INTO DESCRIPTOR und WITH EXECUTE können die Performance verbessern, da sie die erforderliche Client/Server-Kommunikation verringern.

- **WITH VARIABLE RESULT-Klausel** Die WITH VARIABLE RESULT-Klausel wird verwendet, um Prozeduren zu beschreiben, die mehr als eine Ergebnismenge mit unterschiedlichen Anzahlen und Typen von Spalten haben könnten.

Wenn WITH VARIABLE RESULT verwendet wird, stellt der Datenbankserver nach der DESCRIBE-Anweisung SQLCOUNT auf einen der folgenden Werte ein:

- **0** Die Ergebnismenge kann sich ändern: Der Prozeduraufruf sollte nach jeder OPEN-Anweisung erneut beschrieben werden.

- **1** Die Ergebnismenge ist unveränderlich. Eine erneute Beschreibung ist nicht erforderlich.

Hinweis

Aus Kompatibilitätsgründen wird die Vorbereitung der Anweisungen COMMIT, PREPARE TO COMMIT und ROLLBACK immer noch unterstützt. Es wird jedoch empfohlen, dass Sie alle Operationen zur Transaktionsverwaltung mit statischer Embedded SQL vornehmen, da dies von bestimmten Anwendungsumgebungen gefordert wird. Außerdem unterstützen andere Embedded SQL-Systeme keine dynamischen Operationen zur Transaktionsverwaltung.

Bemerkungen

Die PREPARE-Anweisung bereitet eine SQL-Anweisung aus *statement* vor und ordnet die vorbereitete Anweisung *statement-name* zu. Dieser Anweisungsname wird referenziert, um die Anweisung auszuführen oder um einen Cursor zu öffnen, wenn es sich bei der Anweisung um eine SELECT- oder CALL-Anweisung handelt. Der *statement-name* kann eine Hostvariable vom Typ `a_sql_statement_number` sein, die in der Header-Datei *sqlca.h* definiert wird, welche automatisch einbezogen wird. Wenn ein Bezeichner für *statement-name* verwendet wird, kann nur eine Anweisung pro Modul mit *statement-name* vorbereitet werden.

Wenn eine Hostvariable für *statement-name* verwendet wird, muss sie vom Typ SHORT INT sein. Es gibt eine Typendefinition für diesen Typ in *sqlca.h* mit dem Namen `a_sql_statement_number`. Dieser Typ wird vom SQL-Präprozessor erkannt und kann in einem DECLARE-Abschnitt verwendet werden. Die Hostvariable wird im Rahmen der PREPARE-Anweisung von der Datenbank definiert und Sie müssen sie nicht initialisieren.

Privilegien

Keine.

Nebenwirkungen

Jede zuvor vorbereitete Anweisung mit demselben Namen geht verloren.

Die Anweisung wird nach der Verwendung nur dann gelöscht, wenn Sie WITH EXECUTE angeben, und wenn sie erfolgreich ausgeführt wurde. Stellen Sie sicher, die Anweisung mit DROP nach der Verwendung unter anderen Bedingungen zu löschen. Wenn Sie dies nicht tun, wird der dieser Anweisung zugeordnete Speicher nicht wieder beansprucht.

Siehe auch

- „DECLARE CURSOR-Anweisung [ESQL] [SP]“ auf Seite 778
- „DESCRIBE-Anweisung [ESQL]“ auf Seite 794
- „OPEN-Anweisung [ESQL] [SP]“ auf Seite 964
- „EXECUTE-Anweisung [ESQL]“ auf Seite 846
- „DROP STATEMENT-Anweisung [ESQL]“ auf Seite 826
- „Anweisung kann nicht ausgeführt werden“ [*Fehlermeldungen*]
- „Dynamic SQL-Anweisungen“ [*SQL Anywhere Server - Programmierung*]

Standards und Kompatibilität

- **SQL/2008** PREPARE ist Teil der optionalen SQL/2008-Sprachenfunktion B031 (Basic Dynamic SQL). Die optionalen Klauseln FOR UPDATE, FOR READ ONLY, DESCRIBE und WITH EXECUTE sind Erweiterungen des Herstellers.

Beispiel

Die folgende Anweisung bereitet eine einfache Abfrage vor:

```
EXEC SQL PREPARE employee_statement FROM  
'SELECT Surname FROM Employees';
```

PREPARE TO COMMIT-Anweisung

Prüft, ob ein COMMIT erfolgreich ausgeführt werden kann.

Syntax

PREPARE TO COMMIT

Bemerkungen

Die PREPARE TO COMMIT-Anweisung testet, ob eine COMMIT-Anweisung erfolgreich ausgeführt werden kann. Die Anweisung verursacht einen Fehler, wenn COMMIT nicht möglich ist, ohne die Integrität der Datenbank zu verletzen.

Die PREPARE TO COMMIT-Anweisung kann nicht in gespeicherten Prozeduren, Triggern, Ereignissen oder Batches verwendet werden.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „COMMIT-Anweisung“ auf Seite 575
- „ROLLBACK-Anweisung“ auf Seite 1015

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Sequenz von Anweisungen führt aufgrund einer Fremdschlüsselüberprüfung in der Tabelle 'Employees' zu einem Fehler.

```
EXECUTE IMMEDIATE  
  "SET OPTION wait_for_commit = 'On';"  
EXECUTE IMMEDIATE "DELETE FROM Employees"
```

```
WHERE EmployeeID = 160";  
EXECUTE IMMEDIATE "PREPARE TO COMMIT";
```

Die folgende Sequenz von Anweisungen verursacht keinen Fehler, wenn die delete-Anweisung ausgeführt wird, auch wenn sie gegen eine Integritätsregel verstößt. Die PREPARE TO COMMIT-Anweisung gibt eine Fehlermeldung zurück.

```
SET OPTION wait_for_commit= 'On';  
DELETE  
FROM GROUPO.Departments  
WHERE DepartmentID = 100;  
PREPARE TO COMMIT;
```

PRINT-Anweisung [T-SQL]

Gibt eine Meldung an den Client zurück oder zeigt eine Meldung im Meldungsfenster des Datenbankservers.

Syntax

PRINT *format-string* [, *arg-list*]

Bemerkungen

Die PRINT-Anweisung gibt eine Meldung im Clientfenster aus, wenn Sie von einer Open Client-Anwendung oder einer jConnect-Anwendung aus verbunden sind. Wenn Sie von einer Embedded SQL- oder ODBC-Anwendung aus verbunden sind, wird die Meldung im Meldungsfenster des Datenbankservers angezeigt.

Die Formatzeichenfolge kann Platzhalter für die Argumente in der optionalen Argumentliste enthalten. Diese Platzhalter haben die Form *%nn!*, wobei *nn* eine Ganzzahl zwischen 1 und 20 ist.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- [„MESSAGE-Anweisung“ auf Seite 959](#)

Standards und Kompatibilität

- **SQL/2008** Transact-SQL-Erweiterung.

Beispiel

Die folgende Anweisung zeigt eine Meldung an:

```
PRINT 'Display this message';
```

Die folgende Anweisung veranschaulicht die Verwendung von Platzhaltern in der PRINT-Anweisung:

```
DECLARE @var1 INT, @var2 INT
SELECT @var1 = 3, @var2 = 5
PRINT 'Variable 1 = %1!, Variable 2 = %2!', @var1, @var2
```

PUT-Anweisung [ESQL]

Fügt eine Zeile in den angegebenen Cursor ein.

Syntax

```
PUT cursor-name
{ USING DESCRIPTOR sqlda-name | FROM hostvar-list }
[ INTO { DESCRIPTOR sqlda-name | hostvar-list } ]
[ ARRAY :row-count ]
```

cursor-name : *identifier* | *hostvar*

sqlda-name : *identifier*

hostvar-list : Kann Indikatorvariablen enthalten

row-count : *integer* | *hostvar*

Bemerkungen

Fügt eine Zeile in den benannten Cursor ein. Werte für die Spalten werden dem SQLDA oder der Hostvariablenliste in einer Übereinstimmung von eins zu eins mit den Spalten in der INSERT-Anweisung (für einen INSERT-Cursor) oder den Spalten in der Auswahlliste (für einen SELECT-Cursor) entnommen.

Die PUT-Anweisung kann nur für einen Cursor über eine INSERT- oder SELECT-Anweisung verwendet werden, die eine einzelne Tabelle in der FROM-Klausel referenziert oder die eine aktualisierbare Ansicht referenziert, die aus einer einzelnen Basistabelle besteht.

Wenn der Zeiger sqldata im SQLDA der Leerzeiger ist, wird für diese Spalte kein Wert angegeben. Wenn der Spalte ein DEFAULT VALUE zugeordnet ist, wird dieser benutzt. Ansonsten wird NULL verwendet.

Der zweite SQLDA oder die Hostvariablenliste enthält die Ergebnisse der PUT-Anweisung.

Die optionale ARRAY-Klausel kann verwendet werden, um breite Ablagen, die mehr als eine Zeile zur gleichen Zeit einfügen, durchzuführen. Dadurch lässt sich die Performance verbessern. Der Ganzzahlwert stellt die Anzahl der einzufügenden Zeilen dar. Der SQLDA muss eine Variable für jeden Eintrag (Anzahl der Zeilen * Anzahl der Spalten) enthalten. Die erste Zeile wird in die SQLDA-Variablen von 0 bis (Spalten pro Zeile)-1 geschrieben, usw.

Hinweis

Bei Scroll-Cursors (wertempfindlich) erscheint die eingefügte Zeile, wenn die neue Zeile mit der WHERE-Klausel übereinstimmt und der keyset-driven Cursor die Datenübernahme nicht abgeschlossen hat. Wenn bei dynamischen Cursors die eingefügte Zeile mit der WHERE-Klausel übereinstimmt, kann die Zeile erscheinen. Unempfindliche Cursors können nicht aktualisiert werden.

Privilegien

Sie müssen Eigentümer der im Cursor referenzierten Tabellen sein, INSERT-Privilegien für die Tabellen haben oder das INSERT ANY TABLE-Systemprivileg haben.

Nebenwirkungen

Wenn Sie Zeilen in einen wertsensitiven (Keyset-gesteuerten) Cursor einfügen, erscheinen die eingefügten Zeilen am Ende der Ergebnismenge, auch wenn sie nicht zur WHERE-Klausel der Abfrage passen oder eine ORDER BY-Klausel sie normalerweise an eine andere Stelle in der Ergebnismenge gesetzt hätte.

Siehe auch

- „UPDATE-Anweisung“ auf Seite 1109
- „UPDATE-Anweisung (positionsbasiert) [ESQL] [SP]“ auf Seite 1103
- „DELETE-Anweisung“ auf Seite 791
- „DELETE-Anweisung (positionsbasiert) [ESQL] [SP]“ auf Seite 789
- „INSERT-Anweisung“ auf Seite 917
- „SET-Anweisung“ auf Seite 1054
- „Für die Zeilenänderung verwendete Cursor“ [*SQL Anywhere Server - Programmierung*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung veranschaulicht die Verwendung der PUT-Anweisung in Embedded SQL:

```
EXEC SQL PUT cur_employee FROM :employeeID, :surname;
```

RAISERROR-Anweisung

Signalisiert einen Fehler und sendet eine Meldung an den Client.

Syntax

```
RAISERROR error-number [ format-string ] [, arg-list ]
```

Parameter

error-number Die *error-number* ist eine fünfstellige Ganzzahl, die größer ist als 17000. Die Fehlernummer wird in der globalen Variablen @@error gespeichert.

format-string *format-string* ist Zeichenfolge mit einer Länge von bis zu 255 Byte. Sie kann Platzhalter für die Argumente in der optionalen Argumentliste enthalten. Diese Platzhalter haben die Form %*nn*!, wobei *nn* eine Ganzzahl zwischen 1 und 20 ist.

Bemerkungen

Mit der RAISERROR-Anweisung können benutzerdefinierte Fehler angezeigt und eine Meldung an den Client gesendet werden.

Um neue Fehlermeldungen zu erstellen, verwenden Sie die CREATE ERROR-Anweisung. Um die Meldungen in der ISYSUSERMESSAGE-Systemtabelle anzuzeigen, fragen Sie die SYSUSERMESSAGE-Systemansicht ab.

Die erweiterten Werte, welche die RAISERROR-Anweisung in Adaptive Server Enterprise unterstützt, werden in SQL Anywhere nicht unterstützt.

Wenn die *format-string* nicht angegeben oder leer ist, wird die Fehlernummer verwendet, um eine Fehlermeldung in den Systemtabellen ausfindig zu machen. Adaptive Server Enterprise erhält die Meldungen 17000-19999 aus der Tabelle SYSMESSAGES. In SQL Anywhere ist diese Tabelle eine leere Ansicht, sodass Fehler in diesem Bereich eine Formatzeichenfolge liefern sollten. Meldungen für die Fehlernummern 20000 und größer werden aus der Systemtabelle ISYSUSERMESSAGE bezogen.

Dazwischenliegende RAISERROR-Statusangaben und -Codes gehen verloren, wenn die Prozedur abgeschlossen ist. Wenn zum Rückgabezeitpunkt zusammen mit RAISERROR ein Fehler auftritt, dann werden die Fehlerinformationen zurückgegeben, und die RAISERROR-Informationen gehen verloren. Die Anwendung kann dazwischenliegende RAISERROR-Statusangaben abfragen, indem sie die globale Variable @@error zu verschiedenen Ausführungspunkten überprüft.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „BEGIN-Anweisung“ auf Seite 557
- „CREATE MESSAGE-Anweisung [T-SQL]“ auf Seite 655
- „SYSUSERMESSAGE-Systemansicht“ auf Seite 1510
- „CREATE TRIGGER-Anweisung [T-SQL]“ auf Seite 769
- „CREATE TRIGGER-Anweisung“ auf Seite 762
- „on_tsql_error-Option“ [SQL Anywhere Server - Datenbankadministration]
- „login_procedure-Option“ [SQL Anywhere Server - Datenbankadministration]
- „continue_after_raiserror-Option“ [SQL Anywhere Server - Datenbankadministration]
- „SIGNAL-Anweisung [SP]“ auf Seite 1061

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** Die RAISERROR-Anweisung kann nicht in zusammengesetzten Anweisungen und Prozeduren in Transact-SQL verwendet werden.

Beispiel

In diesem Beispiel wird RAISERROR verwendet, um keine Verbindungen zuzulassen.

```
CREATE PROCEDURE DBA.login_check()
BEGIN
    // Allow a maximum of 3 concurrent connections
    IF( DB_PROPERTY('ConnCount') > 3 ) THEN
```

```
        RAISERROR 28000
        'User %1! is not allowed to connect -- there are ' ||
        'already %2! users logged on',
        Current User,
        CAST( DB_PROPERTY( 'ConnCount' ) AS INT )-1;
    ELSE
        CALL sp_login_environment;
    END IF;
END
go
GRANT EXECUTE ON DBA.login_check TO PUBLIC
go
SET OPTION PUBLIC.login_procedure='DBA.login_check'
go
```

READ-Anweisung [Interactive SQL]

Liest Interactive SQL-Anweisungen aus einer Datei.

Syntax

READ [**ENCODING** *encoding*] *filename* [*parameter*] ...

encoding : *identifier* | *string*

Parameter

ENCODING Mit der ENCODING-Klausel können Sie die Kodierung angeben, mit der die Datei gelesen werden soll. Die READ-Anweisung verarbeitet beim Lesen einer Datei keine Escapezeichen. Es wird vorausgesetzt, dass die gesamte Datei in der angegebenen Kodierung vorliegt.

Wenn Sie Interactive SQL ausführen, wird die beim Lesen der Daten verwendete Kodierung in der folgenden Reihenfolge ermittelt:

1. Die durch die ENCODING-Klausel angegebene Kodierung (sofern diese Klausel angegeben ist).
2. Die durch die Bytereihenfolge-Markierung (BOM) in der Datei angegebene Kodierung (wenn eine BOM angegeben ist).
3. Die durch die Option `default_isql_encoding` angegebene Kodierung (sofern diese Option festgelegt wurde).
4. Die Standardkodierung für die Plattform, die Sie verwenden. Auf Computern mit englischem Windows ist die Standardkodierung 1252.

Bemerkungen

Die READ-Anweisung liest eine Sequenz der Interactive SQL-Anweisungen aus der benannten Datei. Diese Datei kann jede gültige Interactive SQL-Anweisung einschließlich weiterer READ-Anweisungen enthalten. READ-Anweisungen können beliebig tief ineinander verschachtelt werden.

Wenn der *filename* keine Erweiterung hat, sucht Interactive SQL nach demselben Dateinamen mit der Erweiterung *.sql*.

Wenn *filename* keinen absoluten Pfad enthält, sucht Interactive SQL nach der Datei. Der Speicherort von *filename* wird basierend auf dem Standort der READ-Anweisung wie folgt ermittelt:

- Wenn die READ-Anweisung direkt in Interactive SQL ausgeführt wird, versucht Interactive SQL zunächst, den Pfad von *filename* relativ zu dem Verzeichnis aufzulösen, in dem Interactive SQL ausgeführt wird. Wenn dies nicht gelingt, sucht Interactive SQL in den durch die Umgebungsvariable SQLPATH angegebenen Verzeichnissen nach *filename* und anschließend in den durch die Umgebungsvariable PATH angegebenen Verzeichnissen.
- Wenn die READ-Anweisungen in einer externen Datei enthalten sind (zum Beispiel in einer *.sql*-Datei), versucht Interactive SQL zuerst, den Pfad zu *filename* relativ zum Speicherort der externen Datei aufzulösen. Wenn dies nicht gelingt, sucht Interactive SQL nach *filename* in einem Pfad, der relativ zu dem Verzeichnis liegt, in dem Interactive SQL läuft. Wenn dies weiterhin nicht gelingt, sucht Interactive SQL in den durch die Umgebungsvariable SQLPATH angegebenen Verzeichnissen und anschließend in den durch die Umgebungsvariable PATH angegebenen Verzeichnissen.

Parameter können nach dem Namen der Skriptdatei aufgeführt werden. Diese Parameter entsprechen den Parametern, die in der PARAMETERS-Anweisung zu Beginn der Anweisungsdatei benannt werden.

Parameternamen müssen in eckige Klammern gesetzt werden. Interactive SQL ersetzt dann den entsprechenden Parameter überall dort, wo die Quelldatei { *parameter-name* } enthält, wobei *parameter-name* der Name des entsprechenden Parameters ist.

Die Parameter, die an eine Skriptdatei übergeben werden, können Bezeichner, Zahlen, angeführte Bezeichner oder Zeichenfolgen sein. Wenn ein Parameter in Anführungszeichen gesetzt ist, werden die Anführungszeichen während der Ersetzung in den Text eingegeben. Parameter, die keine Bezeichner, Zahlen oder Zeichenfolgen (mit Leerstellen oder Tabstops) sind, müssen in eckige Klammern ([]) gesetzt werden. Dadurch ist die willkürliche Textersetzung in der Skriptdatei möglich.

Wenn nicht genug Parameter in die Skriptdatei übernommen werden, fordert Interactive SQL die Werte für die fehlenden Parameter an.

Wenn Sie eine *reload.sql*-Datei mit Interactive SQL ausführen, müssen Sie den Chiffrierschlüssel als Parameter angeben. Wenn Sie den Schlüssel nicht in der READ-Anweisung übergeben, werden Sie von Interactive SQL aufgefordert, den Schlüssel bereitzustellen.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „PARAMETERS-Anweisung [Interactive SQL]“ auf Seite 974
- „default_isql_encoding-Option [Interactive SQL]“ [*SQL Anywhere Server - Datenbankadministration*]
- „Interactive SQL“ [*SQL Anywhere Server - Datenbankadministration*]
- „Interactive SQL-Dienstprogramm (dbisql)“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel werden Daten aus den fiktiven Dateien *status.rpt* und *birthday.sql* gelesen:

```
READ status.rpt '160';  
READ birthday.sql [ >= '1988-1-1' ] [ <= '1988-1-30' ];
```

In diesem Beispiel wird Interactive SQL gestartet und angewiesen, eine fiktive Datei zu verarbeiten, die eine bestimmte OEM-Codepage verwendet:

```
dbisql READ ENCODING 'cp437' myfile.sql
```

READTEXT-Anweisung [T-SQL]

Liest ab einem bestimmten Ausgangspunkt Text- und Imagewerte aus der Datenbank und führt dies über die angegebene Byteanzahl fort. Diese Funktion wird nur für die Kompatibilität mit Transact-SQL verfügbar gemacht und ihre Verwendung wird nicht empfohlen.

Syntax

```
READTEXT table-name.column-name  
text-pointer-offset text-size  
[ HOLDLOCK ]
```

Bemerkungen

READTEXT wird verwendet, um CHAR-, NCHAR- und BINARY-Spalten aus einer Datenbank zu lesen. Sie können in Ansichten keine READTEXT-Vorgänge ausführen.

Privilegien

Sie müssen Eigentümer sein, SELECT-Privilegien für die Tabelle haben oder das SELECT ANY TABLE-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- [„WRITETEXT-Anweisung \[T-SQL\]“ auf Seite 1127](#)
- [„GET DATA-Anweisung \[ESQL\]“ auf Seite 876](#)
- [„TEXTPTR-Funktion \[Text und Bild\]“ auf Seite 408](#)

Standards und Kompatibilität

- **SQL/2008** Transact-SQL-Erweiterung.

Beispiel

Das folgende Embedded SQL-Beispiel verwendet TEXTPTR, um die Spalte Description zu finden, die mit ProductID 500 in der Tabelle MarketingInformation verknüpft ist.

Der Textzeiger wird in der Variablen **txtptr** gespeichert und als Parameter an die READTEXT-Anweisung übergeben, die 55 Byte, beginnend beim Spaltenoffset 181, zurückgibt.

```
EXEC SQL BEGIN DECLARE SECTION;
char          hostvar[100];
EXEC SQL END DECLARE SECTION;

EXEC SQL create variable txtptr binary(16);
EXEC SQL set txtptr =
    ( SELECT txtptr(Description)
      FROM GROUPO.MarketingInformation
      WHERE ProductID = '500' );
EXEC SQL PREPARE S1 FROM
    'READTEXT GROUPO.MarketingInformation.Description txtptr 181 55';
EXEC SQL EXECUTE S1 INTO :hostvar;
printf( "hostvar: %s\n", hostvar );
```

READTEXT gibt die folgende Zeichenfolge zurück.

```
Lightweight 100% organically grown cotton construction.
```

REFRESH MATERIALIZED VIEW-Anweisung

Initialisiert oder aktualisiert die Daten in einer materialisierten Ansicht, indem ihre Abfragedefinition ausgeführt wird.

Syntax

```
REFRESH MATERIALIZED VIEW view-list
[ WITH {
    ISOLATION LEVEL isolation-level
    | { EXCLUSIVE | SHARE } MODE } ]
[ FORCE BUILD ]
```

```
view-list :
[ owner.]materialized-view-name [, ... ]
```

```
isolation-level :
READ UNCOMMITTED
| READ COMMITTED
| SERIALIZABLE
| REPEATABLE READ
| SNAPSHOT
```

Parameter

WITH-Klausel Benutzen Sie die WITH-Klausel zur Angabe der Art der Sperre, die Sie während der Aktualisierung für die Basistabellen benutzen möchten. Der Sperrentyp bestimmt, wie die materialisierte Ansicht mit Daten gefüllt wird und wie die Gleichzeitigkeit von Transaktionen betroffen ist. Die Einstellung der WITH-Klausel beeinflusst nicht den Typ der Sperre, die für die materialisierte Ansicht selbst gesetzt wurde. Sie ist immer eine Exklusivsperre. Folgende Klauseln können Sie für Sperren angeben:

- **ISOLATION LEVEL *isolation-level*** Verwenden Sie WITH ISOLATION LEVEL, um die Isolationsstufe für die Ausführung des Aktualisierungsvorgangs zu ändern. Die ursprüngliche Isolationsstufe wird für die Verbindung wiederhergestellt, nachdem die Anweisung ausgeführt wurde.

Bei Sofortansichten kann die *isolation-level* nur SERIALIZABLE sein.

Für die Snapshot-Isolation wird nur die Snapshot-Isolationsstufe von der REFRESH MATERIALIZED VIEW-Anweisung unterstützt. Sie müssen SNAPSHOT als Isolationsstufe angeben. Die Isolationsstufen "statement-snapshot" und "readonly-statement-snapshot" werden nicht unterstützt.

- **EXCLUSIVE MODE** Verwenden Sie WITH EXCLUSIVE MODE, wenn Sie die Isolationsstufe nicht ändern wollen, aber sicherstellen möchten, dass die Daten aktualisiert werden, um mit den festgeschriebenen Daten in den Basistabellen konsistent zu sein. Wenn Sie WITH EXCLUSIVE MODE verwenden, werden exklusive Tabellensperren in allen Basistabellen gesetzt und keine andere Transaktion kann Abfragen, Aktualisierungen oder andere Aktionen mit den Basistabellen durchführen, bis der Aktualisierungsvorgang abgeschlossen ist. Wenn keine exklusiven Tabellensperren erhalten werden können, schlägt die Aktualisierung fehl und es wird ein Fehler zurückgegeben.
- **SHARE MODE** Verwenden Sie WITH SHARE MODE, um anderen Transaktionen Schreibzugriff auf Basistabellen zu gewähren, während der Aktualisierungsvorgang läuft. Wenn diese Klausel angegeben wird, werden vor dem Ausführen des Aktualisierungsvorgangs gemeinsame Tabellensperren für alle Basistabellen eingerichtet, bis der Aktualisierungsvorgang abgeschlossen ist.

FORCE BUILD-Klausel Standardmäßig gilt: Wenn Sie eine REFRESH MATERIALIZED VIEW-Anweisung ausführen, prüft der Datenbankserver, ob die materialisierte Ansicht veraltet ist (d.h. ob die Basistabellen geändert wurden, seit die materialisierte Ansicht aktualisiert wurde). Wenn sie nicht veraltet ist, wird die Aktualisierung nicht durchgeführt. Verwenden Sie die FORCE BUILD-Klausel, um eine Aktualisierung der materialisierten Ansicht unabhängig davon vorzunehmen, ob die materialisierte Ansicht veraltet ist oder nicht.

Bemerkungen

Benutzen Sie die Anweisung, um die materialisierten Ansichten zu initialisieren oder zu aktualisieren, die in der *view-list* enthalten sind.

Wenn eine REFRESH MATERIALIZED VIEW-Anweisung für eine materialisierte Ansicht ausgeführt wird, die nicht veraltet ist, wird keine Aktualisierung vorgenommen, wenn nicht die Klausel FORCE BUILD angegeben ist.

Das Standardaktualisierungsverhalten für Sperren und gleichzeitigen Zugriff auf Daten ist wie folgt:

- Bei einer Sofortansicht ist das Standardaktualisierungsverhalten WITH SHARE MODE, gleichgültig ob die Snapshot-Isolation aktiviert ist oder nicht.
- Bei einer manuellen Ansicht und *aktiver* Snapshot-Isolation ist das Standardverhalten WITH ISOLATION LEVEL SNAPSHOT.
- Bei einer manuellen Ansicht und *nicht aktiver* Snapshot-Isolation ist das Standardverhalten WITH SHARE MODE.

Für verschiedene Optionen müssen bestimmte Werte eingestellt werden, damit eine REFRESH MATERIALIZED VIEW-Anweisung erfolgreich verläuft und die Ansicht für die Optimierung verwendet werden kann. Außerdem werden Optionseinstellungen für jede materialisierte Ansicht gespeichert, wenn sie erstellt wird. Um die Ansicht zu aktualisieren oder die Ansicht in der Optimierung zu verwenden, müssen diese Optionseinstellungen zu den aktuellen Optionen passen.

Wenn eine Aktualisierung nach einer teilweisen Verarbeitung fehlschlägt, verbleibt die Ansicht in einem nicht initialisierten Status und die Daten können nicht mehr in das Stadium zurückversetzt werden, das vor dem Start der Aktualisierung aktiv war. Prüfen Sie den beim Fehlschlag der Aktualisierung aufgetretenen Fehler, beheben Sie die Ursache des Fehlers und führen Sie die REFRESH MATERIALIZED VIEW-Anweisung nochmals aus.

Sie können auch die IMMEDIATE REFRESH-Klausel der ALTER MATERIALIZED VIEW-Anweisung verwenden, um die Ansicht so zu ändern, dass sie sofort aktualisiert wird, wenn die Basisdaten verändert werden.

Diese Anweisung kann nicht ausgeführt werden, wenn die Verbindung mit der WITH HOLD-Klausel geöffnete Cursor hat, die entweder Anweisungs- oder Transaktions-Snapshots verwenden.

Privilegien

Sie müssen Eigentümer der materialisierten Ansicht sein oder das INSERT-Privileg für sie haben. Außerdem müssen Sie Eigentümer der Basistabellen sein, das SELECT-Privileg für sie haben oder das SELECT ANY TABLE-Systemprivileg haben.

Nebenwirkungen

Geöffnete Cursor, die die materialisierte Ansicht referenzieren, werden geschlossen.

Ein Checkpoint wird am Anfang der Ausführung durchgeführt.

Automatische Festschreibungen werden am Anfang und am Ende der Ausführung durchgeführt.

Während der Ausführung wird mithilfe der BLOCKING-Verbindungsoption eine exklusive Schemasperrung für die zu aktualisierende materialisierte Ansicht gesetzt und gemeinsame Tabellensperrungen, ohne Blockierung, werden für alle Tabellen gesetzt, die von der materialisierten Ansicht referenziert werden. Wenn die WITH-Klausel angegeben ist, können zusätzliche Sperren in der Basistabelle gesetzt werden. Überdies ist die materialisierte Ansicht bis zum Abschluss der Aktualisierung in einem nicht initialisierten Status, wodurch sie dem Datenbankserver für die Abfrageoptimierung oder Abfrageausführung nicht zur Verfügung steht.

Wenn die REFRESH MATERIALIZED VIEW-Anweisung mit der Snapshot-Isolation ausgeführt wird, enthält das Transaktionslog der Datenbank sowohl den REFRESH-Anweisungstext und Kopien aller Zeilen, die in die materialisierte Ansicht eingefügt werden. Die einzelnen Zeilen sind erforderlich, um zu gewährleisten, dass bei einer erforderlichen Wiederherstellung der Datenbank der Inhalt der Ansicht nach der Wiederherstellung genau zu dem Inhalt der Ansicht passt, nachdem die ursprüngliche Fertigstellung der Anweisung REFRESH MATERIALIZED VIEW erfolgt ist. Die einzelnen Zeilen im Transaktionslog werden außerdem einzeln angewendet, wenn die Datenbank gespiegelt wird. Aus diesem Grund kann es sinnvoll sein, die Häufigkeit von REFRESH MATERIALIZED VIEW-Anweisungen zu begrenzen, wenn die Snapshot-Isolation verwendet wird, oder das Transaktionslog regelmäßig mit der

BACKUP-Anweisung zu kürzen, um den für das Transaktionslog erforderlichen Speicherplatz zu verringern.

Siehe auch

- „Materialisierte Ansichten“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Einschränkungen für materialisierte Ansichten“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „ALTER MATERIALIZED VIEW-Anweisung“ auf Seite 476
- „BACKUP-Anweisung“ auf Seite 548
- „CREATE MATERIALIZED VIEW-Anweisung“ auf Seite 652
- „sa_refresh_materialized_views-Systemprozedur“ auf Seite 1294
- „sa_materialized_view_info-Systemprozedur“ auf Seite 1258
- „sa_materialized_view_can_be_immediate-Systemprozedur“ auf Seite 1256

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Gesetzt den Fall, Sie erstellen eine materialisierte Ansicht EmployeeConfid99 und füllen sie mit Daten unter Verwendung der folgenden Anweisungen:

```
CREATE MATERIALIZED VIEW EmployeeConfid99 AS
  SELECT EmployeeID, Employees.DepartmentID, SocialSecurityNumber, Salary,
  ManagerID,
  Departments.DepartmentName, Departments.DepartmentHeadID
  FROM GROUPO.Employees, GROUPO.Departments
  WHERE Employees.DepartmentID=Departments.DepartmentID;
REFRESH MATERIALIZED VIEW EmployeeConfid99;
```

Später, nachdem die Ansicht benutzt wurde, möchten Sie die Ansicht mit der Isolationsstufe READ COMMITTED aktualisieren (Isolationsstufe 1) und die Ansicht neu aufbauen. Sie könnten die folgende Anweisung ausführen:

```
REFRESH MATERIALIZED VIEW EmployeeConfid99
  WITH ISOLATION LEVEL READ COMMITTED
  FORCE BUILD;
```

Vorsicht

Wenn Sie dieses Beispiel durchgeführt haben, löschen Sie die von Ihnen erstellte materialisierte Ansicht (DROP MATERIALIZED VIEW EmployeeConfid99). Andernfalls können Sie keine Schemaänderungen an ihren Basistabellen Employees und Departments durchführen, wenn Sie andere Beispiele ausprobieren. Sie können das Schema einer Tabelle nicht ändern, die aktivierte, materialisierte Ansichten haben. Siehe „[Materialisierte Ansichten löschen](#)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

REFRESH TEXT INDEX-Anweisung

Aktualisiert einen Textindex.

Syntax

```

REFRESH TEXT INDEX text-index-name ON [ owner.]table-name
[ WITH {
    ISOLATION LEVEL isolation-level
    | EXCLUSIVE MODE
    | SHARE MODE } ]
[ FORCE { BUILD | INCREMENTAL } ]

```

Parameter

WITH-Klausel Benutzen Sie die WITH-Klausel zur Angabe der Art der Sperre, die Sie während der Aktualisierung für die Basistabellen benutzen möchten. Die Arten der gesetzten Sperren legen fest, wie der Textindex mit Daten gefüllt wird und wie die Gleichzeitigkeit von Transaktionen betroffen ist. Wenn Sie die WITH-Klausel nicht angeben, ist der Standardwert WITH ISOLATION LEVEL READ UNCOMMITTED, unabhängig von der für die Verbindung eingerichteten Isolationsstufe.

Sie können folgende Optionen für die WITH-Klausel festlegen:

- **ISOLATION LEVEL *isolation-level*** Verwenden Sie WITH ISOLATION LEVEL, um die Isolationsstufe für die Ausführung des Aktualisierungsvorgangs zu ändern.

Die ursprüngliche Isolationsstufe der Verbindung wird am Ende der Anweisungsausführung wiederhergestellt.

- **EXCLUSIVE MODE** Verwenden Sie WITH EXCLUSIVE MODE, wenn Sie die Isolationsstufe nicht ändern wollen, aber sicherstellen möchten, dass die Daten aktualisiert werden, um mit den festgeschriebenen Daten in der Basistabelle konsistent zu sein. Wenn Sie WITH EXCLUSIVE MODE verwenden, werden exklusive Tabellensperren in der Basistabelle gesetzt und keine andere Transaktion kann Abfragen, Aktualisierungen oder andere Aktionen mit den Basistabellen durchführen, bis der Aktualisierungsvorgang abgeschlossen ist. Wenn keine Tabellensperren erhalten werden können, schlägt die Aktualisierung fehl und es wird ein Fehler zurückgegeben.
- **SHARE MODE** Verwenden Sie WITH SHARE MODE, um anderen Transaktionen Schreibzugriff auf die Basistabelle zu gewähren, während der Aktualisierungsvorgang läuft. Wenn diese Klausel angegeben wird, werden vor dem Ausführen des Aktualisierungsvorgangs gemeinsame Tabellensperren für die Basistabelle gesetzt und bleiben erhalten, bis der Aktualisierungsvorgang abgeschlossen ist.

FORCE-Klausel Verwenden Sie diese Klausel, um die Aktualisierungsmethode anzugeben. Wenn diese Klausel nicht angegeben ist, entscheidet der Datenbankserver, ob eine inkrementelle Aktualisierung oder ein kompletter Neuaufbau erfolgt. Dabei richtet er sich danach, wie stark die Tabelle geändert wurde.

- **FORCE BUILD-Klausel** Aktualisiert den Textindex durch Neuerstellen. Verwenden Sie diese Klausel, um einen kompletten Neuaufbau des Textindexes zu erzwingen.
- **FORCE INCREMENTAL-Klausel** Aktualisiert den Textindex nur auf der Grundlage der Änderungen an der zugrunde liegenden Tabelle. Eine inkrementelle Aktualisierung nimmt weniger Zeit in Anspruch, wenn keine signifikante Menge von Aktualisierungen an der zugrunde liegenden Tabelle vorgenommen wurden. Verwenden Sie diese Klausel, um eine inkrementelle Aktualisierung des Textindexes zu erzwingen.

Bei einer inkrementellen Aktualisierung werden gelöschte Einträge nicht aus dem Textindex entfernt. Daher ist der Textindex möglicherweise größer als erwartet, da er sowohl die aktuellen als auch historische Daten enthält. In der Regel tritt dieses Problem bei Textindizes auf, die immer mit der FORCE INCREMENTAL-Klausel manuell aktualisiert werden. Bei automatisch aktualisierten Textindizes werden historische Daten automatisch gelöscht, wenn sie 50% der Gesamtgröße des Textindex ausmachen.

Bemerkungen

Diese Anweisung kann nur für Textindizes benutzt werden, die als MANUAL REFRESH oder AUTO REFRESH definiert sind.

Wenn Sie die FORCE-Klausel verwenden, können Sie die Ergebnisse der Systemprozedur `sa_text_index_stats` prüfen, um zu entscheiden, ob ein kompletter Neuaufbau (FORCE BUILD) oder eine inkrementelle Aktualisierung (FORCE INCREMENTAL) besser geeignet ist.

Sie können die REFRESH TEXT INDEX-Anweisung nicht für einen Textindex ausführen, der als IMMEDIATE REFRESH definiert ist.

Für MANUAL REFRESH-Textindizes verwenden Sie die Systemprozedur `sa_text_index_stats`, um zu ermitteln, ob der Textindex aktualisiert werden muss. Teilen Sie `pending_length` durch `doc_length` und verwenden Sie den Prozentsatz als Richtlinie für die Entscheidung, ob eine Aktualisierung erforderlich ist. Um zu ermitteln, welche Art des Neuaufbaus erforderlich ist, verwenden Sie dieselbe Vorgehensweise für `deleted_length` und `doc_count`.

Diese Anweisung kann nicht ausgeführt werden, wenn mit der WITH HOLD-Klausel geöffnete Cursor vorhanden sind, die entweder Anweisungs- oder Transaktions-Snapshots verwenden.

Privilegien

Sie müssen Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- REFERENCES-Privileg für die Tabelle
- CREATE ANY INDEX-Systemprivileg
- ALTER ANY INDEX-Systemprivileg
- CREATE ANY OBJECT-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Konzepte und Referenz zu Textindizes“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Transaktionen und Isolationsstufen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Isolationsstufen und Konsistenz“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „CREATE TEXT INDEX-Anweisung“ auf Seite 759
- „ALTER TEXT INDEX-Anweisung“ auf Seite 536
- „DROP TEXT INDEX-Anweisung“ auf Seite 834
- „TRUNCATE TEXT INDEX-Anweisung“ auf Seite 1091
- „Tabellensperren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Snapshot-Isolation“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Textindex-Aktualisierungstypen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „sa_refresh_text_indexes-Systemprozedur“ auf Seite 1295
- „sa_text_index_stats-Systemprozedur“ auf Seite 1342

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung aktualisiert einen fiktiven Textindex namens MarketingTextIndex und erzwingt seinen Neuaufbau.

```
REFRESH TEXT INDEX MarketingTextIndex ON GROUP0.MarketingInformation
FORCE BUILD;
```

REFRESH TRACING LEVEL-Anweisung

Lädt erneut die Protokollierungsstufen aus der sa_diagnostic_tracing_level-Tabelle, während eine Protokollierungssitzung läuft.

Syntax

```
REFRESH TRACING LEVEL
```

Bemerkungen

Diese Anweisung wird verwendet, um die Protokollierungsstufen erneut aus der sa_diagnostic_tracing_level-Tabelle zu laden. Sie muss aus einer Datenbank heraus aufgerufen werden, deren Profil erstellt wird.

Wenn eine Protokollierungssitzung zum ersten Mal gestartet wird, werden Zeilen aus der sa_diagnostic_tracing_level-Tabelle in den Serverspeicher geladen, um zu steuern, welche Art von Informationen protokolliert wird. Wenn Sie die Typen der zu protokollierenden Daten ändern möchten, ohne dafür die Protokollierungssitzung zu stoppen und neu zu starten, können Sie die entsprechenden Zeilen in der sa_diagnostic_tracing_level-Tabelle manuell löschen bzw. einfügen und anschließend die REFRESH TRACING LEVEL-Anweisung ausführen, um die Einstellungen neu zu laden.

Um die aktuellen Protokollierungsstufen anzuzeigen, fragen Sie die sa_diagnostic_tracing_level-Tabelle folgendermaßen ab:

```
SELECT * FROM sa_diagnostic_tracing_level WHERE enabled = 1;
```

Angenommen, Sie möchten ein Performanceproblem beheben. Sie verwenden eine hohe Protokollierungsstufe für die gesamte Datenbank, um die Abfragen zu erfassen, die das Problem verursachen. Nachdem Sie die Protokollierungssitzung gestartet haben, stellen Sie fest, dass das Erfassen aller Abfragen für alle Benutzer Ihre Datenbank allzu sehr verlangsamt. Daher beschließen Sie, die Protokollierung auf einen Benutzer zu beschränken und darauf zu warten, dass dieser Benutzer ein Problem meldet. Die Protokollierungssitzung wollen Sie allerdings nicht stoppen, um die Einstellungen zu ändern.

Sie können dies in Sybase Central tun, indem Sie den Assistenten zur Datenbankprotokollierung verwenden, was die empfohlene Methode ist. Sie können dies allerdings auch von der Befehlszeile aus tun, indem Sie Zeilen in der `sa_diagnostic_tracing_level`-Tabelle, in denen `scope=DATABASE` und `enabled=1` gilt, durch die entsprechenden Zeilen ersetzen, in denen `scope=USER`, `identifizier=Benutzer-ID` `enabled=1` usw. gilt. Dann führen Sie eine `REFRESH TRACING LEVEL`-Anweisung aus, um mit der Protokollierung und den neuen Einstellungen fortzufahren.

Privilegien

Sie müssen das `MANAGE PROFILING`-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „[ATTACH TRACING-Anweisung](#)“ auf Seite 546
- „[DETACH TRACING-Anweisung](#)“ auf Seite 801
- „[Diagnoseprotokollierung](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „[sa_diagnostic_tracing_level-Tabelle](#)“ auf Seite 1156

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesen Beispiel wird die Protokollierungsstufe aktualisiert:

```
REFRESH TRACING LEVEL;
```

RELEASE SAVEPOINT-Anweisung

Gibt einen Savepoint innerhalb der aktuellen Transaktion frei.

Syntax

```
RELEASE SAVEPOINT [ savepoint-name ]
```

Bemerkungen

Gibt einen Savepoint frei. Der *savepoint-name* ist ein Bezeichner, der für eine `SAVEPOINT`-Anweisung innerhalb der aktuellen Transaktion angegeben wurde. Wenn der *savepoint-name* weggelassen wird, wird der neueste Savepoint freigegeben.

Bei der Freigabe eines Savepoints wird kein FESTSCHREIBEN (COMMIT) vorgenommen. Der Savepoint wird einfach aus der Liste der gegenwärtig aktiven Savepoints entfernt.

Innerhalb der aktuellen Transaktion muss es einen entsprechenden SAVEPOINT geben.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „BEGIN TRANSACTION-Anweisung [T-SQL]“ auf Seite 561
- „COMMIT-Anweisung“ auf Seite 575
- „ROLLBACK-Anweisung“ auf Seite 1015
- „ROLLBACK TO SAVEPOINT-Anweisung“ auf Seite 1015
- „SAVEPOINT-Anweisung“ auf Seite 1019
- „Savepoints innerhalb von Transaktionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** RELEASE SAVEPOINT ist Teil der optionalen SQL/2008-Sprachenfunktion T271 (Savepoints).

REMOTE RESET-Anweisung [SQL Remote]

Startet alle Subskriptionen eines entfernten Benutzers in einer einzigen Transaktion in benutzerdefinierten Datenbankextraktionsprozeduren.

Syntax

REMOTE RESET *userid*

Bemerkungen

Diese Anweisung startet alle Subskriptionen eines entfernten Benutzers in einer einzigen Transaktion. Er setzt die log_sent- und confirm_sent-Werte in der ISYSREMOTEUSER-Tabelle auf die aktuelle Position im Transaktionslog. Er setzt auch bei allen Subskriptionen für diesen entfernten Benutzer die CREATED- und STARTED-Werte in ISYSSUBSCRIPTION auf die aktuelle Position im Transaktionslog. Die Anweisung führt kein COMMIT durch. Sie müssen ein explizites COMMIT nach diesem Aufruf vornehmen.

Um einen für die aktive Datenbank unbedenklichen Extraktionsprozess zu schreiben, müssen die Daten auf Isolationsstufe 3 in derselben Transaktion extrahiert werden, in der die Subskriptionen gestartet werden.

Diese Anweisung ist eine Alternative zu START SUBSCRIPTION. START SUBSCRIPTION hat ein implizites COMMIT als Nebenwirkung. Daher ist es unmöglich, mit START SUBSCRIPTION mehrere Subskriptionen eines entfernten Benutzers in einer Transaktion zu starten.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Bei dieser Anweisung wird kein automatisches COMMIT durchgeführt.

Siehe auch

- „START SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 1069
- „SYSREMOTEUSER-Systemansicht“ auf Seite 1476
- „Mehrere entfernte Datenbanken erstellen“ [SQL Remote]
- „Starten von Subskriptionen“ [SQL Remote]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung setzt die Subskriptionen für den entfernten Benutzer Sam_Singer zurück:

```
REMOTE RESET Sam_Singer;  
COMMIT;
```

REMOVE EXTERNAL OBJECT-Anweisung

Entfernt ein externes Objekt aus der Datenbank.

Syntax

REMOVE EXTERNAL OBJECT *object-name*

Parameter

object-name Der Name des externen Objekts.

Bemerkungen

Weitere Hinweise zu externen Umgebungen finden Sie unter „Unterstützung für externe Umgebungen in SQL Anywhere“ [SQL Anywhere Server - Programmierung].

Privilegien

Sie müssen Eigentümer des externen Objekts sein oder das MANAGE ANY EXTERNAL OBJECT-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Unterstützung für externe Umgebungen in SQL Anywhere“ [*SQL Anywhere Server - Programmierung*]
- „ALTER EXTERNAL ENVIRONMENT-Anweisung“ auf Seite 463
- „INSTALL EXTERNAL OBJECT-Anweisung“ auf Seite 923
- „START EXTERNAL ENVIRONMENT-Anweisung“ auf Seite 1066
- „STOP EXTERNAL ENVIRONMENT-Anweisung“ auf Seite 1075
- „SYSEXTERNENV-Systemansicht“ auf Seite 1448
- „SYSEXTERNENVOBJECT-Systemansicht“ auf Seite 1450

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

REMOVE JAVA-Anweisung

Entfernt eine Klasse oder eine JAR-Datei aus einer Datenbank.

Syntax

```
REMOVE JAVA
{ CLASS java-class-name [ , java-class-name ... ]
| JAR jar-name [ , jar-name ... ] }
```

Parameter

CLASS-Klausel Der *java-class-name*-Parameter ist der Name einer oder mehrerer Javaklassen, die entfernt werden sollen. Bei diesen Klassen muss es sich um in der aktuellen Datenbank installierte Klassen handeln.

JAR-Klausel Der *jar-name* ist ein in Apostrophe eingeschlossener Zeichenfolgenwert mit einer maximalen Länge von 255. Jeder *jar-name* muss mit dem *jar-name*-Wert einer in der aktuellen Datenbank gespeicherten JAR-Datei übereinstimmen. Die Gleichheit von *jar-name* wird durch die Vergleichsregeln für Zeichenfolgen des SQL-Systems bestimmt.

Bemerkungen

Entfernt eine Klasse oder eine JAR-Datei aus der Datenbank. Die Klasse bzw. JAR-Datei muss bereits installiert sein.

Diese Anweisung wird unter Windows Mobile nicht unterstützt.

Privilegien

Sie müssen Eigentümer der Klasse bzw. JAR-Datei sein oder das MANAGE ANY EXTERNAL OBJECT-Systemprivileg haben.

Siehe auch

- „INSTALL JAVA-Anweisung“ auf Seite 925
- „SYSJAR-Systemansicht“ auf Seite 1458
- „SYSJARCOMPONENT-Systemansicht“ auf Seite 1459
- „SYSJAVACLASS-Systemansicht“ auf Seite 1460

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird eine fiktive Java-Klasse namens Demo aus der aktuellen Datenbank entfernt:

```
REMOVE JAVA CLASS Demo;
```

In diesem Beispiel wird eine fiktive Java-JAR-Datei namens myJar aus der aktuellen Datenbank entfernt:

```
REMOVE JAVA JAR 'myJar';
```

REORGANIZE TABLE-Anweisung

Defragmentiert Tabellen, wenn eine vollständiger Neuerstellung nicht möglich ist, weil der fortlaufende Zugriff auf die Datenbank gewährleistet sein muss.

Syntax

```
REORGANIZE TABLE [ owner.]table-name  
[ { PRIMARY KEY  
| FOREIGN KEY foreign-key-name  
| INDEX index-name } ]
```

Parameter

Reorganisieren Sie die Tabelle gemäß den nachstehend angegebenen Werten:

PRIMARY KEY-Klausel Organisiert den Primärschlüsselindex für die Tabelle neu

FOREIGN KEY-Klausel Organisiert den angegebenen Fremdschlüssel neu

INDEX-Klausel Organisiert den angegebenen Index neu

Bemerkungen

Die Tabellenfragmentierung kann die Performance beeinträchtigen. Mit dieser Anweisung können Sie die Zeilen in einer Tabelle defragmentieren oder Indizes komprimieren, die aufgrund von DELETE-Anweisungen klein geworden sind. Sie kann auch die Gesamtanzahl der Seiten reduzieren, die benutzt werden, um die Tabelle und ihre Indizes zu speichern, sowie eventuell die Ebenen in einem Indexbaum verringern. Sie führt allerdings nicht zu einer Verringerung der Gesamtgröße der Datenbankdatei. Es wird empfohlen, dass Sie die Systemprozeduren `sa_table_fragmentation` und `sa_index_density` verwenden, um Tabellen auszuwählen, für die sich eine Verarbeitung lohnt.

Wenn weder ein Index noch ein Schlüssel angegeben sind, werden Zeilen in der Tabelle beim Neuorganisieren defragmentiert, indem Gruppen von Zeilen gelöscht und wieder eingefügt werden. Für

jede Gruppe wird eine Exklusivsperrung in der Tabelle gesetzt. Nachdem die Gruppe verarbeitet wurde, wird die Sperrung freigegeben und erneut erworben (ggf. nach Wartezeit), sodass andere Verbindungen Gelegenheit erhalten, auf die Tabelle zuzugreifen. Checkpoints werden unterbrochen, während eine Gruppe verarbeitet wird. Wenn eine Gruppe fertig gestellt ist, kann ein Checkpoint auftreten. Die Zeilen werden in der Reihenfolge des Clustered-Indexes verarbeitet, wenn einer existiert. Sonst werden sie in der Reihenfolge des Primärschlüssels verarbeitet. Wenn die Tabelle keinen Clustered-Index oder einen Primärschlüssel hat, wird ein Fehler zurückgegeben. Die verarbeiteten Zeilen werden am Ende der Tabelle wieder eingefügt, sodass die Zeilen am Ende des Verfahrens in Clustern nach Primärschlüssel vorliegen. Der erforderliche Aufwand ist der gleiche, unabhängig vom ursprünglichen Grad der Zeilenfragmentierung.

Wenn ein Index oder ein Schlüssel angegeben wird, verarbeitet das System den angegebenen Index. Für die Dauer des Vorgangs gilt eine Exklusivsperrung für die Tabelle, und Checkpoints werden unterbrochen. Versuche, über andere Verbindungen auf die Tabelle zuzugreifen, werden blockiert oder schlagen fehl, je nach Einstellung der Blockierungsoption. Durch vorbereitendes Lesen der Indexseiten vor dem Setzen der Exklusivsperrung wird die Dauer der Sperrung minimiert.

Da die Neuorganisation viele Seiten ändern kann, wird das Checkpoint-Log unter Umständen sehr groß. Dies kann zu einem Anwachsen der Datenbankdatei führen. Dieses Anwachsen ist allerdings nur temporär, da das Checkpoint-Log beim Herunterfahren gelöscht und die Datei dabei gekürzt wird.

Diese Anweisung wird im Transaktionslog nicht protokolliert.

Diese Anweisung kann nicht ausgeführt werden, wenn mit der WITH HOLD-Klausel geöffnete Cursor vorhanden sind, die entweder Anweisungs- oder Transaktions-Snapshots verwenden.

Während des Ausführens dieser Anweisung können Sie Meldungen zum Verarbeitungsfortschritt anfordern.

Sie können auch mithilfe der Verbindungseigenschaft "Progress" feststellen, wie viel von der Anweisung ausgeführt wurde.

Diese Anweisung wird unter Windows Mobile nicht unterstützt.

Privilegien

Sie müssen Eigentümer der Tabelle sein oder das REORGANIZE ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Vor dem Start der Neuorganisation wird versucht, mit einem Checkpoint die Anzahl der freien Seiten zu maximieren. Außerdem wird beim Ausführen der REORGANIZE TABLE-Anweisung eine implizierte Festschreibung bei ca. jeder 100. Zeile durchgeführt. Daher finden bei der Neuorganisation einer großen Tabelle vielfache Festschreibungen statt.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Siehe auch

- [Progress-Verbindungseigenschaft](#) [*SQL Anywhere Server - Datenbankadministration*]
- [„progress_messages-Option“](#) [*SQL Anywhere Server - Datenbankadministration*]
- [„Snapshot-Isolation“](#) [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Beispiele

Die folgende Anweisung organisiert den Primärschlüsselindex für die Tabelle 'Employees' neu:

```
REORGANIZE TABLE GROUPO.Employees  
PRIMARY KEY;
```

Die folgende Anweisung organisiert die Tabellenseiten der Tabelle 'Employees' neu:

```
REORGANIZE TABLE GROUPO.Employees;
```

Die folgende Anweisung organisiert den Index 'IX_product_name' der Tabelle 'Products' neu:

```
REORGANIZE TABLE GROUPO.Products  
INDEX IX_product_name;
```

Die folgende Anweisung organisiert den Fremdschlüssel 'FK_DepartmentID_DepartmentID' für die Tabelle 'Employees' neu:

```
REORGANIZE TABLE GROUPO.Employees  
FOREIGN KEY FK_DepartmentID_DepartmentID;
```

RESIGNAL-Anweisung [SP]

Signalisiert eine Ausnahmebedingung erneut.

Syntax

```
RESIGNAL [ exception-name ]
```

Bemerkungen

In einer Ausnahmeroutine können Sie mit RESIGNAL die zusammengesetzte Anweisung verlassen, und die Ausnahmebedingung bleibt weiterhin aktiv, oder Sie können die Anweisung verlassen und über eine weitere Ausnahmebedingung berichten. Die Ausnahmebedingung wird von einer anderen Ausnahmeroutine verarbeitet oder wieder an die Anwendung zurückgegeben.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „[SIGNAL-Anweisung \[SP\]](#)“ auf Seite 1061
- „[BEGIN-Anweisung](#)“ auf Seite 557
- „[Ausnahmeroutinen](#)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „[RAISERROR-Anweisung](#)“ auf Seite 982

Standards und Kompatibilität

- **SQL/2008** Die RESIGNAL-Anweisung ist Teil der optionalen SQL/2008-Sprachenfunktion P002 (Verarbeitungsvollständigkeit).
- **Transact-SQL** Die RESIGNAL-Anweisung kann nicht in zusammengesetzten Anweisungen und Prozeduren in Transact-SQL verwendet werden.

Beispiel

Das Beispielfragment gibt alle Ausnahmebedingungen außer SQLSTATE 52003 an die Anwendung zurück.

```
...
DECLARE COLUMN_NOT_FOUND EXCEPTION
    FOR SQLSTATE '52003';
...
EXCEPTION
WHEN COLUMN_NOT_FOUND THEN
SET message='Column not found';
WHEN OTHERS THEN
RESIGNAL;
```

RESTORE DATABASE-Anweisung

Stellt eine gesicherte Datenbank aus einem Archiv wieder her.

Syntax

```
RESTORE DATABASE filename
FROM archive-root
[ CATALOG ONLY
  | [ RENAME dbspace-name TO new-dbspace-name ] ... ]
[ HISTORY { ON | OFF } ]
[ KEY encryption-key ]
```

filename : *string* | *variable*
archive-root : *string* | *variable*
new-dbspace-name : *string* | *variable*

Parameter

FROM-Klausel Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

CATALOG ONLY-Klausel Ruft Informationen über das benannte Archiv ab und stellt sie in die Sicherungsverlaufsdatei (*backup.syb*), stellt aber keine Daten vom Archiv wieder her.

RENAME-Klausel Ermöglicht Ihnen die Angabe eines neuen Speicherorts für jeden DBSpace. Sie können die RENAME-Klausel nicht verwenden, um den DBSpace-Namen zu ändern. Sie können jedoch den Dateinamen mit der RENAME-Klausel ändern.

HISTORY-Klausel Sie können damit steuern, ob der RESTORE DATABASE-Vorgang in der Verlaufsdatei *backup.syb* gespeichert wird.

KEY-Klausel Hiermit können Sie den Chiffrierschlüssel angeben, wenn Sie eine stark verschlüsselte Datenbank wiederherstellen, die mit aktivierter Eliminierung freier Seiten gesichert wurde. Wenn die Sicherung mit deaktivierter Eliminierung freier Seiten durchgeführt wurde, brauchen Sie zum Wiederherstellen der Datenbank den Chiffrierschlüssel nicht anzugeben. Der Schlüssel kann entweder eine Zeichenfolge oder ein Variablenname sein.

Ab Version 12 können Sie keine Archivsicherungen wiederherstellen, die mit Version 11 oder früheren Datenbankservern erstellt wurden.

Bemerkungen

Wenn nicht HISTORY OFF angegeben ist, aktualisiert jeder RESTORE DATABASE-Vorgang eine Sicherungsverlaufsdatei namens *backup.syb*. Die Datei protokolliert die BACKUP- und RESTORE-Vorgänge, die auf einem Datenbankserver ausgeführt wurden. Es kann sinnvoll sein, zu verhindern, dass der RESTORE DATABASE-Vorgang in *backup.syb* gespeichert wird, wenn folgende Bedingungen vorliegen:

- Sie führen häufig RESTORE DATABASE-Vorgänge durch.
- Es gibt keine Prozedur, um die *backup.syb*-Datei regelmäßig zu archivieren oder zu löschen.
- Der Speicherplatz ist sehr beschränkt.

Hinweis

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „Sandboxing“ [[SQL Anywhere Server - Datenbankadministration](#)].

RESTORE DATABASE ersetzt die wiederherzustellende Datenbank. Wenn Sie Sicherungen benötigen, die inkrementell durchgeführt werden, verwenden Sie das Imageformat des BACKUP-Befehls und sichern Sie nur das Transaktionslog. Sicherungskopien auf Band werden jedoch nicht unterstützt.

Während des Ausführens dieser Anweisung können Sie Meldungen zum Verarbeitungsfortschritt anfordern.

Sie können auch mithilfe der Verbindungseigenschaft "Progress" feststellen, wie viel von der Anweisung ausgeführt wurde.

Sie können nicht mit der Datenbank verbunden sein, die Sie wiederherstellen möchten. Sie müssen mit einer anderen Datenbank verbunden sein, z.B. mit der Dienstprogrammdateiabank. Die Datenbank, die Sie verschlüsseln möchten, darf nicht laufen.

Diese Anweisung wird unter Windows Mobile nicht unterstützt.

Privilegien

Ob Sie diese Anweisung ausführen können, hängt von der Einstellung der Datenbankoption -gu ab und davon, ob Sie das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „Vordefinierte DBSpaces“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „BACKUP-Anweisung“ auf Seite 548
- „Sicherung und Datenwiederherstellung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „SALOGDIR-Umgebungsvariable“ [[SQL Anywhere Server - Datenbankadministration](#)]
- FREE PAGE ELIMINATION-Klausel, BACKUP-Anweisung auf Seite 553
- „Datenbankserveroption -gu“ [[SQL Anywhere Server - Datenbankadministration](#)]
- Progress-Verbindungseigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]
- „progress_messages-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Windows Mobile** Unter Windows Mobile nicht unterstützt.

Beispiel

Im folgenden Beispiel wird eine fiktive Datenbank von einem Bandlaufwerk wiederhergestellt. Die Anzahl der erforderlichen Backslashes hängt davon ab, mit welcher Datenbank eine Verbindung besteht, wenn Sie RESTORE DATABASE ausführen. Die Datenbank beeinflusst die Einstellung der Option escape_character. Sie ist gewöhnlich auf ON, nur in utility_db auf OFF gesetzt. Wenn eine Verbindung zu einer anderen Datenbank als utility_db besteht, sind zusätzliche Backslashes erforderlich.

```
RESTORE DATABASE 'd:\\dbhome\\mydatabase.db'
FROM '\\\\.\\tape0';
```

RESUME-Anweisung

Nimmt die Ausführung eines Cursors, der Ergebnismengen zurückgibt, wieder auf.

Syntax

RESUME *cursor-name*

cursor-name : *identifier* | *hostvar*

Bemerkungen

Diese Anweisung nimmt die Ausführung einer Prozedur, die Ergebnismengen zurückgibt, wieder auf. Diese Prozedur wird solange ausgeführt, bis die nächste Ergebnismenge (SELECT-Anweisung ohne INTO-Klausel) feststeht. Wenn die Prozedur abgearbeitet ist und keine Ergebnismenge ermittelt wurde, wird die Warnung SQLSTATE_PROCEDURE_COMPLETE gesetzt. Diese Warnung wird ebenfalls gesetzt, wenn Sie einen Cursor für eine SELECT-Anweisungszeile wieder aufnehmen.

Die RESUME-Anweisung wird in Interactive SQL nicht unterstützt.

Der Cursor muss zuvor geöffnet worden sein.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „DECLARE CURSOR-Anweisung [ESQL] [SP]“ auf Seite 778
- „FETCH-Anweisung [ESQL] [SP]“ auf Seite 853
- „Ergebnismengen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Es folgen Beispiele für Embedded SQL.

```
EXEC SQL RESUME cur_employee;  
EXEC SQL RESUME :cursor_var;
```

RETURN-Anweisung

Mit dieser Anweisung beenden Sie eine Funktion, eine Prozedur oder einen Batch bedingungslos und geben optional einen Rückgabewert aus.

Syntax

RETURN [*expression*]

Bemerkungen

Eine RETURN-Anweisung verursacht die sofortige Beendigung eines SQL-Blocks. Wenn *expression* angegeben ist, wird der Wert für *expression* als Wert der Funktion oder der Prozedur zurückgegeben. In *expression* können keine Unterabfragen verwendet werden.

Wenn RETURN in einem inneren BEGIN-Block erscheint, ist es der äußere BEGIN-Block, der beendet wird.

Anweisungen, die auf RETURN folgen, werden nicht ausgeführt.

Innerhalb einer Funktion sollte der Ausdruck denselben Datentyp haben wie der Rückgabedatentyp der Funktion.

Innerhalb von Prozeduren wird RETURN aus Gründen der Transact-SQL-Kompatibilität verwendet und dient dazu, einen Ganzzahlfehlercode zurückzugeben.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „CREATE FUNCTION-Anweisung“ auf Seite 633
- „CREATE FUNCTION-Anweisung [externer Aufruf]“ auf Seite 617
- „CREATE FUNCTION-Anweisung [Webdienst]“ auf Seite 624
- „CREATE PROCEDURE-Anweisung“ auf Seite 681
- „CREATE PROCEDURE-Anweisung [externer Aufruf]“ auf Seite 661
- „CREATE PROCEDURE-Anweisung [Webdienste]“ auf Seite 670
- „BEGIN-Anweisung“ auf Seite 557
- „Einen Wert mit der RETURN-Anweisung zurückgeben“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Kernfunktion.

Beispiel

Die folgende Funktion gibt das Produkt aus drei Zahlen zurück:

```
CREATE FUNCTION product (
  a NUMERIC,
  b NUMERIC,
  c NUMERIC )
RETURNS NUMERIC
BEGIN
  RETURN ( a * b * c );
END;
```

Produkt aus drei Zahlen errechnen:

```
SELECT product(2, 3, 4);
```

product(2, 3, 4)
24.000000

Die folgende Prozedur verwendet die RETURN-Anweisung, um die Ausführung einer komplexen Abfrage zu vermeiden, wenn sie bedeutungslos ist:

```
CREATE PROCEDURE customer_products
( in customer_ID integer DEFAULT NULL)
RESULT ( ID integer, quantity_ordered integer )
BEGIN
  IF customer_ID NOT IN (SELECT ID FROM Customers)
  OR customer_ID IS NULL THEN
    RETURN
  ELSE
    SELECT Products.ID,sum(
```

```
        SalesOrderItems.Quantity )
FROM    GROUPO.Products,
        SalesOrderItems,
        SalesOrders
WHERE   SalesOrders.CustomerID=customer_ID
AND     SalesOrders.ID=SalesOrderItems.ID
AND     SalesOrderItems.ProductID=Products.ID
GROUP BY Products.ID
END IF
END;
```

REVOKE CONSOLIDATE-Anweisung [SQL Remote]

Verhindert, dass eine konsolidierte Datenbank SQL Remote-Nachrichten von dieser Datenbank empfängt.

Syntax

REVOKE CONSOLIDATE FROM *userid*

Bemerkungen

CONSOLIDATE-Privilegien müssen in einer entfernten Datenbank für die Benutzer-ID erteilt werden, die die konsolidierte Datenbank darstellt. Die REVOKE CONSOLIDATE-Anweisung entfernt die Benutzer-ID der konsolidierten Datenbank aus der Liste der Benutzer, die Nachrichten von der aktuellen Datenbank erhalten.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Löscht alle Subskriptionen des Benutzers.

Siehe auch

- „GRANT CONSOLIDATE-Anweisung [SQL Remote]“ auf Seite 889
- „GRANT ROLE SYS_REPLICATION_ADMIN_ROLE-Anweisung [MobiLink] [SQL Remote]“ auf Seite 896
- „REVOKE PUBLISH-Anweisung [SQL Remote]“ auf Seite 1007
- „REVOKE REMOTE-Anweisung [SQL Remote]“ auf Seite 1008

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

- Die folgende Anweisung entzieht der Benutzer-ID "Sam_Singer" den konsolidierten Status:

```
REVOKE CONSOLIDATE FROM Sam_Singer;
```

REVOKE PUBLISH-Anweisung [SQL Remote]

Beendet die Kennzeichnung der benannten Benutzer-ID als aktueller Publikationseigentümer. Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben, um Publikationseigentümerrechte entziehen zu können.

Syntax

```
REVOKE PUBLISH FROM userid
```

Bemerkungen

Jede Datenbank in einer SQL Remote-Installation wird in ausgehenden Nachrichten durch eine Publikationseigentümer-Benutzer-ID gekennzeichnet. Sie können die Benutzer-ID des aktuellen Publikationseigentümers ermitteln, indem Sie folgendermaßen den CURRENT PUBLISHER-Spezialwert abfragen:

```
SELECT CURRENT PUBLISHER;
```

Die REVOKE PUBLISH-Anweisung hebt die Kennzeichnung der benannten Benutzer-ID als Publikationseigentümer auf. Wenn Sie den Publikationseigentümer ändern möchten, müssen Sie die Publikationseigentümerrechte dem aktuellen Publikationseigentümer entziehen (REVOKE PUBLISH) und anschließend dem neuen Publikationseigentümer erteilen.

Wenn Sie die Benutzer-ID des Publikationseigentümers für eine konsolidierte oder entfernte Datenbank in einer SQL Remote-Installation ändern, müssen Sie sicherstellen, dass der Benutzer-ID des neuen Publikationseigentümers das REMOTE-Privileg für alle Datenbanken erteilt wird, die Nachrichten von der Datenbank erhalten. Um diese Änderung vornehmen zu können, müssen Sie alle Subskriptionen löschen und neu erstellen.

Entziehen Sie nicht die Publikationseigentümerrechte, während für die betreffende Datenbank aktive SQL Remote-Publikationen oder Subskriptionen vorhanden sind.

Wenn Sie Publikationseigentümerrechte entziehen und keinem neuen Benutzer erteilen, hat dies Auswirkungen auf eine SQL Remote-Installation:

- Sie können keine Daten in Tabellen einfügen, die eine CURRENT PUBLISHER-Spalte als Teil des Primärschlüssels enthalten. Ausgehende Nachrichten werden nicht durch die Benutzer-ID eines Publikationseigentümers identifiziert und deshalb von den Empfängerdatenbanken nicht akzeptiert.

Durch Ausführen dieser Anweisung wird der Wert der PUBLIC.db_publisher-Datenbankoption geändert.

Privilegien

Sie müssen das SET ANY SYSTEM OPTION-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „GRANT ROLE SYS_REPLICATION_ADMIN_ROLE-Anweisung [MobiLink] [SQL Remote]“ auf Seite 896
- „db_publisher-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „GRANT PUBLISH-Anweisung [SQL Remote]“ auf Seite 894
- „REVOKE REMOTE-Anweisung [SQL Remote]“ auf Seite 1008
- „REVOKE CONSOLIDATE-Anweisung [SQL Remote]“ auf Seite 1006
- „CURRENT PUBLISHER-Spezialwert“ auf Seite 71

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Entziehen Sie publisher_ID den Status als aktueller Publikationseigentümer.

```
REVOKE PUBLISH FROM publisher_ID;
```

REVOKE REMOTE-Anweisung [SQL Remote]

Beendet für einen Benutzer die Möglichkeit, SQL Remote-Nachrichten von dieser Datenbank zu empfangen.

Syntax

```
REVOKE REMOTE FROM userid, ...
```

Bemerkungen

Das REMOTE-Privileg ist für eine Benutzer-ID erforderlich, um Nachrichten in einer SQL Remote-Replikationsinstallation empfangen zu können. Die REVOKE REMOTE-Anweisung entfernt eine Benutzer-ID aus der Liste der Benutzer, die Nachrichten von der aktuellen Datenbank erhalten.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit). Löscht alle Subskriptionen des Benutzers.

Siehe auch

- „REVOKE PUBLISH-Anweisung [SQL Remote]“ auf Seite 1007
- „GRANT REMOTE-Anweisung [SQL Remote]“ auf Seite 900
- „REVOKE ROLE SYS_REPLICATION_ADMIN_ROLE-Anweisung [MobiLink] [SQL Remote]“ auf Seite 1013
- „REVOKE CONSOLIDATE-Anweisung [SQL Remote]“ auf Seite 1006
- „REMOTE-Privileg entziehen“ [*SQL Remote*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

```
REVOKE REMOTE FROM Sam_Singer;
```

REVOKE-Anweisung

Entzieht Benutzern und Rollen erteilte Rollen und Privilegien.

Hinweis

Die in Versionen der Software vor 16.0 verwendete REVOKE-Syntax für Berechtigungen und Gruppen wird weiterhin unterstützt, jedoch nicht mehr empfohlen. Weitere Hinweise zu dieser Syntax finden Sie unter „[REVOKE-Anweisung \(Berechtigungen und Gruppen\) \(nicht mehr empfohlen\)](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Zum Vergleich finden Sie Informationen zur REVOKE-Anweisungssyntax in SQL Anywhere 12.0.1 unter <http://dcx.sybase.com/index.html#1201/en/dbreference/revoke-statement.html>.

Systemrollen entziehen

```
REVOKE ROLE system-role
FROM grantee, ...
```

```
grantee :
{ system-role | userid }
```

```
system-role :
dbo
| DIAGNOSTICS
| PUBLIC
| rs_systabgroup
| SA_DEBUG
| SYS
| SYS_REPLICATION_ADMIN_ROLE
| SYS_RUN_REPLICATION_ROLE
| SYS_SAMONITOR_ROLE
| SYS_SPATIAL_ADMIN_ROLE
```

Benutzerdefinierte Rollen entziehen

```
REVOKE [ ADMIN OPTION FOR ] ROLE user-defined-role
FROM grantee, ...
```

```
grantee :
{ system-role | userid }
```

Kompatibilitätsrollen entziehen

```
REVOKE [ ADMIN OPTION FOR ] ROLE compatibility-role-name
FROM grantee, ...
```

compatibility-role-name :
SYS_AUTH_BACKUP_ROLE
SYS_AUTH_DBA_ROLE
SYS_AUTH_PROFILE_ROLE
SYS_AUTH_READCLIENTFILE_ROLE
SYS_AUTH_READFILE_ROLE
SYS_AUTH_RESOURCE_ROLE
SYS_AUTH_SA_ROLE
SYS_AUTH_SSO_ROLE
SYS_AUTH_VALIDATE_ROLE
SYS_AUTH_WRITECLIENTFILE_ROLE
SYS_AUTH_WRITEFILE_ROLE

grantee :
{ *system-role* | *userid* }

Systemprivilegien entziehen

REVOKE [**ADMIN OPTION FOR**] *privilege*
FROM *grantee*, ...

grantee :
{ *system-role* | *userid* }

Privilegien auf Objektebene entziehen

REVOKE *object-level-privilege*[,...]
ON [*owner.*] *table-or-view*
FROM *userid*[,...]

object-level-privilege :
ALL [**PRIVILEGES**]
ALTER
DELETE
INSERT
LOAD
REFERENCES [(*column-name*[,...])]
SELECT [(*column-name*[,...])]
TRUNCATE
UPDATE [(*column-name*[,...])]

CONNECT, INTEGRATED LOGIN und KERBEROS LOGIN entziehen

REVOKE *capability* **FROM** *userid*[,...]

capability :
CONNECT
INTEGRATED LOGIN
KERBEROS LOGIN

EXECUTE für eine Prozedur entziehen

REVOKE EXECUTE
ON [*owner.*] *procedure-name*[,...]
FROM *userid*[,...]

USAGE für eine Sequenz entziehen

```
REVOKE USAGE ON SEQUENCE sequence-name[,...]
FROM userid[,...]
```

Parameter

- **ADMIN OPTION FOR-Klausel** Geben Sie die ADMIN OPTION FOR-Klausel an, um Administrationsrechte zu entziehen. Diese Klausel entzieht nur die Administrationsrechte, während die betreffende Rolle weiterhin erteilt wird.
- **REVOKE CONNECT** REVOKE CONNECT entfernt eine Benutzer-ID aus der Datenbank und löscht auch alle Objekte (Tabellen, Ansichten, Prozeduren usw.), deren Eigentümer der betreffende Benutzer ist. Von dem Benutzer erteilte Systemprivilegien bleiben gültig, während entsprechende Privilegien auf Objektebene entzogen werden.

Sie können keine REVOKE CONNECT-Anweisung für einen Benutzer ausführen, wenn der zu löschende Benutzer Eigentümer einer Tabelle ist, die von einer Ansicht referenziert wird, deren Eigentümer ein anderer Benutzer ist.

Wenn Sie mit der Dienstprogrammdatenbank verbunden sind und REVOKE CONNECT FROM DBA ausführen, werden zukünftige Verbindungen mit der Dienstprogrammdatenbank deaktiviert. Das bedeutet, dass keine weiteren Verbindungen mit der Dienstprogrammdatenbank hergestellt werden können, es sei denn, Sie verwenden eine Verbindung, die vor dem Ausführen von REVOKE CONNECT bestanden hat, oder Sie starten den Datenbankserver neu.

- **REVOKE USAGE ON SEQUENCE** Geben Sie diese Syntax an, um das Privileg zum Auswerten des aktuellen oder nächsten Werts in einer Sequenz zu entfernen.

Bemerkungen

Wenn zu entziehende Rollen oder Privilegien nicht dem *grantee* erteilt wurden, hat die Anweisung keine Auswirkungen und es wird kein Fehler zurückgegeben.

REVOKE ROLE schlägt mit einem Fehler fehl, wenn durch das Ausführen der Anweisung die Anzahl von Administratoren für zu entziehende Rollen oder Privilegien unter den erforderlichen Mindestwert fallen würde, der durch die *min_role_admins*-Datenbankoption festgelegt ist.

Wenn Sie Privilegien auf Objektebene einer Rolle entziehen, erben Berechtigungsempfänger dieser Rolle die entsprechenden Änderungen.

Wenn Sie ein Privileg auf Objektebene für einen Benutzer entziehen, der auch Administrationsrechte für dieses Privileg hatte, wird das Privileg auch allen entzogen, denen es dieser Benutzer erteilt hat, sowie allen, denen es die Berechtigungsempfänger erteilt haben, usw.

Wenn Sie einem Benutzer verbindungsbezogene Privilegien entziehen möchten, darf der Benutzer nicht mit der Datenbank verbunden sein.

Privilegien

Sie benötigen die Administrationsrechte für die Systemprivilegien oder Rollen, die Sie entziehen möchten.

Wenn Sie Privilegien auf Objektebene entziehen möchten, müssen Sie über eine der folgenden Berechtigungen verfügen:

- Eigentum am Objekt
- Administrationsrechte für das Privileg auf Objektebene für dieses Objekt
- **MANAGE ANY OBJECT PRIVILEGE**-Systemprivileg

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Rollen und Privilegien für einen Benutzer oder eine Rolle anzeigen (Sybase Central)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Systemprivilegien“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Objektprivilegien“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Rollen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Berechtigungen auf Datenbankebene werden zu Kompatibilitätsrollen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „min_role_admins-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „GRANT-Anweisung“ auf Seite 881

Standards und Kompatibilität

- **SQL/2008** **REVOKE** *capability* ist eine Erweiterung des Herstellers. **REVOKE** *object-level-privilege* und **REVOKE** **EXECUTE** sind Kernfunktionen des SQL/2008-Standards. Bei **REVOKE** **ALL** (alle Privilegien auf Objektebene entziehen) ist das **PRIVILEGES**-Schlüsselwort in SQL Anywhere optional, während es im SQL/2008-Standard obligatorisch ist.

REVOKE **USAGE** **ON** **SEQUENCE** ist Teil der optionalen SQL/2008-Sprachenfunktion T176 (Sequenzgenerator-Unterstützung).

Beispiel

In diesem Beispiel wird verhindert, dass der Benutzer Dave die Tabelle "Employees" aktualisiert:

```
REVOKE UPDATE ON GROUPO.Employees FROM Dave;
```

In diesem Beispiel wird die **SYS_AUTH_VALIDATE_ROLE**-Kompatibilitätsrolle dem fiktiven Benutzer Jim entzogen.

```
REVOKE ROLE SYS_AUTH_VALIDATE_ROLE FROM Jim;
```

In diesem Beispiel wird die **DIAGNOSTICS**-Systemrolle einem fiktiven Benutzer namens Administrator entzogen.

```
REVOKE ROLE DIAGNOSTICS FROM Administrator;
```

In diesem Beispiel wird verhindert, dass eine fiktive benutzererweiterte Rolle namens Finance die ShowCustomers-Prozedur ausführt.

```
REVOKE EXECUTE ON ShowCustomers FROM Finance;
```

In diesem Beispiel wird die Benutzer-ID "FranW" aus der Datenbank gelöscht. Diese Syntax wird nicht mehr empfohlen. Sie sollten stattdessen die DROP USER-Anweisung verwenden.

```
REVOKE CONNECT FROM FranW;
```

Dieses Beispiel entzieht das Datenbank-Login-Privileg einem fiktiven Kerberos-Benutzer namens pchin.

```
REVOKE KERBEROS LOGIN  
FROM "pchin@MYREALM.COM";
```

REVOKE ROLE SYS_REPLICATION_ADMIN_ROLE- Anweisung [MobiLink] [SQL Remote]

Entzieht die SYS_REPLICATION_ADMIN_ROLE-Systemrolle einem Benutzer (Berechtigungsempfänger). Wenn diese Rolle dem Benutzer entzogen wird, hat er nicht mehr die Möglichkeit, Administrationsaufgaben im Zusammenhang mit der Replikation auszuführen, z.B. Erteilen von Replikationsrollen, Verwalten von Publikationen, Subskriptionen, Synchronisationsbenutzern und Profilen, Verwalten von Nachrichtentypen, Festlegen von Optionen für die Replikation.

Syntax

```
REVOKE ROLE SYS_REPLICATION_ADMIN_ROLE  
FROM grantee[, ...]
```

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Rollen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „GRANT ROLE SYS_REPLICATION_ADMIN_ROLE-Anweisung [MobiLink] [SQL Remote]“ auf Seite 896
- „REVOKE PUBLISH-Anweisung [SQL Remote]“ auf Seite 1007
- „REVOKE REMOTE-Anweisung [SQL Remote]“ auf Seite 1008
- „REVOKE CONSOLIDATE-Anweisung [SQL Remote]“ auf Seite 1006
- „Initiieren einer Synchronisation“ [[MobiLink - Clientadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Die folgende Anweisung entzieht SYS_REPLICATION_ADMIN_ROLE dem Benutzer Sam_Singer.

```
REVOKE ROLE SYS_REPLICATION_ADMIN_ROLE FROM Sam_Singer;
```

REVOKE ROLE SYS_RUN_REPLICATION_ROLE- Anweisung [MobiLink] [SQL Remote]

Entzieht die SYS_RUN_REPLICATION_ROLE-Systemrolle einem Benutzer (Berechtigungsempfänger).
Wenn diese Rolle entzogen wird, hat der betreffende Benutzer folgende Möglichkeiten nicht mehr:

- Das Dienstprogramm dbremote für die Replikation ausführen
- Das Dienstprogramm dbmlsync für die Synchronisation ausführen

Syntax

```
REVOKE ROLE SYS_RUN_REPLICATION_ROLE  
FROM grantee[, ...]
```

Bemerkungen

In MobiLink umfasst SYS_RUN_REPLICATION_ROLE Privilegien, die vom SQL Anywhere-Synchronisationsclient (dbmlsync) benötigt werden.

In SQL Remote gewährt SYS_RUN_REPLICATION_ROLE dem SQL Remote-Nachrichtenagenten vollen Zugriff auf die Datenbank, um in den Nachrichten enthaltene Änderungen vorzunehmen.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Rollen“ [*SQL Anywhere Server - Datenbankadministration*]
- „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ auf Seite 898
- „REVOKE PUBLISH-Anweisung [SQL Remote]“ auf Seite 1007
- „REVOKE REMOTE-Anweisung [SQL Remote]“ auf Seite 1008
- „REVOKE CONSOLIDATE-Anweisung [SQL Remote]“ auf Seite 1006
- „Initiieren einer Synchronisation“ [*MobiLink - Clientadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Die folgende Anweisung entzieht SYS_RUN_REPLICATION_ROLE dem Benutzer Sam_Singer.

```
REVOKE ROLE SYS_RUN_REPLICATION_ROLE FROM Sam_Singer;
```

ROLLBACK-Anweisung

Beendet eine Transaktion und macht die Änderungen seit dem letzten COMMIT- oder ROLLBACK-Vorgang rückgängig

Syntax

ROLLBACK [WORK]

Bemerkungen

Eine Transaktion ist die logische Arbeitseinheit, die über eine Datenbankverbindung zu einer Datenbank zwischen COMMIT- oder ROLLBACK-Anweisungen ausgeführt wird. Die ROLLBACK-Anweisung beendet die aktuelle Transaktion und setzt alle Änderungen zurück, die seit dem vorhergehenden COMMIT oder ROLLBACK gemacht wurden.

Privilegien

Keine.

Nebenwirkungen

Schließt alle Cursors, die nicht mit WITH HOLD geöffnet wurden

Siehe auch

- „COMMIT-Anweisung“ auf Seite 575
- „Tastenkürzel für Interactive SQL“ [*SQL Anywhere Server - Datenbankadministration*]
- „ROLLBACK TO SAVEPOINT-Anweisung“ auf Seite 1015
- „Interactive SQL-Optionen“ [*SQL Anywhere Server - Datenbankadministration*]
- „Rückgängigmachen von Änderungen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Kernfunktion.

ROLLBACK TO SAVEPOINT-Anweisung

Hebt alle Änderungen auf, die seit einem SAVEPOINT vorgenommen wurden

Syntax

ROLLBACK TO SAVEPOINT [*savepoint-name*]

Bemerkungen

Die ROLLBACK TO SAVEPOINT-Anweisung macht alle Änderungen rückgängig, die vorgenommen wurden, seit der SAVEPOINT gesetzt wurde. Änderungen, die vor dem SAVEPOINT vorgenommen wurden, werden nicht rückgängig gemacht und bleiben weiterhin gültig.

Der *savepoint-name* ist ein Bezeichner, der für eine SAVEPOINT-Anweisung innerhalb der aktuellen Transaktion angegeben wurde. Wenn *savepoint-name* weggelassen wird, kommt der neueste Savepoint zur Anwendung. Alle Savepoints seit dem benannten Savepoint werden automatisch freigegeben.

Innerhalb der aktuellen Transaktion muss es einen entsprechenden SAVEPOINT geben.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „BEGIN TRANSACTION-Anweisung [T-SQL]“ auf Seite 561
- „COMMIT-Anweisung“ auf Seite 575
- „RELEASE SAVEPOINT-Anweisung“ auf Seite 994
- „ROLLBACK-Anweisung“ auf Seite 1015
- „SAVEPOINT-Anweisung“ auf Seite 1019
- „Savepoints innerhalb von Transaktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** ROLLBACK TO SAVEPOINT ist Teil der optionalen SQL-Sprachenfunktion T271 des SQL/2008-Standards.

ROLLBACK TRANSACTION-Anweisung [T-SQL]

Hebt alle Änderungen auf, die seit einer SAVE TRANSACTION-Anweisung vorgenommen wurden.

Syntax

ROLLBACK TRANSACTION [*savepoint-name*]

Bemerkungen

Die ROLLBACK TRANSACTION-Anweisung macht alle Änderungen rückgängig, die seit einem mit SAVE TRANSACTION gesetzten Savepoint durchgeführt wurden. Änderungen, die vor der SAVE TRANSACTION-Anweisung vorgenommen wurden, werden nicht rückgängig gemacht und bleiben weiterhin gültig.

Der *savepoint-name* ist ein Bezeichner, der für eine SAVE TRANSACTION-Anweisung innerhalb der aktuellen Transaktion angegeben wurde. Wenn kein *savepoint-name* angegeben ist, werden alle ausstehenden Änderungen zurückgesetzt. Alle Savepoints seit dem benannten Savepoint werden automatisch freigegeben.

Innerhalb der aktuellen Transaktion muss es eine entsprechende SAVE TRANSACTION-Anweisung geben.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „ROLLBACK TO SAVEPOINT-Anweisung“ auf Seite 1015
- „BEGIN TRANSACTION-Anweisung [T-SQL]“ auf Seite 561
- „COMMIT-Anweisung“ auf Seite 575
- „SAVE TRANSACTION-Anweisung [T-SQL]“ auf Seite 1018

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel zeigt fünf Zeilen mit den Werten "10", "20", usw. an. Die Auswirkung des Vorgangs DELETE wird durch die ROLLBACK TRANSACTION-Anweisung rückgängig gemacht, jedoch nicht die Auswirkung der vorherigen Vorgänge INSERT oder UPDATE.

```
BEGIN
    SELECT row_num INTO #tmp
    FROM sa_rowgenerator( 1, 5 )
    UPDATE #tmp SET row_num=row_num*10
    SAVE TRANSACTION before_delete
    DELETE FROM #tmp WHERE row_num >= 3
    ROLLBACK TRANSACTION before_delete
    SELECT * FROM #tmp
END
```

ROLLBACK TRIGGER-Anweisung

Macht alle Änderungen in einem Trigger rückgängig.

Syntax

ROLLBACK TRIGGER [**WITH** *raiserror-statement*]

Bemerkungen

Die ROLLBACK TRIGGER-Anweisung setzt die in einem Trigger durchgeführte Aktion zurück, einschließlich der Datenänderung, die den Trigger ausgelöst hat.

Wahlweise kann eine RAISERROR-Anweisung ausgeführt werden. Wenn eine RAISERROR-Anweisung ausgeführt wird, wird eine Fehlermeldung an die Anwendung zurückgegeben. Wenn keine RAISERROR-Anweisung ausgeführt wird, erfolgt keine Fehlermeldung.

Wenn eine ROLLBACK TRIGGER-Anweisung innerhalb eines verschachtelten Triggers und ohne RAISERROR-Anweisung verwendet wird, werden nur der innerste Trigger und die Anweisung, die ihn ausgelöst hat, rückgängig gemacht.

Privilegien

Keine.

Nebenwirkungen

Keine

Siehe auch

- „CREATE TRIGGER-Anweisung“ auf Seite 762
- „ROLLBACK-Anweisung“ auf Seite 1015
- „ROLLBACK TO SAVEPOINT-Anweisung“ auf Seite 1015
- „RAISERROR-Anweisung“ auf Seite 982

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.
- **Transact-SQL** ROLLBACK TRIGGER wird in gespeicherten Prozeduren sowohl in Watcom-SQL als auch in Transact-SQL unterstützt. ROLLBACK TRIGGER wird von Adaptive Server Enterprise unterstützt.

SAVE TRANSACTION-Anweisung [T-SQL]

Richtet einen Savepoint innerhalb der aktuellen Transaktion ein.

Syntax

SAVE TRANSACTION *savepoint-name*

Bemerkungen

Richtet einen Savepoint innerhalb einer aktuellen Transaktion ein. Der *savepoint-name* ist ein Bezeichner, der in einer ROLLBACK TRANSACTION-Anweisung verwendet werden kann. Alle Savepoints werden automatisch freigegeben, wenn die Transaktion endet.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „SAVEPOINT-Anweisung“ auf Seite 1019
- „Savepoints innerhalb von Transaktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „BEGIN TRANSACTION-Anweisung [T-SQL]“ auf Seite 561
- „COMMIT-Anweisung“ auf Seite 575
- „ROLLBACK TRANSACTION-Anweisung [T-SQL]“ auf Seite 1016

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel zeigt fünf Zeilen mit den Werten "10", "20", usw. an. Die Auswirkung des Vorgangs DELETE wird durch die ROLLBACK TRANSACTION-Anweisung rückgängig gemacht, jedoch nicht die Auswirkung der vorherigen Vorgänge INSERT oder UPDATE.

```
BEGIN
  SELECT row_num INTO #tmp
  FROM sa_rowgenerator( 1, 5 )
  UPDATE #tmp SET row_num=row_num*10
  SAVE TRANSACTION before_delete
  DELETE FROM #tmp WHERE row_num >= 3
  ROLLBACK TRANSACTION before_delete
  SELECT * FROM #tmp
END
```

SAVEPOINT-Anweisung

Richtet einen Savepoint innerhalb der aktuellen Transaktion ein.

Syntax

SAVEPOINT [*savepoint-name*]

Bemerkungen

Richtet einen Savepoint innerhalb einer aktuellen Transaktion ein. Der *savepoint-name* ist ein Bezeichner, der in einer RELEASE SAVEPOINT- oder ROLLBACK TO SAVEPOINT-Anweisung verwendet werden kann. Alle Savepoints werden automatisch freigegeben, wenn die Transaktion endet.

Savepoints, die während der Ausführung eines Triggers oder einer unteilbaren, zusammengesetzten Anweisung eingerichtet werden, werden automatisch freigegeben, wenn der unteilbare Vorgang endet.

Sie können Daten in einer Proxytabelle nicht von einem Savepoint aus ändern.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „RELEASE SAVEPOINT-Anweisung“ auf Seite 994
- „ROLLBACK TO SAVEPOINT-Anweisung“ auf Seite 1015
- „SAVE TRANSACTION-Anweisung [T-SQL]“ auf Seite 1018
- „Savepoints innerhalb von Transaktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Die SAVEPOINT-Anweisung ist Teil der optionalen SQL/2008-Sprachenfunktion T271 (Savepoints).

- **Transact-SQL** In Transact-SQL erfolgt die Erstellung eines Savepoint mithilfe der SAVE TRANSACTION-Anweisung.

SELECT-Anweisung

Ruft Informationen aus der Datenbank ab

Syntax

```
[ WITH temporary-views ]
SELECT [ ALL | DISTINCT ] [ row-limitation-option-1 ] select-list
[ INTO { hostvar-list | variable-list | table-name } ]
[ INTO LOCAL TEMPORARY TABLE { table-name } ]
[ FROM from-expression ]
[ WHERE search-condition ]
[ GROUP BY group-by-expression ]
[ HAVING search-condition ]
[ WINDOW window-expression ]
[ ORDER BY { expression | integer } [ ASC | DESC ], ... ]
[ FOR READ ONLY | for-update-clause ]
[ FOR XML xml-mode ]
[ FOR JSON json-mode ]
[ row-limitation-option-2 ]
[ OPTION( query-hint, ... ) ]
```

temporary-views :
regular-view, ...
| **RECURSIVE** { *regular-view* | *recursive-view* }, ...

regular-view :
view-name [(*column-name*, ...)]
AS (*subquery*)

recursive-view :
view-name (*column-name*, ...)
AS (*initial-subquery* **UNION ALL** *recursive-subquery*)

row-limitation-option-1 :
FIRST
| **TOP** { **ALL** | *limit-expression* } [**START AT** *startat-expression*]

row-limitation-option-2 :
LIMIT { [*offset-expression*,] *limit-expression* | *limit-expression* **OFFSET** *offset-expression* }

limit-expression : *simple-expression*

startat-expression : *simple-expression*

offset-expression : *simple-expression*

simple-expression :
integer
| *variable*

| (*simple-expression*)
 | (*simple-expression* { + | - | * } *simple-expression*)

select-list :
expression [[**AS**] *alias-name*], ...
 | *
 | *window-function* **OVER** { *window-name* | *window-spec* }
 | [[**AS**] *alias-name*]
 | *sequence-expression*

sequence-expression
sequence-name [**CURRVAL** | **NEXTVAL**]
FROM *table-name*

sequence-expression : Siehe „Ausdrücke“ auf Seite 22.

from_expression: Siehe „FROM-Klausel“ auf Seite 863.

group-by-expression : Siehe „GROUP BY-Klausel“ auf Seite 903.

search-condition : Siehe „Suchbedingungen“ auf Seite 42.

window-name : *identifier*

window-expression : Siehe „WINDOW-Klausel“ auf Seite 1124.

window-spec : Siehe „WINDOW-Klausel“ auf Seite 1124.

window-function :
RANK()
 | **DENSE_RANK()**
 | **PERCENT_RANK()**
 | **CUME_DIST()**
 | **ROW_NUMBER()**
 | *aggregate-function*

for-update-clause
FOR UPDATE
 | **FOR UPDATE** *cursor-concurrency*
 | **FOR UPDATE OF** [(*column-name*, ...)]

cursor-concurrency :
BY { **VALUES** | **TIMESTAMP** | **LOCK** }

xml-mode :
RAW [, **ELEMENTS**]
 | **AUTO** [, **ELEMENTS**]
 | **EXPLICIT**

query-hint :
MATERIALIZED VIEW OPTIMIZATION *option-value*
 | **FORCE OPTIMIZATION**
 | **FORCE NO OPTIMIZATION**
 | *option-name*=*option-value*

option-name : *identifier*

option-value :
hostvar (Bezeichner zulässig)
| *string*
| *identifizier*
| *number*

Parameter

WITH- oder WITH RECURSIVE-Klausel Definition von einem oder mehreren gemeinsamen Tabellenausdrücken, auch als temporäre Ansichten bekannt, die im Rest der Anweisung verwendet werden können. Diese Ausdrücke können nicht-rekursiv oder selbstrekursiv sein. Rekursive gemeinsame Tabellenausdrücke können allein oder mit nicht-rekursiven Tabellenausdrücken gemischt erscheinen, wenn das Schlüsselwort RECURSIVE angegeben wird. Gegenseitig rekursive gemeinsame Tabellenausdrücke werden nicht unterstützt.

Diese Klausel ist nur gestattet, wenn der SELECT-Abfrageblock an einer der folgenden Positionen erscheint:

- Innerhalb des SELECT-Abfrageblocks der obersten Ebene einschließlich des obersten SELECT-Abfrageblocks einer Ansichtsdefinition
- In einer SELECT-Anweisung auf der obersten Ebene in einem INSERT-Abfrageblock
- Innerhalb eines verschachtelten SELECT-Abfrageblocks, der eine abgeleitete Tabelle in einem beliebigen SQL-Anweisungstyp definiert

Rekursive Ausdrücke bestehen aus einer ersten Unterabfrage und einer rekursiven Unterabfrage. Die erste Abfrage definiert implizit das Schema der Ansicht. Die rekursive Unterabfrage muss eine Referenz auf die Ansicht in der FROM-Klausel enthalten. Während jeder Wiederholung bezieht sich diese Referenz nur auf die in der vorherigen Iteration in die Ansicht eingefügten Zeilen. Die Referenz darf nicht auf der Nullwert-liefernden Seite eines Outer-Join erscheinen. Ein rekursiver gemeinsamer Tabellenausdruck darf keine Aggregatfunktionen verwenden und darf keine GROUP BY-, ORDER BY- oder DISTINCT-Klausel enthalten.

Die WITH-Klausel wird bei entfernten Tabellen nicht unterstützt. Die WITH-Klausel kann auch in INTERSECT, UNION und EXCEPT-Abfrageblöcken verwendet werden.

ALL- oder DISTINCT-Klausel ALL (der Standardwert) gibt alle Zeilen zurück, welche die Klauseln der SELECT-Anweisung erfüllen. Wenn DISTINCT angegeben ist, werden mehrfach vorhandene Zeilen entfernt. Da die Ausführung vieler Anweisungen bedeutend länger dauert, wenn DISTINCT angegeben ist, sollten Sie DISTINCT nur dann verwenden, wenn es unbedingt notwendig ist.

row-limitation-Klauseln Die Zeilenbeschränkungsklauseln ermöglichen es Ihnen, eine Teilmenge der Zeilen zurückzugeben, die von der WHERE-Klausel erfasst werden. Es kann nur eine Zeilenbeschränkungsklausel (*row-limitation*) gleichzeitig angegeben werden. Wenn diese Klauseln angegeben werden, ist auch eine ORDER BY-Klausel erforderlich, um die Zeilen sinnvoll zu ordnen.

- **row-limitation-option-1** Die Argumente TOP und START AT können einfache arithmetische Ausdrücke über Hostvariablen, Ganzzahlkonstanten oder Ganzzahlvariablen sein. Das TOP-Argument muss jedoch mit einem Wert größer oder gleich 0 ausgewertet werden. Das START AT-Argument mit einem Wert größer als 0 ausgewertet werden.

Wenn *startat-expression* nicht angegeben wird, ist der Standardwert 1. Wenn das TOP-Argument ALL lautet, werden alle Zeilen zurückgegeben, die bei *startat-expression* beginnen. Die TOP limit-expression START AT *startat-expression*-Klausel ist äquivalent mit LIMIT (*startat-expression* -1), limit-expression oder LIMIT limit-expression OFFSET (*startat-expression* -1).

- **row-limitation-option-2** Die Argumente LIMIT und OFFSET können einfache arithmetische Ausdrücke über Hostvariablen, Ganzzahlkonstanten oder Ganzzahlvariablen sein. Das LIMIT-Argument muss jedoch mit einem Wert größer oder gleich 0 ausgewertet werden. Das OFFSET-Argument muss jedoch mit einem Wert größer oder gleich 0 ausgewertet werden. Wenn *offset-expression* nicht angegeben wird, ist der Standardwert 0.

Die Zeilenbeschränkungsklausel LIMIT *offset-expression*, limit-expression entspricht LIMIT limit-expression OFFSET *offset-expression*. Beide Konstrukte entsprechen TOP limit-expression START AT (*offset-expression* + 1).

Das LIMIT-Schlüsselwort ist standardmäßig deaktiviert. Verwenden Sie die reserved_keywords-Option, um das LIMIT Schlüsselwort zu aktivieren.

select-list-Klausel Die *select-list* ist eine Liste von Ausdrücken, die durch Kommata getrennt werden und angeben, was aus der Datenbank abgerufen wird. Ein Stern (*) bedeutet, dass sämtliche Spalten aller Tabellen in der FROM-Klausel ausgewählt werden.

Aggregatfunktionen sind in der *select-list* zulässig. Unterabfragen sind in der *select-list* ebenfalls zulässig. Jede Unterabfrage muss in Klammern gesetzt werden.

Aliasnamen können in der Abfrage verwendet werden, um den Ausdruck mit einem Alias darzustellen.

Aliasnamen werden auch von Interactive SQL ganz oben in jeder Spalte angezeigt, die von der SELECT-Anweisung ausgegeben wurde. Wenn der optionale Aliasname nach einem Ausdruck nicht angegeben wird, zeigt Interactive SQL den Ausdruck an.

Hinweis

Die folgenden Zeichen sind in Aliasnamen nicht zulässig:

- Anführungszeichen
- Steuerzeichen (alle Zeichen unter 0x20)
- Backslashes
- Eckige Klammern
- Invertierte Hochkommata

INTO-Klausel Nachstehend werden drei Verwendungsmöglichkeiten für die INTO-Klausel angeführt:

- **INTO hostvar-list-Klausel** Diese Klausel wird nur in Embedded SQL verwendet. Sie gibt an, wo die Ergebnisse der SELECT-Anweisung abgelegt werden. Für jedes Element in der *select-list* muss

ein Hostvariablenelement vorhanden sein. Die Elemente der *select-list* werden in die Hostvariablen einsortiert. Mit jeder Hostvariablen ist außerdem eine Indikator-Hostvariable zulässig, sodass das Programm erfährt, ob die Elemente in der *select-list* NULL waren.

Wenn die Abfrage dazu führt, dass keine Zeilen ausgewählt werden, können die Variablen nicht aktualisiert werden und die Warnung "Zeile nicht gefunden" wird angezeigt.

- **INTO *variable-list*-Klausel** Diese Klausel wird nur in Prozeduren und Triggern verwendet. Sie gibt an, wo die Ergebnisse der SELECT-Anweisung abgelegt werden. Für jedes Element in der *select-list* muss eine Variable vorhanden sein. Die Elemente in der *select-list* werden in die Variablen einsortiert.
- **INTO *table-name*-Klausel** Diese Klausel wird zur Erstellung von Tabellen und zum Füllen der Tabellen mit Daten verwendet.

Um permanente Tabellen zu erstellen, muss die Abfrage eine der folgenden Bedingungen erfüllen:

- Die *select-list* enthält mehr als ein Element und das Ziel INTO ist ein einziger *table-name*-Bezeichner.
- Die *select-list* enthält ein * und das Ziel INTO ist als *Eigentümer.Tabelle* angegeben.

Um eine permanente Tabelle mit einer Spalte zu erstellen, muss der Tabellename als *Eigentümer.Tabelle* angegeben sein.

Diese Anweisung verursacht vor der Ausführung ein COMMIT. Dies ist eine Nebenwirkung bei der Erstellung der Tabelle. Für die neue Tabelle werden keine Privilegien erteilt: Die Anweisung ist eine Kurzform für CREATE TABLE, gefolgt von INSERT...SELECT.

Tabellen, die mit dieser Klausel erstellt werden, haben keinen definierten Primärschlüssel. Sie können mithilfe von ALTER TABLE einen Primärschlüssel hinzufügen. Ein Primärschlüssel sollte hinzugefügt werden, bevor Aktualisierungen oder Löschungen auf die Tabelle angewendet werden. Andernfalls werden bei diesen Vorgängen alle Spaltenwerte für die betroffenen Zeilen im Transaktionslog protokolliert.

INTO LOCAL TEMPORARY TABLE Diese Klausel wird benutzt, um eine lokale temporäre Tabelle zu erstellen und mit den Ergebnissen der Abfrage zu füllen. Wenn Sie diese Klausel verwenden, muss der temporäre Tabellename nicht mit # beginnen.

FROM-Klausel Zeilen werden aus den in *table-expression* angegebenen Tabellen und Ansichten abgerufen. Eine SELECT-Anweisung ohne FROM-Klausel kann verwendet werden, um die Werte von Ausdrücken anzuzeigen, die nicht von Tabellen abgeleitet werden. Diese zwei Anweisungen z.B. sind äquivalent und zeigen den Wert der globalen Variablen @@version an.

```
SELECT @@version;  
SELECT @@version FROM DUMMY;
```

WHERE-Klausel Diese Klausel gibt an, welche Zeilen aus den benannten Tabellen in der FROM-Klausel ausgewählt werden. Sie kann benutzt werden, um Joins zwischen mehreren Tabellen einzurichten, als Alternative zur ON-Phrase (Teil der FROM-Klausel).

GROUP BY-Klausel Sie können nach Spalten, Aliasnamen oder Funktionen gruppieren. Das Ergebnis der Abfrage enthält eine Zeile für jede unterschiedliche Menge von Werten in den benannten Spalten, Aliasen oder Funktionen. Wie auch die DISTINCT und die Mengenoperationen UNION, INTERSECT und EXCEPT behandelt die GROUP BY-Klausel NULL auf dieselbe Weise wie die anderen Werte in den jeweiligen Domänen. Das bedeutet, dass mehrere NULL-Werte in einem Gruppierungsattribut eine einzige Gruppe bilden. Aggregatfunktionen können dann für diese Gruppen angewendet werden, um sinnvolle Ergebnisse zu erhalten.

Wenn GROUP BY verwendet wird, können die *select-list*, HAVING-Klausel und ORDER BY-Klausel nur die Bezeichner referenzieren, die in der GROUP BY-Klausel aufgeführt werden. Die Ausnahme ist, dass die *select-list* und die HAVING-Klausel Aggregatfunktionen enthalten können.

HAVING-Klausel Diese Klausel wählt die Zeile auf der Grundlage der Gruppenwerte und nicht der einzelnen Zeilenwerte aus. Die HAVING-Klausel kann nur verwendet werden, wenn entweder die Anweisung eine GROUP BY-Klausel umfasst oder die *select-list* nur aus Aggregatfunktionen besteht. Sämtliche Spaltennamen, die in der HAVING-Klausel referenziert werden, müssen entweder in der GROUP BY-Klausel enthalten sein oder in der HAVING-Klausel als Parameter für eine Aggregatfunktion verwendet werden.

WINDOW-Klausel Diese Klausel bestimmt das ganze Fenster oder einen Fensterabschnitt für die Verwendung mit Fensterfunktionen wie AVG und RANK.

ORDER BY-Klausel Diese Klausel sortiert die Ergebnisse einer Abfrage. Jedes Element in der ORDER BY-Liste kann als ASC für aufsteigende Sortierfolge (der Standardwert) oder DESC für absteigende Sortierfolge benannt werden. Wenn der Ausdruck eine Ganzzahl n ist, dann werden die Abfrageergebnisse nach dem n -ten Element in der *select-list* sortiert.

Die einzige Möglichkeit, sicherzustellen, dass Zeilen in einer bestimmten Reihenfolge zurückgegeben werden, ist die Verwendung von ORDER BY. Ohne die Klausel ORDER BY gibt SQL Anywhere die Zeilen in der jeweils effizientesten Reihenfolge zurück. Das Erscheinungsbild der Ergebnismengen hängt davon ab, wann Sie zuletzt auf die Zeile zugegriffen haben, und kann durch andere Faktoren beeinflusst werden.

In Embedded SQL wird die SELECT-Anweisung dafür verwendet, Ergebnisse aus der Datenbank abzurufen und die Werte über die INTO-Klausel in Hostvariablen abzulegen. Die SELECT-Anweisung darf nur eine Zeile zurückgeben. Für Abfragen mehrerer Zeilen müssen Sie Cursor verwenden.

FOR UPDATE- oder FOR READ ONLY-Klausel Diese Klauseln geben an, ob Aktualisierungen über einen von der Abfrage aus geöffneten Cursor zulässig sind und, falls dem so ist, welche Parallelitätssemantik verwendet werden kann. Diese Klausel kann nicht mit der FOR XML-Klausel verwendet werden.

Wenn Sie in der SELECT-Anweisung keine FOR-Klausel verwenden, hängt die Aktualisierbarkeit eines Cursors von der Deklaration des Cursors ab sowie von der Angabe der Cursorparallelität durch die API. In ODBC, JDBC, OLE DB, ADO.NET und Embedded SQL sind die Anweisungen explizit aktualisierbar und es wird ein schreibgeschützter Cursor verwendet, es sei denn, die Anwendung setzt dies außer Kraft. In Open Client und innerhalb von gespeicherten Prozeduren muss die Aktualisierbarkeit eines Cursor nicht angegeben werden, und der Standardwert ist FOR UPDATE.

Bei Open Client und gespeicherten Prozeduren hängt die Aktualisierbarkeit von Cursor und Anweisungen von der Einstellung der Datenbankoption `ansi_update_constraints` und den spezifischen Merkmalen der

Anweisung ab, z.B. davon, ob die Anweisung ORDER BY, DISTINCT, GROUP BY, HAVING, UNION, Aggregatfunktionen, Joins oder nicht aktualisierbare Ansichten enthält. Cursor gelten standardmäßig als FOR UPDATE bei Einzeltabellenabfragen ohne ORDER BY-Klausel bzw. wenn die Option ansi_update_constraints auf OFF gesetzt ist. Wenn die Option ansi_update_constraints auf "Cursors" oder "Strict" gesetzt ist, gelten Cursor über einer Abfrage, die eine ORDER BY-Klausel enthält, standardmäßig als READ ONLY. Sie können jedoch Cursor explizit als aktualisierbar markieren, indem Sie die FOR UPDATE-Klausel verwenden. Da es kostenträchtig ist, Aktualisierungen über Cursor mit einer ORDER BY-Klausel oder einem Join zuzulassen, sind Cursor über eine Abfrage, die einen Join von zwei oder mehr Tabellen enthält, READ ONLY und können nicht aktualisierbar gemacht werden, es sei denn, die Datenbankoption ansi_update_constraints ist auf OFF gesetzt.

Ein als FOR READ ONLY deklarierter Cursor kann nicht in (positionsbasierten) UPDATE- und DELETE- sowie in PUT-Anweisungen verwendet werden. FOR READ ONLY ist die Standardeinstellung für Embedded SQL.

Die FOR UPDATE-Klausel macht einen Cursor explizit aktualisierbar. Die Verwendung von FOR UPDATE hat alleine keine Auswirkungen auf die Parallelitätssteuerung für die Zeilen in der Ergebnismenge der Anweisung. Für diesen Zweck müssen Sie entweder FOR UPDATE BY LOCK oder FOR UPDATE BY [VALUES | TIMESTAMP] angeben.

- **FOR UPDATE BY LOCK-Klausel** Der Datenbankserver erwirbt Absichtszeilensperren für abgerufene Zeilen der Ergebnismenge. Dies sind langfristige Sperren, die gehalten werden, bis die Transaktion festgeschrieben oder zurückgesetzt wird. Absichtszeilensperren werden nicht gesetzt, wenn die SELECT-Anweisung eine INTO-Klausel verwendet, da kein positionsbasiertes Update ausgeführt werden kann.
- **FOR UPDATE BY { VALUES | TIMESTAMP }** Wenn Sie FOR UPDATE BY TIMESTAMP oder FOR UPDATE BY VALUES angeben, verwendet der Datenbankserver eine optimistische Parallelität, indem ein durch Keyset gesteuerter (wertsensitiver) Cursor eingesetzt wird. In diesem Fall können verlorene Aktualisierungen auftreten, wenn die Anwendung eine Zeile außerhalb des Cursors ändert (mit einer separaten Anweisung) oder wenn die Anwendung die vom Server generierten Warnungen und/oder Fehler nicht beachtet, die darauf hinweist, dass die Zeile von einer anderen Verbindung geändert wurde.

Um zu gewährleisten, dass eine Anweisung eine Absichtssperre erhält, führen Sie einen der folgenden Schritte durch:

- Geben Sie FOR UPDATE BY LOCK in der Abfrage an
- Geben Sie HOLDLOCK, WITH (HOLDLOCK), WITH (UPDLOCK) oder WITH (XLOCK) in der FROM-Klausel der Abfrage an
- Öffnen Sie den Cursor mit API-Aufrufen, geben Sie CONCUR_LOCK an
- Die Zeilen mit Attributen abrufen, die einen Abruf zum Aktualisieren angeben

Die FOR UPDATE OF-Klausel benennt explizit die Spalten, die mit einer (positionsbasierten) UPDATE- oder DELETE oder einer PUT-Anweisung geändert werden können. Sie können diese Klausel nicht in Kombination mit anderen FOR UPDATE-, FOR READ ONLY-, oder FOR XML-Klauseln verwenden.

- **FOR UPDATE OF *column-list*-Klausel** Wenn Sie die FOR UPDATE OF-Klausel angeben, beschränkt der Datenbankserver die Spalten, die mit einer positionsbasierten UPDATE- oder DELETE-Anweisung geändert werden können, auf die in dieser Klausel explizit benannten Spalten. Der Versuch, ein anderes Spaltenergebnis zu ändern, führt zu einem Fehler, der angibt, dass die Spalte nicht gefunden werden kann. Es wird nicht geprüft, ob eine in der Liste referenzierte Spalte tatsächlich vorhanden ist oder ob die Tabelle der Spalte eine aktualisierbare Tabelle ist.

FOR XML-Klausel Diese Klausel legt fest, dass die Ergebnismenge als XML-Dokument zurückgegeben werden muss. Das XML-Format hängt von dem angegebenen Modus ab. Diese Klausel kann nicht mit der FOR UPDATE oder FOR READ ONLY-Klausel verwendet werden. Mit FOR XML deklarierte Cursor sind implizit READ ONLY.

Wenn Sie den RAW-Modus festlegen, wird jede Zeile in der Ergebnismenge als XML <row>-Element dargestellt, und jede Spalte als Attribut des <row>-Elements.

AUTO-Modus gibt die Abfrageergebnisse als verschachtelte XML-Elemente zurück. Jede in der *select-list* referenzierte Tabelle wird als Element in XML angegeben. Die Reihenfolge der Verschachtelung der Elemente basiert auf der Reihenfolge, in der Tabellen in der *select-list* referenziert werden.

Der EXPLICIT-Modus ermöglicht es Ihnen, die Form des generierten XML-Dokuments zu steuern. Mit dem EXPLICIT-Modus haben Sie mehr Flexibilität bei der Benennung der Elemente und Angabe der Verschachtelungsstruktur als bei RAW oder AUTO.

FOR JSON-Klausel Diese Klausel gibt an, dass die Ergebnismenge im JSON-Format zurückgegeben werden soll. Die JSON-Format hängt vom angegebenen Modus ab. Diese Klausel kann nicht mit der FOR UPDATE oder FOR READ ONLY-Klausel verwendet werden. Mit FOR JSON deklarierte Cursor sind implizit READ ONLY.

Wenn Sie den RAW-Modus angeben, wird jede Zeile in der Ergebnismenge als entschachtelte JSON-Darstellung zurückgegeben.

Im AUTO-Modus werden die Abfrageergebnisse als verschachtelte JSON-Objekte zurückgegeben, basierend auf Abfrage-Joins.

Der EXPLICIT-Modus ermöglicht es Ihnen, die Form der generierten JSON-Objekte zu steuern. Mit dem EXPLICIT-Modus haben Sie mehr Flexibilität beim Angeben von Spalten und verschachtelten hierarchischen Objekten, um einheitliche oder heterogene Arrays zu erzeugen.

OPTION-Klausel Diese Klausel liefert Hints, wie die Abfrage verarbeitet werden soll. Die folgenden Abfrage-Hints werden unterstützt:

- **MATERIALIZED VIEW OPTIMIZATION-Klausel** Verwenden Sie die MATERIALIZED VIEW OPTIMIZATION-Klausel, um anzugeben, wie der Optimierer bei der Verarbeitung einer Abfrage materialisierte Ansichten einsetzen soll. Der angegebene *option-value* setzt die Datenbankoption *materialized_view_optimization* nur bei dieser Abfrage außer Kraft. Mögliche Werte für *option-value* sind dieselben wie für die Datenbankoption *materialized_view_optimization*.
- **FORCE OPTIMIZATION-Klausel** Wenn eine Abfragespezifikation nur einfache Abfragen enthält (Einzelblock- und Einzeltabellenabfragen mit Gleichheitsbedingungen in der WHERE-Klausel, die eine bestimmte Zeile eindeutig identifizieren), umgeht sie während der Verarbeitung üblicherweise

die kostenbasierte Optimierung. Es kann aber sein, dass Sie eine kostenbasierte Optimierung wünschen. Wenn Sie z.B. materialisierte Ansichten in die Abfragenverarbeitung einbeziehen möchten, muss es eine Ansichtenübereinstimmung geben. Eine Ansichtenübereinstimmung findet jedoch nur während einer kostenbasierten Optimierung statt. Wenn Sie wollen, dass eine kostenbasierte Optimierung bei einer Abfrage durchgeführt wird, aber die Abfragespezifikation nur einfache Abfragen enthält, geben Sie die **FORCE OPTIMIZATION**-Option an, um zu gewährleisten, dass der Optimierer eine kostenbasierte Optimierung bei der Abfrage durchführt.

Auf ähnliche Weise erzwingt die Angabe der **FORCE OPTIMIZATION**-Option in einer **SELECT**-Anweisung innerhalb einer Prozedur die Verwendung des Optimierers bei jedem Aufruf der Prozedur. In diesem Fall werden Pläne für die Anweisung nicht im Cache gespeichert.

- **FORCE NO OPTIMIZATION-Klausel** Geben Sie die **FORCE NO OPTIMIZATION**-Klausel an, wenn Sie möchten, dass die Anweisung den Optimierer umgeht. Wenn die Anweisung zu komplex für eine Verarbeitung auf diese Weise ist – möglicherweise aufgrund der festgelegten Datenbankoptionen oder Eigenschaften des Schemas bzw. der Abfrage – schlägt die Anweisung fehl und der Datenbankserver gibt einen Fehler zurück.
- **option-name = option-value** Legen Sie eine Optionseinstellung fest. Die von Ihnen angegebene Einstellung gilt nur für die aktuelle Anweisung und hat Vorrang vor allen öffentlichen und temporären Optionseinstellungen, einschließlich durch ODBC-fähige Anwendungen festgelegter Einstellungen.

Die unterstützten Optionen sind folgende:

- „isolation_level-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „max_query_tasks-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „optimization_goal-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „optimization_level-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „optimization_workload-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „user_estimates-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Wenn Sie die **isolation_level**-Option in einer Abfrage angeben, hat der in der Abfrage angegebene Wert Vorrang vor allen anderen Einstellungen für die Isolationsstufe der aktuellen Abfrage (z.B. dem Setzen der **isolation_level**-Option für die Datenbank oder der Einstellung für den Cursor).

sequence-expression Sie können den aktuellen Wert (**CURRVAL**) oder den nächsten Wert (**NEXTVAL**) aus einem Sequenzgenerator wählen.

Bemerkungen

Die **SELECT**-Anweisung kann wie folgt verwendet werden:

- Zum Abrufen von Ergebnissen aus der Datenbank.
- In Interactive SQL, um Daten in einer Datenbank zu durchsuchen oder um Daten aus einer Datenbank in eine externe Datei zu exportieren.
- In Prozeduren und Triggern oder in Embedded SQL. Eine **SELECT**-Anweisung mit einer **INTO**-Klausel wird für das Abrufen von Ergebnissen aus der Datenbank verwendet, wenn die **SELECT**-Anweisung nur eine Zeile zurückgibt. Für Abfragen mehrerer Zeilen müssen Sie Cursor verwenden.

- Zum Zurückgeben einer Ergebnismenge aus einer Prozedur.

Privilegien

Sie benötigen die entsprechenden SELECT-Privilegien für die Objekte, die in der SELECT-Anweisung referenziert werden.

Um den CURRVAL- oder NEXTVAL-Wert aus einem Sequenzgenerator auswählen zu können, müssen Sie das USE ANY SEQUENCE-Systemprivileg haben oder der Eigentümer der Sequenz sein oder Ihnen müssen die für die Verwendung des Sequenzgenerators erforderlichen Privilegien erteilt worden sein.

Nebenwirkungen

Keine.

Siehe auch

- „Eignung für ein Überspringen der Abfrageverarbeitungsphase“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Fortgeschrittene Aufgaben: Phasen der Abfrageverarbeitung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Ausdrücke“ auf Seite 22
- „FROM-Klausel“ auf Seite 863
- „GROUP BY-Klausel“ auf Seite 903
- „WINDOW-Klausel“ auf Seite 1124
- „Suchbedingungen“ auf Seite 42
- „DECLARE CURSOR-Anweisung [ESQL] [SP]“ auf Seite 778
- „UNION-Anweisung“ auf Seite 1096
- „EXCEPT-Anweisung“ auf Seite 841
- „INTERSECT-Anweisung“ auf Seite 927
- WITH Tabellen-Hint-Klausel, FROM-Klausel auf Seite 869
- „materialized_view_optimization-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „ansi_update_constraints-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „reserved_keywords-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Spalte '%1' nicht gefunden“ [*Fehlermeldungen*]
- „Zeilenbeschränkungsklauseln in SELECT-, UPDATE- und DELETE-Abfrageblöcken“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Joins: Daten aus mehreren Tabellen abrufen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Allgemeine Tabellenausdrücke“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Verwendung einer Sequenz zum Generieren von eindeutigen Werten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „FOR XML EXPLICIT“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Abrufen von Abfrageergebnissen als XML mit der FOR XML-Klausel“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Abfrageergebnisse mit der FOR JSON-Klausel als JSON abrufen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „ODBC-Cursormerkmale“ [*SQL Anywhere Server - Programmierung*]
- „Aktualisierbare Anweisungen“ [*SQL Anywhere Server - Programmierung*]
- „SQL Anywhere-Cursor“ [*SQL Anywhere Server - Programmierung*]
- „Transact-SQL-Outer-Joins (* = oder = *)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Mit Transact-SQL kompatible Abfragen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Kernfunktion. Wegen der Komplexität der SELECT-Anweisung sollten Sie individuelle Klauseln auf den Standard abprüfen. Zum Beispiel ist das ROLLUP-Schlüsselwort, das in einer GROUP BY-Klausel angegeben werden kann, Teil der optionalen SQL/2008-Sprachenfunktion T431. Zu den optionalen SQL/2008-Sprachenfunktionen, die von SQL Anywhere unterstützt werden, gehören folgende:
 - Die WINDOW-Klausel und die WINDOW-Aggregatfunktionen umfassen die optionalen SQL/2008-Sprachenfunktionen T611 und T612.
 - Sequenzausdrücke sind Teil der Funktion T176.

- Allgemeine Tabellenausdrücke sind die optionale SQL/2008-Sprachenfunktion T121. Ein in einer verschachtelten Abfrage enthaltener gemeinsamer Tabellenausdruck ist Funktion T122. WITH RECURSIVE ist die optionale SQL/2008-Sprachenfunktion T131, bei Einbindung in eine verschachtelte Abfrage Funktion T132.
- Die Möglichkeit zum Angeben einer ORDER BY-Klausel mit einem Abfrageausdruck, der UNION, EXCEPT oder INTERSECT erfordert, ist die optionale Funktion F850. Die Möglichkeit zum Angeben von ORDER BY in einer Unterabfrage ist Funktion F851.
- Im SQL-Standard sind FOR UPDATE und FOR READ ONLY Teil einer Cursordeklaration.

SQL Anywhere bietet Unterstützung für viele Erweiterungen für die SQL/2008-Definition der Anweisung SELECT. Dazu gehören folgende:

- Die optionale Klausel für die *cursor-concurrency* (FOR UPDATE BY { LOCK | VALUES | TIMESTAMP}) ist eine Erweiterung des Herstellers.
- Die Klauseln FOR XML, OPTION und die INTO sind Erweiterungen des Herstellers.
- Die Zeilenbeschränkungsklausel ist eine Erweiterung des Herstellers. Im SQL/2008-Standard wird die Zeileneinschränkung mithilfe der FETCH FIRST-Syntax unterstützt, bei der es sich um die optionale Sprachenfunktion F856 handelt. Die Syntax für die Funktion F856 wird nicht von SQL Anywhere unterstützt.
- Die Möglichkeit zum Angeben von ORDER BY *n* ist eine Erweiterung des Herstellers.
- In SQL/2008 sind alle Cursor mit Ausnahme von INSENSITIVE-Cursor standardmäßig aktualisierbar. Die schreibgeschützte Standard bei Embedded SQL-Programmen ist eine Erweiterung des Herstellers.
- **Transact-SQL** Bei der Unterstützung für die SELECT-Anweisung gibt es erhebliche Unterschiede zwischen SQL Anywhere und Adaptive Server Enterprise. Einige Funktionen der SELECT-Anweisung werden von Adaptive Server Enterprise nicht unterstützt.

Zu diesen Unterschieden gehören folgende:

- Sybase ASE unterstützt nicht die Cursorparallelitätsklausel von SQL Anywhere. Um eine Sperre auf eine abgerufene Zeile zu setzen, müssen Sie den Tabellen-Hint HOLDLOCK verwenden.
- Adaptive Server Enterprise unterstützt weder rekursive Abfragen noch allgemeine Tabellenausdrücke.
- Es gibt Unterschiede zwischen Adaptive Server Enterprise und SQL Anywhere in Bezug auf Outer Joins in Transact-SQL.

In Transact-SQL verwenden Sie die SELECT-Anweisung zum Zuordnen eines Werts zu einer Variablen, ähnlich wie bei der Watcom-SQL-Anweisung SET.

Beispiel

Diese Abfrage gibt die Gesamtzahl der Mitarbeiter in der Employees-Tabelle zurück.

```
SELECT COUNT(*)  
FROM GROUPO.Employees;
```

In dieser Abfrage werden alle Kunden (customers) und der Gesamtwert ihrer Bestellungen (orders) aufgelistet.

```
SELECT CompanyName,  
       CAST( SUM( SalesOrderItems.Quantity *  
                Products.UnitPrice ) AS INTEGER ) VALUE  
FROM GROUPO.Customers  
  JOIN GROUPO.SalesOrders  
  JOIN GROUPO.SalesOrderItems  
  JOIN GROUPO.Products  
GROUP BY CompanyName  
ORDER BY VALUE DESC;
```

Die folgende Anweisung zeigt eine Embedded SQL-SELECT-Anweisung, in der die Anzahl von Mitarbeitern in der Tabelle Employee in die :size-Hostvariable selektiert wird:

```
SELECT COUNT(*) INTO :size  
FROM GROUPO.Employees;
```

Die folgende Anweisung ist dahingehend optimiert, die erste Zeile der Ergebnismenge schnell zurückzugeben:

```
SELECT Name  
FROM GROUPO.Products  
GROUP BY Name  
HAVING COUNT( * ) > 1  
AND MAX( UnitPrice ) > 10  
OPTION( optimization_goal = 'first-row' );
```

SET CONNECTION-Anweisung [Interactive SQL] [ESQL]

Ändert die aktive Datenbankverbindung.

Syntax

```
SET CONNECTION [ connection-name ]
```

```
connection-name :  
identifier  
| string  
| hostvar
```

Bemerkungen

Die SET CONNECTION-Anweisung ändert die aktive Datenbankverbindung auf Verbindungsname. Der aktuelle Zustand einer Verbindung wird gespeichert und wieder angenommen, wenn sie wieder eine aktive Verbindung wird. Wenn kein Verbindungsname angegeben ist und es eine Verbindung gibt, die nicht benannt worden ist, dann wird diese Verbindung die aktive Verbindung.

Wenn Cursor in Embedded SQL geöffnet werden, werden sie der aktuellen Verbindung zugeordnet. Wenn die Verbindung geändert wird, sind die Cursornamen der vorhergehenden aktiven Verbindung nicht mehr zugänglich. Diese Cursor bleiben aktiv und an ihrer Position, und werden zugänglich, wenn die zugeordnete Verbindung wieder aktiv wird.

Diese SQL-Anweisung wird nicht für SAP HANA-Datenbanken unterstützt.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „CONNECT-Anweisung [ESQL] [Interactive SQL]“ auf Seite 578
- „DISCONNECT-Anweisung [ESQL] [Interactive SQL]“ auf Seite 802
- „Interactive SQL“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** SET CONNECTION ist Teil der optionalen SQL/2008-Sprachenfunktion F771 (Verbindungsverwaltung). Die Verwendung innerhalb einer Interactive SQL-Sitzung ist eine Erweiterung des Herstellers.

Beispiel

Das folgende Beispielfragment ist Embedded SQL-Code.

```
EXEC SQL SET CONNECTION :conn_name;
```

Setzen Sie in Interactive SQL die aktuelle Verbindung auf die fiktive Verbindung conn1.

```
SET CONNECTION conn1;
```

SET DESCRIPTOR-Anweisung [ESQL]

Beschreibt die Variablen in einem SQL-Deskriptorbereich und platziert Daten in den Deskriptorbereich.

Syntax

```
SET DESCRIPTOR descriptor-name
{ COUNT = { integer | hostvar }
| VALUE { integer | hostvar } assignment, ... }
```

```
assignment :
{ TYPE | SCALE | PRECISION | LENGTH | INDICATOR }
= { integer | hostvar }
| DATA = hostvar
```

```
descriptor-name : identifier
```

Bemerkungen

Die SET DESKRIPTOR-Anweisung wird verwendet, um die Variablen in einem Deskriptorbereich zu beschreiben und um Daten in diesem Deskriptorbereich abzulegen.

Die SET...COUNT-Anweisung stellt die Anzahl der beschriebenen Variablen im Deskriptorbereich ein. Der Wert für "count" darf die Anzahl der bei Zuweisung des Deskriptorbereichs angegebenen Variablen nicht überschreiten.

Der Wert { *integer* | *hostvar* } gibt die Variable im Deskriptorbereich an, für die die Zuordnungen ausgeführt werden.

Mit der Durchführung von SET...DATA findet eine Typüberprüfung statt, um sicherzustellen, dass die Variable innerhalb des Deskriptorbereichs und die Hostvariable den gleichen Datentyp haben. LONG VARCHAR und LONG BINARY werden von SET DESCRIPTOR...DATA nicht unterstützt.

Wenn ein Fehler auftritt, wird er in dem SQLCA zurückgegeben.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „ALLOCATE DESCRIPTOR-Anweisung [ESQL]“ auf Seite 449
- „DEALLOCATE DESCRIPTOR-Anweisung [ESQL]“ auf Seite 777
- „Der SQL-Deskriptor-Bereich (SQLDA)“ [*SQL Anywhere Server - Programmierung*]

Standards und Kompatibilität

- **SQL/2008** SET DESCRIPTOR ist Teil der optionalen SQL/2008-Sprachenfunktion B031 (Basic Dynamic SQL).

Beispiel

Mit dem folgenden Beispiel wird der Typ einer Spalte mit der Position "Spaltennummer" in SQLDA festgelegt.

```
void set_type( SQLDA *sqlda, int col_num, int new_type )
{
    EXEC SQL BEGIN DECLARE SECTION;
    INT new_type1 = new_type;
    INT col = col_num;
    EXEC SQL END DECLARE SECTION;
    EXEC SQL SET DESCRIPTOR sqlda VALUE :col TYPE = :new_type1;
}
```

SET MIRROR OPTION-Anweisung

Hinweis

Scale-Out mit Schreibschutz und Datenbankspiegelung erfordern jeweils eine getrennte Lizenz. Siehe „Getrennt lizenzierbare Komponenten“ [*SQL Anywhere 16 - Einführung*].

Ändert die Werte der Optionen zum Steuern der Einstellungen für Datenbankspiegelung und Scale-Out mit Schreibschutz.

Syntax

SET MIRROR OPTION *option-name*={ *option-value* | **NULL** }

option-name :

authentication_string
auto_add_fan_out
auto_add_server
auto_failover
child_creation
page_timeout
max_disconnected_time
max_retry_connect_time
synchronization_mode

Parameter

NULL Gibt den Standardwert für die Option an. Wenn der *option-name* auf NULL gesetzt ist, wird der Optionswert auf den Standardwert gesetzt.

<i>option-name</i>	Gilt für	Werte	Standardwert	Beschreibung
authentication_string	Datenbankspiegelung	Zeichenfolge	NULL	Gibt die Authentifizierungszeichenfolge an, die von allen Servern im Datenbankspiegelungssystem verwendet wird. Die Authentifizierungszeichenfolge ist bei der Datenbankspiegelung erforderlich.
auto_add_fan_out	Scale-Out mit Schreibschutz	Ganzzahl	10	Gibt die maximale Anzahl von untergeordneten Elementen für jede Verzweigung an. Der Mindestwert, der definiert werden kann, ist 2.
auto_add_server	Scale-Out mit Schreibschutz	Zeichenfolge	NULL	Gibt den Namen des Datenbankservers an, der als das übergeordnete Objekt der automatischen Zuordnungsstruktur fungiert.

option-name	Gilt für	Werte	Standard wert	Beschreibung
auto_failover	Datenbankspiegelung	On, Off	NULL	<p>Gibt an, ob der Spiegelserver automatisch als Primärserver übernimmt, wenn der aktuelle Primärserver ausfällt. Diese Option gilt nicht im synchronus-Modus.</p> <p>Diese Option akzeptiert Boolesche Werte. (Automatischer Failover wird mit YES, ON, TRUE oder 1 aktiviert und mit NO, OFF, FALSE und 0 deaktiviert.) Bei den Parametern wird die Groß-/Kleinschreibung nicht berücksichtigt.</p> <p>Wenn Sie den asynchronen oder Asynchron-Ganzseiten-Modus verwenden, ist es empfehlenswert, die auto_failover-Option auf ON zu setzen. Dadurch übernimmt der Spiegelserver automatisch als Primärserver, wenn der ursprüngliche Primärserver ausfällt.</p>
child_creation	Scale-Out mit Schreibschutz	automatic, off, manual	automatic	Steuert, ob Kopieknoten automatisch erstellt werden.
page_timeout	Datenbankspiegelung	Ganzzahl in Sekunden	5	Gibt an, mit welcher Häufigkeit (in Sekunden) Transaktionslog-Seiten an den Spiegelserver gesendet werden, gleichgültig ob sie voll sind oder nicht. Diese Option gilt nur im Asynchron-Ganzseiten-Modus.

<i>option-name</i>	Gilt für	Werte	Standardwert	Beschreibung
max_disconnected_time	Scale-Out mit Schreibschutz	Ganzzahl, in Sekunden, größer oder gleich max_retry_connect_time	Keine Zeitbegrenzung	Gibt die Zeitspanne an, die nach der letzten Verbindung des Kopieknotens mit dem übergeordneten Knoten, dem alternativen übergeordneten Knoten oder der Stammdatenbank verstreichen darf, bevor der Kopieknoten heruntergefahren wird.
max_retry_connect_time	Scale-Out mit Schreibschutz	Ganzzahl in Sekunden	120	Gibt die Zeitspanne an, für die ein Kopieknoten versucht, eine Verbindung zum übergeordneten Server herzustellen, nachdem der übergeordnete Server ausgefallen ist.

option-name	Gilt für	Werte	Standard wert	Beschreibung
promotion_time	Scale-Out mit Schreibschutz	Ganzzahl, in Sekunden, größer oder gleich max_retry_connect_time	3600	<p>Gibt die Zeitspanne an, für die ein Kopieknoten mit dem Stammdatenbankserver verbunden bleibt, nachdem eine Verbindung mit dem übergeordneten Knoten verloren gegangen ist, bevor er selbst zum übergeordneten Knoten wird (damit die Scale-Out-Struktur keinen getrennten übergeordneten Knoten enthält). Mithilfe dieser Option können Sie vermeiden, dass die Scale-Out-Struktur angepasst wird, wenn ein Kopieknoten für kurze Zeit nicht aktiv ist (was zu einer flachen Scale-Out-Struktur führen kann), während versucht wird, eine über längere Zeiträume gesteigerte Auslastung der Primärdatenbank zu verhindern. Damit der Kopieknoten nie zum übergeordneten Knoten wird, setzen Sie diese Option auf 315,360,000 (10 Jahre) oder höher.</p> <p>Diese Option wird nur in Datenbanken der Version 16 unterstützt.</p>
synchronizati- on_mode	Datenbankspiegelung	synchronous, asynchronous, asyncfullpage	synchronous	Gibt den Synchronisationsmodus für die Datenbankspiegelung an: synchronous (sync), asynchronous (async) oder asyncfullpage (page). Der Synchronisationsmodus steuert, wann und wie Transaktionen auf dem Spiegelserver aufgezeichnet werden.

Bemerkungen

Wenn Sie einen Datenbankserver für ein Datenbankspiegelungssystem oder ein Scale-Out-System mit Schreibschutz mithilfe der CREATE MIRROR SERVER-Anweisung erstellen, können Sie mit der SET MIRROR OPTION-Anweisung die Einstellungen für das System konfigurieren.

Privilegien

Sie müssen den Server gestartet haben oder das MANAGE ANY MIRROR SERVER-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Automatisches Zuweisen des übergeordneten Knotens eines Kopieknotens“ [*SQL Anywhere Server - Datenbankadministration*]
- „So handhaben Scale-Out-Systeme mit Schreibschutz den Verlust der Verbindung mit einem übergeordneten Knoten“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankspiegelungsmodi“ [*SQL Anywhere Server - Datenbankadministration*]
- „Hinzufügen von untergeordneten Kopieknoten“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -xa“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankspiegelung“ [*SQL Anywhere Server - Datenbankadministration*]
- „SQL Anywhere-Scale-Out mit Schreibschutz“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE MIRROR SERVER-Anweisung“ auf Seite 656
- „ALTER MIRROR SERVER-Anweisung“ auf Seite 480
- „DROP MIRROR SERVER-Anweisung“ auf Seite 815
- „SYSMIRROROPTION-Systemansicht“ auf Seite 1463

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung setzt die Authentifizierungszeichenfolge für ein Datenbankspiegelungssystem auf abc:

```
SET MIRROR OPTION authentication_string = 'abc';
```

SET OPTION-Anweisung

Ändert die Werte von Datenbank- und Verbindungsoptionen.

Syntax 1

```
SET [ EXISTING ] [ TEMPORARY ] OPTION  
[ userid. ] PUBLIC. ]option-name = [ option-value ]
```

Syntax 2 (nicht mehr empfohlen)

SET [EXISTING] [TEMPORARY] OPTION
[*userid*.| **PUBLIC**.] *option-name* = [*identifier*]

userid : *identifier*

option-name : *identifier*

option-value : **ON**, **OFF**, **NULL**, *string literal*, *number*, *hostvar*, oder *@variable-name*

Parameter

option-value Bei Syntax 1 kann für *option-value* Folgendes angegeben werden:

- die Schlüsselwörter **ON**, **OFF** oder **NULL**
- ein Zeichenfolgenliteral-Wert, in Apostrophen
- eine Zahl in einem beliebigen gültigen Format, einschließlich **NUMERIC**
- innerhalb eines Embedded SQL-Programms der Wert für eine Hostvariable
- der Wert einer SQL-Variablen, deren Variablenname mit einem @-Zeichen beginnen muss

Bei Syntax 2 können Sie jeden gültigen Bezeichner als Optionswert angeben. Zudem behandelt der Datenbankserver den Namen des Bezeichners, als wäre er ein in Apostrophe eingeschlossenes Zeichenfolgenliteral. Die folgenden Anweisungen sind beispielsweise gleichwertig:

```
SET TEMPORARY OPTION ansi_update_constraints = 'strict';
```

```
SET TEMPORARY OPTION ansi_update_constraints = strict;
```

Bemerkungen

Die SET OPTION-Anweisung wird verwendet, um Optionen zu ändern, welche das Verhalten des Datenbankservers beeinflussen. Die Einstellung eines Optionswerts kann das Verhalten für alle Benutzer (**PUBLIC**), für einen einzelnen Benutzer oder für die aktuelle Verbindung ändern. Die neue Einstellung kann entweder temporär oder permanent gelten.

Die folgenden Optionsklassen können mit der SET OPTION-Anweisung eingestellt werden:

- Transact-SQL-Kompatibilitätsoptionen.
- Verbindungs- und Datenbankoptionen.
- Synchronisationsoptionen.
- Benutzerdefinierte Optionen.

Optionsbereich Bei den meisten Optionen können Sie die Werte in drei Bereichsstufen einstellen: öffentlich, Benutzer und Verbindung. Einige spezifische Optionen, z.B. `login_mode`, sind auf die öffentliche Stufe beschränkt. Eine Verbindungsoption hat den Vorrang vor den beiden anderen Stufen und Benutzeroptionen haben Vorrang vor öffentlichen Optionen. Zum Festlegen einer Option auf Verbindungsebene verwenden Sie das **TEMPORARY**-Schlüsselwort. Wenn Sie eine Benutzeroption für den aktuellen Benutzer einstellen, wird gleichzeitig die entsprechende Option auf Verbindungsebene eingestellt.

Standardmäßig gilt der Optionswert für die gegenwärtig angemeldete Benutzer-ID, die die SET OPTION-Anweisung ausgeführt hat. Wenn Sie eine Benutzer-ID angeben, gilt der Optionswert für diesen Benutzer.

Wenn Sie PUBLIC angeben, gilt der Optionswert für alle Benutzer, die keine individuelle Einstellung für die Option haben.

TEMPORARY-Optionen Standardmäßig ist ein neuer Optionswert permanent, es sei denn, das TEMPORARY-Schlüsselwort ist angegeben. Das Hinzufügen des Schlüsselworts TEMPORARY zur SET OPTION-Anweisung verändert die Dauer des Änderungsvorgangs.

Wenn die SET TEMPORARY OPTION-Anweisung nicht mit einer Benutzer-ID qualifiziert wird, ist der neue Optionswert nur für die aktuelle Verbindung wirksam.

Wenn SET TEMPORARY OPTION für die PUBLIC-Rolle verwendet wird, gilt die Änderung so lange, wie die Datenbank läuft. Wenn die Datenbank heruntergefahren wird, werden TEMPORARY-Optionen für die PUBLIC-Rolle wieder auf ihren dauerhaften Wert zurückgesetzt.

Das Setzen von temporären Optionen für die PUBLIC-Rolle bietet Sicherheitsvorteile. Wenn zum Beispiel die login_mode-Option aktiviert ist, richtet sich die Datenbank nach der Login-Sicherheit des Systems, auf dem sie läuft. Diese temporär zu aktivieren bedeutet, dass eine Datenbank, die sich auf die Sicherheit einer Windows-Domäne bezieht, nicht gefährdet wird, wenn sie heruntergefahren und auf ein lokales System kopiert wird. In diesem Fall wird die temporäre Aktivierung der Option login_mode wieder auf ihren permanenten Wert zurückgesetzt, der "Standard" sein kann, ein Modus, in dem integrierte Logins nicht zulässig sind.

Entfernen von Optionseinstellungen Wenn *option-value* nicht angegeben ist, wird die angegebene Optionseinstellung aus der Datenbank gelöscht. Wenn es sich um eine Optionseinstellung auf Benutzerebene gehandelt hat, wird der Wert wieder auf die PUBLIC-Einstellung zurückgesetzt. Beim Löschen der TEMPORARY-Option wird die Optionseinstellung auf die permanente Einstellung für den Benutzer zurückgesetzt.

Optionsdatentypen Optionen können boolesche, numerische, oder Zeichenfolgenwerte haben, werden jedoch in der Datenbank immer als Zeichenfolgen gespeichert. Optionseinstellungen werden immer als Zeichenfolgen zurückgegeben, unabhängig davon, ob sie das Ergebnis einer Eigenschaftsfunktion, einer Funktion oder einer Systemprozedur sind. Optionswerte können nicht länger sein als die Seitengröße der Datenbank.

Benutzerdefinierte Optionen Jede Option, ob benutzerdefiniert oder nicht, muss eine öffentliche Einstellung haben, bevor ein benutzerspezifischer Wert zugeordnet werden kann. Der Datenbankserver unterstützt bei benutzerdefinierten Optionen nicht das Setzen von TEMPORARY-Werten. Wenn Sie zum Beispiel eine benutzerdefinierte Option mit dem Namen ApplicationControl erstellen möchten, müssen Sie zuerst die folgende Anweisung ausführen:

```
SET OPTION PUBLIC.ApplicationControl = 'Default';
```

Diese Anweisung setzt die ApplicationControl-Option für alle Benutzer auf "Default" und wird bei jeder neuen Verbindung mit dem Server wirksam. Danach kann ein einzelner Benutzer seine eigene Einstellung für diese Option festlegen, indem er eine separate SET OPTION-Anweisung ausführt.

Einschränkungen Wenn Sie das EXISTING-Schlüsselwort verwenden, können Optionswerte nicht für eine einzelne Benutzer-ID gesetzt werden, es sei denn, es gibt bereits eine PUBLIC-Einstellung für die betreffende Option.

Vorsicht

Ändern Sie Optionswerte nicht, während ein Cursor geöffnet ist. Das Ändern der Optionswerte bei geöffnetem Cursor kann zu inkonsistenten Ergebnissen in dem Cursor führen. Wenn Sie beispielsweise die `date_format`-Option ändern, während ein Cursor geöffnet ist, kann dies dazu führen, dass einige Zeilen im alten Format zurückgegeben werden und einige Zeilen im neuen Format. Um zu gewährleisten, dass die Zeilen in der Ergebnismenge konsistent mit dem neuen Optionswert berechnet werden, öffnen Sie den Cursor erst nach dem Ändern des Optionswerts.

Es gibt mehrere Möglichkeiten, den Wert von spezifischen Optionen für eine Verbindung oder einen Benutzer abzufragen.

Die `SET OPTION`-Anweisung wird vom SQL Flagger ignoriert.

Privilegien

Jeder Benutzer kann seine eigenen Optionen setzen.

Um Datenbankoptionen für andere Benutzer oder Rollen, einschließlich der PUBLIC-Rolle, setzen zu können, benötigen Sie eines der folgenden Systemprivilegien, je nachdem, welches Privileg die betreffende Option erfordert:

- `SET ANY SYSTEM OPTION`
- `SET ANY PUBLIC OPTION`
- `SET ANY SECURITY OPTION`
- `SET ANY USER DEFINED OPTION`

Nebenwirkungen

Wenn `TEMPORARY` nicht angegeben ist, erfolgt ein automatisches Festschreiben.

Siehe auch

- „Datenbankoptionen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Alphabetische Liste der Datenbankoptionen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Einstellungen für die entfernte ID“ [[MobiLink - Clientadministration](#)]
- „Kompatibilitätsoptionen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Synchronisationsoptionen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „SQL Remote-Optionen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankoptionen anzeigen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankoptionen mit der `SET OPTION`-Anweisung setzen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „SYSOPTION-Systemansicht“ auf Seite 1467
- „sa_conn_options-Systemprozedur“ auf Seite 1188
- „sa_conn_options-Systemprozedur“ auf Seite 1188
- „`CONNECTION_PROPERTY`-Funktion [System]“ auf Seite 199
- „`GET OPTION`-Anweisung [ESQL]“ auf Seite 880
- „`SET OPTION`-Anweisung [Interactive SQL]“ auf Seite 1043
- „`SET`-Anweisung [T-SQL]“ auf Seite 1056
- „`SET REMOTE OPTION`-Anweisung [SQL Remote]“ auf Seite 1046

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das Datumsformat wird für alle Benutzer ohne individuelle Einstellung festgelegt:

```
SET OPTION PUBLIC.date_format = 'Mmm dd yyyy';
```

Die Option wait_for_commit wird auf On gesetzt:

```
SET OPTION wait_for_commit = 'On';
```

Das folgende Fragment ist ein Embedded SQL-Beispiel:

```
EXEC SQL SET TEMPORARY OPTION date_format = :value;
```

Legen Sie die date_format-Option für den Benutzer fest, der derzeit verbunden ist. Zukünftige Verbindung für dieselbe Benutzer-ID verwenden diesen Optionswert.

```
SET OPTION date_format = 'yyyy/mm/dd';
```

Die folgende Anweisung entfernt die Einstellung der Option date_format für die aktuelle Benutzer-ID. Nach dem Ausführen der Anweisung wird stattdessen die date_format-Einstellung für PUBLIC verwendet.

```
SET OPTION date_format=;
```

Die folgende Anweisung ändert die login_mode-Option für Berechtigungsempfänger der PUBLIC-Rolle in "Standard":

```
SET OPTION PUBLIC.login_mode = 'Standard';
```

SET OPTION-Anweisung [Interactive SQL]

Ändert die Werte von Interactive SQL-Optionen.

Syntax 1 - Setzt eine Interactive SQL-Option

```
SET OPTION option-name = [ option-value ] | SET TEMPORARY OPTION option-name = [ option-value ]
```

option-name : identifier | string | hostvar

option-value : string | identifier | number

Syntax 2 - Speichert die aktuellen Optionen permanent

```
SET PERMANENT
```

Syntax 3 - Listet die aktuellen Optionseinstellungen der Datenbank auf

```
SET
```

Bemerkungen

Wenn Sie eine Option mit der SET OPTION-Syntax einrichten, wird die Optionseinstellung permanent gespeichert und ändert sich nur, wenn eine andere SET OPTION-Anweisung sie ändert.

Mit der SET TEMPORARY OPTION-Syntax können Sie temporär eine Optionseinstellung ändern. Die temporäre Einstellung bleibt wirksam, bis Sie Interactive SQL schließen. Wenn Sie Interactive SQL das nächste Mal starten, erhält die Option wieder ihre ursprüngliche Einstellung.

Verwenden Sie die SET PERMANENT-Syntax, um alle aktuellen Interactive SQL-Optionseinstellungen permanent zu speichern (temporäre Einstellungen werden permanent).

Wenn der *option-value* nicht vorhanden ist, wird die angegebene Option auf den Standardwert gesetzt.

Verwenden Sie die Syntax 3, um alle aktuellen Optionseinstellungen der *Datenbank* anzuzeigen. Wenn temporäre Optionseinstellungen für den Datenbankserver vorhanden sind, werden diese anstelle der permanenten Einstellungen angezeigt.

Optionseinstellungen für Interactive SQL werden im Clientcomputer gespeichert, nicht in der Datenbank.

Die folgende Tabelle enthält eine Liste der Optionen für Interactive SQL.

Option	Werte	Standardwert
„auto_commit-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	On, Off	Off
„auto_refetch-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	On, Off	On
„bell-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbankadministration]	On, Off	On
„command_delimiter-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	Zeichenfolge	" ; "
„commit_on_exit-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	On, Off	On
„default_isql_encoding-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	Zeichenfolge	Leere Zeichenfolge
„echo-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	On, Off	On

Option	Werte	Standardwert
„input_format-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	TEXT, FIXED	TEXT
„isql_allow_read_client_file-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbankadministration]	On, Off, Prompt	Prompt
„isql_allow_write_client_file-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbankadministration]	On, Off, Prompt	Prompt
„isql_command_timing-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	On, Off	On
„isql_escape_character-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	Zeichen	" \"
„isql_field_separator-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	Zeichenfolge	" , "
„isql_maximum_displayed_rows-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbankadministration]	"All" oder eine nicht negative Ganzzahl	500
„isql_print_result_set-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	Last, All, None	Last
„isql_quote-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	Zeichenfolge	'
„isql_show_multiple_result_sets-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbankadministration]	On, Off	Off
„nulls-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	Zeichenfolge	"(NULL)"

Option	Werte	Standardwert
„on_error-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	Stop, Continue, Prompt, Exit, Notify_Continue, Notify_Stop, Notify_Exit	Prompt
„output_format-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	TEXT, FIXED, HTML, SQL, XML	TEXT
„output_length-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	Ganzzahl	0
„output_nulls-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	Zeichenfolge	Leere Zeichenfolge
„truncation_length-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration]	Ganzzahl	256

Privilegien

Keine.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Siehe auch

- „Interactive SQL-Optionen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Im folgenden Beispiel wird der Wert der on_error-Option in "continue" geändert:

```
SET OPTION on_error='continue';
```

SET REMOTE OPTION-Anweisung [SQL Remote]

Legt einen Nachrichtensteuerungsparameter für eine SQL Remote-Nachrichtenverbindung fest.

Syntax

```
SET REMOTE link-name OPTION  
[ userid. PUBLIC. ] link-option-name = link-option-value
```

link-name :
file

ftp
http
smtp

link-option-name :

common-options

file-options

ftp-options

smtp-options

common-options :

debug

encode_dll

max_retries

output_log_send_on_error

output_log_send_limit

output_log_send_now

pause_after_failure

file-options :

directory

invalid_extensions

unlink_delay

ftp-options :

active_mode

host

invalid_extensions

password

port

root_directory

reconnect_retries

reconnect_pause

suppress_dialogs

user

http-options :

certificate

client_port

https

password

proxy

reconnect_retries

reconnect_pause

root_directory

url

user

smtp-options :

local_host

pop3_host

pop3_password

pop3_port

pop3_userid

smtp_authenticate

smtp_option

smtp_password

```
| smtp_port  
| smtp_userid  
| suppress_dialogs  
| top_supported
```

link-option-value : *string*

Parameter

userid Wenn Sie keine *userid* eingeben, wird der aktuelle Publikationseigentümer genommen.

common-options Folgende Optionen gelten für die FILE-, FTP-, HTTP- und SMTP-Nachrichtensysteme gleichermaßen:

- **debug** Dieser Parameter ist entweder auf YES oder NO gesetzt. Der Standardwert ist NO. Wenn auf YES gesetzt, werden spezifische Debug-Ausgaben des Nachrichtensystems angezeigt. Diese Informationen können verwendet werden, um Probleme im Nachrichtensystem zu beheben.
- **max_retries** Standardmäßig gilt: Wenn SQL Remote im kontinuierlichen Modus ausgeführt wird und ein Fehler tritt auf, wenn auf das Nachrichtensystem zugegriffen wird, wird es automatisch nach den Sende- und/oder Empfangsphasen heruntergefahren. Verwenden Sie diesen Parameter, um die Häufigkeit festzulegen, mit der Sie aus SQL Remote zu wiederholen Sie den Sende- und/oder von der Synchronisation, bevor sie beendet.
- **output_log_send_on_error** Sendet Loginformationen, wenn ein Fehler auftritt.
- **output_log_send_limit** Begrenzt die Menge der Informationen, die an die konsolidierte Datenbank gesendet werden. Die Option *output_log_send_limit* gibt die Anzahl von Bytes am Ende des Ausgabeblogs an (d.h. die aktuellsten Einträge), die an die konsolidierte Datenbank gesendet wird. Der Standardwert ist 5 kB.
- **output_log_send_now** Wenn auf YES gesetzt, werden Ausgabeloginformationen an die konsolidierte Datenbank gesendet. Beim nächsten Abruf sendet die entfernte Datenbank die Ausgabeloginformationen und setzt anschließend die Option *output_log_send_now* auf NO.
- **pause_after_failure** Dieser Parameter gilt, wenn der *max_retries*-Parameter auf einen anderen Wert als Null gesetzt ist, und SQL Remote im kontinuierlichen Modus ausgeführt wird. Wenn ein Fehler im Nachrichtensystem auftritt, definiert dieser Parameter die Anzahl der Sekunden, die SQL Remote wartet, bis die Sende- und/oder Empfangsphasen wiederholt werden.
- **encode_dll** Wenn Sie ein benutzerdefiniertes Kodierungsschema implementiert haben, müssen Sie dies auf den vollständigen Pfad der von Ihnen erstellten angepassten Kodierungs-DLL einstellen.

file-options Folgende Optionen gelten nur für das FILE-Nachrichtensystem:

- **directory** Das Verzeichnis, in dem die Nachrichten gespeichert werden. Dieser Parameter ist eine Alternative zur SQLREMOTE-Umgebungsvariablen.
- **invalid_extensions** Eine durch Kommas getrennte Liste von Dateierweiterungen, die vom SQL Remote-Nachrichtenagenten (dbremote) beim Generieren von Dateien im Nachrichtensystem nicht verwendet werden sollen.

- **unlink_delay** Die Sekunden, die abgewartet werden sollen, bevor eine Datei gelöscht wird, falls der vorherige Versuch, die Datei zu löschen, fehlgeschlagen ist. Wenn für unlink_delay kein Wert definiert ist, wird als Standardverhalten 1 Sekunde nach dem ersten fehlgeschlagenen Versuch, 2 Sekunden nach dem zweiten, 3 Sekunden nach dem dritten und 4 Sekunden nach dem vierten gewartet.

ftp-options Folgende Optionen gelten nur für das FTP-Nachrichtensystem:

- **active_mode** Dieser Parameter steuert, wie SQL Remote die Client-/Server-Verbindung herstellt. Dieser Parameter ist entweder auf YES oder NO gesetzt. Der Standardwert ist NO (passiver Modus). Der passive Modus ist der bevorzugte Übermittlungsmodus und der Standardwert für die FTP-Nachrichtenverbindung. Im passiven Modus werden alle Datenübertragungsverbindungen durch den Client initiiert, in diesem Fall durch die Nachrichtenverbindung. Im aktiven Modus initiiert der FTP-Server alle Datenverbindungen.
- **host** Der Hostname des Computers, auf dem der FTP-Server läuft. Dieser Parameter kann ein Hostname (z.B. FTP.ianywhere.com) oder eine IP-Adresse (z.B. 192.138.151.66) sein.
- **invalid_extensions** Eine durch Kommas getrennte Liste von Dateierweiterungen, die von dbremote beim Generieren von Dateien im Nachrichtensystem nicht verwendet werden sollen.
- **password** Das Kennwort für den Zugriff auf den FTP-Host
- **port** Die IP-Portnummer, die für die FTP-Verbindung verwendet wird. Dieser Parameter wird gewöhnlich nicht benötigt.
- **reconnect_retries** Gibt an, wie häufig die Verbindung versuchen soll, einen Socket zu öffnen, bevor der Versuch abgebrochen wird. Der Standardwert ist "4". Wenn Sie diesen Parameter setzen, sind nur erneute Verbindungen betroffen. Die ursprünglich vom FTP-Link erstellte Verbindung ist nicht betroffen.
- **reconnect_pause** Die Zeit in Sekunden, die zwischen den Verbindungsversuchen abgewartet wird. Der Standardwert ist 30 Sekunden. Wenn Sie diesen Parameter setzen, sind nur erneute Verbindungen betroffen. Die ursprünglich vom FTP-Link erstellte Verbindung ist nicht betroffen.
- **root_directory** Das Stammverzeichnis auf der FTP-Host-Site, unter dem die Nachrichten gespeichert werden.
- **suppress_dialogs** Dieser Parameter ist auf TRUE oder FALSE gesetzt. Wenn dieser Parameter auf TRUE gesetzt ist, wird das Fenster **Verbinden** nach gescheiterten Verbindungsversuchen zum FTP-Server nicht angezeigt. Stattdessen wird eine Fehlermeldung ausgegeben.
- **user** Der Benutzername für den Zugriff auf den FTP-Host

http-options Diese Optionen gelten nur für das HTTP-Nachrichtensystem:

- **certificate** Um eine sichere (HTTPS-)Anforderung durchführen zu können, muss der Client Zugriff auf das vom HTTPS-Server verwendete Zertifikat haben. Die benötigten Informationen werden in einer Zeichenfolge von durch Semikola getrennten Schlüssel/Werte-Paaren angegeben. Sie können mithilfe des Dateischlüssels den Dateinamen für das Zertifikat angeben. Sie können nicht

sowohl einen Dateischlüssel als auch einen Zertifikatschlüssel angeben. Folgende Schlüssel sind verfügbar:

Schlüssel	Abkürzung	Beschreibung
file		The file name of the certificate
certificate	cert	The certificate itself
company	co	The company specified in the certificate
unit		The company unit specified in the certificate
name		The common name specified in the certificate

Zertifikate sind nur bei Anforderungen erforderlich, die entweder an den HTTPS-Server gerichtet sind oder von einem nicht-sicheren zu einem sicheren Server umgeleitet werden können. Nur PEM-formatierte Zertifikate werden unterstützt. **certificate**='Datei=Dateiname'

Zertifikatnamen in SQL Anywhere-Datenbanken erstellen

```
CREATE OR REPLACE CERTIFICATE certificate_name FROM FILE
'certificate_file';
```

Zertifikatnamen für einen HTTPS-Nachrichtentyp verwenden

```
SET REMOTE HTTP OPTION user_name.certificate =
'cert_name=certificate_name';
```

- **client_port** Kennzeichnet die Portnummer, auf der SQL Remote unter Verwendung von HTTP kommuniziert. Sie wird nur für Verbindungen über Firewalls bereitgestellt und empfohlen, die "ausgehende" TCP/IP-Verbindungen filtern. Sie können eine einzelne Portnummer, Bereiche von Portnummern oder eine Kombination aus beiden angeben. Die Angabe einer niedrigen Anzahl von Clientports kann dazu führen, dass SQL Remote nicht in der Lage ist, Nachrichten zu senden und zu empfangen, wenn das Betriebssystem die Ports nicht rechtzeitig freigibt, nachdem SQL Remote den Port bei einer früheren Ausführung geschlossen hat.
- **debug** Wenn dieser Parameter auf YES eingestellt ist, werden alle HTTP-Befehle und -Antworten im Ausgabelog angezeigt. Diese Informationen können zur Fehlerbehandlung von HTTP-Problemen verwendet werden. Der Standardwert ist NO.
- **https** Geben Sie an, ob HTTPS (**https=yes**) oder HTTP (**https=no**) verwendet werden soll.
- **password** Das Kennwort für die Nachrichtenserver-Datenbank. Das Kennwort authentifiziert bei HTTP-Servern und Gateways anderer Hersteller, die RFC 2617 Basic-Authentifizierung verwenden.
- **proxy_host** Gibt den URI eines Proxyservers an. Wird verwendet, wenn SQL Remote über einen Proxyserver auf das Netzwerk zugreifen muss. Zeigt an, dass SQL Remote eine Verbindung zum Proxyserver herstellen soll, um die Anforderung durch ihn an den Nachrichtenserver zu senden.
- **reconnect_retries** Gibt an, wie häufig die Verbindung versuchen soll, einen Socket zu öffnen, bevor der Versuch abgebrochen wird. Der Standardwert ist "4". Wenn Sie diesen Parameter setzen,

sind nur erneute Verbindungen betroffen. Die ursprünglich vom FTP-Link erstellte Verbindung ist nicht betroffen.

- **reconnect_pause** Die Zeit in Sekunden, die zwischen den Verbindungsversuchen abgewartet wird. Der Standardwert ist 30 Sekunden. Wenn Sie diesen Parameter setzen, sind nur erneute Verbindungen betroffen. Die ursprünglich vom FTP-Link erstellte Verbindung ist nicht betroffen.
- **root_directory** Dieser HTTP-Steuerungsparameter wird ignoriert, wenn er auf Clientseite festgelegt wurde. Sie müssen diesen Steuerungsparameter im Nachrichtenserver festlegen, bevor Sie die gespeicherte Prozedur `sr_add_message_server` oder `sr_update_message_server` aufrufen. Bei der Verwendung des HTTP-Nachrichtensystems kann die für einen entfernten Benutzer oder Publikationseigentümer angegebene Adresse nur ein einziges Unterverzeichnis enthalten und nicht mehrere Unterverzeichnisse.
- **url** Geben Sie den Servernamen oder die IP-Adresse sowie optional die Portnummer für den verwendeten HTTP-Server an, durch Semikola getrennt. Wenn Anforderungen durch den Relay Server übergeben werden, können Sie optional eine URL-Erweiterung hinzufügen, um anzuzeigen, an welche Serverfarm die Anforderung übergeben werden soll.
- **user** Die Benutzer-ID für die Nachrichtenserver-Datenbank. Dient zum Authentifizieren bei HTTP-Servern und Gateways anderer Hersteller, die RFC 2617 Basic-Authentifizierung verwenden.

smtp-options Folgende Optionen gelten nur für das SMTP-Nachrichtensystem:

- **local_host** Der Name des lokalen Computers. Er ist auf Computern nützlich, bei denen SQL Remote nicht in der Lage ist, den lokalen Hostnamen zu bestimmen. Der lokale Hostname wird benötigt, um eine Sitzung mit einem SMTP-Server zu initialisieren. In den meisten Netzwerkumgebungen kann der Hostname automatisch ermittelt werden. In diesem Fall wird dieser Eintrag nicht gebraucht.
- **pop3_host** Der Name des Computers, auf dem der POP3-Host läuft. Üblicherweise ist dies derselbe Name wie der SMTP-Host. Er entspricht dem POP3-Hostfeld im SMTP/POP3-Login-Fenster.
- **pop3_password** Das Kennwort, das zum Abrufen der Mail verwendet wird. Der Wert entspricht dem Kennwortfeld im SMTP/POP3-Login-Fenster.
- **pop3_port** Die Nummer des Ports, den der POP3-Server abhört. Der Standardwert ist 110. Dies entspricht dem Portfeld im SMTP/POP3-Login-Fenster.
- **pop3_userid** Die Benutzer-ID, die zum Abrufen der Mail verwendet wird. Die POP3-Benutzer-ID entspricht dem Benutzer-ID-Feld im SMTP/POP3-Login-Fenster. Sie erhalten eine Benutzer-ID von Ihrem POP3-Host-Administrator.
- **smtp_host** Der Name des Computers, auf dem der SMTP-Server läuft. Er entspricht dem SMTP-Hostfeld im SMTP/POP3-Login-Fenster
- **top_supported** SQL Remote verwendet einen POP3-Befehl namens TOP, um eintreffende Nachrichten zu verarbeiten. Der TOP-Befehl wird möglicherweise nicht von allen POP3-Servern unterstützt. Wenn Sie den Parameter "top_supported" auf NO setzen, verwendet SQL Remote den RETR-Befehl, der weniger effizient ist, aber mit allen POP-Servern funktioniert. Der Standardwert ist YES.

- **smtp_authenticate** Legt fest, ob die SMTP-Verbindung den Benutzer authentifiziert. Der Standardwert ist YES. Setzen Sie diesen Parameter auf NO, um die SMTP-Authentifizierung zu deaktivieren.
- **smtp_userid** Die Benutzer-ID für die SMTP-Authentifizierung. Standardmäßig übernimmt dieser Parameter denselben Wert wie der Parameter pop3_userid. Der Parameter "smtp_userid" muss nur gesetzt werden, wenn die Benutzer-ID von der des POP-Servers abweicht.
- **smtp_password** Das Kennwort für die SMTP-Authentifizierung. Standardmäßig übernimmt dieser Parameter denselben Wert wie der Parameter pop3_password. Der Parameter "smtp_password" muss nur gesetzt werden, wenn die Benutzer-ID von der des POP-Servers abweicht.
- **smtp_port** Die Nummer des Ports, den der SMTP-Server derzeit abhört. Standardwert ist "25". Dies entspricht dem Portfeld im SMTP/POP3-Login-Fenster.
- **suppress_dialogs** Wenn dieser Parameter auf TRUE gesetzt ist, wird das Fenster **Verbinden** nach gescheiterten Verbindungsversuchen zum Mail-Server nicht angezeigt. Stattdessen wird eine Fehlermeldung ausgegeben.

Bemerkungen

Der SQL Remote (dbremote)-Nachrichtenagent speichert Nachrichtenverbindungsparameter, die der Benutzer in das Fenster für den Nachrichtenagenten eingibt, wenn die Nachrichtenverbindung das erste Mal verwendet wird. In diesem Fall ist es nicht nötig, diese Anweisung explizit zu verwenden. Diese Anweisung ist besonders von Nutzen, wenn eine konsolidierte Datenbank für das Extrahieren von zahlreichen Datenbanken vorbereitet wird.

Bei den Optionsnamen wird die Groß-/Kleinschreibung berücksichtigt. Ob die Optionswerte auf Groß-/Kleinschreibung reagieren, hängt von der Option ab: Boolesche Werte reagieren nicht, während Kennwörter, Verzeichnisnamen und andere Zeichenfolgen abhängig von der Reaktion auf Groß-/Kleinschreibung des Dateisystems (bei Verzeichnisnamen) oder der Datenbank (bei Benutzer-IDs und Kennwörtern) sind.

Privilegien

Der Publikationseigentümer kann seine eigenen Optionen setzen. Andernfalls müssen Sie die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Sammeln von Fehlern aus der entfernten Datenbank“ [[SQL Remote](#)]
- „Festlegen von Steuerungsparametern für den entfernten Nachrichtentyp“ [[SQL Remote](#)]
- „Benutzerdefinierte Kodierungsschemata“ [[SQL Remote](#)]
- „SET OPTION-Anweisung“ auf Seite 1039
- „Das FTP-Nachrichtensystem“ [[SQL Remote](#)]
- „Das FILE-Nachrichtensystem“ [[SQL Remote](#)]
- „Das HTTP-Nachrichtensystem“ [[SQL Remote](#)]
- „Praktische Einführung: Einrichten eines Replikationssystems unter Verwendung des HTTP-Nachrichtensystems“ [[SQL Remote](#)]
- „Das SMTP-Nachrichtensystem“ [[SQL Remote](#)]
- „CREATE CERTIFICATE-Anweisung“ auf Seite 582
- „SET REMOTE OPTION-Anweisung [[SQL Remote](#)]“ auf Seite 1046

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Die folgende Anweisung legt den FTP-Host für die FTP-Verbindung des Benutzers Sam_Singer auf *ftp.mycompany.com* fest:

```
SET REMOTE FTP OPTION Sam_Singer.host = 'ftp.mycompany.com';
```

Die folgende Anweisung verhindert, dass SQL Remote die angegebenen Dateierweiterungen für Meldungen verwendet, die generiert werden:

```
SET REMOTE FTP OPTION  
"Public"."invalid_extensions"='exe,pif,dll,bat,cmd,vbs';
```

Die folgende Anweisung legt die URL so fest, dass sie auf den Localhost für die HTTP-Verbindung des Benutzers Sam_Singer zeigt:

```
SET REMOTE HTTP OPTION Sam_Singer.url='localhost:8033';
```

Die folgende Anweisung legt die HTTP-URL so fest, dass sie auf einen Relay Server verweist, der die Anforderung an die SRHTTP-Farm weiterleitet:

```
SET REMOTE HTTP OPTION "public"."url"='iis7.company.com:80/rs/client/  
rs_client.dll/srhttp';
```

SET SQLCA-Anweisung [ESQL]

Weist den SQL-Präprozessor an, einen anderen SQLCA als den standardmäßigen globalen *sqlca* zu verwenden.

Syntax

```
SET SQLCA sqlca
```

sqlca : *identifier* | *string*

Bemerkungen

Die SET SQLCA-Anweisung weist den SQL-Präprozessor an, einen anderen SQLCA zu verwenden als den standardmäßigen globalen *sqlca*. Der *sqlca* muss ein Bezeichner oder eine Zeichenfolge sein, der bzw. die eine C-Sprachenreferenz zu einem SQLCA-Zeiger ist.

Der aktuelle SQLCA-Zeiger wird bei jeder Embedded SQL-Anweisung implizit an die Interface-Bibliothek der Datenbank übergeben. Alle Embedded SQL-Anweisungen, die in der C-Quelldatei auf diese Anweisung folgen, verwenden den neuen SQLCA.

Diese Anweisung ist nur notwendig, wenn Sie einen Code schreiben, der ablaufinvariant ist.

Der *sqlca* sollte eine lokale Variable referenzieren. Jede globale oder statische Variable in einem Modul kann von einem anderen Thread geändert werden.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „SQLCA-Verwaltung für Code mit mehreren Threads oder "reentrant"-Code“ [[SQL Anywhere Server - Programmierung](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die Eigentümerfunktion könnte in einer Windows DLL enthalten sein. Jede Anwendung, die diese DLL verwendet, hat ihren eigenen SQLCA.

```
an_sql_code FAR PASCAL ExecutesQL( an_application *app, char *com )
{
    EXEC SQL BEGIN DECLARE SECTION;
    char *sqlcommand;
    EXEC SQL END DECLARE SECTION;
    EXEC SQL SET SQLCA "&app->.sqlca";
    sqlcommand = com;
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL EXECUTE IMMEDIATE :sqlcommand;
    return( SQLCODE );
}
```

SET-Anweisung

Weist einer SQL-Variablen einen Wert zu.

Syntax

SET *identifier* = *expression*

Bemerkungen

Die SET-Anweisung weist einer Variablen einen neuen Wert zu. Die Variable muss zuvor mit einer CREATE VARIABLE-Anweisung oder einer DECLARE-Anweisung erstellt worden sein, oder ein OUTPUT-Parameter einer Prozedur sein. Die Variable kann optional die Transact-SQL-Konvention eines @-Zeichens vor einem Namen verwenden. Beispiel: `SET @localvar = 42`

Eine Variable kann in einer SQL-Anweisung überall dort verwendet werden, wo ein Spaltenname zugelassen ist. Wenn ein Spaltenname mit demselben Namen wie der der Variablen besteht, wird der Wert der Variablen verwendet.

Variablen gehören lokal zur aktuellen Verbindung und verschwinden, sobald Sie die Verbindung zur Datenbank trennen oder die DROP VARIABLE-Anweisung verwenden. Sie werden von den Anweisungen COMMIT oder ROLLBACK nicht beeinflusst.

Variablen werden für die Erstellung von umfangreichen Text- oder Binärobjekten für die INSERT- oder UPDATE -Anweisungen von Embedded SQL-Programmen benötigt, da Embedded SQL-Hostvariablen auf 32767 Byte begrenzt sind.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „CREATE VARIABLE-Anweisung“ auf Seite 771
- „DECLARE-Anweisung“ auf Seite 786
- „SET-Anweisung [T-SQL]“ auf Seite 1056
- „DROP VARIABLE-Anweisung“ auf Seite 839
- „Ausdrücke“ auf Seite 22
- „Hostvariablen verwenden“ [*SQL Anywhere Server - Programmierung*]

Standards und Kompatibilität

- **SQL/2008** Die SET-Anweisung ist Teil der optionalen SSQL/2008-Sprachenfunktion P002 (Verarbeitungsvollständigkeit).
- **Transact-SQL** Die SET-Anweisung wird von Adaptive Server Enterprise unterstützt. In ASE kann eine einzelne SET-Anweisung verwendet werden, um Werte mehreren Variablen zuzuordnen, wobei die einzelnen Zuordnungsklauseln durch Kommas getrennt werden.

Beispiel

Dieses einfache Beispiel zeigt die Erstellung einer Variablen namens 'birthday' und verwendet SET, um das Datum auf CURRENT DATE zu setzen.

```
CREATE VARIABLE @birthday DATE;  
SET @birthday = CURRENT DATE;
```

Mit dem folgenden Codefragment wird ein großer Textwert in die Datenbank eingefügt.

```
EXEC SQL BEGIN DECLARE SECTION;
DECL_VARCHAR( 5000 ) buffer;
/* Note: maximum DECL_VARCHAR size is 32765 */
EXEC SQL END DECLARE SECTION;

EXEC SQL CREATE VARIABLE hold_blob LONG VARCHAR;
EXEC SQL SET hold_blob = '';
for(;;) {
    /* read some data into buffer ... */
    size = fread( buffer, 1, 5000, fp );
    if( size <= 0 ) break;
    /* Does not work if data contains null chars */
    EXEC SQL SET hold_blob = hold_blob || :buffer;
}
EXEC SQL INSERT INTO some_table VALUES( 1, hold_blob );
EXEC SQL DROP VARIABLE hold_blob;
```

Mit dem folgenden Codefragment wird ein großer Binärwert in die Datenbank eingefügt.

```
EXEC SQL BEGIN DECLARE SECTION;
DECL_BINARY( 5000 ) buffer;
EXEC SQL END DECLARE SECTION;

EXEC SQL CREATE VARIABLE hold_blob LONG BINARY;
EXEC SQL SET hold_blob = '';
for(;;) {
    /* read some data into buffer ... */
    size = fread( &(buffer.array), 1, 5000, fp );
    if( size <= 0 ) break;
    buffer.len = size;
    /* add data to blob using concatenation */
    EXEC SQL SET hold_blob = hold_blob || :buffer;
}
EXEC SQL INSERT INTO some_table VALUES ( 1, hold_blob );
EXEC SQL DROP VARIABLE hold_blob;
```

SET-Anweisung [T-SQL]

Legt Datenbankoptionen für die aktuelle Verbindung auf Adaptive Server Enterprise-kompatible Art fest.

Syntax

SET *option-name option-value*

Bemerkungen

Die folgenden Optionen stehen zur Verfügung:

Name der Option	Wert der Option
ansinull	On oder Off
ansi_permissions	On oder Off
close_on_endtrans	On oder Off

Name der Option	Wert der Option
datefirst	1, 2, 3, 4, 5, 6 oder 7 Die Einstellung dieser Option beeinflusst das Ergebnis von DATEPART beim Abrufen eines Wochentagwerts.
quoted_identifier	On Off
rowcount	<i>Ganzzahl</i>
self_recursion	On Off
string_truncation	On Off
textsize	<i>Ganzzahl</i>
transaction isolation level	0, 1, 2, 3, snapshot, statement-snapshot oder readonly-statement-snapshot

Datenbankoptionen werden in SQL Anywhere mit der SET OPTION-Anweisung eingestellt. SQL Anywhere unterstützt jedoch auch die in Adaptive Server Enterprise verwendete SET-Anweisung für Optionen, die für die Kompatibilität nützlich sind.

Die folgenden Optionen können mithilfe der Transact-SQL-Anweisung SET in SQL Anywhere und in Adaptive Server Enterprise gesetzt werden:

- **SET ansinull** Das standardmäßige Verhalten zum Vergleichen von Werten mit NULL in SQL Anywhere und Adaptive Server Enterprise ist unterschiedlich. Wenn ansinull auf OFF gesetzt wird, können Transact-SQL-kompatible Vergleiche mit NULL ausgeführt werden.

SQL Anywhere unterstützt auch die folgende Syntax:

SET ansi_nulls { On | Off }

- **SET ansi_permissions** Das Standardverhalten in Bezug auf erforderliche Privilegien zum Ausführen von UPDATE oder DELETE mit einer Spaltenreferenz ist in SQL Anywhere und Adaptive Server Enterprise unterschiedlich. Wenn ansi_permissions auf OFF gesetzt wird, ergeben sich Transact-SQL-kompatible Privilegien für UPDATE und DELETE.
- **SET close_on_endtrans** Das Standardverhalten in SQL Anywhere und Adaptive Server Enterprise zum Schließen von Cursors am Ende einer Transaktion ist unterschiedlich. Wenn close_on_endtrans auf OFF gesetzt wird, ergibt sich ein Transact-SQL-kompatibles Verhalten.
- **SET datefirst** Der Standardwert ist 7, d.h., der erste Tag der Woche ist standardmäßig Sonntag.
- **SET quoted_identifier** Damit wird gesteuert, ob Zeichenfolgen in Anführungszeichen als Bezeichner (On) oder als Literale (Off) interpretiert werden.
- **SET rowcount** *Ganzzahl*. Die Transact-SQL-Option ROWCOUNT begrenzt die Anzahl der Zeilen, die für einen beliebigen Cursor abgerufen werden, auf die angegebene Ganzzahl. Dazu

gehören Zeilen, die durch Neupositionierung des Cursors abgerufen werden. Jegliche Abrufe über dieses Maximum hinaus führen zu einer Warnmeldung. Die Optionseinstellung wird berücksichtigt, wenn die Schätzung der Zeilenanzahl für einen Cursor bei einer OPEN-Anforderung zurückgegeben wird.

SET ROWCOUNT begrenzt auch die Anzahl der Zeilen, die von einer UPDATE- oder DELETE-Anweisung betroffen sind, auf eine *Ganzzahl*. Dies kann z.B. eingesetzt werden, um zuzulassen, dass COMMIT-Anweisungen in regelmäßigen Intervallen ausgeführt werden, um die Größe des Rollback-Logs und der Sperrentabelle einzuschränken. Die Anwendung (bzw. die Prozedur) müsste mit einer Schleife ausgeführt werden, damit UPDATE/DELETE für die Zeilen erneut ausgeführt werden, die vom ersten Vorgang nicht betroffen sind. Es folgt ein einfaches Beispiel:

```
BEGIN
  DECLARE @count INTEGER
  SET rowcount 20
  WHILE(1=1) BEGIN
    UPDATE GROUPO.Employees SET Surname='new_name'
    WHERE Surname <> 'old_name'
    /* Stop when no rows changed */
    SELECT @count = @@rowcount
    IF @count = 0 BREAK
    PRINT string('Updated ',
                @count, ' rows; repeating...')
    COMMIT
  END
  SET rowcount 0
END
```

Wenn die ROWCOUNT-Einstellung größer ist als die Anzahl der Zeilen, die Interactive SQL anzeigen kann, führt Interactive SQL möglicherweise einige zusätzliche Abrufvorgänge aus, um den Cursor neu zu positionieren. So kann die Anzahl der tatsächlich angezeigten Zeilen niedriger sein als die angeforderte Anzahl. Wenn darüber hinaus Zeilen aufgrund von Kürzungswarnungen erneut abgerufen werden, kann die Zählung ungenau sein.

Ein Wert von Null setzt die Option zurück, sodass alle Zeilen abgerufen werden.

- **SET self_recursion** Die Option self_recursion wird innerhalb von Triggern eingesetzt, um zu ermöglichen (On) bzw. zu verhindern (Off), dass Vorgänge in der dem Trigger zugeordneten Tabelle andere Trigger auslösen.
- **SET string_rtruncation** Das Standardverhalten in SQL Anywhere und Adaptive Server Enterprise ist unterschiedlich, wenn beim Zuordnen von SQL-Textdaten Zeichen gekürzt werden, die keine Leerstellen sind. Wenn string_rtruncation auf On gesetzt wird, ergeben sich Transact-SQL-kompatible Zeichenfolgenvergleiche.
- **SET textsize** Dient zur Angabe der maximalen Größe (in Byte) für Daten vom Typ Text oder Bild, die mit einer SELECT-Anweisung zurückgegeben werden sollen. In der globalen Variablen @@textsize wird die derzeitige Einstellung gespeichert. Mit dem folgenden Befehl wird die Standardgröße (32 kB) wiederhergestellt:
- **SET transaction isolation level** Legt die Sperr-Isolationsstufe für die aktuelle Verbindung fest.

Bei Adaptive Server Enterprise sind nur 1 und 3 gültige Optionen. Bei SQL Anywhere sind 0, 1, 2, 3, snapshot, statement-snapshot oder readonly-statement-snapshot gültige Optionen.

Die SET-Anweisung ist in SQL Anywhere für die prefetch-Option aus Kompatibilitätsgründen zulässig, hat aber keine Auswirkungen.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „ansinull-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Isolationsstufen und Konsistenz“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Optionen für die Transact-SQL-Kompatibilität“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Kompatibilitätsoptionen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „isolation_level-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „ansi_permissions-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „close_on_endtrans-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „quoted_identifier-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „string_rtruncation-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „first_day_of_week-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „DATEPART-Funktion [Datum und Uhrzeit]“ auf Seite 221
- „SET OPTION-Anweisung“ auf Seite 1039

Standards und Kompatibilität

- **SQL/2008** Transact-SQL-Erweiterung.

SETUSER-Anweisung

Erlaubt es einem Benutzer, die Identität eines anderen autorisierten Benutzers anzunehmen (diesen zu impersonieren).

Syntax

```
{ SETUSER | SET SESSION AUTHORIZATION }
[ [ WITH OPTION ] userid ]
```

Parameter

SETUSER oder SET SESSION AUTHORIZATION SETUSER und SET SESSION AUTHORIZATION sind semantisch gleichwertig. Beachten Sie jedoch, dass der von Ihnen eingegebene Wert für SETUSER als Bezeichner formatiert sein muss (z.B. SETUSER JoeS oder SETUSER "JoeS"), der Wert für SET SESSION AUTHORIZATION dagegen als Zeichenfolge (z.B. SETUSER 'JoeS').

WITH OPTION-Klausel Während einer Impersonierungssitzung weichen Datenbankoptionseinstellungen für den Impersonierer möglicherweise von denjenigen für *userid* ab, was Auswirkungen auf die Ergebnisse haben kann. Geben Sie WITH OPTION an, um die Datenbankoptionen so zu ändern, dass sie die für *userid* geltenden Optionen widerspiegeln.

Bemerkungen

Die SETUSER-Anweisung dient administrativen Zwecken und sollte nicht für das Verbindungspooling verwendet werden. Nachdem Sie eine SETUSER-Anweisung ausgeführt haben, können Sie mit einem der folgenden Befehle überprüfen, welche Benutzerautorisierung Sie verwendet haben:

- SELECT USER
- SELECT CURRENT USER

SETUSER ohne Angabe einer Benutzer-ID macht alle vorherigen SETUSER-Anweisungen rückgängig.

Eine erfolgreiche Impersonierung bleibt wirksam, bis sie manuell beendet wird (durch Ausführen einer SETUSER-Anweisung ohne ID) oder die Sitzung beendet wird.

Die Anweisung SETUSER ist in gespeicherten Prozeduren, Triggern, Ereignisverarbeitungsroutinen oder Batches nicht zulässig.

Zu den zahlreichen Anwendungsmöglichkeiten der SETUSER-Anweisung gehören auch die folgenden:

- **Objekte erstellen** Mit SETUSER können Sie ein Datenbankobjekt erstellen, dessen Eigentümer ein anderer Benutzer ist.
- **Privilegienüberprüfung** Indem ein Benutzer einen anderen Benutzer impersoniert, kann er Privilegien und Namensauflösung für Abfragen, Prozeduren, Ansichten usw. überprüfen.
- **Sichere Umgebung für Administratoren bereitstellen** Der Datenbankadministrator ist berechtigt, Handlungen jeder Art in der Datenbank vorzunehmen. Wenn Sie sicherstellen möchten, dass Sie nicht versehentlich eine unbeabsichtigte Aktion ausführen, können Sie mit SETUSER zu einer anderen Benutzer-ID mit weniger Privilegien wechseln.

Privilegien

Sie müssen das SET USER-Systemprivileg haben. Ob Sie jedoch eine SETUSER-Anweisung erfolgreich ausführen (eine Impersonierungssitzung starten) können, hängt davon ab, ob Sie die Mindestkriterien für die Person erfüllen, die Sie impersonieren möchten. Die SETUSER-Anweisung schlägt fehl, wenn diese Bedingung nicht erfüllt ist.

Siehe auch

- „Impersonierung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Details zu den Mindestkriterien für die Impersonierung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „EXECUTE IMMEDIATE-Anweisung [SP]“ auf Seite 843
- „GRANT-Anweisung“ auf Seite 881
- „REVOKE-Anweisung“ auf Seite 1009
- „SET OPTION-Anweisung“ auf Seite 1039
- „Bezeichner“ auf Seite 4
- „Zeichenfolgen“ auf Seite 6

Standards und Kompatibilität

- **SQL/2008** Die SET SESSION AUTHORIZATION-Syntax ist Teil der optionalen SQL/2008-Sprachenfunktion F321 (Benutzerautorisierung). Die SETUSER-Syntax ist eine Erweiterung des

Herstellers. Sie können die WITH OPTION-Syntax bei beiden Varianten verwenden, aber WITH OPTION ist eine Erweiterung des Herstellers.

Beispiel

Mit der ersten Anweisung in diesem Beispiel (SETUSER "Joe") impersoniert ein Benutzer, der das SET USER-Systemprivileg hat, den Benutzer Joe, um einige Vorgänge mit dessen Privilegien auszuführen. Mit der zweiten Anweisung (SETUSER WITH OPTION "Jane") impersoniert der Benutzer Jane, um einige Vorgänge mit deren Privilegien und den derzeit für sie geltenden Datenbankoptionen auszuführen. Mit der dritten Anweisung (SETUSER) kehrt der Benutzer zu seiner eigenen Benutzer-ID sowie den dazugehörigen Privilegien und Datenbankoptionen zurück.

```
SETUSER "Joe"  
// Some operations are run using Joes privileges ...  
SETUSER WITH OPTION "Jane"  
// Some operations are run using Jane's privileges, and the  
// database options in effect are changed to the current  
// database options for Jane  
SETUSER
```

SIGNAL-Anweisung [SP]

Signalisiert eine Ausnahmebedingung.

Syntax

SIGNAL *exception-name*

Bemerkungen

SIGNAL zeigt eine Ausnahmebedingung an.

Der *exception-name*, der mit einer DECLARE-Anweisung am Anfang der aktuellen zusammengesetzten Anweisung deklariert wurde. Die Ausnahmebedingung muss einem system-definierten SQLSTATE oder einem benutzerdefinierten SQLSTATE entsprechen. Benutzerdefinierte SQLSTATE-Werte müssen im Bereich von 99000 und 99999 liegen.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „RESIGNAL-Anweisung [SP]“ auf Seite 1000
- „BEGIN-Anweisung“ auf Seite 557
- „Ausnahmeroutinen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „RAISERROR-Anweisung“ auf Seite 982

Standards und Kompatibilität

- **SQL/2008** Die SIGNAL-Anweisung ist Teil der optionalen SQL/2008-Sprachenfunktion P002 (Verarbeitungsvollständigkeit).
- **Transact-SQL** Die SIGNAL-Anweisung kann nicht in zusammengesetzten Anweisungen und Prozeduren in Transact-SQL verwendet werden.

Beispiel

Die folgende zusammengesetzte Anweisung deklariert eine benutzerdefinierte zusammengesetzte Anweisung und zeigt sie an. Wenn Sie dieses Beispiel von Interactive SQL ausführen, wird die Meldung **My exception signaled** auf der Registerkarte **Meldungen** im Bereich **Ergebnisse** angezeigt.

```
BEGIN
  DECLARE myexception EXCEPTION
  FOR SQLSTATE '99001';
  SIGNAL myexception;
  EXCEPTION
    WHEN myexception THEN
      MESSAGE 'My exception signaled'
      TO CLIENT;
END
```

START DATABASE-Anweisung

Startet eine Datenbank auf dem aktuellen Datenbankserver.

Syntax

START DATABASE *database-file* [*start-options ...*]

start-options :

```
[ AS database-name ]
[ WITH TRUNCATE AT CHECKPOINT ]
[ FOR READ ONLY ]
[ AUTOSTOP { ON | OFF } ]
[ KEY key ]
[ WITH SERVER NAME alternative-database-server-name ]
[ DIRECTORY dbspace-directory ]
[ CHECKSUM { ON | OFF } ]
[ DISKSANDBOX { ON | OFF | DEFAULT } ]
[ MIRROR ON ]
```

Parameter

database-file Der Parameter *database-file* ist eine Zeichenfolge. Wenn ein relativer Pfad in *database-file* angegeben wird, ist er relativ zum Startverzeichnis des Datenbankservers.

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

start-options-Klausel Die *start-options*-Klauseln können in beliebiger Reihenfolge aufgelistet sein:

- **AS-Klausel** Wenn *database-name* nicht angegeben ist, wird der Datenbank ein Standardname zugeordnet. Dieser Standardname ist der Stamm der Datenbankdatei. Zum Beispiel würde eine Datenbank in der Datei *C:\Database Files\demo.db* den Standardnamen *demo* erhalten. Der Parameter *database-name* ist ein Bezeichner.
- **WITH TRUNCATE AT CHECKPOINT-Klausel** Startet eine Datenbank mit aktivierter Logkürzung am Checkpoint.
- **FOR READ ONLY-Klausel** Startet eine Datenbank im schreibgeschützten Modus. Wenn diese Klausel mit einer Datenbank verwendet wird, die einer Wiederherstellung bedarf, schlägt die Anweisung fehl und die Fehlermeldung wird zurückgegeben.
- **AUTOSTOP-Klausel** Die Standardeinstellung für die AUTOSTOP-Klausel ist ON. Wenn AUTOSTOP auf ON eingestellt ist, wird die Datenbank entladen, sobald die letzte Verbindung gelöscht wird. Wenn AUTOSTOP auf OFF eingestellt ist, wird die Datenbank nicht entladen.

In Interactive SQL können Sie YES oder NO als Alternativen zu ON und OFF verwenden.
- **KEY-Klausel** Wenn die Datenbank stark verschlüsselt ist, geben Sie den Wert KEY (Kennwort) mit dieser Klausel ein.
- **WITH SERVER NAME-Klausel** Verwenden Sie diese Klausel, um bei der Verbindungsaufnahme zu dieser Datenbank einen alternativen Namen für den Datenbankserver anzugeben.

Hinweis

Bei einer Datenbankspiegelung muss ein alternativer Servername angegeben werden, damit Clientanwendungen in der Lage sind, sich mit dem aktuellen Primärserver zu verbinden, ohne im Voraus zu wissen, welcher Server der Primärserver und welcher der Spiegelserver ist. Beide Partnerserver müssen denselben Namen als alternativen Servernamen benutzen. Verwenden Sie die CREATE MIRROR SERVER-Anweisung, um die alternativen Servernamen für den Primär- und Spiegelserver in einem Spiegelungssystem zu erstellen.

- **DIRECTORY-Klausel** Verwenden Sie diese Klausel, um das Verzeichnis anzugeben, in dem sich die DBSpace-Dateien für die Datenbank befinden, die gestartet wird. Beispiel: Wenn der Datenbankserver in demselben Verzeichnis wie die DBSpaces gestartet wird und Sie die DIRECTORY ' . ' -Klausel aufnehmen, wird der Datenbankserver angewiesen, alle DBSpaces im aktuellen Verzeichnis zu suchen.

Hinweis

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „[Sandboxing](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

- **CHECKSUM-Klausel** Verwenden Sie diese Klausel, um Prüfsummen für neu geschriebene Seiten für Datenbanken zu aktivieren, die nicht mit aktivierten globalen Prüfsummen erstellt wurden. Diese Klausel hat dasselbe Verhalten wie die Datenbankoption -wc.

Der Unterschied zwischen der CHECKSUM-Klausel und dem Erstellen einer Datenbank mit aktivierten globalen Prüfsummen ist folgender: Wenn Sie CHECKSUM ON angeben, werden

Prüfsummen für Datenbankseiten nur beim Schreiben auf die Festplatte berechnet. Seiten, die von der Festplatte gelesen werden, werden nur überprüft, wenn ein Prüfsummenwert berechnet wurde, bevor die Seiten geschrieben wurden. Wenn für eine Datenbank globale Prüfsummen aktiviert sind, werden Prüfsummen für alle Seiten während des Schreibens berechnet und während des Lesens überprüft.

Falls der Datenbankserver feststellt, dass die Datenbank unter Windows Mobile oder auf einem Wechseldatenträger ausgeführt wird, z.B. einer Netzwerkfreigabe oder einem USB-Gerät, aktiviert der Datenbankserver automatisch Schreib-Prüfsummen für alle Datenbankseiten.

Standardmäßig sind für Datenbanken, die mit Version 10 und 11 von SQL Anywhere erstellt wurden, keine globalen Prüfsummen aktiviert. Wenn Sie eine Datenbank, die mit SQL Anywhere 10 oder 11 erstellt wurde, auf einem Datenbankserver der Version 12 oder später starten, erstellt der Datenbankserver standardmäßig Schreib-Prüfsummen für Seiten, wenn diese auf die Festplatte geschrieben werden (CHECKSUM ON). Datenbanken der Version 12 oder später haben globale Prüfsummen standardmäßig aktiviert, sodass der Datenbankserver bei diesen Datenbanken die Standardeinstellung CHECKSUM OFF verwendet, weil standardmäßig alle Datenbankseiten Prüfsummen haben. Sie können entweder die Option -wc oder die START DATABASE-Anweisung verwenden, um das Prüfsummenverhalten des Datenbankservers zu ändern, wenn Sie nicht die Standardeinstellungen für Prüfsummen verwenden wollen.

Sie können überprüfen, ob eine Datenbank mit aktivierten globalen Prüfsummen erstellt wurde, indem Sie die folgende Anweisung ausführen:

```
SELECT DB_PROPERTY ( 'Checksum' );
```

Sie können überprüfen, ob Schreib-Prüfsummen aktiviert sind, indem Sie die folgende Anweisung ausführen:

```
SELECT DB_PROPERTY ( 'WriteChecksum' );
```

- **DISKSANDBOX-Klausel** Setzen Sie DISKSANDBOX auf ON, um Dateivorgänge der Datenbank mit Lese-/Schreibzugriff auf das Verzeichnis zu beschränken, in dem sich die Hauptdatenbankdatei befindet. Setzen Sie DISKSANDBOX auf OFF, um den Zugriff auf alle Verzeichnisse zu erlauben. Wenn DISKSANDBOX auf DEFAULT gesetzt ist, werden die durch die Datenbankserveroption -sbx festgelegten Sandboxing-Einstellungen verwendet. Siehe „[Datenbankserveroption -sbx](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Hinweis

Wenn Sie einen Datenbankserver mit der Datenbankserveroption -sbx starten, müssen Sie den Schlüssel für die gesicherte Funktion manage_disk_sandbox angeben, um eine Datenbank mit DISKSANDBOX OFF starten zu können.

- **MIRROR ON-Klausel** Verwenden Sie die MIRROR ON-Klausel, um eine zusätzliche gespiegelte Datenbank zu einem Datenbankserver hinzuzufügen, der bereits läuft und möglicherweise bereits als Host einer gespiegelten Datenbank fungiert. Sie müssen die AUTOSTOP OFF-Klausel angeben, wenn Sie diese Klausel verwenden.

Bemerkungen

Startet eine bestimmte Datenbank auf dem aktuellen Datenbankserver

Die Anweisung `START DATABASE` baut keine Verbindung zwischen der aktuellen Anwendung und der angegebenen Datenbank auf: Die Verbindung muss explizit ausgeführt werden.

Wenn Sie nicht mit einer Datenbank verbunden sind und die `START DATABASE`-Anweisung verwenden wollen, müssen Sie sich zuerst mit einer Datenbank, z.B. mit der Dienstprogrammdatenbank, verbinden.

Sie können nur den Datenbanknamen `utility_db` verwenden, um eine Verbindung mit der Dienstprogrammdatenbank herzustellen.

Privilegien

Jeder Benutzer kann eine Datenbank auf dem Personal Server (dbeng16) starten.

Welche Privilegien zum Starten einer Datenbank auf einem Netzwerkserver erforderlich sind, wird durch die Datenbankserveroption `-gd` festgelegt. Standardmäßig ist das `SERVER OPERATOR`-Systemprivileg erforderlich, um eine Datenbank auf dem Netzwerkserver starten zu können.

Nebenwirkungen

Keine.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Siehe auch

- „`STOP DATABASE`-Anweisung“ auf Seite 1074
- „`CREATE MIRROR SERVER`-Anweisung“ auf Seite 656
- „`CONNECT`-Anweisung [ESQL] [Interactive SQL]“ auf Seite 578
- „`VALIDATE`-Anweisung“ auf Seite 1118
- „Datenbankserveroption `-gd`“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankoption `-ds`“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankoption `-ek`“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankoption `-m`“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankoption `-r`“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankoption `-n`“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankoption `-wc`“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankoption `-sn`“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankspiegelungssysteme einrichten“ [*SQL Anywhere Server - Datenbankadministration*]
- „Die Dienstprogrammdatenbank“ [*SQL Anywhere Server - Datenbankadministration*]
- „Erkennung von Beschädigungen mithilfe von Prüfsummen“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

In diesem Beispiel wird eine fiktive Datenbankdatei `C:\temp\sample_2.db` auf dem aktuellen Server gestartet:

```
START DATABASE 'c:\\temp\\sample_2.db';
```

In diesem Beispiel wird dieselbe Datenbank gestartet, jedoch als "sam2":

```
START DATABASE 'c:\\temp\\sample_2.db'  
AS sam2;
```

START SERVER-Anweisung [Interactive SQL]

Startet einen Datenbankserver.

Syntax

START SERVER AS *database-server-name* [**STARTLINE** *command-string*]

Bemerkungen

Die START SERVER-Anweisung startet einen Datenbankserver. Wenn Sie Optionen für den Datenbankserver angeben möchten, müssen Sie das STARTLINE-Schlüsselwort zusammen mit einer Befehlszeichenfolge verwenden.

START ENGINE wird aus Kompatibilitätsgründen akzeptiert, aber nicht mehr empfohlen.

Diese SQL-Anweisung wird nicht für SAP HANA-Datenbanken unterstützt.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „STOP SERVER-Anweisung“ auf Seite 1077
- „Syntax des SQL Anywhere-Datenbankservers“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird ein Datenbankserver namens sample gestartet, ohne darauf Datenbanken zu starten.

```
START SERVER AS sample;
```

Dieses Beispiel zeigt die Verwendung der STARTLINE-Klausel.

```
START SERVER AS eng1 STARTLINE 'dbsrv16 -c 8M';
```

START EXTERNAL ENVIRONMENT-Anweisung

Startet eine externe Umgebung.

Syntax

START EXTERNAL ENVIRONMENT *environment-name*

environment-name :

```

JAVA
PERL
PHP
CLR
C_ESQL32
C_ESQL64
C_ODBC32
C_ODBC64

```

Parameter

environment-name Der Name der externen Umgebung, die gestartet werden soll.

Bemerkungen

Weitere Hinweise zu externen Umgebungen finden Sie unter „[Unterstützung für externe Umgebungen in SQL Anywhere](#)“ [[SQL Anywhere Server - Programmierung](#)].

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „[Unterstützung für externe Umgebungen in SQL Anywhere](#)“ [[SQL Anywhere Server - Programmierung](#)]
- „[ALTER EXTERNAL ENVIRONMENT-Anweisung](#)“ auf Seite 463
- „[STOP EXTERNAL ENVIRONMENT-Anweisung](#)“ auf Seite 1075
- „[INSTALL EXTERNAL OBJECT-Anweisung](#)“ auf Seite 923
- „[REMOVE EXTERNAL OBJECT-Anweisung](#)“ auf Seite 996
- „[SYSEXTERNENV-Systemansicht](#)“ auf Seite 1448
- „[SYSEXTERNENVOBJECT-Systemansicht](#)“ auf Seite 1450

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Startet die externe Perl-Umgebung.

```
START EXTERNAL ENVIRONMENT PERL;
```

START JAVA-Anweisung

Startet die Java VM.

Syntax

START JAVA

Bemerkungen

Die Anweisung START JAVA startet die Java VM. Diese Anweisung wird hauptsächlich zum Laden der Java VM zu einem geeigneten Zeitpunkt verwendet, damit keine Unterbrechung durch das Laden der Java VM entsteht, wenn der Benutzer beginnt, Java-Funktionen zu verwenden.

Der Datenbankserver muss so eingerichtet sein, dass er eine Java VM finden kann. Da Sie verschiedene Java VMs bei jeder Datenbank angeben können, kann die `java_location`-Option verwendet werden, um den Speicherort (Pfad) der Java VM anzuzeigen.

Eine Java VM muss installiert sein.

Diese Anweisung wird unter Windows Mobile nicht unterstützt.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „STOP JAVA-Anweisung“ auf Seite 1076
- „Java VM starten und stoppen“ [[SQL Anywhere Server - Programmierung](#)]
- „java_location-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird die Java VM gestartet.

```
START JAVA;
```

START LOGGING-Anweisung [Interactive SQL]

Startet die Protokollierung von ausgeführten SQL-Anweisungen in einer Logdatei.

Syntax

START LOGGING *filename*

Bemerkungen

Die Anweisung START LOGGING beginnt mit dem Kopieren aller folgenden ausgeführten SQL-Anweisungen in eine von Ihnen angegebene Logdatei. Wenn die Datei nicht existiert, wird sie von Interactive SQL erstellt. Die Protokollierung wird so lange fortgeführt, bis Sie den

Protokollierungsprozess explizit mit der STOP LOGGING-Anweisung stoppen oder bis Sie die aktuelle Interactive SQL-Sitzung beenden.

Sie können die Protokollierung auch starten bzw. stoppen, indem Sie auf Folgendes klicken:
SQL » Protokollierung starten bzw. **SQL » Protokollierung stoppen**.

Die Ausführungszeiten sind im Log enthalten, wenn die Funktionen für die Protokollierung und die Berichterstellung über die Ausführungszeit aktiviert sind.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „STOP LOGGING-Anweisung [Interactive SQL]“ auf Seite 1077
- „isql_command_timing-Option [Interactive SQL]“ [*SQL Anywhere Server - Datenbankadministration*]
- „Anweisungen in Interactive SQL protokollieren“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird die Protokollierung in der Datei `c:\temp\filename.sql` gestartet:

```
START LOGGING 'c:\\temp\\filename.sql';
```

START SUBSCRIPTION-Anweisung [SQL Remote]

Startet eine Subskription einer Publikation für einen Benutzer.

Syntax

```
START SUBSCRIPTION  
TO publication-name [ ( subscription-value ) ]  
FOR subscriber-id, ...
```

Parameter

publication-name Der Name der Publikation, die für den Benutzer subskribiert ist. Darin kann der Eigentümer der Publikation enthalten sein.

subscription-value Eine Zeichenfolge, die mit dem Subskriptionsausdruck der Publikation verglichen wird. Dieser Wert ist hier erforderlich, da jeder Subskribent mehr als eine Subskription für eine Publikation haben kann.

subscriber-id Die Benutzer-ID des Subskribenten für die Publikation. Der Benutzer muss eine Subskription für die Publikation haben.

Bemerkungen

Eine SQL Remote-Subskription wird gestartet, wenn die Publikationsaktualisierungen von der konsolidierten Datenbank an die entfernte Datenbank gesendet werden.

Die START SUBSCRIPTION-Anweisung ist eine aus einer Reihe von Anweisungen, die Subskriptionen verwalten. Die CREATE SUBSCRIPTION-Anweisung definiert die Daten, die der Subskribent empfangen soll. Die SYNCHRONIZE SUBSCRIPTION-Anweisung stellt sicher, dass die konsolidierten und entfernten Datenbanken konsistent sind. Die START SUBSCRIPTION-Anweisung ist erforderlich, damit Nachrichten, die an den Subskribenten versendet werden, gestartet werden können.

Die Daten auf beiden Seiten der Subskription müssen konsistent sein, bevor die Subskription gestartet wird. Es wird empfohlen, dass Sie das Extraktionsdienstprogramm für Datenbanken verwenden, um das Erstellen, Synchronisieren und das Starten von Subskriptionen zu verwalten. Wenn Sie das Extraktionsdienstprogramm für Datenbanken verwenden, müssen Sie keine START SUBSCRIPTION-Anweisung explizit ausführen. Außerdem startet der Nachrichtenagent Subskriptionen, sobald sie synchronisiert sind.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 730
- „REMOTE RESET-Anweisung [SQL Remote]“ auf Seite 995
- „SYNCHRONIZE SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 1086
- „STOP SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 1078
- „Extraktionsdienstprogramm (dbxtract)“ [*SQL Remote*]
- „Starten von Subskriptionen“ [*SQL Remote*]
- „Mehrere entfernte Datenbanken erstellen“ [*SQL Remote*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung startet die Subskription des Benutzers Sam_Singer für die pub_contact-Publikation.

```
START SUBSCRIPTION TO pub_contact  
FOR Sam_Singer;
```

START SYNCHRONIZATION DELETE-Anweisung [MobiLink]

Startet die Protokollierung von Löschungen für die MobiLink-Synchronisation

Syntax

START SYNCHRONIZATION DELETE

Bemerkungen

Normalerweise protokollieren SQL Anywhere und UltraLite automatisch alle Änderungen in Tabellen oder Spalten, die Teil einer Synchronisation sind, und laden diese Änderungen während der nächsten Synchronisation in die konsolidierte Datenbank hoch. Sie können die automatische Protokollierung von Löschvorgängen vorübergehend aufheben, indem Sie die Anweisung STOP SYNCHRONIZATION DELETE verwenden. Mit der Anweisung START SYNCHRONIZATION DELETE können Sie die automatische Protokollierung erneut starten.

Bei der Ausführung einer STOP SYNCHRONIZATION DELETE-Anweisung wird kein Löschvorgang synchronisiert, der über diese Verbindung ausgeführt wird. Die Wirkung hält so lange an, bis eine START SYNCHRONIZATION DELETE-Anweisung ausgeführt wird. Die Wiederholung der Anweisung STOP SYNCHRONIZATION DELETE hat keine zusätzliche Auswirkung.

Eine einzelne START SYNCHRONIZATION DELETE-Anweisung startet die Protokollierung erneut, unabhängig von der Anzahl der vorhergehenden STOP SYNCHRONIZATION DELETE-Anweisungen.

Verwenden Sie START SYNCHRONIZATION DELETE nicht, wenn Ihre Anwendung keine Daten synchronisiert.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Keine.

Siehe auch

- „STOP SYNCHRONIZATION DELETE-Anweisung [MobiLink]“ auf Seite 1079
- [ULConnection.StartSynchronizationDelete-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Mit der folgenden Sequenz von SQL-Anweisungen wird veranschaulicht, wie START SYNCHRONIZATION DELETE und STOP SYNCHRONIZATION DELETE eingesetzt werden:

```
-- Prevent deletes from being sent
-- to the consolidated database
```

```
STOP SYNCHRONIZATION DELETE;

-- Remove all records older than 1 month
-- from the remote database,
-- NOT the consolidated database
DELETE FROM PROPOSAL
WHERE last_modified < months( CURRENT_TIMESTAMP, -1 )

-- Re-enable all deletes to be sent
-- to the consolidated database
-- DO NOT FORGET to start this
START SYNCHRONIZATION DELETE;

-- Commit the entire operation,
-- otherwise rollback everything
-- including the stopping of the deletes
commit;
```

START SYNCHRONIZATION SCHEMA CHANGE- Anweisung [MobiLink]

Startet eine Änderung des MobiLink-Synchronisationsschemas.

Syntax

START SYNCHRONIZATION SCHEMA CHANGE
FOR TABLES *table-list*

set-script-version
| *set-script-version-on-subscription*, ...

set-script-version :
SET SCRIPT VERSION = *script-version*

set-script-version-on-subscription :
SET SCRIPT VERSION = *script-version* **ON SUBSCRIPTION** *subscription_name*

script-version : *string* | **NULL**

subscription-name : *identifier*

Parameter

FOR TABLES-Klausel Diese Klausel gibt die Tabellen an, die von der Schemaänderung betroffen sind.

SET SCRIPT VERSION-Klausel Gibt die neue Skriptversion für alle Subskriptionen an, die eine beliebige in der FOR TABLES-Klausel angegebene Tabelle enthalten. Die neue Skriptversion kann mit der vorhandenen Skriptversion übereinstimmen.

SET SCRIPT VERSION ... ON SUBSCRIPTION-Klausel Gibt die neue Skriptversion für die angegebene Subskription an. Wenn diese Klausel verwendet wird, muss sie, durch Kommata getrennt, für jede Subskription wiederholt werden, die eine in der FOR TABLES-Klausel angegebene Tabelle enthält. Die neue Skriptversion kann mit der vorhandenen Skriptversion übereinstimmen.

Bemerkungen

Alle Tabellen, auf die Sie eine Schemaänderung anwenden möchten, müssen in der *table-list* aufgeführt werden. Eine Tabelle kann nicht mehr als einmal aufgeführt werden. Eine Fehlermeldung erscheint, wenn eine Sperre für eine der Tabellen in *table-list* vorhanden ist.

In einer Datenbank kann jeweils nur eine Änderung des Synchronisationsschemas ausgeführt werden. Die START SYNCHRONIZATION SCHEMA CHANGE-Anweisung schlägt fehl, wenn eine andere Schemaänderung ausgeführt wird.

Der Datenbankserver setzt Sperren für alle in der *table-list* angegebenen Tabellen. Der Datenbankserver ignoriert beim Setzen der Sperren die Einstellung für die BLOCKING-Option. Wenn eine Sperre nicht gesetzt werden kann, werden alle zuvor gesetzten Sperren freigegeben und eine Fehlermeldung wird angezeigt.

Während einer Änderung des Synchronisationsschemas gilt:

- Sie können keine Datenmanipulationsanweisung ausführen.
- Sie können keine weiteren START SYNCHRONIZATION SCHEMA CHANGE-Anweisungen ausführen.
- Sie können eine Publikation ändern, um die Erstellung von Spaltenteilmengen für eine beliebige Tabelle in der *table-list* zu ändern.
- Sie können eine Publikation ändern, um eine beliebige Tabelle aus der *table-list* zu löschen.
- Sie können eine beliebige Tabelle in der *table-list* ändern.

Ein implizites COMMIT erfolgt sowohl vor als auch nach der START SYNCHRONIZATION SCHEMA CHANGE-Anweisung. Eine Änderung des Synchronisationsschemas endet mit der Ausführung einer STOP SYNCHRONIZATION SCHEMA CHANGE-Anweisung. Beim Ausführen der STOP SYNCHRONIZATION SCHEMA CHANGE-Anweisung werden alle Tabellensperren freigegeben.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Keine.

Siehe auch

- „STOP SYNCHRONIZATION SCHEMA CHANGE-Anweisung [MobiLink]“ auf Seite 1085
- SynchronizationSchemaChangeActive-Datenbankeigenschaft [SQL Anywhere Server - Datenbankadministration]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Mit der folgenden Sequenz von SQL-Anweisungen wird veranschaulicht, wie START SYNCHRONIZATION SCHEMA CHANGE und STOP SYNCHRONIZATION SCHEMA CHANGE eingesetzt werden:

```
START SYNCHRONIZATION SCHEMA CHANGE
FOR TABLES GROUPO.SalesOrders, GROUPO.Products
SET SCRIPT VERSION = 'version_2' ON SUBSCRIPTION sub1,
SET SCRIPT VERSION = 'version_2' ON SUBSCRIPTION sub2;
ALTER TABLE GROUPO.SalesOrders ADD SUBTOTAL NUMERIC (10,2);
ALTER TABLE GROUPO.Products ALTER QUANTITY BIGINT;
STOP SYNCHRONIZATION SCHEMA CHANGE;
```

STOP DATABASE-Anweisung

Stoppt eine bestimmte Datenbank auf dem aktuellen Datenbankserver.

Syntax

```
STOP DATABASE database-name
[ ON database-server-name ]
[ UNCONDITIONALLY ]
```

Parameter

STOP DATABASE-Klausel Der *database-name* ist der Name einer Datenbank (nicht die aktuelle Datenbank), die auf dem aktuellen Server läuft.

ON-Klausel Diese Klausel wird nur von Interactive SQL unterstützt. Wenn *database-server-name* in Interactive SQL nicht angegeben ist, werden alle laufenden Server nach der Datenbank mit dem angegebenen Namen durchsucht.

Wird diese Anweisung nicht in Interactive SQL verwendet, wird die Datenbank nur gestoppt, wenn sie auf dem aktuellen Datenbankserver gestartet wurde.

UNCONDITIONALLY-Klausel Stoppt die Datenbank, auch wenn Verbindungen zur Datenbank bestehen. Standardmäßig wird die Datenbank nicht angehalten, wenn Verbindungen zu ihr bestehen.

Bemerkungen

Die STOP DATABASE-Anweisung hält eine angegebene Datenbank auf dem aktuellen Datenbankserver an.

Sie können STOP DATABASE nicht auf die Datenbank anwenden, mit der Sie gerade verbunden sind.

Privilegien

Jeder Benutzer kann eine Datenbank auf dem Personal Server stoppen.

Welche Privilegien zum Stoppen einer Datenbank auf dem Netzwerkserver erforderlich sind, wird durch die Datenbankserveroption -gd bestimmt. Standardmäßig ist zum Stoppen einer Datenbank auf dem Netzwerkserver das SERVER OPERATOR-Systemprivileg erforderlich.

Nebenwirkungen

Keine

Siehe auch

- „START DATABASE-Anweisung“ auf Seite 1062
- „DISCONNECT-Anweisung [ESQL] [Interactive SQL]“ auf Seite 802
- „Datenbankserveroption -gd“ [*SQL Anywhere Server - Datenbankadministration*]
- „Serverstopp-Dienstprogramm (dbstop)“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die Datenbank namens *sample* wird auf dem aktuellen Server gestoppt.

```
STOP DATABASE sample;
```

STOP EXTERNAL ENVIRONMENT-Anweisung

Stoppt eine externe Umgebung.

Syntax

STOP EXTERNAL ENVIRONMENT *environment-name*

environment-name :

```
JAVA
| PERL
| PHP
| CLR
| C_ESQL32
| C_ESQL64
| C_ODBC32
| C_ODBC64
```

Parameter

environment-name Der Name der externen Umgebung, die beendet werden soll.

Bemerkungen

Weitere Hinweise zu externen Umgebungen finden Sie unter „Unterstützung für externe Umgebungen in SQL Anywhere“ [*SQL Anywhere Server - Programmierung*].

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Unterstützung für externe Umgebungen in SQL Anywhere“ [[SQL Anywhere Server - Programmierung](#)]
- „ALTER EXTERNAL ENVIRONMENT-Anweisung“ auf Seite 463
- „START EXTERNAL ENVIRONMENT-Anweisung“ auf Seite 1066
- „INSTALL EXTERNAL OBJECT-Anweisung“ auf Seite 923
- „REMOVE EXTERNAL OBJECT-Anweisung“ auf Seite 996
- „SYSEXTERNENV-Systemansicht“ auf Seite 1448
- „SYSEXTERNENVOBJECT-Systemansicht“ auf Seite 1450

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird die externe Perl-Umgebung gestoppt.

```
STOP EXTERNAL ENVIRONMENT PERL;
```

STOP JAVA-Anweisung

Stoppt die Java VM.

Syntax

STOP JAVA

Bemerkungen

Mit der Anweisung STOP JAVA wird die Java VM entladen, wenn sie nicht mehr benötigt wird. Die Anweisung wird hauptsächlich verwendet, um nicht benötigte Systemressourcen freizugeben.

Diese Anweisung wird unter Windows Mobile nicht unterstützt.

Privilegien

Keine.

Nebenwirkungen

Keine

Siehe auch

- „START JAVA-Anweisung“ auf Seite 1067

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesem Beispiel wird die Java VM gestoppt.

```
STOP JAVA;
```

STOP LOGGING-Anweisung [Interactive SQL]

Stoppt das Protokollieren von SQL-Anweisungen in der aktuellen Sitzung.

Syntax

STOP LOGGING

Bemerkungen

Die Anweisung STOP LOGGING bewirkt, dass Interactive SQL nicht mehr jede SQL-Anweisung, die Sie ausführen, in eine Logdatei schreibt. Sie können die Protokollierung mit der Anweisung START LOGGING starten.

Sie können die Protokollierung auch stoppen, indem Sie auf **SQL » Protokollierung stoppen** klicken.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „START LOGGING-Anweisung [Interactive SQL]“ auf Seite 1068
- „Anweisungen in Interactive SQL protokollieren“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Mit dem folgenden Beispiel wird die aktuelle Protokollsitzung gestoppt.

```
STOP LOGGING;
```

STOP SERVER-Anweisung

Stoppt einen Datenbankserver.

Syntax

STOP SERVER [*database-server-name*] [**UNCONDITIONALLY**]

Parameter

UNCONDITIONALLY-Klausel Wenn Sie die einzige Verbindung zum Datenbankserver sind, müssen Sie UNCONDITIONALLY nicht verwenden. Wenn es noch andere Verbindungen gibt, stoppt der Datenbankserver nur dann, wenn Sie das Schlüsselwort UNCONDITIONALLY verwenden.

Bemerkungen

Sie können nur in Interactive SQL *database-server-name* verwenden. Wenn Sie diese Anweisung nicht in Interactive SQL ausführen, wird der aktuelle Datenbankserver gestoppt.

Standardmäßig wird der Datenbankserver nicht angehalten, wenn andere Verbindungen zu ihm bestehen. Wenn die UNCONDITIONALLY-Klausel verwendet wird, wird der Datenbankserver angehalten, auch wenn andere Verbindungen zum Datenbankserver bestehen. Wenn die STOP SERVER-Anweisung auf einer Clientverbindung ausgeführt und der Server erfolgreich beendet wird, tritt ein Verbindungsfehler auf.

Die STOP SERVER-Anweisung kann nicht in gespeicherten Prozeduren, Triggern, Ereignissen oder Batches verwendet werden.

STOP ENGINE wird aus Kompatibilitätsgründen akzeptiert, aber nicht mehr empfohlen.

Privilegien

Jeder Benutzer kann den Personal Server (dbeng16) herunterfahren.

Welche Privilegien zum Herunterfahren eines Netzwerkservers (dbsrv16) erforderlich sind, wird durch die Einstellung -gk in der Befehlszeile des Datenbankservers bestimmt. Standardmäßig ist das SERVER OPERATOR-Systemprivileg erforderlich, um einen Netzwerkservers herunterzufahren.

Nebenwirkungen

Keine

Siehe auch

- „START SERVER-Anweisung [Interactive SQL]“ auf Seite 1066
- „Datenbankserveroption -gk “ [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Der aktuelle Datenbankserver wird gestoppt, wenn keine anderen Verbindungen bestehen.

```
STOP SERVER;
```

STOP SUBSCRIPTION-Anweisung [SQL Remote]

Stoppt eine Subskription einer Publikation für einen Benutzer.

Syntax

```
STOP SUBSCRIPTION  
TO publication-name [ ( subscription-value ) ]  
FOR subscriber-id, ...
```

Parameter

publication-name Der Name der Publikation, die für den Benutzer subskribiert ist. Darin kann der Eigentümer der Publikation enthalten sein.

subscription-value Eine Zeichenfolge, die mit dem Subskriptionsausdruck der Publikation verglichen wird. Dieser Wert ist hier erforderlich, da jeder Subskribent mehr als eine Subskription für eine Publikation haben kann.

subscriber-id Die Benutzer-ID des Subskribenten für die Publikation. Der Benutzer muss eine Subskription für die Publikation haben.

Bemerkungen

Eine SQL Remote-Subskription wird gestartet, wenn die Publikationsaktualisierungen von der konsolidierten Datenbank an die entfernte Datenbank gesendet werden.

Die STOP SUBSCRIPTION-Anweisung verhindert, dass weitere Nachrichten an den Subskribenten gesendet werden. Die START SUBSCRIPTION-Anweisung ist erforderlich, um Nachrichten, die an den Subskribenten gesendet werden, erneut zu starten. Sie sollten aber darauf achten, dass die Subskription korrekt synchronisiert wurde, bevor der Neustart durchgeführt wird: Damit gehen Sie sicher, dass keine Nachrichten verloren gegangen sind.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „DROP SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 828
- „START SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 1069
- „SYNCHRONIZE SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 1086
- „Extraktionsdienstprogramm (dbxtract)“ [SQL Remote]
- „Stoppen von Subskriptionen“ [SQL Remote]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung stoppt die Subskription des Benutzers SamS für die pub_contact-Publikation.

```
STOP SUBSCRIPTION TO pub_contact  
FOR Sam_Singer;
```

STOP SYNCHRONIZATION DELETE-Anweisung [MobiLink]

Stoppt die Protokollierung von Löschungen für die MobiLink-Synchronisation vorübergehend

Syntax

STOP SYNCHRONIZATION DELETE

Bemerkungen

Normalerweise protokollieren entfernte SQL Anywhere- und UltraLite-Datenbanken automatisch alle Änderungen in Tabellen oder Spalten, die synchronisiert werden, und laden diese Änderungen während der nächsten Synchronisation in die konsolidierte Datenbank hoch. Mit dieser Anweisung können Sie die Protokollierung von Löschvorgängen in einer entfernten SQL Anywhere- oder UltraLite-Datenbank vorübergehend aufheben.

Keiner der Löschvorgänge, die auf einer Verbindung zwischen den Ausführungszeitpunkten von STOP SYNCHRONIZATION DELETE und START SYNCHRONIZATION DELETE ausgeführt werden, wird synchronisiert.

Eine einzelne START SYNCHRONIZATION DELETE-Anweisung startet die Protokollierung erneut, unabhängig von der Anzahl der vorhergehenden STOP SYNCHRONIZATION DELETE-Anweisungen.

Diese Anweisung kann hilfreich sein, wenn Sie Korrekturen in einer entfernten Datenbank vornehmen. Sie sollte aber mit Vorsicht verwendet werden, da sie die MobiLink-Synchronisation praktisch deaktiviert.

Verwenden Sie STOP SYNCHRONIZATION DELETE nicht, wenn Ihre Anwendung keine Daten synchronisiert.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Keine.

Siehe auch

- [ULConnection.StartSynchronizationDelete-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)
- [ULConnection.StopSynchronizationDelete-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)
- „START SYNCHRONIZATION DELETE-Anweisung [MobiLink]“ auf Seite 1071

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Mit der folgenden Sequenz von SQL-Anweisungen wird veranschaulicht, wie START SYNCHRONIZATION DELETE und STOP SYNCHRONIZATION DELETE eingesetzt werden:

```
-- Prevent deletes from being sent
-- to the consolidated database
STOP SYNCHRONIZATION DELETE;
```

```
-- Remove all records older than 1 month
-- from the remote database,
-- NOT the consolidated database
DELETE FROM PROPOSAL
WHERE last_modified < months( CURRENT_TIMESTAMP, -1 )

-- Re-enable all deletes to be sent
-- to the consolidated database
-- DO NOT FORGET to start this
START SYNCHRONIZATION DELETE;

-- Commit the entire operation,
-- otherwise rollback everything
-- including the stopping of the deletes
commit;
```

SYNCHRONIZE-Anweisung [MobiLink]

Synchronisiert eine SQL Anywhere-Datenbank mit einem MobiLink-Server. Die Synchronisationsoptionen können in der Anweisung selbst angegeben werden.

Syntax

```
SYNCHRONIZE {
PROFILE sync-profile-name [ MERGE sync-option [ ;... ] ]
| USING sync-option [ ;... ]
| START
| STOP
}

[ PORT port-number ]
[ VERBOSITY { LOW | NORMAL | HIGH } ]
[ TIMEOUT timeout ]
[ USER user-name IDENTIFIED BY password ]
```

sync-option : *string*

Parameter

sync-profile-name Der Name des Synchronisationsprofils, das für diese Synchronisation verwendet werden soll.

MERGE-Klausel Verwenden Sie diese Klausel, um Synchronisationsprofiloptionen hinzuzufügen oder zu überschreiben.

USING-Klausel Mit dieser Klausel geben Sie Synchronisationsprofiloptionen an, wenn Sie kein Synchronisationsprofil verwenden.

sync-option Eine Zeichenfolge mit mindestens einem Paar "Synchronisationsprofiloption=Wert", getrennt durch Semikola. Beispiel: 'option1=value1;option2=value2'.

PORT-Klausel Verwenden Sie diese Klausel, um die Portnummer anzugeben, die der Datenbankserver für die Kommunikation mit dem Dienstprogramm dbmlsync verwendet. Standardwert ist "4433".

VERBOSITY-Klausel Diese Klausel steuert den Umfang der Informationen, die den gemeinsam genutzten globalen temporären Tabellen "synchronize_results" und "synchronize_parameters" während der Synchronisation hinzugefügt werden.

Im Folgenden finden Sie eine Liste der Client-API-Ereignisse, die von den einzelnen VERBOSITY-Optionen zurückgegeben werden.

Option	Rückgabe
LOW	<ul style="list-style-type: none"> ○ DBSC_EVENTTYPE_SYNC_START ○ DBSC_EVENTTYPE_SYNC_DONE ○ DBSC_EVENTTYPE_ERROR_MSG ○ DBSC_EVENTTYPE_WARNING_MSG
NORMAL (Standard)	<ul style="list-style-type: none"> ○ DBSC_EVENTTYPE_SYNC_START ○ DBSC_EVENTTYPE_SYNC_DONE ○ DBSC_EVENTTYPE_ERROR_MSG ○ DBSC_EVENTTYPE_WARNING_MSG ○ DBSC_EVENTTYPE_INFO_MSG
HIGH	<ul style="list-style-type: none"> ○ DBSC_EVENTTYPE_SYNC_START ○ DBSC_EVENTTYPE_SYNC_DONE ○ DBSC_EVENTTYPE_ERROR_MSG ○ DBSC_EVENTTYPE_WARNING_MSG ○ DBSC_EVENTTYPE_INFO_MSG ○ DBSC_EVENTTYPE_PROGRESS_INDEX ○ DBSC_EVENTTYPE_PROGRESS_TEXT ○ DBSC_EVENTTYPE_TITLE

Achten Sie darauf, die VERBOSITY-Klausel der SYNCHRONIZE-Anweisung nicht mit der VERBOSITY-Option zu verwechseln, die Sie in einem Synchronisationsprofil angeben können. Die VERBOSITY-Klausel der SYNCHRONIZE-Anweisung steuert den Typ der Ereignisse, die in den Tabellen synchronize_results und synchronize_parameters aufgezeichnet werden können. Die VERBOSITY-Optionen in einem Synchronisationsprofil steuern die Anzahl der DBSC_EVENTTYPE_INFO_MSG-Ereignisse, die während der Synchronisation generiert werden.

```
SYNCHRONIZE PROFILE SalesData VERBOSITY NORMAL;
```

```
SYNCHRONIZE PROFILE SalesData MERGE 'Verbosity=BASIC,ROW_DATA' VERBOSITY  
NORMAL;
```

TIMEOUT-Klausel Diese Klausel gibt an, wie lange (in Sekunden) der Datenbankserver warten soll, bis die Synchronisation abgeschlossen wird, bevor sie abgebrochen wird. Der Standardwert beträgt 240 Sekunden.

USER / IDENTIFIED BY-Klausel Verwenden Sie diese Klausel, um die Datenbankbenutzer-ID und das Kennwort anzugeben, die das Dienstprogramm dbmlsync verwendet, um die Datenbank zu synchronisieren. Die angegebene Benutzer-ID muss die SYS_RUN_REPLICATION_ROLE-Systemrolle haben. Standardmäßig verwendet die Synchronisation die Benutzer-ID für die Datenbankverbindung, die die SYNCHRONIZE-Anweisung ausgeführt hat.

START-Klausel Startet das im Servermodus ausgeführte Dienstprogramm dbmsync und lässt es laufen. Es wird keine Synchronisation durchgeführt. Wenn Sie mehr als eine Synchronisation innerhalb kurzer Zeit durchführen, können Sie die Performance verbessern, indem Sie den dbmsync-Server mit dieser Klausel explizit starten, die Synchronisationen vornehmen und den dbmsync-Server anschließend mit der STOP-Klausel explizit stoppen.

STOP-Klausel Stoppt einen dbmsync-Server, der zuvor mit der START-Klausel gestartet wurde. Es wird keine Synchronisation durchgeführt.

Bemerkungen

Diese Anweisung kann nur verwendet werden, wenn der MobiLink-Client für SQL Anywhere einschließlich Dbmsync C++-API installiert ist.

Der MobiLink-Client für SQL Anywhere steht nicht auf allen Plattformen zur Verfügung, auf denen der Datenbankserver möglicherweise ausgeführt wird. Eine Liste der unterstützten Plattformen finden Sie unter <http://www.sybase.com/detail?id=1061806>.

Wenn die Synchronisation abgeschlossen ist, können Sie die Ergebnisse der Synchronisation in der gemeinsamen globalen temporären Tabelle `synchronize_results` und `synchronize_parameters` anzeigen. Die gemeinsamen globalen temporären Tabellen `synchronize_results` und `synchronize_parameters` speichern die Ergebnisse aller Synchronisationen, die seit dem Start des Datenbankservers mit der SYNCHRONIZE-Anweisung ausgeführt wurden. Die Tabellen `synchronize_results` und `synchronize_parameters` werden jedes Mal gekürzt, wenn der Datenbankserver heruntergefahren wird.

Die Tabelle `synchronize_results` enthält folgende Spalten:

Spaltenname	Datentyp	Beschreibung
<code>row_id</code>	UNSIGNED BIGINT	Der Primärschlüssel der Tabelle, mit dessen Hilfe die Reihenfolge bestimmt wird, in der Zeilen in die Tabelle eingefügt wurden.
<code>conn_id</code>	UNSIGNED INT	Die Verbindungs-ID-Nummer der Verbindung, für die die SYNCHRONIZE-Anweisung ausgeführt wird, die dieses Ereignis generiert hat.
<code>result_time</code>	TIMESTAMP	Der Zeitpunkt, zu dem das Ereignis der Tabelle <code>synchronize_results</code> hinzugefügt wurde.
<code>result_type</code>	CHAR(128)	Der Typ des Ereignisses.

Jedes Ereignis in der Tabelle `synchronize_results` hat 0 oder mehr Parameter zugeordnet, die zusätzliche Informationen über das Ereignis enthalten. Die Parameter werden in der Tabelle `synchronize_parameters` gespeichert, die folgende Spalten enthält:

Spaltenname	Datentyp	Beschreibung
row_id	UNSIGNED BIGINT	Ein Fremdschlüssel für die row_id-Spalte in der synchronize_results-Tabelle. Verwenden Sie diesen Wert, um die einzelnen Parameter wieder dem Ereignis zuzuordnen, zu dem sie gehören.
parm_id	UNSIGNED INT	Die numerische ID des Parameters. Verwenden Sie diesen Wert bei Ereignissen mit mehr als einem Parameter, um den jeweils benötigten Parameter zu finden.
parm_message	LONG VARCHAR	Der dem Parameter zugeordnete Wert.

Um Details früherer oder aktueller Synchronisationen anzuzeigen, können Sie die Systemprozedur "sp_get_last_synchronize_result" als Alternative zum direkten Abfragen der Tabellen "synchronize_results" und "synchronize_parameters" verwenden.

Alternativ können Sie auch die folgende Anweisung verwenden, um die Ergebnisse aller Synchronisationen anzuzeigen, die seit dem Start des Datenbankservers durchgeführt wurden.

```
SELECT *
  FROM synchronize_results sr
  KEY JOIN synchronize_parameters sp
  ORDER BY sr.row_id , sp.parm_id
```

Sie können mithilfe der gemeinsam genutzten globalen temporären Tabellen synchronize_results und synchronize_parameters den Fortschritt einer Synchronisation auf einer Verbindung überwachen, die sich von der aktuellen Verbindung unterscheidet. So überwachen Sie den Fortschritt einer Synchronisation auf einer anderen Verbindung:

- Führen Sie eine SELECT CONNECTION_PROPERTY-Anweisung aus, um die Verbindungs-ID der aktuellen Verbindung zu bestimmen.
- Führen Sie eine SYNCHRONIZE-Anweisung, um die Synchronisation zu starten.
- Verwenden Sie für separate Verbindungen die sp_get_last_synchronize_results-Systemprozedur, um Ergebnisse mit der Verbindungs-ID, die sie zuvor ermittelt haben, abzurufen.

Um die Ergebnisse einer Synchronisation anzuzeigen, die auf einer bestimmten Verbindung abgeschlossen oder aktiv ist, können Sie die sp_get_last_synchronize_results-Systemprozedur verwenden.

Die SYNCHRONIZE-Anweisung ist ähnlich wie die SYNCHRONIZE-Anweisung von UltraLite. Die SYNCHRONIZE-Anweisung von SQL Anywhere startet jedoch zum Ausführen der Synchronisation das Dienstprogramm dbmsync im Servermodus. Die SYNCHRONIZE-Anweisung von UltraLite verwendet die UltraLite-Laufzeitbibliothek.

Der Datenbankserver fungiert als dbmsync-API-Client und verwendet TCP/IP für die Kommunikation mit einem dbmsync-Server. Standardmäßig erfolgt diese Kommunikation auf Port 4433. Verwenden Sie die PORT-Klausel, um einen anderen Port anzugeben.

Verwenden Sie die Anweisungen `SYNCHRONIZE PROFILE` und `SYNCHRONIZE USING`, um eine Synchronisation durchzuführen. Verwenden Sie die Anweisungen `SYNCHRONIZE START` und `SYNCHRONIZE STOP`, um einen dbmlsync-Server zu starten bzw. zu stoppen. Bei der Ausführung einer `SYNCHRONIZE PROFILE`- oder `SYNCHRONIZE USING`-Anweisung versucht der Datenbankserver, eine Verbindung mit einem dbmlsync-Server herzustellen, der bereits ausgeführt wird. Wenn kein bereits laufender dbmlsync-Server gefunden werden kann, wird ein dbmlsync-Server gestartet. Wenn die Synchronisation abgeschlossen ist, fährt der Datenbankserver den gestarteten dbmlsync-Server herunter. Wenn die Anweisung eine Verbindung mit einem bereits laufenden dbmlsync-Server hergestellt hat, wird der dbmlsync-Server nicht heruntergefahren. Wenn Sie mehrere Synchronisationen durchführen und nicht möchten, dass bei jeder Synchronisation der dbmlsync-Server gestartet und gestoppt wird, können Sie eine `SYNCHRONIZE START`-Anweisung ausführen, gefolgt von einer `SYNCHRONIZE PROFILE`- oder `SYNCHRONIZE USING`-Anweisung, und den Vorgang mit einer `SYNCHRONIZE STOP`-Anweisung beenden.

Privilegien

Sie müssen entweder das `MANAGE REPLICATION`-Systemprivileg oder die `SYS_RUN_REPLICATION_ROLE`-Systemrolle haben.

Nebenwirkungen

Keine

Siehe auch

- „`CREATE SYNCHRONIZATION PROFILE`-Anweisung [MobiLink]“ auf Seite 732
- Ereignisse und `parm_id`-Werte aus der `synchronize_parameters`-Tabelle auf Seite 1370
- `DBSC_Event`-Struktur [Dbmlsync .NET] [MobiLink - Clientadministration]
- „MobiLink-Synchronisation“ [MobiLink - Erste Orientierung]
- „MobiLink-Synchronisationsprofile“ [MobiLink - Clientadministration]
- „Bereitstellung von SQL Anywhere MobiLink-Clients“ [MobiLink - Serveradministration]
- „`sp_get_last_synchronize_result`-Systemprozedur“ auf Seite 1368

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel zeigt die Syntax für das Synchronisieren eines Synchronisationsprofils namens `Test1`:

```
SYNCHRONIZE PROFILE Test1;
```

STOP SYNCHRONIZATION SCHEMA CHANGE-Anweisung [MobiLink]

Beendet eine Änderung des MobiLink-Synchronisationsschemas.

Syntax

STOP SYNCHRONIZATION SCHEMA CHANGE

Bemerkungen

Die STOP SYNCHRONIZATION SCHEMA CHANGE-Anweisung beendet eine Schemaänderung, die durch eine START SYNCHRONIZATION SCHEMA CHANGE-Anweisung gestartet wurde. Alle von der START SYNCHRONIZATION SCHEMA CHANGE-Anweisung gesetzten Sperren werden freigegeben.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Keine.

Siehe auch

- „START SYNCHRONIZATION SCHEMA CHANGE-Anweisung [MobiLink]“ auf Seite 1072
- „SYSARTICLE-Systemansicht“ auf Seite 1438
- SynchronizationSchemaChangeActive-Datenbankeigenschaft [SQL Anywhere Server - Datenbankadministration]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Mit der folgenden Sequenz von SQL-Anweisungen wird veranschaulicht, wie START SYNCHRONIZATION SCHEMA CHANGE und STOP SYNCHRONIZATION SCHEMA CHANGE eingesetzt werden:

```
START SYNCHRONIZATION SCHEMA CHANGE
FOR TABLES GROUPO.SalesOrders, GROUPO.Products
SET SCRIPT VERSION = 'version_2' ON SUBSCRIPTION sub1,
SET SCRIPT VERSION = 'version_2' ON SUBSCRIPTION sub2;
ALTER TABLE GROUPO.SalesOrders ADD SUBTOTAL NUMERIC (10,2);
ALTER TABLE GROUPO.Products ALTER QUANTITY BIGINT;
STOP SYNCHRONIZATION SCHEMA CHANGE;
```

SYNCHRONIZE SUBSCRIPTION-Anweisung [SQL Remote]

Synchronisiert eine Subskription einer Publikation für einen Benutzer.

Syntax

```
SYNCHRONIZE SUBSCRIPTION
TO publication-name [ ( subscription-value ) ]
FOR remote-user, ...
```

Parameter

publication-name Der Name der Publikation, die für den Benutzer subskribiert ist. Darin kann der Eigentümer der Publikation enthalten sein.

subscription-value Eine Zeichenfolge, die mit dem Subskriptionsausdruck der Publikation verglichen wird. Dieser Wert ist hier erforderlich, da jeder Subskribent mehr als eine Subskription für eine Publikation haben kann.

remote-user Die Benutzer-ID des Subskribenten für die Publikation. Der Benutzer muss eine Subskription für die Publikation haben.

Bemerkungen

Eine SQL Remote-Subskription gilt als **synchronisiert**, wenn die Daten in der entfernten Datenbank mit denjenigen in der konsolidierten Datenbank konsistent sind, sodass Publikationsaktualisierungen, die von der konsolidierten Datenbank an die entfernte Datenbank gesendet werden, nicht zu Konflikten und Fehlern führen.

Um eine Subskription zu synchronisieren, wird eine Kopie der Daten in der Publikation in der konsolidierten Datenbank an die entfernte Datenbank versendet. Die SYNCHRONIZE SUBSCRIPTION-Anweisung führt dies über das Nachrichtensystem durch. Es wird empfohlen, stattdessen, wenn möglich, das Extraktionsdienstprogramm für Datenbanken (dbxtract) zu verwenden, um die Subskription ohne die Verwendung eines Nachrichtensystems zu synchronisieren.

Privilegien

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „CREATE SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 730
- „START SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 1069
- „STOP SUBSCRIPTION-Anweisung [SQL Remote]“ auf Seite 1078
- „Extraktionsdienstprogramm (dbxtract)“ [SQL Remote]
- „Synchronisieren mit dem SQL Remote-Nachrichtenagenten (dbremote)“ [SQL Remote]
- „Resynchronisation von Subskriptionen“ [SQL Remote]
- „Subskriptionen synchronisieren“ [SQL Remote]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung synchronisiert die Subskription des Benutzers Sam_Singer für die pub_contact-Publikation.

```
SYNCHRONIZE SUBSCRIPTION
  TO pub_contact
  FOR Sam_Singer;
```

SYSTEM-Anweisung [Interactive SQL]

Startet eine Programmdatei von Interactive SQL aus.

Syntax

SYSTEM '[*path*] *filename*'

Bemerkungen

Startet die angegebene Programmdatei. Der Pfad und der Dateiname müssen in Apostrophe eingeschlossen werden.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „CONNECT-Anweisung [ESQL] [Interactive SQL]“ auf Seite 578
- „Interactive SQL“ [*SQL Anywhere Server - Datenbankadministration*]
- „Interactive SQL-Dienstprogramm (dbisql)“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Mit der folgenden Anweisung wird das Programm Notepad gestartet, vorausgesetzt, die Programmdatei befindet sich im Suchpfad.

```
SYSTEM 'notepad.exe' ;
```

TRIGGER EVENT-Anweisung

Löst ein benanntes Ereignis aus. Das Ereignis kann für Ereignistrigger definiert oder ein geplantes Ereignis sein.

Syntax

TRIGGER EVENT *event-name* [(*parm* = *value*, ...)]

Parameter

parm = value Wenn eine Trigger-auslösende Bedingung einen Event-Handler zum Ausführen veranlasst, kann der Datenbankserver Informationen über den Inhalt an den Event-Handler mithilfe der Funktion `event_parameter` liefern. Mit der Anweisung **TRIGGER EVENT** können Sie diese Parameter explizit bereitstellen, um einen Kontext für den Event-Handler zu simulieren.

Bemerkungen

Aktionen werden durch die Anweisung CREATE EVENT an bestimmte Triggerbedingungen oder Abfolgepläne gebunden. Sie können die Anweisung TRIGGER EVENT dazu verwenden, den Event-Handler zum Ausführen zu zwingen, auch wenn die geplante Zeit nicht erreicht oder die Triggerbedingung nicht erfüllt wurde. TRIGGER EVENT führt keine deaktivierten Event-Handler aus.

Jeder *value* ist eine Zeichenfolge. Die maximale Länge jedes *value* ist auf die maximale Seitengröße beschränkt, die durch die -gp-Serveroption festgelegt wird. Wenn die Länge von *value* die Seitengröße überschreitet, wird die Zeichenfolge an der Stelle gekürzt, an der die Seite voll wird.

Privilegien

Sie müssen Eigentümer des Ereignisses sein oder das MANAGE ANY EVENT-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Datenbankserveroption -gp“ [\[SQL Anywhere Server - Datenbankadministration\]](#)
- „ALTER EVENT-Anweisung“ auf Seite 461
- „CREATE EVENT-Anweisung“ auf Seite 606
- „EVENT_PARAMETER-Funktion [System]“ auf Seite 256

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel zeigt, wie ein Zeichenfolgenparameter an ein Ereignis übergeben wird. Das Ereignis zeigt den Zeitpunkt, an dem es ausgelöst wird, im Meldungsfenster des Datenbankservers.

```
CREATE EVENT ev_PassedParameter
HANDLER
BEGIN
    MESSAGE 'ev_PassedParameter - was triggered at ' ||
event_parameter( 'time' );
END;
TRIGGER EVENT ev_PassedParameter( "Time"=string( current timestamp ) );
```

TRUNCATE-Anweisung

Löscht alle Zeilen aus einer Tabelle, ohne die Tabellendefinition zu löschen

Syntax

```
TRUNCATE
TABLE [ owner.]table-name
| MATERIALIZED VIEW [ owner.]materialized-view-name
```

Bemerkungen

Die TRUNCATE-Anweisung löscht alle Zeilen aus der Tabelle oder materialisierten Ansicht.

Hinweis

Die TRUNCATE TABLE-Anweisung sollte bei einer Datenbank, die in Synchronisationen oder Replikationen einbezogen ist, mit Umsicht verwendet werden, da die Anweisung alle Zeilen aus einer Tabelle löscht, ähnlich wie die DELETE-Anweisung ohne WHERE-Klausel. Allerdings werden aufgrund einer TRUNCATE-Anweisung keine Trigger ausgelöst. Überdies werden die einzelnen Zeilenlöschungen nicht ins Transaktionslog eingegeben und daher weder synchronisiert noch repliziert. Dies kann zu Inkonsistenzen führen, was möglicherweise ein Fehlschlagen der Synchronisation oder Replikation bewirkt.

Nach einer TRUNCATE-Anweisung bestehen das Schema des Objekts und alle Indizes solange weiter, bis Sie eine DROP-Anweisung ausführen. Die Schemadefinitionen und Integritätsregeln bleiben intakt und Trigger sowie Privilegien bleiben gültig.

table-name kann der Name einer Basistabelle oder einer temporären Tabelle sein.

Wenn bei TRUNCATE TABLE alle nachstehenden Kriterien erfüllt sind, wird eine schnelle Form der Tabellenkürzung durchgeführt:

- Es gibt keine Fremdschlüssel, weder zu noch von der Tabelle.
- Die Anweisung TRUNCATE TABLE wird nicht innerhalb eines Triggers ausgeführt.
- Die Anweisung TRUNCATE TABLE wird nicht innerhalb einer unteilbaren Anweisung ausgeführt.

Wenn eine schnelle Kürzung durchgeführt wird, werden keine einzelnen DELETE-Vorgänge im Transaktionslog registriert und ein COMMIT wird vor und nach dem Vorgang ausgeführt. Die schnelle Kürzung kann in Snapshot-Transaktionen nicht benutzt werden.

Wenn Sie versuchen, TRUNCATE TABLE mit einer Tabelle zu verwenden, für die ein Soforttextindex aufgebaut wird oder die von einer Sofortansicht referenziert wird, schlägt die Kürzung fehl. Dies kommt bei Nicht-Soforttextindizes oder materialisierten Ansichten nicht vor. Es wird aber empfohlen, die Daten in abhängigen Indizes und materialisierten Ansichten zu kürzen, bevor die TRUNCATE TABLE-Anweisung für eine Tabelle ausgeführt wird, und dann die Indizes und materialisierten Ansichten zu aktualisieren.

Für Basistabellen und materialisierten Ansichten erfordert die TRUNCATE-Anweisung exklusiven Zugriff auf die Tabelle, da es sich um einen unteilbaren Vorgang handelt (entweder werden alle Zeilen gelöscht oder gar keine). Cursor, die zuvor geöffnet wurden und die zu kürzende Tabelle referenzieren, müssen geschlossen werden und ein COMMIT oder ROLLBACK muss ausgeführt werden, damit die Tabellenreferenz freigegeben wird.

Bei temporären Tabellen hat jeder Benutzer seine eigene Datenkopie, daher ist Exklusivzugriff nicht erforderlich, wenn Sie die TRUNCATE-Anweisung ausführen.

Privilegien

Wenn Sie diese Anweisung ausführen möchten, muss eine der folgende Bedingungen erfüllt sein:

- Sie müssen Eigentümer der Tabelle sein
- Sie haben das ALTER-Privileg für die Tabelle
- Sie haben das TRUNCATE-Privileg für die Tabelle
- Sie haben das ALTER ANY TABLE-Systemprivileg
- Sie haben das TRUNCATE ANY TABLE-Systemprivileg
- Sie haben das ALTER ANY OBJECT-Systemprivileg

Nebenwirkungen

- Wenn Sie eine materialisierte Ansicht kürzen, ändern Sie den Status der Ansicht auf "nicht initialisiert".
- Delete-Trigger werden von der TRUNCATE-Anweisung nicht ausgelöst.
- Ein COMMIT wird vor und nach einer TRUNCATE-Anweisung ausgeführt.
- Individuelle Zeilenlöschungen werden nicht ins Transaktionslog eingetragen, daher wird der TRUNCATE-Vorgang nicht repliziert. Verwenden Sie diese Anweisung nicht in einer SQL Remote-Replikation oder mit einer entfernten MobiLink-Datenbank.
- Wenn die Tabelle eine als DEFAULT AUTOINCREMENT oder DEFAULT GLOBAL AUTOINCREMENT definierte Spalte enthält, setzt der Kürzungsvorgang den nächsten verfügbaren Wert für die Spalte zurück.

Siehe auch

- „DELETE-Anweisung“ auf Seite 791
- „TRUNCATE TEXT INDEX-Anweisung“ auf Seite 1091
- „Löschen aller Zeilen aus einer Tabelle“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Snapshot-Isolation“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Fortgeschrittene Aufgaben: Status und Eigenschaften von materialisierten Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** Die TRUNCATE TABLE-Anweisung ist die optionale Sprachenfunktion F200 des SQL/2008-Standards. TRUNCATE MATERIALIZED VIEW ist eine Erweiterung des Herstellers.

Beispiel

Löschen Sie alle Zeilen aus der Tabelle "SalesOrderItems":

```
TRUNCATE TABLE GROUP0.SalesOrderItems;
```

TRUNCATE TEXT INDEX-Anweisung

Löscht die Daten in einem MANUAL- oder einem AUTO REFRESH-Textindex.

Syntax

TRUNCATE TEXT INDEX *text-index-name*
ON [*owner.*] *table-name*

Parameter

ON-Klausel Der Name der Tabelle, für die der Textindex erstellt wird.

Bemerkungen

Verwenden Sie die TRUNCATE TEXT INDEX-Anweisung, wenn Sie Daten aus einem manuellen Textindex löschen möchten, ohne die Textindexdefinition zu löschen. Wenn Sie beispielsweise das Textkonfigurationsobjekt für den Textindex ändern möchten, um die Stoppliste zu ändern, müssen Sie den Textindex kürzen, das referenzierte Textkonfigurationsobjekt ändern und anschließend den Textindex aktualisieren, um ihn mit neuen Daten zu füllen.

Sie können keine TRUNCATE TEXT INDEX-Anweisung für einen Textindex ausführen, der als IMMEDIATE REFRESH (Standardvorgabe) definiert ist. Bei IMMEDIATE REFRESH-Textindizes müssen Sie stattdessen den Index löschen.

TRUNCATE TEXT INDEX erfordert exklusiven Zugriff auf die Tabelle. Geöffnete Cursor, die die zu kürzende Tabelle referenzieren, müssen geschlossen werden und eine COMMIT- oder ROLLBACK-Anweisung muss ausgeführt werden, damit die Tabellenreferenz freigegeben wird.

Privilegien

Sie müssen Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Tabelle
- ALTER ANY INDEX-Systemprivileg
- ALTER ANY TABLE-Systemprivileg

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Volltextsuche“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Konzepte und Referenz zu Textindizes“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „CREATE TEXT INDEX-Anweisung“ auf Seite 759
- „ALTER TEXT INDEX-Anweisung“ auf Seite 536
- „DROP TEXT INDEX-Anweisung“ auf Seite 834
- „REFRESH TEXT INDEX-Anweisung“ auf Seite 990

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die erste Anweisung erstellt den txt_index_manual-Textindex. Die zweite Anweisung füllt den Textindex mit Daten. Die dritte Anweisung kürzt die Textindexdaten.

```
CREATE TEXT INDEX txt_index_manual ON GROUPO.MarketingInformation
( Description )
  MANUAL REFRESH;
REFRESH TEXT INDEX txt_index_manual ON GROUPO.MarketingInformation;
TRUNCATE TEXT INDEX txt_index_manual ON GROUPO.MarketingInformation;
```

Der gekürzte Textindex wird wieder mit Daten gefüllt, wenn er das nächste Mal aktualisiert wird.

TRY-Anweisung

Implementiert Fehlerbehandlung für zusammengesetzte Anweisungen. (Wenn im TRY-Block ein Fehler auftritt, übergibt er die Steuerung an eine andere Gruppe von Anweisungen in einem CATCH-Block.)

Syntax

```
[ statement-label : ]
BEGIN TRY
  [ local-declaration; ... ]
  [ statement-list ]
END TRY
BEGIN CATCH
  [ statement-list ]
END CATCH
```

local-declaration :
variable-declaration
 | *cursor-declaration*
 | *exception-declaration*
 | *temporary-table-declaration*

variable-declaration und *exception-declaration* : Siehe „[DECLARE-Anweisung](#)“ auf Seite 786.

cursor-declaration : Siehe „[DECLARE CURSOR-Anweisung \[ESQL\] \[SP\]](#)“ auf Seite 778.

temporary-table-declaration : Siehe „[DECLARE LOCAL TEMPORARY TABLE-Anweisung](#)“ auf Seite 784.

Parameter

statement-label Wenn am Ende ein *statement-label* angegeben wird, muss es mit dem *statement-label* am Anfang übereinstimmen. Die LEAVE-Anweisung kann verwendet werden, um die Ausführung bei der ersten Anweisung nach der zusammengesetzten Anweisung wieder aufzunehmen. Die zusammengesetzte Anweisung, die den Hauptteil einer Prozedur oder eines Triggers darstellt, hat ein implizites Label, das mit dem Namen der Prozedur oder des Triggers übereinstimmt.

local-declaration Unmittelbar nach BEGIN TRY können in einer zusammengesetzten Anweisung lokale Deklarationen für Objekte folgen, die nur innerhalb der zusammengesetzten Anweisung vorhanden sind. Eine zusammengesetzte Anweisung kann eine lokale Deklaration für eine Variable, einen Cursor, eine temporäre Tabelle oder eine Ausnahmebedingung enthalten. Lokale Deklarationen können von jeder beliebigen Anweisung in dieser zusammengesetzten Anweisung oder in jeder beliebigen darin verschachtelten zusammengesetzten Anweisung referenziert werden. Lokale Deklarationen der zusammengesetzten Anweisung sind für den Ausnahmeroutinen für die Anweisung sichtbar. Lokale

Deklarationen sind für die anderen Prozeduren, die von innerhalb einer zusammengesetzten Anweisung aufgerufen werden, nicht sichtbar.

Bemerkungen

Der CATCH-Block ist die Fehlerbehandlungsroutine für die TRY-Anweisung.

TRY...CATCH-Anweisungen können verschachtelt werden und überall dort stehen, wo auch eine BEGIN...END-Anweisung verwendet werden kann. Anweisungen innerhalb des TRY-Blocks ignorieren die Datenbankoptionen `on_tsql_error` und `continue_after_raisererror` sowie die `ON EXCEPTION RESUME`-Klausel der `CREATE PROCEDURE`-Anweisung. TRY...CATCH-Anweisungen sind nicht atomar.

Wenn im TRY-Block keine Fehler auftreten, wird der CATCH-Block übersprungen und die Steuerung wird an die auf den CATCH-Block folgende Anweisung übergeben bzw. an den Aufrufer, falls keine solche Anweisung vorhanden ist. Wenn in einer der Anweisungen im TRY-Block ein Fehler auftritt, wird die Steuerung an die erste Anweisung im CATCH-Block übergeben. Nach Abschluss des CATCH-Blocks wird die Steuerung an die auf den CATCH-Block folgende Anweisung übergeben bzw. an den Aufrufer, falls keine solche Anweisung vorhanden ist. Die Auswirkungen von Anweisungen, die einer einen Fehler generierenden Anweisung vorangehen, werden nicht rückgängig gemacht, es sei denn, die Ausnahmeroutine generiert einen Fehler und ist innerhalb eines atomaren Blocks verschachtelt oder eine explizite ROLLBACK-Anweisung wird aufgerufen. In diesem Fall werden alle Anweisungen innerhalb des atomaren Transaktionsblocks zurückgesetzt.

Fehler im CATCH-Block werden entsprechend den Verbindungs- und Prozedureinstellungen behandelt, es sei denn, die Anweisungen, die sie generieren, sind von weiteren TRY...CATCH-Anweisungen umschlossen.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „ERROR_LINE-Funktion [Verschiedene]“ auf Seite 244
- „ERROR_MESSAGE-Funktion [Verschiedene]“ auf Seite 245
- „ERROR_PROCEDURE-Funktion [Funktionstyp]“ auf Seite 246
- „ERROR_SQLCODE-Funktion [Verschiedene]“ auf Seite 247
- „ERROR_SQLSTATE-Funktion [Verschiedene]“ auf Seite 248
- „ERROR_STACK_TRACE-Funktion [Verschiedene]“ auf Seite 249
- „STACK_TRACE-Funktion [Verschiedene]“ auf Seite 392
- „sa_error_stack_trace-Systemprozedur“ auf Seite 1216
- „sa_stack_trace-Systemprozedur“ auf Seite 1335
- „Verschachtelte zusammengesetzte Anweisungen und Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Behandlung von Fehlern und Warnungen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- Routine bei Ausnahmefehlern und atomare zusammengesetzte Anweisungen [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Atomare zusammengesetzte Anweisungen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „CONTINUE-Anweisung“ auf Seite 581
- „SIGNAL-Anweisung [SP]“ auf Seite 1061
- „RESIGNAL-Anweisung [SP]“ auf Seite 1000
- „RAISERROR-Anweisung“ auf Seite 982
- „BEGIN-Anweisung [TSQL]“ auf Seite 560

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

In diesen Beispiele wird die folgende Tabelle verwendet:

```
CREATE TABLE t( col1 DOUBLE );
```

Durch Ausführen der folgenden zusammengesetzten Anweisung wird der Wert 6 in die Tabelle "t" eingefügt:

```
BEGIN TRY
    DECLARE val INT;

    SET val = 0;

    INSERT INTO t VALUES( 1 / val );
    -- This statement will not be executed
    INSERT INTO t VALUES( val );
END TRY
BEGIN CATCH
    SET val = 6;
    INSERT INTO t VALUES( val );
END CATCH;
```

Durch Ausführen der folgenden Prozedur mit `CALL proc1(10);` werden die folgenden Werte in die die Tabelle "t" eingefügt:

-10
10

```
CREATE PROCEDURE DBA.proc1( v INTEGER )
BEGIN TRY
    DECLARE local_val INTEGER = 0;

    INSERT INTO t VALUES(-v);

    SET local_val = v / local_val;
    -- This statement will not be executed
    MESSAGE 'The value is ', v;
END TRY
BEGIN CATCH
    INSERT INTO t VALUES(v);
END CATCH;
```

UNION-Anweisung

Kombiniert die Ergebnisse von zwei oder mehr SELECT-Anweisungen oder Abfrageausdrücken.

Syntax

```
[ WITH temporary-views ] query-block
UNION [ ALL | DISTINCT ] query-block
[ ORDER BY [ integer | select-list-expression-name ] [ ASC | DESC ], ... ]
[ FOR XML xml-mode ]
[ OPTION( query-hint, ... ) ]
```

query-block : Siehe „Allgemeine Elemente der SQL-Syntax“ auf Seite 445.

query-hint :

```
MATERIALIZED VIEW OPTIMIZATION option-value
| FORCE OPTIMIZATION
| option-name = option-value
```

option-name : *identifier*

option-value :

```
hostvar (Bezeichner zulässig)
| string
| identifier
| number
```

Parameter

FOR XML-Klausel Hinweise zur Verwendung der FOR XML-Klausel finden Sie unter „SELECT-Anweisung“ auf Seite 1020.

OPTION-Klausel Verwenden Sie diese Klausel, um Hints für das Ausführen der Anweisung anzugeben. Die folgenden Hints werden unterstützt:

- MATERIALIZED VIEW OPTIMIZATION *option-value*
- FORCE OPTIMIZATION
- *option-name* = *option-value*. Die Spezifikation `OPTION(isolation_level = ...)` im Abfragetext hebt alle anderen Festlegungen der Isolationsstufe für eine Abfrage auf.

Bemerkungen

UNION ALL verkettet die Ergebnisse der zwei Abfrageblöcke zu einer einzigen (größeren) Ergebnismenge. Jeder Abfrageblock kann verschachtelt sein. UNION DISTINCT entfernt Duplikatzeilen aus dem endgültigen Ergebnis. Das Entfernen von Duplikaten erfordert einen zusätzlichen Verarbeitungsprozess. Daher sollte, wenn möglich, UNION ALL anstelle von UNION verwendet werden. UNION DISTINCT ist identisch mit UNION.

Die beiden *query-block*-Ergebnismengen müssen UNION-kompatibel sein. Sie müssen dieselbe Anzahl von Elementen in ihren jeweiligen Auswahllisten aufweisen und die Art der einzelnen Ausdrücke sollte vergleichbar sein. Wenn übereinstimmende Elemente in zwei Auswahllisten verschiedene Datentypen umfassen, wählt SQL Anywhere einen Datentyp für die entsprechende Spalte im Ergebnis aus und konvertiert automatisch die Spalten in jedem *query-block*.

Die angezeigten Spaltennamen sind dieselben wie beim ersten *query-block* und mithilfe dieser Namen wird bestimmt, welche Ausdrucknamen mit der ORDER BY-Klausel abgeglichen werden. Eine andere Möglichkeit zum Anpassen von Spaltennamen in der Ergebnismenge ist die Verwendung eines allgemeinen Tabellenausdrucks (der WITH-Klausel).

Privilegien

Sie müssen Eigentümer der Objekte im *query-block* sein oder das SELECT-Privileg für die Objekte haben, für die Sie die Vereinigung definieren.

Nebenwirkungen

Keine.

Siehe auch

- „EXCEPT-Anweisung“ auf Seite 841
- „INTERSECT-Anweisung“ auf Seite 927
- OPTION-Klausel, SELECT-Anweisung auf Seite 1027
- „SELECT-Anweisung“ auf Seite 1020
- „Mengenoperatoren und NULL“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Standards und Kompatibilität

- **SQL/2008** UNION ist eine Kernfunktion des SQL/2008-Standards. Das explizite Angeben des DISTINCT- Schlüsselworts mit UNION ist die optionale SQL-Sprachenfunktion T551. Das Angeben einer ORDER BY-Klausel mit UNION ist SQL-Sprachenfunktion F850. Ein *Abfrageblock* mit einer ORDER BY-Klausel entspricht SQL/2008-Funktion F851. Ein Abfrageblock, der eine Zeilenbegrenzungsklausel (SELECT TOP oder LIMIT) enthält, umfasst je nach Kontext die optionale

SQL-Sprachenfunktion F857 oder F858. Die Klauseln FOR XML und OPTION sind Erweiterungen des Herstellers.

- **Transact-SQL** UNION und UNION ALL werden von Adaptive Server Enterprise unterstützt. Die Klauseln FOR XML und OPTION werden nicht unterstützt.

Beispiel

Listen Sie alle Nachnamen (surnames) von Mitarbeitern (employees) und Kunden (customers) auf:

```
SELECT Surname
FROM GROUPO.Employees
UNION
SELECT Surname
FROM GROUPO.Customers;
```

UNLOAD-Anweisung

Entlädt Daten aus einer Datenquelle in eine Datei.

Syntax

```
UNLOAD data-source
{ TO filename
  | INTO FILE filename
  | INTO CLIENT FILE client-filename
  | INTO VARIABLE variable-name }
[ unload-option ... ]

data-source
[ FROM ] [ TABLE ] [ owner. ] table-name
| [ FROM ] [ MATERIALIZED VIEW ] [ owner. ] materialized-view-name
| select-statement

filename : string | variable

client-filename : string | variable
```

Syntax

```
unload-option :
APPEND { ON | OFF }
| BYTE ORDER MARK { ON | OFF }
| { COMPRESSED | NOT COMPRESSED }
| COLUMN DELIMITED BY string
| DELIMITED BY string
| ENCODING encoding
| { ENCRYPTED KEY 'key' [ ALGORITHM 'algorithm' ] | NOT ENCRYPTED }
| ESCAPE CHARACTER character
| ESCAPES { ON | OFF }
| FORMAT { TEXT | BCP }
| HEXADECIMAL { ON | OFF }
| ORDER { ON | OFF }
| QUOTE string
| QUOTES { ON | OFF }
| ROW DELIMITED BY string
```

encoding : string

Parameter

TO-Klausel Der Name der Datei, in die die Daten entladen werden. Der Pfad von *filename* ist relativ zum Startverzeichnis des Datenbankservers. Wenn die Datei nicht existiert, wird sie erstellt. Wenn sie bereits existiert, wird sie überschrieben, sofern nicht APPEND ON angegeben wird.

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „Sandboxing“ [[SQL Anywhere Server - Datenbankadministration](#)].

INTO FILE-Klausel Semantisch gleichwertig mit TO *filename*.

Wenn Festplatten-Sandboxing aktiviert ist, werden die Vorgänge der Datenbank auf das Verzeichnis beschränkt, in dem sich die Haupt-Datenbankdatei befindet. Siehe „Sandboxing“ [[SQL Anywhere Server - Datenbankadministration](#)].

INTO CLIENT FILE-Klausel Die Datei auf dem Clientcomputer, in die die Daten entladen werden. Wenn die Datei nicht existiert, wird sie erstellt. Wenn sie bereits existiert, wird sie überschrieben, sofern nicht APPEND ON angegeben wird. Der Pfad wird auf dem Clientcomputer relativ zum aktuellen Arbeitsverzeichnis der Clientanwendung aufgelöst.

INTO VARIABLE-Klausel Die Variable, in die die Daten entladen werden sollen. Die Variable muss bereits existieren und vom Typ CHAR, NCHAR oder BINARY sein. Mit der APPEND-Option werden die entladenen Daten an die bestehenden Inhalte der Variablen angehängt.

APPEND-Klausel Wenn die APPEND-Klausel ON ist, werden entladene Daten am Ende der angegebenen Datei angefügt. Wenn die APPEND-Klausel OFF ist, ersetzen entladene Daten den Inhalt der angegebenen Datei. APPEND ist standardmäßig OFF. Diese Klausel kann nicht verwendet werden, wenn die Klauseln COMPRESSED oder ENCRYPTED angegeben wurden, bzw, wenn die Datei, in der die Inhalte angehängt werden sollen, komprimiert oder verschlüsselt ist.

BYTE ORDER MARK-Klausel Verwenden Sie diese Klausel, um anzugeben, ob eine Bytereihenfolge-Markierung (BOM) geschrieben werden soll. Standardmäßig ist diese Option ON, wenn das Ziel des Entladevorgangs eine lokale oder Clientdatei ist. Wenn die BYTE ORDER MARK-Option ON ist und ENCODING UTF-8 oder UTF-16, wird eine BOM geschrieben. Wenn BYTE ORDER MARK auf OFF steht, wird keine BOM entladen.

COMPRESSED-Klausel Legt fest, ob die Daten komprimiert werden sollen. Standardwert ist NOT COMPRESSED. Sie können die Daten nicht komprimieren, wenn sie angehängt werden sollen (APPEND ON).

COLUMN DELIMITED BY- und DELIMITED BY-Klausel Die zwischen Spalten verwendete Zeichenfolge. Das Standardtrennzeichen für Spalten ist ein Komma. Sie können ein alternatives Spaltentrennzeichen angeben, indem Sie eine Zeichenfolge mit einer Länge von bis zu 255 Byte bereitstellen.

ENCODING-Klausel Alle Datenbankdaten werden von der Datenbank-Zeichenkodierung in den angegebenen CHAR- oder NCHAR-Zeichensatz konvertiert. Wenn ENCODING nicht angegeben ist, wird der CHAR-Zeichensatz der Datenbank verwendet.

Wenn während eines Entladevorgangs ein Konvertierungsfehler auftritt, wird er entsprechend der Einstellung der `on_charset_conversion_failure`-Option gemeldet.

Geben Sie die `BYTE ORDER MARK`-Klausel an, um eine BOM (Byte Order Mark) in die Daten einzubeziehen.

ENCRYPTED-Klausel Legt fest, ob die Daten verschlüsselt werden sollen. Wenn Sie `NOT ENCRYPTED` (Standardvorgabe) angeben, werden die Daten nicht verschlüsselt. Wenn Sie `ENCRYPTED KEY` mit einem Schlüssel und keinem Algorithmus angeben, werden die Daten mit AES128 und dem angegebenen Schlüssel verschlüsselt. Der Schlüssel kann entweder eine Zeichenfolge oder ein Variablenname sein. Wenn Sie `ENCRYPTED KEY` mit einem Schlüssel und einem Algorithmus angeben, werden die Daten mit dem angegebenen Schlüssel und Algorithmus verschlüsselt. Der Algorithmus kann jeder Algorithmus sein, der von der `CREATE DATABASE`-Anweisung akzeptiert wird. Sie können keine einfache Verschlüsselung angeben.

Sie können die Daten nicht verschlüsseln, wenn sie angehängt werden sollen (`APPEND ON`).

ESCAPES-Klausel Wenn `ESCAPES` aktiviert ist (Standard), schreibt der Datenbankserver Escapesequenzen. Zeilenendmarken können als `\n` geschrieben werden, und andere Zeichen können als hexadezimale ASCII-Codes in die Daten eingefügt werden, wie zum Beispiel als `\x09` für das Tabulatorzeichen. Eine Sequenz von zwei Backslashes (`\\`) wird als ein einzelner Backslash geschrieben. Ein Backslash gefolgt von einem beliebigen Zeichen außer `n`, `x`, `X` oder `\` wird als zwei separate Zeichen geschrieben. Zum Beispiel werden mit `\q` ein Backslash und der Buchstabe `q` eingefügt. Es wird empfohlen, dass die Zeichenfolge, die Sie für das Escape-Zeichen angeben, nicht länger als ein Mehrbyte-Zeichen ist.

FORMAT-Klausel Gibt Daten entweder im `TEXT`-Format oder im `BCP`-Ausgabeformat aus. Wenn Sie `TEXT` wählen, werden Eingabezeilen als Textzeichen behandelt, eine Tabellenzeile pro Ausgabezeile, wobei die Werte durch die Spalten-Trennzeichenfolge voneinander getrennt werden. Wenn Sie `BCP` wählen, werden Daten einschließlich `BLOBs` als `BCP`-Eingabedateien zur Verwendung mit Adaptive Server Enterprise exportiert. Das Standardformat ist `TEXT`.

HEXADECIMAL-Klausel Standardmäßig ist `HEXADECIMAL` auf `ON` gesetzt. Binärspaltenwerte werden als `0xnnnnnn...` geschrieben, wobei `0x` eine Null, gefolgt von einem `x` ist und jedes `n` ist eine hexadezimale Zahl. Es ist wichtig, `HEXADECIMAL ON` zu verwenden, wenn Sie mit Mehrbyte-Zeichensätzen arbeiten.

Die `HEXADECIMAL`-Klausel kann nur mit der `FORMAT TEXT`-Klausel verwendet werden.

ESCAPE CHARACTER-Klausel Verwenden Sie diese Klausel, um das Escape-Zeichen anzugeben, das in den Daten verwendet wird. Das standardmäßige Escape-Zeichen für Zeichen, die als hexadezimale Codes und Symbole geschrieben werden, ist ein Backslash (`\`), sodass `\x0A` beispielsweise der Zeilenendmarke entspricht. Mit der `ESCAPE CHARACTER`-Klausel kann das geändert werden.

Es wird empfohlen, dass die Zeichenfolge, die Sie für das Escape-Zeichen angeben, nicht länger als ein Mehrbyte-Zeichen ist.

ORDER-Klausel Wenn `ORDER` auf `ON` (Standardwert) gesetzt ist, werden die exportierten Daten nach Clustered-Index geordnet, wenn einer vorhanden ist. Wenn kein Clustered-Index vorhanden ist, werden die exportierten Daten nach Primärschlüsselwerten sortiert. Mit `ORDER OFF` werden die Daten

in derselben Reihenfolge exportiert wie sie bei der Auswahl aus einer Tabelle ohne ORDER BY-Klausel zu sehen ist. Das Exportieren geht langsamer vonstatten, wenn ORDER aktiviert ist. Das Neuladen mit der Anweisung LOAD TABLE erfolgt schneller, weil der Indizierungsschritt einfach ist.

Für UNLOAD *select-statement* wird die ORDER-Klausel ignoriert. Sie können die Daten aber immer noch ordnen, indem Sie eine ORDER BY-Klausel in die SELECT-Anweisung aufnehmen.

QUOTE-Klausel Die QUOTE-Klausel ist nur für TEXT-Daten bestimmt. *string* wird vor und nach den Zeichenfolgenwerten gesetzt. Der Standardwert ist ein Apostroph.

QUOTES-Klausel Wenn QUOTES aktiviert ist (Standard), wird das Hervorhebungszeichen (standardmäßig ein Apostroph) vor und nach allen exportierten Zeichenfolgen gesetzt.

ROW DELIMITED BY-Klausel Verwenden Sie diese Klausel, um die Zeichenfolge anzugeben, die das Ende eines Datensatzes anzeigt. Das Standardtrennzeichen ist eine Zeilenendmarke (\n). Es kann aber jede Zeichenfolge mit bis zu 255 Byte Länge verwendet werden, z.B. . . . ROW DELIMITED BY '###' . . . Wenn Sie durch Tabulatoren getrennte Werte angeben möchten, könnten Sie z.B. die hexadezimale Escapesequenz für das Tabulatorzeichen (9) verwenden, . . . ROW DELIMITED BY '\x09' . . . Wenn Ihre Trennzeichenfolge ein \n enthält, entspricht es entweder \r\n oder \n.

Bemerkungen

Die UNLOAD *select-statement*-Anweisung lässt zu, dass Daten aus einer SELECT-Anweisung in eine kommagetrennte Datei exportiert werden. Die Ergebnismenge wird nur sortiert, wenn die SELECT-Anweisung selbst eine ORDER BY-Klausel enthält.

Die UNLOAD TABLE-Anweisung ermöglicht effiziente Massensexporte aus einer Datenbanktabelle oder materialisierten Ansicht in eine Datei. Die UNLOAD TABLE-Anweisung ist effizienter als die Interactive SQL-Anweisung OUTPUT und kann von jeder Clientanwendung aufgerufen werden.

Der Datenbankserver oder die Clientanwendung (je nachdem, ob INTO FILE oder INTO CLIENT FILE angegeben wurde), müssen Berechtigungen auf der Ebene des Betriebssystems haben, in die angegebene Datei zu schreiben.

Beim Entladen von Tabellenspalten mit binären Datentypen schreibt UNLOAD TABLE hexadezimale Zeichenfolgen in der Form von \xnnnn, wobei *n* eine hexadezimale Ziffer ist.

Wenn Sie eine Datenbank mit Proxytabellen entladen und neu laden, müssen Sie ein externes Login erstellen, um den lokalen Benutzer dem entfernten Benutzer zuzuordnen, selbst wenn der Benutzer für die lokale und die entfernte Datenbank über das gleiche Kennwort verfügt. Sollten Sie kein externes Login haben, kann es sein, dass das erneute Laden am fehlenden Verbindungsaufbau zum Fremdserver scheitert.

Beim Entladen in eine Variable (INTO VARIABLE) wird die Ausgabe wie folgt in einen Zeichensatz umgewandelt:

- **CHAR** Ausgabe in die Variable im CHAR-Zeichensatz. Die ENCODING-Klausel muss dem CHAR-Zeichensatz entsprechen.
- **NCHAR** Ausgabe in die Variable im NCHAR-Zeichensatz. Die ENCODING-Klausel muss dem NCHAR-Zeichensatz entsprechen.

- **BINARY** Ausgabe in die Variable im BINARY-Zeichensatz. Die ENCODING-Klausel muss dem BINARY-Zeichensatz entsprechen. Andernfalls wird der CHAR-Zeichensatz verwendet.

Wenn Sie entscheiden, die entladenen Daten zu komprimieren und zu verschlüsseln, erfolgt erst die Verschlüsselung.

UNLOAD TABLE setzt eine Exklusivsperr für die gesamte Tabelle oder materialisierte Ansicht.

Während des Ausführens dieser Anweisung können Sie Meldungen zum Verarbeitungsfortschritt anfordern.

Sie können auch mithilfe der Verbindungseigenschaft "Progress" feststellen, wie viel von der Anweisung ausgeführt wurde.

Um die maximale Gesamtstellenzahl von Datumswerten beizubehalten, setzen Sie date_format auf YYYY-MM-DD.

Um die maximale Gesamtstellenzahl von TIMESTAMP-Werten beizubehalten, setzen Sie timestamp_format auf YYYY-MM-DD HH:NN:SS.SSSSSS.

Um die maximale Gesamtstellenzahl von TIMESTAMP WITH TIMEZONE-Werten beizubehalten, setzen Sie timestamp_with_time_zone_format auf YYYY-MM-DD HH:NN:SS.SSSSSS+HH:NN.

Privilegien

Beim Entladen in eine Variable sind keine Privilegien erforderlich. Andernfalls hängen die erforderlichen Privilegien wie folgt von der Datenbankserveroption -gl ab:

- Wenn die Option -gl auf ALL gesetzt ist, müssen Sie Eigentümer der Tabellen sein, das SELECT-Privileg für die Tabellen haben oder das SELECT ANY TABLE-Systemprivileg haben.
- Wenn die Option -gl auf DBA gesetzt ist, müssen Sie das SELECT ANY TABLE-Systemprivileg haben.
- Wenn -gl auf NONE gesetzt ist, wird UNLOAD nicht zugelassen.

Beim Entladen in eine Datei auf einem Clientcomputer gilt Folgendes:

- Sie müssen das WRITE CLIENT FILE-Privileg haben.
- Sie müssen Schreibberechtigungen für das Verzeichnis haben, in dem sich die Datei befindet.
- Die allow_write_client_file-Datenbankoption muss aktiviert sein.
- Die gesicherte Funktion write_client_file muss aktiviert sein.

Nebenwirkungen

Keine. Die Abfrage wird auf der aktuellen Isolationsstufe ausgeführt.

Siehe auch

- „CREATE DATABASE-Anweisung“ auf Seite 583
- „LOAD TABLE-Anweisung“ auf Seite 931
- „PASSTHROUGH-Anweisung [SQL Remote]“ auf Seite 975
- „Clustered-Indizes“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Unterstützte Zeichensätze“ [*SQL Anywhere Server - Datenbankadministration*]
- „OUTPUT-Anweisung [Interactive SQL]“ auf Seite 968
- „Tipps zum Exportieren von Daten mit der UNLOAD-Anweisung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Zugriff auf Daten auf Clientcomputern“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Datenimport und -export“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Tipps zum Exportieren von Daten mit der UNLOAD TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Datenbankserveroption -sf“ [*SQL Anywhere Server - Datenbankadministration*]
- „allow_write_client_file-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -gl“ [*SQL Anywhere Server - Datenbankadministration*]
- „timestamp_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „date_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „on_charset_conversion_failure-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- Progress-Verbindungseigenschaft [*SQL Anywhere Server - Datenbankadministration*]
- „progress_messages-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird der Inhalt der Tabelle Products in eine mit UTF-8 kodierte Datei *productsT.dat* entladen:

```
UNLOAD TABLE GROUPO.Products TO 'c:\\temp\\productsT.dat' ENCODING 'UTF-8';
```

Im folgenden Beispiel wird eine Variable @myProducts erstellt. Danach wird die Spalte Products.Name in die Variable entladen:

```
CREATE VARIABLE @myProducts LONG VARCHAR;
UNLOAD SELECT NAME FROM GROUPO.Products INTO VARIABLE @myProducts ESCAPE
CHARACTER '!' ;
```

UPDATE-Anweisung (positionsbasiert) [ESQL] [SP]

Ändert die Daten an der aktuellen Cursorposition.

Syntax 1 [nur ESQL]

```
UPDATE WHERE CURRENT OF cursor-name
{ USING [ SQL ] DESCRIPTOR sqlda-name | { [ FROM ] | [ USING ] } hostvar-list }
```

Syntax 2

UPDATE *update-table*, ...
SET *set-item*, ...
WHERE CURRENT OF *cursor-name*

hostvar-list : *indicator variables allowed*

update-table :
[*owner-name*.] *object-name* [[**AS**] *correlation-name*]

set-item :
[*correlation-name*.] *column-name* = { *expression* | **DEFAULT** }
[[*owner-name*.] *object-name*. *column-name* = { *expression* | **DEFAULT** }

object-name : *identifier* (Tabellen- oder Ansichtsname)

sqlda-name : *identifier*

Parameter

UPDATE-Klausel Jede *update-table* wird mit einer Tabelle in der Abfrage für den Cursor wie folgt verglichen:

1. Wenn ein Korrelationsname angegeben ist, wird sie mit einer Tabelle in der Abfrage für den Cursor verglichen, die denselben *table-or-view-name*- und denselben *correlation-name*-Parameter hat.
2. Wenn die Abfrage des Cursors eine Tabelle enthält, die denselben *table-or-view-name*-Wert aufweist, aber keinen Korrelationsnamen hat, oder einen Korrelationsnamen hat, der mit dem *table-or-view-name*-Wert übereinstimmt, wird die Aktualisierungstabelle mit dieser Tabelle in der Abfrage des Cursors abgeglichen.
3. Wenn die Abfrage des Cursors eine einzige Tabelle enthält, die denselben *table-or-view-name*-Wert aufweist wie die Aktualisierungstabelle, wird die Aktualisierungstabelle mit dieser Tabelle in der Abfrage des Cursors abgeglichen.

Wenn für eine Spalte ein Standardwert definiert ist, können Sie die SET-Klausel verwenden, um eine Spalte auf ihren Standardwert zu setzen. Ein Beispiel dafür finden Sie im Abschnitt "Beispiele" unter [„UPDATE-Anweisung“ auf Seite 1109](#).

USING DESCRIPTOR-Klausel Wenn eine Variable zugeordnet wird, muss diese bereits deklariert sein und ihr Name muss mit dem At-Zeichen (@) beginnen. Variablen- und Spaltenzuordnungen können gemischt werden, und die Verwendung jeglicher Zahl ist möglich.

SET-Klausel Die im *set-item* angegebenen Spalten müssen sich in der zu aktualisierenden Tabelle oder Ansicht befinden. Wenn in der SET-Liste ein Name auf der linken Seite einer Zuordnung sowohl einer Spalte in der aktualisierten Tabelle als auch einem Variablennamen entspricht, wird die Anweisung die Spalte aktualisieren. Das *set-item* kann weder Aliasnamen noch Spalten aus anderen Tabellen oder Ansichten referenzieren. Wenn der zu aktualisierenden Tabelle oder Ansicht in der Cursorspezifikation ein Korrelationsname zugewiesen ist, müssen Sie diesen Korrelationsnamen in der SET-Klausel verwenden.

Jedes *set-item* ist einer einzigen *update-table* zugeordnet, und die entsprechende Spalte der passenden Tabelle in der Cursorabfrage wird geändert. Der *expression* referenziert Spalten der in der UPDATE-Liste

angegebenen Tabellen und kann Konstanten, Hostvariablen, Variablen, Ausdrücke aus der SELECT-Liste der Abfrage oder Kombinationen aus diesen Elementen verwenden, und zwar mit Operatoren wie +, -, ..., COALESCE, IF usw. Der *expression* kann weder Aliasnamen von Ausdrücken aus der Abfrage des Cursors referenzieren noch Spalten aus anderen Tabellen in der Abfrage des Cursors, die nicht in der UPDATE-Liste enthalten sind. Unterabfrage-Bedingungen, Unterabfrageprädikate oder Aggregatfunktionen dürfen nicht in *set-item*-Objekten verwendet werden.

Bemerkungen

Diese Form der UPDATE-Anweisung aktualisiert die aktuelle Zeile des angegebenen Cursors. Die aktuelle Zeile wird als die zuletzt vom Cursor erfolgreich abgerufene Zeile definiert, und der letzte Cursorvorgang darf keine positionsbasierte DELETE-Anweisung gewesen sein.

Bei Syntax 1 entsprechen Spalten aus dem SQLDA oder Werte aus der Hostvariablenliste eins zu eins den vom festgelegten Cursor zurückgegebenen Spalten. Wenn der sqldata-Zeiger im SQLDA der Leerzeiger ist, wird das entsprechende Element der Auswahlliste nicht aktualisiert.

In Syntax 2 werden die angeforderten Spalten auf die angegebenen Werte für die Zeile in der aktuellen Zeile der angegebenen Abfrage gesetzt. Die Spalten müssen nicht in der Auswahlliste des angegebenen und geöffneten Cursors enthalten sein. Dieses Format kann vorbereitet werden.

Wenn eine Variable zugeordnet wird, muss diese bereits deklariert sein und ihr Name muss mit dem At-Zeichen (@) beginnen. Variablen- und Spaltenzuordnungen können gemischt werden, und die Verwendung jeglicher Zahl ist möglich. Wenn in der SET-Liste ein Name auf der linken Seite einer Zuordnung sowohl einer Spalte in der aktualisierten Tabelle als auch einem Variablennamen entspricht, wird die Anweisung die Spalte aktualisieren.

Die Formate USING DESCRIPTOR, FROM *hostvar-list* und *hostvar* gelten nur für Embedded SQL.

Privilegien

Sie müssen Eigentümer der zu aktualisierenden Tabelle sein, das UPDATE-Privileg für die zu ändernden Spalten haben oder das UPDATE ANY TABLE-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „INSERT-Anweisung“ auf Seite 917
- „LOAD TABLE-Anweisung“ auf Seite 931
- „MERGE-Anweisung“ auf Seite 952
- „DELETE-Anweisung“ auf Seite 791
- „DELETE-Anweisung (positionsbasiert) [ESQL] [SP]“ auf Seite 789
- „UPDATE-Anweisung“ auf Seite 1109
- „ansi_update_constraints-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Standards und Kompatibilität

- **SQL/2008** Syntax 1 ist eine Erweiterung des Herstellers. Syntax 2 ist eine Kernfunktion des SQL/2008-Standards. Bei Verwendung innerhalb eines Embedded SQL-Programms umfasst Syntax 2 einen

Teil der optionalen SQL-Sprachenfunktion B031 (Basic Dynamic SQL). Die Möglichkeit zum Angeben mehrerer zu aktualisierender Tabellen ist eine Erweiterung des Herstellers.

Der Bereich der Cursor, die aktualisiert werden können, ist abhängig von der Einstellung der Option `ansi_update_constraints`. Die Möglichkeit zum Ausführen eines positionsbasierten UPDATE über einen sortierten Cursor, d.h., die SQL-Abfrage enthält eine ORDER BY-Klausel, umfasst die optionale SQL/2008-Sprachenfunktion F831 (vollständige Cursoraktualisierung). Das Ausführen eines positionsbasierten UPDATE über komplexere SQL-Konstruktionen erfordert möglicherweise zusätzliche Erweiterungen des Herstellers.

Beispiel

Im folgenden Beispiel wird eine UPDATE-Anweisung in einem fiktiven Cursor namens `emp_cursor` ausgeführt:

```
UPDATE GROUPO.Employees
SET Surname = 'Jones'
WHERE CURRENT OF emp_cursor;
```

UPDATE-Anweisung [SQL Remote]

Ändert Daten in der Datenbank.

Syntax 1

```
UPDATE table-list
SET column-name = expression, ...
[ VERIFY ( column-name, ... ) VALUES ( expression, ... ) ]
[ WHERE search-condition ]
[ ORDER BY expression [ ASC | DESC ], ... ]
```

Syntax 2

```
UPDATE table-name
PUBLICATION publication-name
{ SUBSCRIBE BY subscription-expression |
  OLD SUBSCRIBE BY old-subscription-expression |
  NEW SUBSCRIBE BY new-subscription-expression }
WHERE search-condition
```

expression : *value* | *subquery*

Parameter

table-name Der *table-name* kennzeichnet die Tabelle, die in den entfernten Datenbanken geändert werden muss.

publication-name Der *publication-name* kennzeichnet die Publikation, bei der Subskriptionen geändert werden müssen.

subscription-expression Der *subscription-expression* ist der Wert, den SQL Remote verwendet, um sowohl die neuen als auch die vorhandenen Empfänger der Zeilen zu ermitteln. Der *subscription-expression* ist ein Wert oder eine Unterabfrage. Eine andere Möglichkeit ist es, sowohl die alten (OLD) als auch neuen (NEW) Subskriptionsausdrücke anzuführen.

WHERE-Klausel Die WHERE-Klausel gibt an, welche Zeilen zwischen den subskribierten Datenbanken zu übertragen sind.

Bemerkungen

Die UPDATE-Anweisung wird verwendet, um Zeilen einer oder mehrerer Tabellen zu ändern. Jede benannte Spalte wird auf den Wert des Ausdrucks auf der rechten Seite des Gleichheitszeichens gesetzt. Es gibt keine Einschränkungen für *expression*. Sogar *column-name* kann im Ausdruck verwendet werden —der alte Wert wird verwendet.

Wenn keine WHERE-Klausel angegeben ist, wird jede Zeile aktualisiert. Wenn eine WHERE-Klausel angegeben ist, werden nur jene Zeilen aktualisiert, die der Suchbedingung entsprechen.

Normalerweise ist die Reihenfolge, in der Zeilen aktualisiert werden, unbedeutend. In Verbindung mit der NUMBER(*)-Funktion kann eine Sortierfolge jedoch nützlich sein, damit ansteigende Zahlen in der angegebenen Reihenfolge zu den Zeilen hinzugefügt werden. Wenn Sie beispielsweise 1 zu den Primärschlüsselwerten in einer Tabelle hinzufügen möchten, ist es notwendig, dies in absteigender Sortierfolge der Primärschlüssel zu tun, damit der Vorgang keine mehrfach vorkommenden Primärschlüssel erzeugt.

Ansichten können aktualisiert werden, solange die SELECT-Anweisung, die die Ansicht definiert, keine GROUP BY-Klausel oder Aggregatfunktion enthält, bzw. nicht mit einer UNION-Klausel zusammenhängt.

In Tabellen eingefügte Zeichenfolgen werden immer in der eingegebenen Groß-/Kleinschreibung gespeichert, unabhängig davon, ob die Datenbank auf Groß-/Kleinschreibung reagiert. Daher wird eine Zeichendatentyp-Spalte, die mit der Zeichenfolge "Value" aktualisiert wird, in der Datenbank immer mit einem groß geschriebenen V und den restlichen Buchstaben in Kleinschreibung enthalten sein. SELECT-Anweisungen geben die Zeichenfolge als Value zurück. Wenn die Datenbank die Groß-/Kleinschreibung jedoch nicht berücksichtigt, ist Value dasselbe wie value, VALUE usw. Wenn außerdem ein Primärschlüssel für eine Spalte einen Eintrag mit Value enthält, wird ein INSERT von value zurückgewiesen, da der Primärschlüssel dann nicht eindeutig wäre.

Mit der optionalen FROM-Klausel können Tabellen auf der Grundlage von Joins aktualisiert werden. Wenn die FROM-Klausel vorliegt, qualifiziert die WHERE-Klausel die Zeilen der FROM-Klausel. Daten werden nur in der Tabellenliste aktualisiert, die unmittelbar auf das Schlüsselwort UPDATE folgt.

Wenn eine FROM-Klausel verwendet wird, ist es wichtig, dass der zu aktualisierende Tabellename in beiden Teilen der Anweisung gleich qualifiziert wird. Wenn ein Korrelationsname einmal verwendet wird, muss derselbe Korrelationsname auch beim nächsten Mal verwendet werden. Sonst wird eine Fehlermeldung ausgegeben.

Syntax 1 und Syntax 2 gelten nur für SQL Remote.

Syntax 2 ohne OLD und NEW SUBSCRIBE BY-Ausdrücke muss in einem BEFORE-Trigger verwendet werden.

Syntax 2 mit OLD und NEW SUBSCRIBE BY-Ausdrücken kann überall verwendet werden.

Syntax 1 ist nur für die Verwendung mit SQL Remote für Aktualisierungen von Einzelzeilen vorgesehen, die durch den Nachrichtenagenten ausgeführt werden. Die VERIFY-Klausel enthält eine Wertmenge, die

in den zu aktualisierenden Zeilen erwartet wird. Wenn die Werte nicht übereinstimmen, werden alle RESOLVE UPDATE-Trigger ausgelöst, bevor die Aktualisierung weitergeführt wird. Das UPDATE schlägt nicht fehl, wenn die VERIFY-Klausel nicht zutrifft. Wenn die VERIFY-Klausel angegeben ist, kann immer nur eine Tabelle gleichzeitig aktualisiert werden.

Syntax 2 wird nur mit SQL Remote verwendet. Wenn keine OLD- und NEW-Ausdrücke verwendet werden, muss die Syntax innerhalb eines BEFORE-Triggers benutzt werden, damit Zugriff auf die relevanten Werte besteht. Diese Syntax dient dazu, bei jeder Listenänderung eine vollständige Liste von SUBSCRIBE BY-Werten zu erhalten. Sie wird in SQL Remote-Triggern platziert, damit der Datenbankserver die aktuelle Liste von SUBSCRIBE BY-Werten ermitteln kann. Beide Listen werden im Transaktionslog abgelegt.

Der Nachrichtenagent verwendet diese zwei Listen, um sicherzustellen, dass die Zeile an alle entfernten Datenbanken übermittelt wird, die die Zeile noch nicht haben und jetzt benötigen. Der Nachrichtenagent entfernt die Zeile aus jeder entfernten Datenbank, welche diese Zeile hat, sie aber nicht länger benötigt. Eine entfernte Datenbank, die diese Zeile enthält und weiterhin benötigt, wird von der UPDATE-Anweisung nicht beeinflusst.

Syntax 2 der UPDATE Anweisung ermöglicht die explizite Angabe der alten SUBSCRIBE BY- und der neuen SUBSCRIBE BY-Liste, wodurch SQL Remote-Trigger wirksamer gemacht werden können. Wenn diese Listen nicht vorliegen, errechnet der Datenbankserver die alte SUBSCRIBE BY-Liste aus der Publikationsdefinition. Da sich die neue SUBSCRIBE BY-Liste im Allgemeinen nur leicht von der alten SUBSCRIBE BY-Liste unterscheidet, kann die Ermittlung der alten Liste zwei Mal erfolgen. Indem Sie sowohl die alte als auch die neue Liste angeben, wird dieser zusätzliche Aufwand vermieden.

Die OLD und NEW SUBSCRIBE BY-Syntax ist vor allem dann nützlich, wenn viele Tabellen im selben Trigger mit denselben Subscribe By-Ausdrücken verwendet werden. Dadurch kann die Performance erheblich gesteigert werden.

Der Ausdruck SUBSCRIBE BY ist entweder ein Wert oder eine Unterabfrage.

Syntax 2 der UPDATE-Anweisung wird verwendet, um eine spezifische SQL Remote-Funktion zu implementieren und muss innerhalb eines BEFORE-Triggers verwendet werden.

Bei Publikationen, die mit einer Unterabfrage in einem Subskriptionsausdruck erstellt werden, müssen Sie einen Trigger schreiben, der Syntax 2 der UPDATE-Anweisung enthält. So stellen Sie sicher, dass die Zeilen in den korrekten Subskriptionen bleiben.

Syntax 2 der UPDATE-Anweisung nimmt einen Eintrag in das Transaktionslog vor, ändert aber die Datenbanktabelle nicht.

Privilegien

Sie müssen Eigentümer der zu aktualisierenden Tabelle sein, das UPDATE-Privileg für die zu ändernden Spalten haben oder das UPDATE ANY TABLE-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „BEFORE UPDATE-Trigger“ [[SQL Remote](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Das folgende Beispiel transferiert den Mitarbeiter Philip Chin (employee 129) von der Verkaufsabteilung in die Marketingabteilung.

```
UPDATE GROUPO.Employees
SET DepartmentID = 400
WHERE EmployeeID = 129;
```

UPDATE-Anweisung

Ändert vorhandene Zeilen in Datenbanktabellen

Syntax 1

```
UPDATE [ row-limitation ] table-expression [, ...]
SET set-item[, ...]
[ WHERE search-condition ]
[ ORDER BY expression [ ASC | DESC ] , ...]
[ OPTION( query-hint, ... ) ]
```

table-expression: Ein Tabellenausdruck, der Joins, Outer-Joins, Ansichten und abgeleitete Tabellen enthalten kann. Siehe „[FROM-Klausel](#)“ auf [Seite 863](#).

Syntax 2

```
UPDATE [ row-limitation ] table-name
SET set-item[, ...]
FROM table-expression [, ...] ]
[ WHERE search-condition ]
[ ORDER BY expression [ ASC | DESC ] , ...]
[ OPTION( query-hint, ... ) ]
```

table-name :

```
[ owner.]table-name [ [ AS ] correlation-name ]
| [ owner.]view-name [ [ AS ] correlation-name ]
| derived-table
```

derived-table :

```
( select-statement )
[ AS ] correlation-name [ ( column-name [, ...] ) ]
```

Syntax 3

```
UPDATE table-name
SET set-item, ...
VERIFY ( column-name, ... ) VALUES ( expression, ... )
[ WHERE search-condition ]
```

[**ORDER BY** *expression* [**ASC** | **DESC**], ...]
[**OPTION**(*query-hint*, ...)]

Syntax 4

UPDATE [*owner.*]*table-name*
PUBLICATION *publication*
{ **SUBSCRIBE BY** *expression*
| **OLD SUBSCRIBE BY** *expression* **NEW SUBSCRIBE BY** *expression* }
WHERE *search-condition*

row-limitation :

FIRST
| **TOP** { **ALL** | *limit-expression* } [**START AT** *startat-expression*]

limit-expression : *simple-expression*

startat-expression : *simple-expression*

simple-expression :

integer
| *variable*
| (*simple-expression*)
| (*simple-expression* { + | - | * } *simple-expression*)

set-item :

[*correlation-name.*]*column-name*= { *expression* | **DEFAULT** }
| [*owner-name.*]*table-name.column-name* = { *expression* | **DEFAULT** }
| @*variable-name*=*expression*

query-hint :

MATERIALIZED VIEW OPTIMIZATION *option-value*
| **FORCE OPTIMIZATION**
| **FORCE NO OPTIMIZATION**
| *option-name* = *option-value*

table-name :

[*owner.*]*base-table-name*
| *temporary-table-name*
| *derived-table-name*
| [*owner.*]*view-name*

option-name : *identifier*

option-value :

hostvar (Bezeichner zulässig)
| *string*
| *identifier*
| *number*

Parameter

UPDATE-Klausel Für Syntax 1 und Syntax 2 kann *table-expression* temporäre Tabellen, globale temporäre Tabellen, abgeleitete Tabellen oder Ansichten enthalten. Ansichten und abgeleitete Tabellen können aktualisiert werden, sofern sie aktualisierbar sind. Für Syntax 1 ergibt eine Liste von mehr als *table-expression* ein kartesisches Produkt der Zeilen, das aus den darunterliegenden Tabellenausdrücken gebildet wird und dann über die Verwendung der WHERE-Klausel eingeschränkt werden kann. Syntax 1

und Syntax 2 ermöglichen die Aktualisierung von Joins. Für Syntax 3 und 4 muss *table-name* eine Basistabelle sein.

UPDATE-Anweisungen können in Ansichten nur ausgeführt werden, wenn die Abfragespezifikation, die die Ansicht definiert, aktualisierbar ist.

row-limitation-Klausel Die Zeilenbeschränkungsklausel ermöglicht es Ihnen, die zu aktualisierenden Zeilen auf eine Teilmenge der Zeilen begrenzen, die von der WHERE-Klausel erfasst werden. Die Argumente TOP und START AT können einfache arithmetische Ausdrücke über Hostvariablen, Ganzzahlkonstanten oder Ganzzahlvariablen sein. Das TOP-Argument muss jedoch mit einem Wert größer oder gleich 0 ausgewertet werden. Das START AT-Argument mit einem Wert größer als 0 ausgewertet werden. Eine ORDER BY-Klausel sollte verwendet werden, um die Zeilen sinnvoll zu ordnen.

SET-Klausel Verwenden Sie die SET-Klausel, um Spaltennamen oder Variablen auf den angegebenen Ausdruck zu setzen.

Sie können die SET-Klausel verwenden, um die Spalte auf einen berechneten Spaltenwert zu setzen, indem Sie folgendes Format verwenden:

```
SET column-name = expression, ...
```

Jede angegebene Spalte wird auf den Wert des Ausdrucks gesetzt. Es gibt keine Einschränkungen für den *expression*. Wenn der *expression* ein *column-name* ist, wird der vorherige Wert aus dieser Spalte verwendet.

Wenn für eine Spalte ein Standardwert definiert ist, können Sie die SET-Klausel verwenden, um eine Spalte auf ihren Standardwert zu setzen.

Sie können die SET-Klausel auch verwenden, um eine Variable zuzuordnen, und zwar im folgenden Format:

```
SET @variable-name = expression, ...
```

Bei Zuordnung eines Wert zu einer Variablen muss diese bereits deklariert sein und ihr Name muss mit dem At-Zeichen (@) beginnen. Wenn der Variablenname mit dem Namen einer Spalte in der zu aktualisierenden Tabelle übereinstimmt, aktualisiert die UPDATE-Anweisung den Spaltenwert und lässt die Variable unverändert. Variablen- und Spaltenzuordnungen können in beliebiger Reihenfolge kombiniert werden.

FROM-Klausel Mit der optionalen Klausel FROM *table-expression* können Tabellen auf der Grundlage von Joins aktualisiert werden. *table-expression* kann beliebig komplexe Tabellenausdrücke wie OUTER-, CROSS- und NATURAL-Joins enthalten.

Wenn die FROM-Klausel vorliegt, muss mit *table-name* die einzige Tabelle angegeben werden, die aktualisiert werden soll, und der Name muss auf die gleiche Weise geschrieben sein, wie er in der FROM-Klausel erscheint. Wenn Korrelationsnamen in der FROM-Klausel verwendet werden, muss der identische Korrelationsname als *table-name* angegeben werden. Wenn der Tabellenausdruck, der aktualisiert werden soll, eine abgeleitete Tabelle ist, muss die abgeleitete Tabelle in der *table-name*-Spezifikation wiederholt werden.

Syntax 2 kann nicht benutzt werden, wenn die Option `ansi_update_constraints` auf `Strict` eingestellt ist.

Wenn eine `FROM`-Klausel angegeben wird, kann die `SET`-Klausel nur Spalten aus *table-name* zur Aktualisierung enthalten. Sonst wird eine Fehlermeldung ausgegeben.

Die folgende Anweisung veranschaulicht eine potenzielle Mehrdeutigkeit bei Tabellennamen in `UPDATE`-Anweisungen mit Syntax 2, die Tabellenausdrücke mit Korrelationsnamen enthalten:

```
UPDATE table_1
SET column_1 = ...
FROM table_1 AS alias_1, table_1 AS alias_2
WHERE ...
```

Im folgenden Beispiel hat jede Instanz von `table_1` in der `FROM`-Klausel einen Korrelationsnamen, der einen Selbst-Join von `table_1` mit sich selbst angibt. Die `UPDATE`-Anweisung gibt aber nicht an, welche der Zeilen, die den Selbst-Join darstellen, aktualisiert werden sollen. Diese Auslassung kann folgendermaßen durch Angeben des Korrelationsnamens in der `UPDATE`-Anweisung behoben werden:

```
UPDATE table_1 as alias_1
SET column_1 = ...
FROM table_1 AS alias_1, table_1 AS alias_2
WHERE ...
```

WHERE-Klausel Wenn eine `WHERE`-Klausel angegeben ist, werden nur die Zeilen aktualisiert, welche die Suchbedingung erfüllen. Wenn keine `WHERE`-Klausel angegeben ist, wird jede Zeile aktualisiert.

ORDER BY-Klausel Normalerweise spielt die Reihenfolge, in der Zeilen aktualisiert werden, keine Rolle. In Verbindung mit der `FIRST`- oder `TOP`-Klausel kann die Reihenfolge jedoch wichtig sein.

Sie können Spalten-Ordinalnummern in der `ORDER BY`-Klausel nicht verwenden.

Wenn Sie die `ORDER BY`-Klausel verwenden möchten, darf die `ansi_update_constraints`-Option nicht auf `"Strict"` gesetzt werden.

Wenn Sie Spalten aktualisieren möchten, die in der `ORDER BY`-Klausel erscheinen, muss die `ansi_update_constraints`-Option auf `"Off"` gesetzt werden.

OPTION-Klausel Verwenden Sie diese Klausel, um Hints für das Ausführen der Anweisung anzugeben. Die folgenden Hints werden unterstützt:

- `MATERIALIZED VIEW OPTIMIZATION` *option-value*
- `FORCE OPTIMIZATION`
- `FORCE NO OPTIMIZATION`
- *option-name* = *option-value*. Die Spezifikation `OPTION(isolation_level = ...)` im Abfragetext hebt alle anderen Festlegungen der Isolationsstufe für eine Abfrage auf.

Bemerkungen

In Tabellen eingefügte Zeichenfolgen werden immer in der gleichen Schreibung (groß oder klein) gespeichert, in der sie eingegeben wurden, unabhängig davon, ob die Datenbank die Groß-/Kleinschreibung berücksichtigt oder nicht. Eine Spalte vom Datentyp `CHAR`, die mit der Zeichenfolge `Street` aktualisiert wird, ist in der Datenbank immer mit einem großen `S` gespeichert, die restlichen

Buchstaben sind klein geschrieben. SELECT-Anweisungen geben die Zeichenfolge als Street zurück. Wenn die Datenbank die Groß-/Kleinschreibung nicht berücksichtigt, wird bei Vergleichen Street genau so wie street, STREET, usw. behandelt. Wenn außerdem ein Primärschlüssel für eine Spalte einen Eintrag "Street" enthält, wird ein UPDATE eines Primärschlüssels einer anderen Zeile auf "street" zurückgewiesen, da sonst der Primärschlüssel nicht mehr eindeutig wäre.

Wenn sich der neue Wert nicht vom alten unterscheidet, werden die Daten nicht verändert. BEFORE UPDATE-Trigger werden jedoch jedes Mal ausgelöst, wenn ein UPDATE einer Zeile stattfindet, ganz gleich, ob sich der neue Wert vom alten unterscheidet. AFTER UPDATE-Trigger lösen nur aus, wenn sich die neuen von den alten Werten unterscheiden.

Syntax 3 und 4 gelten nur in SQL Remote.

Syntax 3 ist nur für die Verwendung mit SQL Remote für Aktualisierungen von Einzelzeilen in einer einzigen Tabelle vorgesehen, die durch den Nachrichtenagenten ausgeführt werden. Die VERIFY-Klausel enthält eine Wertmenge, die in den zu aktualisierenden Zeilen erwartet wird. Wenn die Werte nicht übereinstimmen, werden alle vorhandenen RESOLVE UPDATE-Trigger ausgelöst, bevor die Aktualisierung weitergeführt wird. Die UPDATE-Anweisung schlägt nicht fehl, weil es für die VERIFY-Klausel keine Übereinstimmung gab.

Syntax 4 der UPDATE-Anweisung wird verwendet, um eine spezifische SQL Remote-Funktion zu implementieren und muss innerhalb eines BEFORE-Triggers verwendet werden. Diese Syntax dient dazu, bei jeder Listenänderung eine vollständige Liste von SUBSCRIBE BY-Werten bereitzustellen. Sie wird in SQL Remote-Triggern platziert, damit der Datenbankserver die aktuelle Liste von SUBSCRIBE BY-Werten ermitteln kann. Beide Listen werden im Transaktionslog abgelegt.

Der Nachrichtenagent verwendet diese zwei Listen, um sicherzustellen, dass die Zeile an alle entfernten Datenbanken übermittelt wird, die die Zeile noch nicht haben und jetzt benötigen. Der Nachrichtenagent entfernt die Zeile aus jeder entfernten Datenbank, welche diese Zeile hat, sie aber nicht länger benötigt. Eine entfernte Datenbank, die diese Zeile enthält und weiterhin benötigt, wird von der UPDATE-Anweisung nicht beeinflusst.

Bei Publikationen, die mit einer Unterabfrage in einer SUBSCRIBE BY-Klausel erstellt werden, müssen Sie einen Trigger schreiben, der Syntax 4 der UPDATE-Anweisung enthält. So stellen Sie sicher, dass die Zeilen in ihren korrekten Subskriptionen bleiben.

Syntax 4 der UPDATE Anweisung ermöglicht die explizite Angabe der alten SUBSCRIBE BY- und der neuen SUBSCRIBE BY-Liste, wodurch SQL Remote-Trigger wirksamer gemacht werden können. Wenn diese Listen nicht vorliegen, errechnet der Datenbankserver die alte SUBSCRIBE BY-Liste aus der Publikationsdefinition. Da sich die neue SUBSCRIBE BY-Liste im Allgemeinen nur leicht von der alten SUBSCRIBE BY-Liste unterscheidet, kann die Ermittlung der alten Liste zwei Mal erfolgen. Indem Sie sowohl die alte als auch die neue Liste angeben, können Sie diese zusätzliche Arbeit vermeiden.

Der Ausdruck SUBSCRIBE BY ist entweder ein Wert oder eine Unterabfrage.

Syntax 4 der UPDATE-Anweisung nimmt einen Eintrag in das Transaktionslog vor, ändert aber die Datenbanktabelle nicht.

Das Aktualisieren großer Datenmengen mit der UPDATE-Anweisung aktualisiert auch die Spaltenstatistiken.

Privilegien

Sie müssen Eigentümer der zu aktualisierenden Tabelle sein, das UPDATE-Privileg für die zu ändernden Spalten haben oder das UPDATE ANY TABLE-Systemprivileg haben.

Nebenwirkungen

Spaltenstatistiken werden aktualisiert, sodass sie die geänderten Werte darstellen.

Wenn eine Tabelle einen Primärschlüssel, eine UNIQUE-Integritätsregel oder einen UNIQUE-Index hat, kann die Verarbeitung der UPDATE-Anweisung die Verwendung einer temporären Tabelle umfassen, wenn die Tabellenmanipulationen nicht ohne Verletzung der Eindeutigkeits-Integritätsregel erfolgen können. Die temporäre Tabelle speichert von der UPDATE-Anweisung geänderte Zeilen, die eine oder mehrere Eindeutigkeits-Integritätsregeln verletzen. Diese Zeilen werden während des Ausführens der UPDATE-Anweisung aus der Basistabelle gelöscht und anschließend wieder eingefügt. Dieses Verhalten kann Auswirkungen auf AFTER-Trigger und andere gleichzeitige Verbindungen haben.

Siehe auch

- „DELETE-Anweisung“ auf Seite 791
- „DELETE-Anweisung (positionsbasiert) [ESQL] [SP]“ auf Seite 789
- „INSERT-Anweisung“ auf Seite 917
- „UPDATE-Anweisung [SQL Remote]“ auf Seite 1106
- „CREATE TRIGGER-Anweisung“ auf Seite 762
- „CREATE TABLE-Anweisung“ auf Seite 737
- „PUT-Anweisung [ESQL]“ auf Seite 981
- OPTION-Klausel, SELECT-Anweisung auf Seite 1027
- „MERGE-Anweisung“ auf Seite 952
- „ALTER TABLE-Anweisung“ auf Seite 516
- „UPDATE-Anweisung (positionsbasiert) [ESQL] [SP]“ auf Seite 1103
- „UPDATE-Anweisungen mit einer VERIFY-Klausel“ [SQL Remote]
- „TSEQUAL-Funktion [System] (nicht mehr empfohlen)“ auf Seite 418
- „FROM-Klausel“ auf Seite 863
- „Ansichten“ [SQL Anywhere Server - SQL-Benutzerhandbuch]
- „Joins: Daten aus mehreren Tabellen abrufen“ [SQL Anywhere Server - SQL-Benutzerhandbuch]
- „Sperrungen während einer Aktualisierung“ [SQL Anywhere Server - SQL-Benutzerhandbuch]
- „Replikation der UPDATE-Anweisung“ [SQL Remote]
- „Zeilenbeschränkungsklauseln in SELECT-, UPDATE- und DELETE-Abfrageblöcken“ [SQL Anywhere Server - SQL-Benutzerhandbuch]
- „Datenänderungen mit UPDATE“ [SQL Anywhere Server - SQL-Benutzerhandbuch]
- „ansi_update_constraints-Option“ [SQL Anywhere Server - Datenbankadministration]

Standards und Kompatibilität

- **SQL/2008** Syntax 1 der UPDATE-Anweisung ist eine Kernfunktion des SQL/2008-Standards. Syntax 2 ist eine Erweiterung des Herstellers. Syntax 3 und 4 sind Erweiterungen des Herstellers nur für die Verwendung mit SQL Remote.

Syntax 1 enthält Unterstützung für zwei optionale SQL-Sprachenfunktionen:

- Die Unterstützung für die Aktualisierung von Joins, möglicherweise einschließlich einer oder mehrerer abgeleiteter Tabellen, ist Teil der optionalen SQL-Sprachenfunktion T111 (aktualisierbare Joins, Vereinigungen und Spalten).
- Die Unterstützung für das Ändern einer Tabelle, die in einer verschachtelten Unterabfrage referenziert wird, die wiederum Teil der Suchbedingung für die UPDATE-Anweisung ist, umfasst die optionale SQL/2008-Sprachenfunktion F781 (selbstreferenzierende Vorgänge).

Die folgenden Funktionen von Syntax 1 sind Erweiterungen des Herstellers:

- Die Klauseln FROM und ORDER BY.
- Die Klausel für *row-limitation*.
- Die Möglichkeit zur Angabe von mehr als einer *table-list*.
- Die Möglichkeit zum Aktualisieren einer Variablen mit der SET-Klausel.
- Die OPTION-Klausel.

Mit Syntax 1 kann die Einstellung der Option `ansi_update_constraints` gesteuert werden, welche Formen von Tabellenausdrücken geändert werden können. Um Kompatibilität mit SQL/2008-Kernfunktionen zu erzwingen, setzen Sie die Option `ansi_update_constraints` auf `STRICT`.

Beispiele

Bei Verwendung der Beispieldatenbank transferiert das folgende Beispiel den Mitarbeiter Philip Chin (employee 129) von der Verkaufsabteilung in die Marketingabteilung.

```
UPDATE GROUPO.Employees
SET DepartmentID = 400
WHERE EmployeeID = 129;
```

Bei Verwendung der Beispieldatenbank nummeriert dieses Beispiel alle bestehenden Bestellaufträge neu, indem 2000 von der ID abgezogen wird.

```
UPDATE GROUPO.SalesOrders AS orders
SET orders.ID = orders.ID - 2000
ORDER BY orders.ID ASC;
```

Diese Aktualisierung ist nur möglich, wenn der Fremdschlüssel der Tabelle 'SalesOrderItems' (der den Primärschlüssel SalesOrders.ID referenziert) mit der Aktion `ON UPDATE CASCADE` definiert wird. Die Tabelle 'SalesOrderItems' wird dann ebenfalls aktualisiert. Da die Anweisung eine `ORDER BY`-Klausel festlegt und das Sortierfolge Attribut auch in der SET-Klausel angegeben ist, muss die Option `ansi_update_constraints` auf "Off" gesetzt werden, weil sonst ein Fehler zurückgegeben wird.

In diesem Beispiel, für das auf die Beispieldatenbank zurückgegriffen wurde, wird der Preis für ein Produkt auf Isolationsstufe 2 geändert, statt die aktuelle Isolationsstufeneinstellung der Datenbank zu verwenden.

```
UPDATE GROUPO.Products
SET UnitPrice = 7.00
WHERE ID = 501
OPTION( isolation_level = 2 );
```

Für dieses Beispiel muss die `ansi_update_constraints`-Option auf einen anderen Wert als "Strict" gesetzt werden. In diesem Beispiel, für das erneut auf die Beispieldatenbank zurückgegriffen wurde, wird Syntax 2 verwendet, um den Lagerbestand derjenigen T-Shirts zurückzusetzen, für die mindestens eine Bestellung vorhanden ist, deren Stückzahl den aktuellen Lagerbestand überschreitet:

```
UPDATE GROUPO.Products AS a
SET Quantity = 0
FROM GROUPO.Products a JOIN GROUPO.SalesOrderItems b ON a.ID = b.ProductID
WHERE a.Name = 'Tee Shirt' AND b.Quantity > a.Quantity;
```

Für dieses Beispiel muss die `ansi_update_constraints`-Option auf einen anderen Wert als "Strict" gesetzt werden. In diesem Beispiel wird Syntax 1 verwendet, um sowohl den Lagerbestand für diese T-Shirts zurückzusetzen als auch das Lieferdatum (`ShipDate`) für die T-Shirt-Bestellung auf das heutige Datum zurückzusetzen:

```
UPDATE GROUPO.Products a JOIN GROUPO.SalesOrderItems b on a.ID = b.ProductID
SET a.Quantity = 0, b.ShipDate = CAST( NOW() AS DATE)
WHERE a.Name = 'Tee Shirt' AND b.Quantity > a.Quantity
```

Dieses Beispiel zeigt, wie eine Tabelle aktualisiert wird, um eine Spalte auf ihren Standardwert zu setzen. In diesem Beispiel erstellen Sie die Tabelle `MyTable`, füllen sie mit Daten und führen dann eine `UPDATE`-Anweisung aus, in der mit der `SET`-Klausel festgelegt wird, dass die Spaltenwerte auf ihre Standardwerte gesetzt werden.

```
CREATE TABLE MyTable(
    PK INT PRIMARY KEY DEFAULT AUTOINCREMENT,
    TableName CHAR(128) NOT NULL,
    TableNameLen INT DEFAULT 20,
    LastUser CHAR(10) DEFAULT last user,
    LastTime TIMESTAMP DEFAULT TIMESTAMP,
    LastTimestamp TIMESTAMP DEFAULT @@dbts );

INSERT INTO MyTable WITH AUTO NAME
SELECT
    LENGTH(t.table_name) AS TableNameLen,
    t.table_name AS TableName
FROM SYS.SYSTAB t
WHERE table_id<=10;

UPDATE MyTable SET LastTime = DEFAULT, LastTimestamp = DEFAULT
WHERE TableName LIKE '%sys%';
```

VALIDATE LDAP SERVER-Anweisung

Validiert ein LDAP-Serverkonfigurationsobjekt.

Syntax

```
VALIDATE LDAP SERVER { ldapua-server-name | ldapua-server-attrs }
[ CHECK user-id [ user-dn-string ] ]
```

Parameter

ldapua-server-name Der Name des zu validierenden LDAP-Serverkonfigurationsobjekts. Eine vollständige Beschreibung dieser Klausel finden Sie unter der `CREATE LDAP SERVER`-Anweisung.

ldapua-server-attrs Bei der Validierung eines LDAP-Serverkonfigurationsobjekt mit *ldapua-server-attrs*-Angabe werden die angegebenen Attribute validiert. Die URLs werden syntaktisch analysiert, um Syntaxfehler zu ermitteln. Die Validierung wird gestoppt und ein Fehler wird zurückgegeben, wenn ein Syntaxfehler auftritt.

Eine vollständige Beschreibung dieser Klausel finden Sie unter der CREATE LDAP SERVER-Anweisung.

CHECK-Klausel Verwenden Sie diese Klausel, um eine Benutzer-ID anzugeben, nach der auf dem LDAP-Server gesucht werden soll.

Bemerkungen

Beim Ausführen einer VALIDATE LDAP SERVER-Anweisung wird versucht, eine Verbindung mit dem LDAP-Server herzustellen. Wenn ACCESS ACCOUNT und ein Kennwort angegeben sind, werden die Werte verwendet, um die Verbindung mit SEARCH DN URL herzustellen, wobei SEARCH DN URL, ACCESS ACCOUNT und ACCESS ACCOUNT-Kennwort validiert werden.

Beim Einrichten eines neuen Servers für die LDAP-Benutzerauthentifizierung ist diese Anweisung nützlich, um Änderungen an einem LDAP-Serverkonfigurationsobjekt vor dem Übernehmen zu validieren und um Probleme zwischen dem Datenbankserver und dem LDAP-Server zu diagnostizieren.

Privilegien

Sie müssen das MANAGE ANY LDAP SERVER-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „LDAP-Benutzerauthentifizierung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „CREATE LDAP SERVER-Anweisung“ auf Seite 642
- „ALTER LDAP SERVER-Anweisung“ auf Seite 468
- „DROP LDAP SERVER-Anweisung“ auf Seite 811

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird ein LDAP-Serverkonfigurationsobjekt erstellt und eine Verbindung mit dem LDAP-Server hergestellt, und zwar mithilfe des Hostnamens voyager, der Portnummer 389 sowie der in der Definition für apps_primary angegebenen Werte für ACCESS ACCOUNT und Kennwort.

```
CREATE LDAP SERVER apps_primary2
  SEARCH DN
    URL 'ldap://voyager:389/dc=MyCompany,dc=com??sub?cn=*'
    ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'
    IDENTIFIED BY 'Secret99Password'
  AUTHENTICATION URL 'ldap://voyager:389/'
  CONNECTION TIMEOUT 3000
  WITH ACTIVATE;
VALIDATE LDAP SERVER apps_primary2;
```

Im folgenden Beispiel wird eine Verbindung mit dem LDAP-Server hergestellt, und zwar mithilfe des Hostnamens voyager, der Portnummer 389 sowie der in der Definition für apps_primary2 angegebenen Werte für ACCESS ACCOUNT und Kennwort. Außerdem wird überprüft, ob die Benutzer-ID "myusername" gültig ist und mit dem erwarteten Benutzer-DN übereinstimmt:

```
VALIDATE LDAP SERVER apps_primary2
CHECK myusername 'cn=myusername,cn=Users,dc=mycompany,dc=com' ;
```

Wenn das LDAP-Serverkonfigurationsobjekt nicht definiert wurde, können dieselben Überprüfungen durch Angeben der Attribute ausgeführt werden:

```
VALIDATE LDAP SERVER
SEARCH DN
URL 'ldap://voyager:389/dc=MyCompany,dc=com??sub?cn=*'
ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'
IDENTIFIED BY 'Secret99Password'
AUTHENTICATION URL 'ldap://voyager:389/'
CONNECTION TIMEOUT 3000
CHECK myusername 'cn=myusername,cn=Users,dc=mycompany,dc=com' ;
```

VALIDATE-Anweisung

Validiert die aktuelle Datenbank oder eine einzelne Tabelle oder materialisierte Ansicht bzw. einen einzelnen Index in der aktuellen Datenbank.

Syntax 1 - Eine Datenbank validieren

```
VALIDATE { CHECKSUM | DATABASE }
```

Syntax 2 - Tabellen und materialisierte Ansichten validieren

```
VALIDATE {
TABLE [ owner.]table-name
| MATERIALIZED VIEW [ owner.]materialized-view-name }
[ WITH EXPRESS CHECK ]
```

Syntax 3 - Indizes validieren

```
VALIDATE {
INDEX index-name
| [ INDEX ] FOREIGN KEY role-name
| [ INDEX ] PRIMARY KEY }
ON [ owner.]object-name
```

object-name :
table-name
materialized-view-name

Syntax 4 - Textindizes validieren

```
VALIDATE TEXT INDEX index-name
ON [ owner.]table-name
```

Bemerkungen

Verwenden Sie die VALIDATE CHECKSUM-Anweisung, um die Prüfsumme für jede Seite der Datenbank zu validieren. Die VALIDATE CHECKSUM-Anweisung stellt sicher, dass Datenbankseiten

nicht auf der Festplatte geändert wurden. Wenn bei der Datenbankerstellung Prüfsummen aktiviert wurden, wird für jede Datenbankseite eine Prüfsumme ermittelt, bevor sie auf die Festplatte geschrieben wird. `VALIDATE CHECKSUM` liest jede Datenbankseite direkt von der Festplatte (nicht über den Datenbankserver-Cache) und berechnet die Prüfsumme für jede Seite. Wenn die berechnete Prüfsumme für eine Seite nicht mit der gespeicherten Prüfsumme für diese Seite übereinstimmt, tritt ein Fehler auf und das Meldungsfenster des Datenbankservers informiert Sie über die ungültige Seite. Die `VALIDATE CHECKSUM`-Anweisung wird nicht für Datenbanken mit deaktivierten Prüfsummen empfohlen, da sie die gesamte Datenbank von der Festplatte liest.

Verwenden Sie die `VALIDATE DATABASE`-Anweisung, um sicherzustellen, dass die freie Zuordnung Seiten richtig identifiziert (entweder als zugewiesen oder als frei) und dass keine BLOBs verwaist wurden. `VALIDATE DATABASE` führt auch eine Prüfsummenvalidierung und prüft, ob jede Datenbankseite zum richtigen Objekt gehört. Beispielsweise muss auf einer Tabellenseite die Tabellen-ID eine gültige Tabelle identifizieren, deren Definition die aktuelle Seite in der dazugehörigen Menge von Tabellenseiten enthalten muss. Die `VALIDATE DATABASE`-Anweisung bringt Seiten in sequenzieller Reihenfolge in den Datenbankserver-Cache. Dadurch werden diese Seiten validiert, da der Datenbankserver stets den Inhalt und die Prüfsummen von Seiten überprüft, die in den Cache gebracht werden. Wenn Sie die Datenbankvalidierung starten, während der Datenbankaufräumvorgang ausgeführt wird, kann die Validierung erst ausgeführt werden, wenn der Datenbankaufräumvorgang abgeschlossen ist.

Die `VALIDATE TABLE`-Anweisung validiert die angegebene Tabelle und alle dazugehörigen Indizes, indem sie prüft, ob die Menge aller Zeilen und Werte in der Basistabelle mit der Menge von Zeilen und Werten in den einzelnen Indizes übereinstimmt. `VALIDATE TABLE` durchläuft ebenfalls alle BLOBs der Tabelle, überprüft BLOB-Zuordnungen und erkennt verwaiste BLOBs. Die `VALIDATE TABLE`-Anweisung prüft die physische Struktur der Indexseiten der Tabelle und überprüft die Reihenfolge der Index-Hash-Werte und die Eindeutigkeit des Indexes (falls diesbezüglich Anforderungen vorgegeben wurden). Bei Fremdschlüsselindizes wird jeder Wert in der Primärschlüsseltabelle nachgeschlagen, um zu überprüfen, ob die referenzielle Integrität intakt ist, wenn die `WITH EXPRESS CHECK`-Klausel nicht angegeben ist. Da die `VALIDATE TABLE`-Anweisung, z.B. `VALIDATE DATABASE`, den Datenbankserver-Cache verwendet, überprüft der Datenbankserver auch die Prüfsummen und die grundlegende Gültigkeit aller Seiten, die von einer Tabelle und den dazugehörigen Indizes verwendet werden.

Die `VALIDATE INDEX`-Anweisung führt dieselben Vorgänge durch wie die `VALIDATE TABLE`-Anweisung, außer dass sie nur den angegebenen Index und die Basistabelle validiert, während andere Indizes nicht geprüft werden. Bei Fremdschlüsselindizes wird jeder Wert in der Primärschlüsseltabelle nachgeschlagen, um zu überprüfen, ob die referenzielle Integrität intakt ist, wenn die `WITH EXPRESS CHECK`-Klausel nicht angegeben ist. Durch Angabe der `WITH EXPRESS CHECK`-Klausel deaktivieren Sie die Prüfung der referenziellen Integrität. Damit können Sie die Performance signifikant steigern. Wenn der angegebene Index kein Fremdschlüsselindex ist, hat `WITH EXPRESS CHECK` keine Auswirkungen.

Verwenden Sie die `VALIDATE INDEX`-Anweisung, um zu überprüfen, ob die Positionsdaten für die Begriffe im Textindex intakt sind. Wenn die Positionsdaten nicht intakt sind, wird ein Fehler generiert und Sie müssen den Textindex neu erstellen. Wenn der Textindex entweder automatisch oder manuell ist, können Sie ihn neu erstellen, indem Sie die `REFRESH TEXT INDEX`-Anweisung ausführen.

Wenn der generierte Fehler einen Soforttextindex betrifft, müssen Sie diesen löschen und einen neuen erstellen.

Vorsicht

Die Validierung einer Tabelle oder einer ganzen Datenbank darf nur durchgeführt werden, wenn keine Verbindungen Änderungen in der Datenbank durchführen, weil sonst möglicherweise Fehler über eine Datenbankbeschädigung gemeldet werden, obwohl eine solche nicht vorliegt.

Privilegien

Sie müssen das VALIDATE ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „REFRESH TEXT INDEX-Anweisung“ auf Seite 990
- „DROP TEXT INDEX-Anweisung“ auf Seite 834
- „CREATE TEXT INDEX-Anweisung“ auf Seite 759
- „Validierungs-Dienstprogramm (dbvalid)“ [*SQL Anywhere Server - Datenbankadministration*]
- „sa_validate-Systemprozedur“ auf Seite 1352
- „sa_clean_database-Systemprozedur“ auf Seite 1174
- „Datenbankvalidierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE DATABASE-Anweisung“ auf Seite 583
- „CREATE INDEX-Anweisung“ auf Seite 638

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Im folgenden Beispiel wird die Tabelle "Products" validiert:

```
VALIDATE TABLE GROUP0.Products;
```

WAITFOR-Anweisung

Verzögert die Verarbeitung für die aktuelle Verbindung um einen bestimmten Zeitraum oder bis zu einem bestimmten Zeitpunkt.

Syntax

```
WAITFOR {  
  DELAY time  
  | TIME time }  
[ CHECK EVERY integer ]  
[ AFTER MESSAGE BREAK ]
```

time : *string*

Parameter

DELAY-Klausel Wenn DELAY verwendet wird, unterbricht das System die Verarbeitung für das angegebene Intervall.

TIME-Klausel Wenn TIME angegeben wird, unterbricht das System die Verarbeitung, bis die Datenbankserverzeit den angegebenen Zeitpunkt erreicht. Wenn die angegebene Serverzeit die angegebene Zeit überschreitet, wird die Verarbeitung bis zum selben Zeitpunkt des folgenden Tages unterbrochen.

CHECK EVERY-Klausel Diese optionale Klausel steuert, wie oft die WAITFOR-Anweisung aktiviert wird. Standardmäßig wird sie alle 5 Sekunden aktiviert. Der Wert wird in Millisekunden angegeben, und der Mindestwert beträgt 250 Millisekunden.

AFTER MESSAGE BREAK-Klausel Die WAITFOR-Anweisung kann zum Warten auf eine Nachricht von einer anderen Verbindung verwendet werden. Wenn eine Nachricht empfangen wird, wird sie normalerweise an die Anwendung weitergeleitet, die die WAITFOR-Anweisung ausgeführt hat, und die WAITFOR-Anweisung wartet weiter. Wenn die AFTER MESSAGE BREAK-Klausel angegeben ist und eine Nachricht von einer anderen Verbindung empfangen wird, schließt die WAITFOR-Anweisung die Ausführung ab. Der Nachrichtentext wird nicht an die Anwendung weitergeleitet, aber Sie können auf ihn zugreifen, indem Sie die MessageReceived-Verbindungseigenschaft verwenden.

Bemerkungen

Die WAITFOR-Anweisung wird regelmäßig aktiviert (standardmäßig alle 5 Sekunden), um zu überprüfen, ob sie abgebrochen wurde oder ob Nachrichten empfangen wurden. Wenn keines von beiden eingetreten ist, wartet die Anweisung weiter.

Da geplante Ereignisse auf ihrer eigenen Verbindung ausgeführt werden, sind geplante Ereignisse oft eine bessere Wahl als die Verwendung von WAITFOR TIME.

Privilegien

Keine

Nebenwirkungen

Die Implementierung der WAITFOR-Anweisung führt dazu, dass der Worker, der die Anweisung bearbeitet, während der Wartezeit blockiert. Dies reduziert die Anzahl der verfügbaren Worker im Worker-Pool.

Siehe auch

- „SQL Anywhere-Threading“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „CREATE EVENT-Anweisung“ auf Seite 606
- „MESSAGE-Anweisung“ auf Seite 959
- MessageReceived-Verbindungseigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiele

Mit dem folgenden Beispiel wird eine Wartezeit von drei Sekunden festgelegt:

```
WAITFOR DELAY '00:00:03';
```

Mit dem folgenden Beispiel wird eine Wartezeit von 0,5 Sekunden (500 Millisekunden) festgelegt:

```
WAITFOR DELAY '00:00:00:500';
```

Mit dem folgenden Beispiel wird festgelegt, dass die Verarbeitung bis 20 Uhr warten soll:

```
WAITFOR TIME '20:00';
```

Im folgenden Beispiel schließt die WAITFOR-Anweisung der 'connection 1' die Ausführung ab, wenn sie die Nachricht von 'connection 2' empfängt:

```
// connection 1:
BEGIN
  DECLARE msg LONG VARCHAR;
  LOOP // forever
    WAITFOR DELAY '00:05:00' AFTER MESSAGE BREAK;
    SET msg = CONNECTION_PROPERTY('MessageReceived');
    IF msg != '' THEN
      MESSAGE 'Msg: ' || msg TO CONSOLE;
    END IF;
  END LOOP
END;
// connection 2:
MESSAGE 'here it is' FOR connection 1
```

WHENEVER-Anweisung [ESQL]

Gibt die Fehlerbehandlung in Embedded SQL-Programmen an.

Syntax

```
WHENEVER {
  SQLERROR
| SQLWARNING
| NOTFOUND }
GOTO
  label
| STOP
| CONTINUE
| { C-code; }
```

label : *identifizier*

Bemerkungen

Die WHENEVER-Anweisung wird verwendet, um Fehlermeldungen, Warnungen und besondere Bedingungen abzufangen, auf welche die Datenbank bei der Verarbeitung von SQL-Anweisungen trifft. Die Anweisung kann in einem Embedded SQL-Programm an eine beliebige Stelle gesetzt werden und erzeugt keinen Code. Der Präprozessor erzeugt nach jeder aufeinanderfolgenden SQL-Anweisung Codes. Die Fehleraktion bleibt für alle Embedded SQL-Anweisungen aus der Quellzeile der WHENEVER-

Anweisung bis zur nächsten WHENEVER-Anweisung mit demselben Fehlerzustand oder bis zum Ende der Quelldatei gültig.

Hinweis

Fehlerbedingungen basieren auf der Positionierung in der Quelldatei in der C-Sprache und nicht auf der Ausführung der Anweisungen.

Die Standardaktion ist CONTINUE.

Diese Anweisung wird aus Gründen der Benutzerfreundlichkeit in einfachen Programmen bereitgestellt. Meistens ist es am einfachsten, das sqlcode-Feld des SQLCA (SQLCODE) direkt zu überprüfen, um Fehlerzustände zu prüfen. In diesem Fall würde die WHENEVER-Anweisung nicht verwendet werden. Die WHENEVER-Anweisung veranlasst nämlich den Präprozessor nur dazu, nach jeder Anweisung einen *if(SQLCODE)*-Test auszugeben.

Privilegien

Keine.

Nebenwirkungen

Keine.

Standards und Kompatibilität

- **SQL/2008** Eine Ausnahmebedingungsdeklaration, die mit der WHENEVER-Anweisung durchgeführt wird, stellt eine Kernfunktion des SQL/2008-Standards dar. Der Standard verwendet das Schlüsselwort SQLEXCEPTION anstelle von SQLERROR. Die Möglichkeit, C-Code direkt in die WHENEVER-Anweisung einzubeziehen, und nicht nur ein Anweisungslabel, ist eine Erweiterung des Herstellers. Die Aktion STOP ist ebenfalls eine Erweiterung des Herstellers.

Beispiel

```
EXEC SQL WHENEVER NOTFOUND GOTO done;
EXEC SQL WHENEVER SQLERROR
{
    PrintError( &sqlca );
    return( FALSE );
};
```

WHILE-Anweisung [T-SQL]

Ermöglicht die wiederholte Ausführung einer Anweisung oder zusammengesetzten Anweisung.

Syntax

WHILE *search-condition statement*

Bemerkungen

Die WHILE-Bedingung hat nur Einfluss auf die Ausführung einer einzelnen SQL-Anweisung, es sei denn, die Anweisungen sind in einer zusammengesetzten Anweisung zwischen den Schlüsselwörtern BEGIN und END zusammengefasst.

Die BREAK-Anweisung und die CONTINUE-Anweisung können verwendet werden, um die Ausführung der Anweisungen in der zusammengesetzten Anweisung zu steuern. Die BREAK-Anweisung beendet die Schleife, und die Ausführung wird nach dem Schlüsselwort END, welches das Ende der Schleife markiert, wieder aufgenommen. Die CONTINUE-Anweisung bewirkt, dass die WHILE-Schleife wieder neu beginnt und alle Anweisungen nach CONTINUE übersprungen werden.

Privilegien

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „LOOP-Anweisung“ auf Seite 950
- „CONTINUE-Anweisung“ auf Seite 581

Standards und Kompatibilität

- **SQL/2008** Transact-SQL-Erweiterung. Die WHILE-Anweisung ist Teil der optionalen SQL/2008-Sprachenfunktion P002 (Verarbeitungsvollständigkeit). Die Transact-SQL-Variante der WHILE-Anweisung enthält END WHILE nicht.

Beispiel

Der folgende Beispielcode veranschaulicht den Gebrauch von WHILE:

```
WHILE ( SELECT AVG(UnitPrice) FROM GROUP0.Products ) < $30
BEGIN
    UPDATE GROUP0.Products
    SET UnitPrice = UnitPrice + 2
    IF ( SELECT MAX(UnitPrice) FROM GROUP0.Products ) > $50
        BREAK
END
```

Die BREAK-Anweisung unterbricht die WHILE-Schleife, wenn der Preis für das teuerste Produkt über \$50 liegt. Sonst setzt sich die Schleife fort, bis der Durchschnittspreis größer oder gleich \$30 ist.

WINDOW-Klausel

Bestimmt das ganze Fenster oder einen Fensterabschnitt für die Verwendung mit Fensterfunktionen wie AVG und RANK in einer SELECT-Anweisung.

Syntax

WINDOW *window-expression*, ...

window-expression : *new-window-name* **AS** (*window-spec*)

window-spec :

[*existing-window-name*]

[**PARTITION BY** *expression*, ...]

```
[ ORDER BY expression [ ASC | DESC ], ... ]
[ { ROWS | RANGE } { window-frame-start | window-frame-between } ]
```

```
window-frame-start :
{ UNBOUNDED PRECEDING
| unsigned-integer PRECEDING
| CURRENT ROW }
```

```
window-frame-between :
BETWEEN window-frame-bound1 AND window-frame-bound2
```

```
window-frame-bound :
window-frame-start
| UNBOUNDED FOLLOWING
| unsigned-integer FOLLOWING
```

Parameter

PARTITION BY-Klausel Die PARTITION BY-Klausel organisiert die Ergebnismenge in logische Gruppen, basierend auf den eindeutigen Werten im angegebenen Ausdruck. Wenn diese Klausel mit Fensterfunktionen verwendet wird, werden die Funktionen jeweils an den einzelnen Fensterabschnitten angewendet. Wenn Sie z.B. nach PARTITION BY einen Spaltennamen angeben, wird die Ergebnismenge anhand von unterschiedlichen Werten in der Spalte aufgeteilt.

Wenn diese Klausel weggelassen wird, wird die gesamte Ergebnismenge als ein Fensterabschnitt angesehen.

PARTITION BY *expression* kann kein Ganzzahlliteral sein.

ORDER BY-Klausel Die ORDER BY-Klausel legt fest, wie die Zeilen in jedem Fensterabschnitt der Ergebnismenge sortiert werden. Sie können die Sortierfolge weiter steuern, indem Sie ASC für eine aufsteigende Sortierfolge (Standardwert) oder DESC für eine absteigende Sortierfolge angeben.

ORDER BY *expression* kann keine Ganzzahl sein.

Wenn diese Klausel weggelassen wird, gibt SQL Anywhere Zeilen in der effizientesten Reihenfolge zurück. Die Präsentation der Ergebnismengen kann unterschiedlich sein, je nachdem, wann Sie zuletzt auf die Zeile zugegriffen haben.

ROWS-Klausel und RANGE-Klausel Verwenden Sie die ROWS- oder RANGE-Klausel, um die Größe des Fensters auszudrücken. Die Fenstergröße kann eine, viele oder alle Zeilen eines Fensterabschnitts umfassen. Sie können die Größe des Fensters entweder als Bereich von Datenwerten ab der aktuellen Zeile (RANGE) oder als Anzahl der physischen Zeilen ab der aktuellen Zeile (ROWS) ausdrücken.

Wenn Sie die RANGE-Klausel verwenden, müssen Sie auch eine ORDER BY-Klausel angeben, weil Bereichsberechnungen ein Sortieren der Werte erfordern. Die ORDER BY-Klausel für Bereiche muss einen Ausdruck enthalten und dieser Ausdruck muss entweder ein Datum oder einen numerischen Wert ergeben.

Wenn Sie keine ROWS- oder RANGE-Klausel angeben, benutzt der Datenbankserver Standardfenstergrößen basierend auf einer eventuell vorhandenen ORDER BY-Klausel.

- **PRECEDING-Klausel** Verwenden Sie die PRECEDING-Klausel, um die erste Zeile des Fensters zu bestimmen, wobei die aktuelle Zeile als Bezugspunkt verwendet wird. Die Startzeile wird als die Anzahl der Zeilen ausgedrückt, die der aktuellen Zeile vorhergehen. Beispiel: 5 PRECEDING bewirkt, dass das Fenster mit der fünften Zeile beginnt, die der aktuellen Zeile vorhergeht.

Verwenden Sie UNBOUNDED PRECEDING, um die erste Zeile im Fenster als die erste Zeile im Fensterabschnitt festzulegen.

- **BETWEEN-Klausel** Verwenden Sie die BETWEEN-Klausel, um die erste und letzte Zeile des Fensters zu bestimmen, wobei die aktuelle Zeile als Bezugspunkt verwendet wird. Erste und letzte Zeilen werden als die Anzahl der Zeilen ausgedrückt, die der aktuellen Zeile vorhergehen bzw. auf sie folgen. Beispiel: BETWEEN 3 PRECEDING AND 5 FOLLOWING legt fest, dass das Fenster mit der dritten Zeile vor der aktuellen Zeile beginnt und mit der fünften Zeile nach der aktuellen Zeile endet.

Verwenden Sie BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING, um die erste und die letzte Zeile im Fenster als die erste bzw. letzte Zeile im Fensterabschnitt festzulegen. Dies entspricht dem Standardverhalten, wenn keine ROW- oder RANGE-Klausel angegeben ist.

- **FOLLOWING-Klausel** Verwenden Sie die FOLLOWING-Klausel, um die letzte Zeile des Fensters zu bestimmen, wobei die aktuelle Zeile als Bezugspunkt verwendet wird. Die letzte Zeile wird als die Anzahl der Zeilen ausgedrückt, die auf die aktuelle Zeile folgen.

Verwenden Sie UNBOUNDED FOLLOWING, um die letzte Zeile im Fenster als die letzte Zeile im Fensterabschnitt festzulegen.

Bemerkungen

Die WINDOW-Klausel muss in einer SELECT-Anweisung vor der ORDER BY-Klausel erscheinen.

Mit Ausnahme der LIST-Funktion können alle Aggregatfunktionen als Fensterfunktionen verwendet werden. Rangfolge-Aggregatfunktionen (RANK, DENSE_RANK, PERCENT_RANK, CUME_DIST und ROW_NUMBER) jedoch erfordern eine ORDER BY-Klausel und erlauben keine ROW- oder RANGE-Klausel in der WINDOW-Klausel bzw. in der Inline-Definition. Für alle anderen Fensterfunktionen können Sie jede der Klauseln verwenden.

Privilegien

Keine.

Siehe auch

- „SELECT-Anweisung“ auf Seite 1020
- „OLAP-Unterstützung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Fensterdefinitionen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Standards und Kompatibilität

- **SQL/2008** Die WINDOW-Klausel und die Fenster-Aggregatfunktionen umfassen die optionalen SQL/2008-Sprachenfunktionen T611 (grundlegende OLAP-Vorgänge) und T612 (erweiterte OLAP-

Vorgänge). Das Fensterfunktionen FIRST_VALUE und LAST_VALUE sind Erweiterungen des Herstellers.

Beispiel

Das folgende Beispiel gibt das Gehalt eines Mitarbeiters und das Durchschnittsgehalt aller Mitarbeiter im gewählten Bundesstaat zurück. Die Ergebnisse werden nach Bundesstaat (State) und dann nach Nachnamen (Surname) sortiert.

```
SELECT EmployeeID, Surname, Salary, State,  
       AVG( Salary ) OVER Salary_Window  
FROM GROUPO.Employees  
WINDOW Salary_Window AS ( PARTITION BY State )  
ORDER BY State, Surname;
```

WRITETEXT-Anweisung [T-SQL]

Ermöglicht das nicht protokollierte Aktualisieren einer CHAR-, NCHAR- oder BINARY-Spalte. Diese Funktion wird nur für die Kompatibilität mit Transact-SQL verfügbar gemacht und ihre Verwendung wird nicht empfohlen.

Syntax

```
WRITETEXT table-name.column-name  
text-pointer [ WITH LOG ] data
```

Bemerkungen

Aktualisiert einen vorhandenen Spaltenwert. Die Aktualisierung wird nicht in das Transaktionslog eingetragen, es sei denn, die Option WITH LOG wird bereitgestellt. Sie können in Ansichten keine WRITETEXT-Vorgänge ausführen.

Privilegien

Keine.

Nebenwirkungen

WRITETEXT löst keine Trigger aus, und standardmäßig werden WRITETEXT-Vorgänge nicht in ein Transaktionslog eingetragen.

Siehe auch

- [„READTEXT-Anweisung \[T-SQL\]“](#) auf Seite 986
- [„TEXTPTR-Funktion \[Text und Bild\]“](#) auf Seite 408

Standards und Kompatibilität

- **SQL/2008** Transact-SQL-Erweiterung.

Beispiel

Das folgende Embedded-SQL-Codefragment veranschaulicht die Verwendung der WRITETEXT-Anweisung. Die SELECT-Anweisung in diesem Beispiel gibt eine einzelne Zeile zurück. Das Beispiel ersetzt den Inhalt der Spalte "Description" in der angegebenen Zeile durch einen neuen Wert.

```
EXEC SQL create variable txtptr binary(16);
EXEC SQL set txtptr =
    ( SELECT textptr(Description)
      FROM MarketingInformation
      WHERE ProductID = '500' );
EXEC SQL writetext MarketingInformation.Description
    txtptr 'newdata';
```

Tabellen

Systemtabellen

Die Struktur jeder Datenbank ist in einer Reihe von Systemtabellen beschrieben. Systemtabellen gehören dem SYS-Benutzer. Der Inhalt dieser Tabellen kann nur durch den Datenbankserver geändert werden. Die Anweisungen UPDATE, DELETE und INSERT können nicht zum Ändern des Inhalts dieser Tabellen verwendet werden. Außerdem kann die Struktur dieser Tabellen nicht mithilfe der Anweisungen ALTER TABLE und DROP verändert werden. Systemansichten werden aktualisiert, wenn ein Checkpoint auftritt.

Daten in Systemtabellen werden über ihre zugewiesenen **Systemansichten** angezeigt. Jeder Systemtabellenname beginnt mit einem großen Buchstaben I. Jede entsprechende Systemansicht hat den gleichen Namen, jedoch ohne das I am Anfang.

Siehe auch

- „Systemansichten“ auf Seite 1437

DUMMY-Systemtabelle

Spaltenname	Spaltentyp	Spalten-Integritätsregel	Tabellen-Integritätsregeln
dummy_col	INTEGER	NOT NULL	

Die DUMMY-Tabelle wird als eine schreibgeschützte Tabelle vorgegeben, die immer nur genau eine Zeile hat. Damit können Informationen über die Datenbank geholt werden, wie im folgenden Beispiel, das die aktuelle Benutzer-ID und das aktuelle Datum aus der Datenbank holt.

```
SELECT USER, today(*) FROM SYS.DUMMY;
```

```
SELECT USER, today(*);
```

dummy_col Diese Spalte wird nicht verwendet. Es gibt sie, da eine Tabelle nicht ohne Spalten erstellt werden kann.

Es ist günstiger, die Kosten aus der Tabelle SYS.DUMMY zu lesen, als die Kosten aus einer ähnlichen, vom Benutzer erstellten Tabelle, zu lesen, weil für die Tabellenseite von SYS.DUMMY keine Sperre besteht.

Zugriffspläne werden ohne Scans der SYS.DUMMY-Tabelle erstellt. Stattdessen werden Referenzen zu SYS.DUMMY durch einen Zeilenkonstruktor-Algorithmus ersetzt, der die Tabellenreferenz virtualisiert. Dadurch werden Konflikte in Zusammenhang mit der Verwendung von SYS.DUMMY vermieden. DUMMY erscheint in kurzen, ausführlichen und grafischen Plänen weiterhin als Tabellen- bzw. Korrelationsname.

ISYSARTICLE-Systemtabelle

Jede Zeile in der ISYSARTICLE-Systemtabelle beschreibt einen Artikel in einer Publikation.

Siehe auch

- [„SYSARTICLE-Systemansicht“ auf Seite 1438](#)

ISYSARTICLECOL-Systemtabelle

Jede Zeile in der ISYSARTICLECOL-Systemtabelle kennzeichnet eine Spalte in einem Artikel.

Siehe auch

- [„SYSARTICLECOL-Systemansicht“ auf Seite 1438](#)

ISYSATTRIBUTE-Systemtabelle

Diese Tabelle dient ausschließlich der internen Verwendung.

ISYSATTRIBUTENAME-Systemtabelle

Diese Tabelle dient ausschließlich der internen Verwendung.

ISYSCAPABILITY-Systemtabelle

Jede Zeile in der ISYSCAPABILITY-Systemtabelle kennzeichnet eine Fähigkeit auf einem entfernten Server.

Siehe auch

- [„YSYSCAPABILITY-Systemansicht“ auf Seite 1439](#)

ISYSCERTIFICATE-Systemtabelle

In jeder Zeile der ISYSCERTIFICATE-Systemtabelle wird ein Zertifikat als Text im PEM-Format gespeichert.

Siehe auch

- [„SYSCERTIFICATE-Systemansicht“ auf Seite 1440](#)

ISYSCHECK-Systemtabelle

Jede Zeile in der ISYSCHECK-Systemtabelle kennzeichnet eine benannte Prüf-Integritätsregel in einer Tabelle.

Siehe auch

- [„SYSCHECK-Systemansicht“ auf Seite 1440](#)

ISYSCOLPERM-Systemtabelle

Jede Zeile in der ISYSCOLPERM-Systemtabelle beschreibt ein UPDATE-, SELECT- oder REFERENCES-Privileg für eine Spalte.

Siehe auch

- [„SYSCOLPERM-Systemansicht“ auf Seite 1441](#)

ISYSCOLSTAT-Systemtabelle

Die ISYSCOLSTAT-Systemtabelle enthält die vom Optimierer verwendeten Spaltenstatistiken.

Hinweis

Bei Datenbanken, die ab SQL Anywhere 16 erstellt werden, ist diese Tabelle immer verschlüsselt, um die Daten vor nicht autorisiertem Zugriff zu schützen.

Siehe auch

- [„SYSCOLSTAT-Systemansicht“ auf Seite 1441](#)

ISYSCONSTRAINT-Systemtabelle

Jede Zeile in der ISYSCONSTRAINT-Systemtabelle beschreibt eine benannte Integritätsregel für alle Tabellen außer Systemtabellen.

Siehe auch

- [„SYSCONSTRAINT-Systemansicht“ auf Seite 1442](#)

ISYSDEPENDENCY-Systemtabelle

Jede Zeile in der ISYSDEPENDENCY-Systemtabelle beschreibt eine Tabellen- oder Ansichtsabhängigkeit.

Siehe auch

- [„SYSDEPENDENCY-Systemansicht“ auf Seite 1445](#)

ISYSDBFILE-Systemtabelle

Jede Zeile in der Systemtabelle ISYSDBFILE beschreibt einen DBSpace.

Siehe auch

- [„SYSDBFIL-Systemansicht“ auf Seite 1443](#)

ISYSDBSpace-Systemtabelle

Jede Zeile in der Systemtabelle ISYSDBSpace beschreibt einen DBSpace.

Siehe auch

- [„SYSDBSpace-Systemansicht“ auf Seite 1444](#)

ISYSDBSpacePERM-Systemtabelle

Jede Zeile in der ISYSDBSpacePERM-Systemtabelle beschreibt ein Privileg für einen DBSpace.

Siehe auch

- [„SYSDBSpacePERM-Systemansicht“ auf Seite 1445](#)

ISYSDOMAIN-Systemtabelle

Jedem der vordefinierten Datentypen (auch **Domäne** genannt) ist eine Nummer zugewiesen. Die Tabelle ISYSDOMAIN ist zur Information vorgesehen, um die Zuordnung zwischen diesen Nummern und den entsprechenden Datentypen zu zeigen. Diese Tabelle wird nie geändert.

Siehe auch

- [„SYSDOMAIN-Systemansicht“ auf Seite 1446](#)

ISYSEVENT-Systemtabelle

Jede Zeile in der ISYSEVENT-Systemtabelle beschreibt ein Ereignis, das mit CREATE EVENT erstellt wurde.

Siehe auch

- [„SYSEVENT-Systemansicht“ auf Seite 1446](#)

ISYSEXTERNENV-Systemtabelle

Jede Zeile in der ISYSEXTERNENV-Systemtabelle beschreibt die Informationen, die benötigt werden, um die einzelnen externen Umgebungen zu identifizieren und zu starten.

Siehe auch

- [„SYSEXTERNENV-Systemansicht“ auf Seite 1448](#)

ISYSEXTERNENVOBJECT-Systemtabelle

Jede Zeile in der ISYSEXTERNENVOBJECT-Systemtabelle beschreibt ein installiertes externes Objekt.

Siehe auch

- [„SYSEXTERNENVOBJECT-Systemansicht“ auf Seite 1450](#)

ISYSEXTERNLOGIN-Systemtabelle

Jede Zeile in der ISYSEXTERNLOGIN-Systemtabelle beschreibt ein externes Login für Ferndatenzugriff.

Hinweis

Bei Datenbanken, die ab SQL Anywhere 16 erstellt werden, ist diese Tabelle immer verschlüsselt, um die Daten vor nicht autorisiertem Zugriff zu schützen.

Siehe auch

- [„SYSEXTERNLOGIN-Systemansicht“ auf Seite 1451](#)

ISYSFKEY-Systemtabelle

Jede Zeile in der ISYSFKEY-Systemtabelle beschreibt einen Fremdschlüssel in der Datenbank

Siehe auch

- [„SYSFKEY-Systemansicht“ auf Seite 1452](#)

ISYSHISTORY-Systemtabelle

Jede Zeile in der ISYSHISTORY-Systemtabelle gibt eine Uhrzeit an, zu der die Datenbank mit einer anderen Version der Software bzw. auf einer anderen Plattform gestartet wurde.

Siehe auch

- [„SYSHISTORY-Systemansicht“ auf Seite 1453](#)

ISYSIDX-Systemtabelle

Jede Zeile in der ISYSIDX-Systemtabelle beschreibt einen Index in der Datenbank.

Siehe auch

- [„SYSIDX-Systemansicht“ auf Seite 1456](#)

ISYSIDXCOL-Systemtabelle

Jede Zeile in der ISYSIDXCOL-Systemtabelle beschreibt eine Spalte in einem Index.

Siehe auch

- „SYSIDXCOL-Systemansicht“ auf Seite 1457

ISYSJAR-Systemtabelle

Jede Zeile in der ISYSJAR-Systemtabelle definiert eine JAR-Datei im System.

Siehe auch

- „SYSJAR-Systemansicht“ auf Seite 1458

ISYSJARCOMPONENT-Systemtabelle

Jede Zeile in der ISYSJARCOMPONENT-Systemtabelle definiert eine JAR-Dateikomponente.

Siehe auch

- „SYSJARCOMPONENT-Systemansicht“ auf Seite 1459

ISYSJAVACLASS-Systemtabelle

Jede Zeile in der ISYSJAVACLASS-Systemtabelle beschreibt eine Java-Klasse.

Siehe auch

- „SYSJAVACLASS-Systemansicht“ auf Seite 1460

ISYSLDAPSERVER-Systemtabelle

Die SYSLDAPSERVER-Systemansicht enthält eine Zeile für jedes in der Datenbank konfigurierte LDAP SERVER-Objekt. Ein LDAP SERVER-Objekt enthält die Konfigurationsinformationen, die benötigt werden, um eine Verbindung mit einem LDAP-Server außerhalb von SQL Anywhere herzustellen.

Siehe auch

- „SYSLDAPSERVER-Systemansicht“ auf Seite 1461

ISYSLOGINMAP-Systemtabelle

Die ISYSLOGINMAP-Systemtabelle enthält alle Benutzerprofilnamen, die verwendet werden können, um Verbindungen mit der Datenbank mithilfe eines integrierten Logins oder eines Kerberos-Logins herzustellen.

Siehe auch

- [„SYSLOGINMAP-Systemansicht“ auf Seite 1462](#)

ISYSLOGINPOLICY-Systemtabelle

Jede Zeile in der ISYSLOGINPOLICY-Systemtabelle beschreibt eine Login-Richtlinie.

Siehe auch

- [„SYSLOGINPOLICY-Systemansicht“ auf Seite 1463](#)

ISYSLOGINPOLICYOPTION-Systemtabelle

Jede Zeile in der ISYSLOGINPOLICYOPTION-Systemtabelle beschreibt eine Option für eine Login-Richtlinie.

Siehe auch

- [„SYSLOGINPOLICYOPTION-Systemansicht“ auf Seite 1463](#)

ISYSMIRROROPTION-Systemtabelle

Jede Zeile in der ISYSMIRROROPTION-Systemtabelle beschreibt eine Option für einen Spiegelserver.

Siehe auch

- [„SYSMIRROROPTION-Systemansicht“ auf Seite 1463](#)

ISYSMIRRORSERVER-Systemtabelle

Jede Zeile in der ISYSMIRRORSERVER-Systemtabelle beschreibt eine Option für einen Spiegelserver.

Siehe auch

- [„SYSMIRRORSERVER-Systemansicht“ auf Seite 1464](#)

ISYSMIRRORSERVEROPTION-Systemtabelle

Jede Zeile in der SYSMIRRORSERVEROPTION-Systemtabelle beschreibt eine Option für einen Spiegelserver.

Siehe auch

- [„SYSMIRRORSERVEROPTION-Systemansicht“ auf Seite 1465](#)

ISYSMVOPTION-Systemtabelle

Jede Zeile in der ISYSMVOPTION-Systemtabelle enthält den Wert einer Erstellungsoption für eine materialisierte Ansicht oder einen Textindex in der Datenbank.

Siehe auch

- [„SYSMVOPTION-Systemansicht“ auf Seite 1465](#)

ISYSMVOPTIONNAME-Systemtabelle

Jede Zeile in der ISYSMVOPTIONNAME-Systemtabelle enthält den Namen einer in ISYSMVOPTION aufgeführten Erstellungsoption für eine materialisierte Ansicht oder einen Textindex.

Siehe auch

- [„SYSMVOPTIONNAME-Systemansicht“ auf Seite 1466](#)

ISYSOBJECT-Systemtabelle

Jede Zeile in der ISYSOBJECT-Systemansicht beschreibt eine Prozedur in der Datenbank. Beispiele für Datenbankobjekte sind Tabellen, Ansichten, Spalten, Indizes und Prozeduren.

Siehe auch

- [„SYSOBJECT-Systemansicht“ auf Seite 1466](#)

ISYSOPTION-Systemtabelle

Jede Zeile in der ISYSOPTION-Systemtabelle beschreibt die Einstellungen für eine Option für eine Benutzer-ID. Optionseinstellungen werden in der ISYSOPTION-Tabelle durch die SET-Anweisung gespeichert und jeder Benutzer kann seine eigenen Einstellungen für jede Option haben.

Siehe auch

- [„SYSOPTION-Systemansicht“ auf Seite 1467](#)

ISYSOPTSTAT-Systemtabelle

Die ISYSOPTSTAT-Systemtabelle speichert Informationen über die Kostenmodellkalibrierung, wie sie von der ALTER DATABASE CALIBRATE-Anwendung berechnet werden.

Siehe auch

- [„SYSOPTSTAT-Systemansicht“ auf Seite 1468](#)

ISYSPHYSIDX-Systemtabelle

Jede Zeile in der ISYSPHYSIDX-Systemtabelle beschreibt einen physischen Index in der Datenbank.

Siehe auch

- [„SYSPHYSIDX-Systemansicht“ auf Seite 1468](#)

ISYSPROCEDURE-Systemtabelle

Jede Zeile in der ISYSPROCEDURE-Systemtabelle beschreibt eine Prozedur in der Datenbank.

Siehe auch

- [„SYSPROCEDURE-Systemansicht“ auf Seite 1469](#)

ISYSPROCPARM-Systemtabelle

Jede Zeile in der ISYSPROCPARM-Systemtabelle beschreibt einen Parameter für eine Prozedur in der Datenbank.

Siehe auch

- [„SYSPROCPARM-Systemansicht“ auf Seite 1470](#)

ISYSPROCPERM-Systemtabelle

Jede Zeile in der ISYSPROCPERM-Systemtabelle beschreibt einen Benutzer, dem das Privileg zum Aufrufen einer Prozedur erteilt wurde.

Siehe auch

- [„SYSPROCPERM-Systemansicht“ auf Seite 1472](#)

ISYSPROXYTAB-Systemtabelle

Jede Zeile in der ISYSPROXYTAB-Systemtabelle beschreibt eine Proxy-Tabelle.

Siehe auch

- [„SYSPROXYTAB-Systemansicht“ auf Seite 1472](#)

ISYSPUBLICATION-Systemtabelle

Jede Zeile in der ISYSPUBLICATION-Systemtabelle beschreibt eine SQL Remote- oder MobiLink-Publikation.

Siehe auch

- [„SYSADVERTISEMENT-Systemansicht“ auf Seite 1473](#)

ISYSREMARK-Systemtabelle

Jede Zeile in der ISYSREMARK-Systemtabelle beschreibt eine Bemerkung (oder Kommentar) für ein Objekt.

Siehe auch

- [„SYSREMARK-Systemansicht“ auf Seite 1474](#)

ISYSREMOTEOPTION-Systemtabelle

Jede Zeile in der ISYSREMOTEOPTION-Systemtabelle beschreibt die Werte eines SQL Remote-Nachrichtenverbindungsparameters.

Siehe auch

- [„SYSREMOTEOPTION-Systemansicht“ auf Seite 1475](#)

ISYSREMOTEOPTIONTYPE-Systemtabelle

Jede Zeile in der ISYSREMOTEOPTIONTYPE-Systemtabelle beschreibt einen der SQL Remote-Nachrichtenverbindungsparameter.

Siehe auch

- [„SYSREMOTEOPTIONTYPE-Systemansicht“ auf Seite 1475](#)

ISYSREMOTETYPE-Systemtabelle

Die ISYSREMOTETYPE-Systemtabelle enthält Informationen über SQL Remote.

Siehe auch

- [„SYSREMOTETYPE-Systemansicht“ auf Seite 1476](#)

ISYSREMOTEUSER-Systemtabelle

Jede Zeile in der ISYSREMOTEUSER-Systemtabelle beschreibt eine Benutzer-ID mit REMOTE-Privilegien (einen Subskribenten) zusammen mit dem Status von SQL Remote-Nachrichten, die an diesen Benutzer und von diesem Benutzer gesendet wurden.

Siehe auch

- „SYSREMOTEUSER-Systemansicht“ auf Seite 1476
- „REMOTE RESET-Anweisung [SQL Remote]“ auf Seite 995

ISYSROLEGRANT-Systemtabelle

Zeilen in der ISYSROLEGRANT-Systemtabelle enthalten Informationen zu Rollenmitgliedschaft und Typ der Mitgliedschaft.

Siehe auch

- „SYSROLEGRANT-Systemansicht“ auf Seite 1478
- „Konsolidierte Ansicht SYSROLEGRANTS“ auf Seite 1480

ISYSROLEGRANTEXT-Systemtabelle

Die ISYSROLEGRANTEXT-Systemtabelle speichert die Syntaxerweiterungen für die Systemprivilegien SET USER und CHANGE PASSWORD.

Siehe auch

- „SYSROLEGRANTEXT-Systemansicht“ auf Seite 1479

ISYSSCHEDULE-Systemtabelle

Jede Zeile in der ISYSSCHEDULE-Systemtabelle gibt einen Zeitpunkt an, zu dem ein Ereignis ausgelöst werden soll, wie in der SCHEDULE-Klausel des CREATE EVENT festgelegt.

Siehe auch

- „SYSSCHEDULE-Systemansicht“ auf Seite 1481

ISYSSEQUENCE-Systemtabelle

Die ISYSSEQUENCE-Systemtabelle enthält eine Zeile für jede benutzerdefinierte Sequenz.

Siehe auch

- „SYSSEQUENCE-Systemansicht“ auf Seite 1483

ISYSSEQUENCEPERM-Systemtabelle

Die ISYSSEQUENCEPERM-Systemtabelle enthält die Privilegien, die Benutzer oder Gruppen für Sequenzen haben.

Siehe auch

- [„SYSEQUENCEPERM-Systemansicht“ auf Seite 1483](#)

ISYSSERVER-Systemtabelle

Jede Zeile in der ISYSSERVER-Systemtabelle beschreibt einen entfernten Server.

Siehe auch

- [„SYSSERVER-Systemansicht“ auf Seite 1484](#)

ISYSSOURCE-Systemtabelle

Jede Zeile in der ISYSSOURCE-Systemtabelle enthält die Quelle für ein Objekt, das in der ISYBJECT-Systemtabelle aufgelistet ist.

Siehe auch

- [„SYSSOURCE-Systemansicht“ auf Seite 1485](#)

ISYSSPATIALREFERENCESYSTEM-Systemtabelle

Jede Zeile der ISYSSPATIALREFERENCESYSTEM-Systemtabelle beschreibt ein räumliches Bezugssystem, das in der Datenbank festgelegt ist.

Siehe auch

- [„SYSSPATIALREFERENCESYSTEM-Systemansicht“ auf Seite 1485](#)

ISYSSQLSERVERTYPE-Systemtabelle

Diese ISYSSQLSERVERTYPE-Systemtabelle enthält Informationen, die sich auf die Kompatibilität zu Adaptive Server Enterprise beziehen.

Siehe auch

- [„SYSSQLSERVERTYPE-Systemansicht“ auf Seite 1488](#)

ISYSSUBSCRIPTION-Systemtabelle

Jede Zeile in der ISYSSUBSCRIPTION-Systemtabelle beschreibt eine Subskription einer Benutzer-ID (die REMOTE-Privilegien haben muss) für eine Publikation.

Siehe auch

- [„SYSSUBSCRIPTION-Systemansicht“ auf Seite 1489](#)

ISYSSYNC-Systemtabelle

Diese Tabelle enthält Informationen, die sich auf die MobiLink-Synchronisation beziehen.

Siehe auch

- „SYSSYNC-Systemansicht“ auf Seite 1489

ISYSSYNCPROFILE-Systemtabelle

Diese Tabelle enthält Informationen im Zusammenhang mit Synchronisationsprofile für MobiLink.

Siehe auch

- „SYSSYNCPROFILE-Systemansicht“ auf Seite 1491

ISYSSYNCSCRIPT-Systemtabelle

Diese Tabelle enthält Informationen, die sich auf die MobiLink-Synchronisationsskripts beziehen.

Siehe auch

- „SYSSYNCSCRIPT-Systemansicht“ auf Seite 1491

ISYSTAB-Systemtabelle

Jede Zeile in der ISYSTAB-Systemtabelle beschreibt eine Tabelle in der Datenbank.

Siehe auch

- „SYSTAB-Systemansicht“ auf Seite 1492

ISYSTABCOL-Systemtabelle

Jede Zeile in der ISYSTABCOL-Systemtabelle beschreibt eine Spalte einer Tabelle in der Datenbank.

Siehe auch

- „SYSTABCOL-Systemansicht“ auf Seite 1495

ISYTEXTCONFIG-Systemtabelle

Jede Zeile in der ISYTEXTCONFIG-Systemtabelle beschreibt eine Textkonfiguration zur Verwendung mit der Volltextsuchfunktion.

Siehe auch

- „SYTEXTCONFIG-Systemansicht“ auf Seite 1499

ISYTEXTIDX-Systemtabelle

Jede Zeile in der ISYTEXTIDX-Systemtabelle beschreibt einen Textindex zur Verwendung mit der Volltextsuchfunktion.

Siehe auch

- [„SYTEXTIDX-Systemansicht“ auf Seite 1501](#)

ISYTEXTIDXTAB-Systemtabelle

Jede Zeile in der ISYTEXTIDXTAB-Systemtabelle beschreibt einen Textindex zur Verwendung mit der Volltextsuchfunktion.

Siehe auch

- [„SYTEXTIDXTAB-Systemansicht“ auf Seite 1502](#)

ISYSTABLEPERM-Systemtabelle

Jede Zeile in der ISYSTABLEPERM-Systemtabelle entspricht einer Tabelle, einer Benutzer-ID, die das Privileg erteilt (**Berechtigungsgeber**), und einer Benutzer-ID, der die Berechtigung erteilt wird (**Berechtigungsempfänger**).

Siehe auch

- [„SYSTABLEPERM-Systemansicht“ auf Seite 1498](#)

ISYSTRIGGER-Systemtabelle

Jede Zeile in der ISYSTRIGGER-Systemtabelle beschreibt einen Trigger in der Datenbank.

Siehe auch

- [„SYSTRIGGER-Systemansicht“ auf Seite 1503](#)

ISYSTYPEMAP-Systemtabelle

Die SYSTYPEMAP-Systemtabelle enthält die Kompatibilitäts-Zuordnungs-Werte für die ISYSSQLSERVERTYPE-Systemtabelle.

Siehe auch

- [„SYSTYPEMAP-Systemansicht“ auf Seite 1505](#)

ISYSUNITOFMEASURE-Systemtabelle

Jede Zeile der ISYSUNITOFMEASURE-Systemansicht beschreibt eine Maßeinheit, die in der Datenbank festgelegt ist.

Siehe auch

- „[ISYSUNITOFMEASURE-Systemansicht](#)“ auf Seite 1505

ISYSUSER-Systemtabelle

Jede Zeile in der ISYSUSER-Systemtabelle beschreibt einen Benutzer im System.

Hinweis

Seit SQL Anywhere 12 ist diese Tabelle immer verschlüsselt, um die Daten vor unberechtigtem Zugriff zu schützen.

Siehe auch

- „[ISYSUSER-Systemansicht](#)“ auf Seite 1506

ISYSUSERMESSAGE-Systemtabelle

Jede Zeile in der ISYSUSERMESSAGE-Systemtabelle enthält eine benutzerdefinierte Meldung für eine Fehlerbedingung.

Siehe auch

- „[ISYSUSERMESSAGE-Systemansicht](#)“ auf Seite 1510

ISYSUSERTYPE-Systemtabelle

Jede Zeile in der ISYSUSERTYPE-Systemtabelle beschreibt einen benutzerdefinierten Datentyp. \\

Siehe auch

- „[ISYSUSERTYPE-Systemansicht](#)“ auf Seite 1510

ISYSVIEW-Systemtabelle

Jede Zeile in der ISYSVIEW-Systemtabelle beschreibt eine Ansicht in der Datenbank. \\

Siehe auch

- „[ISYSVIEW-Systemansicht](#)“ auf Seite 1511

ISYSWEBSERVICE-Systemtabelle

Jede Zeile in der ISYSWEBSERVICE-Systemtabelle beschreibt einen Webdienst. \\

Siehe auch

- „SYSWEBSERVICE-Systemansicht“ auf Seite 1513

Diagnoseprotokollierungstabellen

Es folgen die wichtigsten Tabellen für die Anwendungsprofilerstellung und die Diagnoseprotokollierung. Diese Tabellen gehören dem dbo-Benutzer. Bei vielen dieser Tabellen gibt es eine gemeinsame, globale, temporäre Tabelle mit ähnlichem Namen und Schema. Die Tabelle sa_diagnostic_blocking beispielsweise hat ein globales temporäres Gegenstück, die Tabelle sa_tmp_diagnostic_blocking mit demselben Schema. Während einer Protokollierungssitzung werden Diagnosedaten in diese temporären Tabellen geschrieben. Da temporäre Tabellen nicht protokolliert werden, bieten sie eine erhöhte Performance während einer Protokollierungssitzung, bei der es wichtig ist, die Auswirkungen auf den Server zu minimieren.

Siehe auch

- „Anwendungsprofilerstellung“ [SQL Anywhere Server - SQL-Benutzerhandbuch]
- „Diagnoseprotokollierung“ [SQL Anywhere Server - SQL-Benutzerhandbuch]

sa_diagnostic_auxiliary_catalog-Tabelle

Die Tabelle sa_diagnostic_auxiliary_catalog gehört dem Benutzer dbo und wird verwendet, um Datenbankobjekte der Produktionsdatenbank bzw. der Protokollierungsdatenbank zuzuordnen. Objekte sind Benutzertabellen, Prozeduren und Funktionen. Diese Tabelle wird hauptsächlich vom Indexberater und der TRACED_PLAN-Funktion verwendet.

Spalten

Spaltenname	Spaltentyp	Beschreibung
original_object_id	UNSIGNED BIGINT	Die Objekt-ID dieses Objekts in der Hauptprotokollierungsdatenbank
local_object_id	UNSIGNED BIGINT	Die Objekt-ID dieses Objekts in der Hilfsprotokollierungsdatenbank
pages_if_table	UNSIGNED INT	Wenn das Objekt eine Tabelle ist, ist dies die Anzahl der Seiten in der Tabelle. Wenn das Objekt keine Tabelle ist, ist dieser Wert NULL.
rows_if_table	UNSIGNED BIGINT	Wenn das Objekt eine Tabelle ist, ist dies die Anzahl der Zeilen in der Tabelle. Wenn das Objekt keine Tabelle ist, ist dieser Wert NULL.

Siehe auch

- „TRACED_PLAN-Funktion [Verschiedene]“ auf Seite 413
- „Indexberater“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

sa_diagnostic_blocking-Tabelle

Die sa_diagnostic_blocking-Tabelle gehört dem dbo-Benutzer und zeichnet blockierende Ereignisse auf. Wenn die Protokollierung von blockierenden Ereignissen aktiviert ist, wird in diese Tabelle jedesmal eine Zeile eingefügt, wenn eine Verbindung beim Versuch, auf eine Ressource zuzugreifen, blockiert wird. Gewöhnlich wird dies durch eine Tabellen- oder eine Zeilensperre bewirkt. Eine große Anzahl von Sperren kann darauf hinweisen, dass Sie die Parallelität in Ihrer Anwendung überprüfen sollten, um Konflikte in Tabellen und Zeilen zu vermindern.

Es gibt zwei Versionen dieser Tabelle: sa_diagnostic_blocking und sa_tmp_diagnostic_blocking.

Spalten

Spaltenname	Spaltentyp	Beschreibung
logging_session_id	UNSIGNED INT	Eine Nummer, die eindeutig die Protokollierungssitzung kennzeichnet, während der die Diagnose-Informationen gesammelt wurden.
lock_id	UNSIGNED BIGINT	Die ID der Sperre, die die Blockierung verursachte, wenn eine Zeilen- oder Tabellensperre die Blockierung verursachte. Sonst NULL.
request_id	UNSIGNED BIGINT	Die ID der Anforderung, die blockiert war, wenn die Blockierung nicht aufgrund eines Cursors aufgetreten ist, sonst NULL. Dieser Wert entspricht der ID, die der Anforderung in sa_diagnostic_request zugeordnet wurde.
cursor_id	UNSIGNED BIGINT	Die ID des Cursors, wenn die Blockierung aufgrund eines Cursors aufgetreten ist, sonst NULL. Dieser Wert entspricht der ID, die dem Cursor in sa_diagnostic_cursor zugeordnet wurde.
original_table_object_id	UNSIGNED BIGINT	Wenn die Blockierung aufgrund einer Tabellensperre aufgetreten ist, die ID der Tabelle, auf der die Blockierung aufgetreten ist. Sonst NULL.
rowid	UNSIGNED BIGINT	Wenn die Blockierung aufgrund einer Zeilensperre aufgetreten ist, die ID der Zeile, auf der die Blockierung aufgetreten ist. Sonst NULL.
block_time	TIMESTAMP	Die Uhrzeit, zu der die Blockierung auftrat
unblock_time	TIMESTAMP	Die Uhrzeit, zu der die Blockierung endete

Spaltenname	Spaltentyp	Beschreibung
blocked_by	UNSIGNED INT	Die ID der Verbindung, die die Sperre besaß und die Blockierung bewirkte

Siehe auch

- „Transaktion blockieren und Deadlock“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Funktionsweise von Sperren“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

sa_diagnostic_cachecontents-Tabelle

Die sa_diagnostic_cachecontents-Tabelle gehört dem dbo-Benutzer. Wenn Diagnoseprotokollierung aktiviert ist, werden periodisch Snapshots vom Cache-Inhalt genommen. Die Tabelle sa_diagnostic_cachecontents zeichnet die Anzahl der Tabellenseiten bei jeder Tabelle im Cache zum Zeitpunkt des Snapshots auf sowie die Anzahl der Zeilen in jeder Tabelle. Der Optimierer kann diese Informationen verwenden, um die Bedingungen wiederherzustellen, unter denen eine Abfrage optimiert wurde, um dann Optimierungsentscheidungen zu treffen.

Daten in der sa_diagnostic_cachecontents-Tabelle werden alle 20 Sekunden aktualisiert, solange es Abfrageaktivitäten gibt.

Es gibt zwei Versionen dieser Tabelle: sa_diagnostic_cachecontents und sa_tmp_diagnostic_cachecontents.

Spalten

Spaltenname	Spaltentyp	Beschreibung
logging_session_id	UNSIGNED INT	Eine Nummer, die eindeutig die Protokollierungssitzung kennzeichnet, während der die Diagnose-Informationen gesammelt wurden.
"time"	TIMESTAMP	Die Uhrzeit, zu der der Snapshot des Caches gemacht wurde
original_table_object_id	UNSIGNED BIGINT	Die Objekt-ID jeder Tabelle, die im Snapshot dargestellt wird
pages_in_cache	UNSIGNED INT	Für jede im Snapshot angegebene Tabelle die Gesamtzahl der Seiten im Cache zum Zeitpunkt des Snapshots
num_table_pages	UNSIGNED INT	Für eine im Snapshot angegebene Tabelle die Gesamtzahl der Seiten der Tabelle
num_table_rows	UNSIGNED BIGINT	Für eine im Snapshot angegebene Tabelle die Gesamtzahl der Zeilen der Tabelle

sa_diagnostic_connection-Tabelle

Die sa_diagnostic_connection-Tabelle gehört dem dbo-Benutzer und hat eine Zeile für jede Datenbankverbindung, die während der Protokollierungssitzung aktiv ist. Verbindungsherstellungs- und Verbindungstrennzeiten, wenn sie während der Protokollierungssitzung auftreten, können anhand der sa_diagnostic_request-Tabelle abgeleitet werden.

Die meisten Werte in dieser Tabelle spiegeln Werte von Verbindungseigenschaften wider.

Es gibt zwei Versionen dieser Tabelle: sa_diagnostic_connection und sa_tmp_diagnostic_connection.

Spalten

Spaltenname	Spaltentyp	Beschreibung
logging_session_id	UNSIGNED INT	Eine Nummer, die eindeutig die Protokollierungssitzung kennzeichnet, während der die Diagnose-Informationen gesammelt wurden.
connection_number	UNSIGNED INT	Eine dem Datenbankserver zugeordnete Nummer, um die Verbindung eines bestimmten Benutzers mit der Datenbank zu identifizieren. Dieser Wert gibt den Wert der Verbindungseigenschaft 'Number' wieder.
connection_name	LONG VARCHAR	Optionale Namenseigenschaft für die Verbindung. Dieser Wert gibt den Wert der Verbindungseigenschaft 'Name' wieder.
user_name	LONG VARCHAR	Der Name des mit der Datenbank verbundenen Benutzers
comm_link	CHAR(40)	Gibt die clientseitigen Netzwerkprotokolloptionen an. Dieser Wert gibt den Wert der Verbindungseigenschaft 'CommLinks' wieder.
node_address	LONG VARCHAR	Der Knoten für den Client in einer Client/Server-Verbindung. Dieser Wert gibt den Wert der Verbindungseigenschaft 'NodeAddress' wieder.

Spaltenname	Spaltentyp	Beschreibung
appinfo	LONG VARCHAR	Informationen über den Clientprozess, wie z.B. die IP-Adresse des Clientcomputers, das Betriebssystem, auf dem er läuft, usw. Dieser Wert gibt den Wert der Verbindungseigenschaft 'AppInfo' wieder.

Siehe auch

- „Liste der Verbindungseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]

sa_diagnostic_cursor-Tabelle

Die Tabelle sa_diagnostic_cursor gehört dem dbo-Benutzer. Jede Zeile beschreibt einen internen oder externen Cursor, der während der Protokollierungssitzung geöffnet wurde.

Es gibt zwei Versionen dieser Tabelle: sa_diagnostic_cursor und sa_tmp_diagnostic_cursor.

Spalten

Spaltenname	Spaltentyp	Beschreibung
logging_session_id	UNSIGNED INT	Eine Nummer, die eindeutig die Protokollierungssitzung kennzeichnet, während der die Diagnose-Informationen gesammelt wurden.
cursor_id	UNSIGNED BIGINT	Eine eindeutige Nummer, die den Cursor kennzeichnet
query_id	UNSIGNED BIGINT	Kennzeichnet die Abfrage, auf die sich dieser Cursor bezieht
isolation_level	TINYINT	Isolationsstufe, auf der dieser Cursor geöffnet wurde
flags	UNSIGNED INT	Interne Verwendung
forward_fetches	UNSIGNED INT	Anzahl der den Cursor betreffenden Vorwärts-Abrufe, einschließlich Prefetch-Vorgänge
reverse_fetches	UNSIGNED INT	Anzahl der den Cursor betreffenden Rückwärts-Abrufe, einschließlich Prefetch-Vorgänge
absolute_fetches	UNSIGNED INT	Anzahl der den Cursor betreffenden absoluten Abrufe
first_fetch_time_ms	UNSIGNED INT	Dauer des Abrufs der ersten Zeile

Spaltenname	Spaltentyp	Beschreibung
total_fetch_time_ms	UNSIGNED INT	Dauer des Abrufens. Dieser Wert enthält nicht die Verarbeitungszeit der Anwendung zwischen den tatsächlichen Abrufen (Think-Time).
plan_xml	LONG VARCHAR	Detaillierter Plan für Cursor, die zum Zeitpunkt gesichert wurden, als der Cursor geschlossen wurde. Diese Pläne enthalten ggf. detaillierte Statistiken.

Siehe auch

- „Cursorverwendung“ [[SQL Anywhere Server - Programmierung](#)]

sa_diagnostic_deadlock-Tabelle

Die sa_diagnostic_deadlock-Tabelle gehört dem dbo-Benutzer. Wenn Diagnoseprotokollierung aktiviert und dazu eingestellt ist, die Protokollierung von Deadlock-Ereignissen aufzunehmen, wird eine Reihe von Zeilen jedesmal in die Tabelle eingefügt, wenn ein Deadlock auftritt (es wird eine Zeile für jede Verbindung, die Teil des Deadlocks war, eingefügt). Die Reihe aller Zeilen, die ein einziges Deadlock-Ereignis darstellen, wird durch eine snapshot_id eindeutig gekennzeichnet.

Spalten

Spaltenname	Spaltentyp	Beschreibung
logging_session_id	UNSIGNED INT	Eine Nummer, die eindeutig die Protokollierungssitzung kennzeichnet, während der die Diagnose-Informationen gesammelt wurden.
snapshot_id	UNSIGNED BIGINT	Eine Nummer, die kennzeichnet, zu welchem Deadlock-Ereignis diese Zeile gehört. Diese Spalte hat nichts mit der Snapshot-Isolation zu tun.
snapshot_at	TIMESTAMP	Die Uhrzeit, zu der der Deadlock auftrat
waiter	UNSIGNED INT	Die Verbindungsnummer der Verbindung, für die diese Zeile steht
request_id	UNSIGNED BIGINT	Die ID der Anforderung, die diese Verbindung verarbeitete, als der Deadlock auftrat
original_table_object_id	UNSIGNED BIGINT	Die Objekt-ID der Tabelle, auf der die Verbindung blockiert war
rowid	UNSIGNED BIGINT	Die Aufzeichnungs-ID der Zeile, auf der die Verbindung blockiert war

Spaltenname	Spaltentyp	Beschreibung
Eigentümer	UNSIGNED INT	Die Verbindungsnummer der Verbindung, die die gewünschte Zeile sperrte
rollback_operation_count	UNSIGNED INT	Die Anzahl der nicht festgeschriebenen Vorgänge

Siehe auch

- „Transaktion blockieren und Deadlock“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

sa_diagnostic_hostvariable-Tabelle

Die Tabelle sa_diagnostic_hostvariable gehört dem dbo-Benutzer und enthält die Werte von Hostvariablen, die vom angegebenen Cursor verwendet werden.

Es gibt zwei Versionen dieser Tabelle: sa_diagnostic_hostvariable und sa_tmp_diagnostic_hostvariable.

Spalten

Spaltenname	Spaltentyp	Beschreibung
logging_session_id	UNSIGNED INT	Eine Nummer, die eindeutig die Protokollierungssitzung kennzeichnet, während der die Diagnose-Informationen gesammelt wurden.
request_id	UNSIGNED BIGINT	Die ID der Anforderung, zu der die Hostvariable gehört
cursor_id	UNSIGNED BIGINT	Die ID des Cursors, auf die sich die Hostvariable bezieht
hostvar_num	UNSIGNED SMALLINT	Die Ordinalposition der Hostvariablen in der SQL-Anweisung
hostvar_type	UNSIGNED TINYINT	Wird nur intern verwendet.
hostvar_value	LONG NVAR-CHAR	Eine Zeichenfolge, die den Wert der Hostvariablen darstellt. Auch wenn die Hostvariable eine Ganzzahl oder ein float-Wert ist, wird der Wert hier als eine Zeichenfolge dargestellt.

Siehe auch

- „Hostvariablen in Embedded SQL“ [[SQL Anywhere Server - Programmierung](#)]

sa_diagnostic_internalvariable-Tabelle

Die Tabelle sa_diagnostic_internalvariable gehört dem dbo-Benutzer und enthält die Werte von internen (lokalen) Variablen, die von einer gegebenen Anweisung verwendet werden. Diese Tabelle wird hauptsächlich vom Indexberater und der traced_plan-Funktion verwendet.

Es gibt zwei Versionen dieser Tabelle: sa_diagnostic_internalvariable und sa_tmp_diagnostic_internalvariable.

Spalten

Spaltenname	Spaltentyp	Beschreibung
logging_session_id	UNSIGNED INT	Eine Nummer, die eindeutig die Protokollierungssitzung kennzeichnet, während der die Diagnose-Informationen gesammelt wurden.
request_id	UNSIGNED BIGINT	Die ID der Anforderung, die die interne Variable enthält
rowvariable_id	UNSIGNED INT	Die Spaltennummer in der Zeilenvariablen dieses Werts
variable_domain	UNSIGNED SMALLINT	Wird nur intern verwendet.
variable_name	CHAR(128)	Der Name der internen Variablen
variable_value	LONG NVARCHAR	Eine Zeichenfolge, die den Wert der internen Variablen darstellt

Siehe auch

- [„Lokale Variablen“ auf Seite 86](#)

sa_diagnostic_query-Tabelle

Die Tabelle sa_diagnostic_query gehört dem dbo-Benutzer und speichert Optimierungsinformationen für Abfragen, besonders den Kontext, in dem sie optimiert wurden. Eine Zeile in dieser Tabelle stellt einen Aufruf des Optimierers für eine Abfrage dar. Zur Optimierungszeit aufgezeichnete Pläne werden hier gespeichert.

Manche der Werte in dieser Tabelle spiegeln Datenbankoptionswerte.

Es gibt zwei Versionen dieser Tabelle: sa_diagnostic_query und sa_tmp_diagnostic_query.

Spalten

Spaltenname	Spaltentyp	Beschreibung
logging_session_id	UNSIGNED INT	Die ID der Protokollierungssitzung, während der die Abfrage oder die Anforderung aufgetreten ist
query_id	UNSIGNED BIGINT	Eine Nummer, die die Abfrage eindeutig kennzeichnet
statement_id	UNSIGNED BIGINT	Eine Nummer, die eine Anweisung in der Abfrage eindeutig kennzeichnet
user_object_id	UNSIGNED BIGINT	Die Objekt-ID des Benutzers, unter dem diese Abfrage ausgeführt wurde. Wenn die Abfrage von einer Prozedur ausgeführt wurde, ist dies die Benutzer-ID des Prozedureigentümers.
start_time	TIMESTAMP	Die Uhrzeit, zu der diese Abfrage optimiert wurde
cache_size_bytes	UNSIGNED BIGINT	Die Größe, in Byte, des Caches zum Zeitpunkt, als diese Abfrage optimiert wurde
optimization_goal	TINYINT	Legt fest, ob die Abfrageverarbeitung dahingehend optimiert wird, die erste Zeile schnell zurückzugeben, oder ob die Kosten für die Ausgabe der vollständigen Ergebnismenge minimiert werden sollen. Dieser Wert gibt den Wert der optimization_goal-Datenbankoption wieder.
optimization_level	TINYINT	Steuert den Aufwand, den der SQL Anywhere-Abfrageoptimierer betreibt, um einen Zugriffsplan für eine SQL-Anweisung zu finden. Dieser Wert gibt den Wert der optimization_level-Datenbankoption wieder.
user_estimates	TINYINT	Steuert, ob Benutzer-Selektivitätseinschätzungen in Abfrageprädikaten vom Abfrageoptimierer respektiert oder ignoriert werden. Dieser Wert gibt den Wert der user_estimates-Datenbankoption wieder.
optimization_workload	TINYINT	Legt fest, ob die Abfrageverarbeitung für eine Arbeitslast optimiert wird, die aus einer Mischung von Aktualisierungen und Lesevorgängen besteht, oder für eine Arbeitslast, die hauptsächlich auf Lesevorgängen basiert. Dieser Wert gibt den Wert der optimization_workload-Datenbankoption wieder.
available_requests	TINYINT	Wird intern zur Berechnung der Stufe der abfrageinternen Parallelität verwendet

Spaltenname	Spaltentyp	Beschreibung
active_requests	TINYINT	Wird intern zur Berechnung der Stufe der abfrageinternen Parallelität verwendet
max_tasks	TINYINT	Wird intern zur Berechnung der Stufe der abfrageinternen Parallelität verwendet
used_bypass	TINYINT	Ob ein einfacher Abfragen-Bypass verwendet wurde. Ein Wert von "1" zeigt an, dass ein Bypass verwendet wurde. Ein Wert von "0" zeigt an, dass die Abfrage voll optimiert wurde.
estimated_cost_ms	TINYINT	Die geschätzten Kosten, in Millisekunden
plan_explain	LONG VAR-CHAR	Eine Textplandarstellung dieser Abfrage
plan_xml	LONG VAR-CHAR	Eine Darstellung dieser Abfrage als grafischer Plan (falls einer aufgezeichnet wurde)
sql_rewritten	LONG VAR-CHAR	Text einer Abfrage, nachdem Optimierungen angewendet wurden. In dieser Spalte wird nur ein Wert angezeigt, wenn die Optimierungsprotokollierung aktiviert ist.

Siehe auch

- „Datenbankoptionen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „So funktioniert der Optimierer“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „optimization_level-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „optimization_workload-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „user_estimates-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

sa_diagnostic_request-Tabelle

Die sa_diagnostic_request-Tabelle gehört dem dbo-Benutzer und ist die Mastertabelle für alle Anforderungen. Eine Anforderung ist ein mit Abfragenverarbeitung zusammenhängendes Ereignis und umfasst gewöhnlich Folgendes:

- Ereignisse der Verbindungsherstellung und -trennung
- Anweisungsausführungen
- Anweisungsvorbereitungen
- Ereignisse der Cursoröffnung und -löschung

Es gibt zwei Versionen dieser Tabelle: sa_diagnostic_request und sa_tmp_diagnostic_request.

Spalten

Spaltenname	Spaltentyp	Beschreibung
logging_session_id	UNSIGNED INT	Die Protokollierungssitzung, während der die Anforderung aufgetreten ist
request_id	UNSIGNED BIGINT	Eine Nummer, die die Anforderung eindeutig kennzeichnet
start_time	TIMESTAMP	Die Uhrzeit, zu der das Ereignis begann
finish_time	TIMESTAMP	Bei Anweisungsausführung die Uhrzeit, zu der die Anweisung abgeschlossen wurde, sonst NULL.
duration_ms	UNSIGNED INT	Die Dauer des Ereignisses in Millisekunden
connection_number	UNSIGNED INT	Die ID der Verbindung, die das Eintreten des Ereignisses bewirkt hat
request_type	UNSIGNED SMALLINT	Der Anforderungstyp. Werte umfassen: Neue Diagnoseprotokollierungssitzung gestartet, SQL-Anweisung ausgeführt, Cursor geöffnet, Cursor geschlossen, Clientverbindung hergestellt, Clientverbindung getrennt und Checkpoint gesetzt.
statement_id	UNSIGNED BIGINT	Wenn das Ereignis anweisungsbezogen war, die der Anweisung zu Protokollierungszwecken zugewiesene ID
query_id	UNSIGNED BIGINT	Wenn das Ereignis abfragebezogen war, die der Abfrage zu Protokollierungszwecken zugewiesene ID
cursor_id	UNSIGNED BIGINT	Wenn das Ereignis cursorbezogen war, die dem Cursor zu Protokollierungszwecken zugewiesene ID
sql_code	SMALLINT	Da die Zeilen in dieser Tabelle Vorgänge auf Anweisungen, Cursorn oder Abfragen darstellen, geben die meisten einen SQL-Code zurück. Diese Spalte enthält den zurückgegebenen SQL-Code. Wenn ein SQL-Code von "0" zurückgegeben wird, enthält die Spalte NULL.

sa_diagnostic_statement-Tabelle

Die Tabelle sa_diagnostic_statement gehört dem dbo-Benutzer und speichert Text von Anweisungen. Eine Zeile in dieser Tabelle stellt eine SQL-Anweisung dar, die vom Server ausgeführt wurde. Solche Anweisungen können von einer externen Quelle, wie einer Clientanforderung, oder von einer internen Quelle, wie einer Prozedur, einem Trigger oder einer benutzerdefinierten Funktion, ausgegeben worden sein. Interne Anweisungen werden hier einmal pro Sitzung angezeigt.

Es gibt zwei Versionen dieser Tabelle: sa_diagnostic_statement und sa_tmp_diagnostic_statement.

Spalten

Spaltenname	Spaltentyp	Beschreibung
logging_session_id	UNSIGNED INT	Die Protokollierungssitzung, während der die Anweisung übermittelt wurde.
statement_id	UNSIGNED BIGINT	Eine eindeutige Nummer, die der Anweisung zu Protokollierungszwecken zugeordnet wurde
database_object	UNSIGNED BIGINT	Wenn die Anweisung von einer Prozedur, einem Trigger oder einer Funktion stammte, ist dies die ID, wie sie in der ISYSOBJECT-Systemtabelle angegeben ist.
line_number	UNSIGNED SMALLINT	Wenn die Anweisung Teil einer zusammengesetzten Anweisung war, spiegelt dies die Ordinalposition der Anweisung innerhalb der zusammengesetzten Anweisung wieder.
signature	UNSIGNED INT	Wird intern verwendet, um ähnliche Abfragen zu gruppieren
statement_text	LONG VARCHAR	Der Anweisungstext

sa_diagnostic_statistics-Tabelle

Die Tabelle sa_diagnostic_statistics gehört dem dbo-Benutzer und enthält einen Verlauf von Performance-Zählern, die im Server geführt werden. Jede Zeile stellt den Wert eines gegebenen Performance-Zählers zu einem gegebenen Zeitpunkt dar.

Es gibt zwei Versionen dieser Tabelle: sa_diagnostic_statistics und sa_tmp_diagnostic_statistics.

Spalten

Spaltenname	Spaltentyp	Beschreibung
logging_session_id	UNSIGNED INT	Eine Nummer, die eindeutig die Protokollierungssitzung kennzeichnet, während der die Diagnose-Informationen gesammelt wurden.
"time"	TIMESTAMP	Die Uhrzeit, zu der der Performance-Zählerwert aufgezeichnet wurde

Spaltenname	Spaltentyp	Beschreibung
counter_id	UNSIGNED SMALLINT	Eine Nummer, die den Performance-Zähler eindeutig kennzeichnet. Sie können den Namen der Eigenschaft, die von dieser counter_id dargestellt wird, mit der PROPERTY_NAME-Funktion abrufen.
type	TINYINT	Gibt an, ob dies eine Datenbank-, Server- oder Verbindungsstatistik ist. Mögliche Werte sind "0" für Server, "1" für Datenbank, "2" für Verbindung und "4" für externe Datenbank.
connection_number	UNSIGNED INT	Im Fall einer Verbindungsstatistik die Verbindungsnummer, von der diese Eigenschaft aufgezeichnet wurde. Im Fall einer erweiterten Datenbankstatistik die Dateinummer der Datei, von der diese Eigenschaft aufgezeichnet wurde. Andernfalls ist der Wert "0".
counter_value	UNSIGNED INT	Der Wert des Performance-Zählers

Siehe auch

- [„PROPERTY_NAME-Funktion \[System\]“ auf Seite 342](#)

sa_diagnostic_tracing_level-Tabelle

Die Tabelle sa_diagnostic_tracing_level gehört dem dbo-Benutzer. Jede Zeile in dieser Tabelle stellt eine Bedingung dar, die festlegt, welche Art von Diagnoseinformationen an die Protokollierungsdatenbank gesendet werden sollen. Wenn ein Abschnitt der Protokollierungsdaten die Bedingungen einer oder mehrerer Zeilen in dieser Tabelle erfüllt, werden die entsprechenden Daten protokolliert.

Daten werden in diese Tabelle mit der CONNECT TRACING- oder der REFRESH TRACING LEVEL-Anweisung eingegeben.

Spalten

Spaltenname	Spaltentyp	Beschreibung
id	UNSIGNED INT	Wird nur intern verwendet.

Spaltenname	Spaltentyp	Beschreibung
scope	CHAR(32)	<p>Der Bereich der Diagnoseprotokollierung, wie unten aufgelistet.</p> <ul style="list-style-type: none"> • DATABASE • ORIGIN • USER • CONNECTION_NAME • CONNECTION_NUMBER • FUNCTION • PROCEDURE • EVENT • TRIGGER • TABLE
identifier	CHAR(128)	<p>Der Bezeichner für den Bereich. Dieser Wert ändert sich, abhängig vom angegebenen Bereich. Beispiel:</p> <ul style="list-style-type: none"> • Wenn <i>scope</i> DATABASE ist, muss kein <i>identifier</i> angegeben sein. • Wenn <i>scope</i> ORIGIN ist, muss <i>identifier</i> entweder Internal oder External sein. • Wenn <i>scope</i> USER ist, ist <i>identifier</i> die ID des Benutzers. • Wenn <i>scope</i> CONNECTION_NAME oder CONNECTION_NUMBER ist, ist <i>identifier</i> der Name bzw. die Nummer der Verbindung. • Wenn <i>scope</i> FUNCTION, PROCEDURE, EVENT, TRIGGER oder TABLE ist, ist <i>identifier</i> der voll qualifizierte Bezeichner für das Objekt.
trace_type	CHAR(32)	<p>Der Typ der Daten, die für den angegebenen Bereich protokolliert werden, wie unten aufgelistet.</p> <ul style="list-style-type: none"> • VOLATILE_STATISTICS • NONVOLATILE_STATISTICS • CONNECTION_STATISTICS • BLOCKING • PLANS • PLANS_WITH_STATISTICS • STATEMENTS • STATEMENTS_WITH_VARIABLES • OPTIMIZATION_LOGGING • OPTIMIZATION_LOGGING_WITH_PLANS

Spaltenname	Spaltentyp	Beschreibung
trace_condition	CHAR(32)	<p>Gilt nur für Pläne und steuert, ob große, kostenträchtige Abfragen oder Abfragen, bei denen der Optimierer eine nicht optimale Auswahl getroffen hat, protokolliert werden. Mögliche Werte sind unten aufgelistet.</p> <ul style="list-style-type: none"> • NONE oder NULL • SAMPLE_EVERY • ABSOLUTE_COST • RELATIVE_COST_DIFFERENCE
value	UNSIGNED INT	<p>Der <i>condition</i> zugeordnete Wert. Wenn <i>condition</i> beispielsweise SAMPLE_EVERY ist, ist <i>condition_value</i> eine positive Ganzzahl, die die Zeit in Millisekunden darstellt. Zusätzliche Regeln sind folgende:</p> <ul style="list-style-type: none"> • Wenn <i>condition</i> NULL oder NONE ist, gibt es keinen <i>condition_value</i>. • Wenn <i>condition</i> ABSOLUTE_COST ist, gibt <i>condition_value</i> die aktuellen Gesamtkosten der Durchführung der Anweisung in Millisekunden wieder. • Wenn <i>condition</i> RELATIVE_COST_DIFFERENCE ist, gibt <i>condition_value</i> die Kosten der Ausführung als einen Prozentsatz der geschätzten Kosten wieder.
enabled	BIT	<p>Ob die Zeile aktiviert ist. D.h., ob die Protokollierungseinstellungen in der Zeile aktiv sind. "1" bedeutet "aktiviert" und "0" bedeutet "deaktiviert".</p>

Siehe auch

- „Bedingungen für die Diagnoseprotokollierung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Bereiche der Diagnoseprotokollierung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „ATTACH TRACING-Anweisung“ auf Seite 546
- „REFRESH TRACING LEVEL-Anweisung“ auf Seite 993

Andere Tabellen

Es folgen Informationen zu anderen Tabellen, wie z.B. Systemtabellen, die von Java in der Datenbank und SQL Remote verwendet werden.

RowGenerator-Tabelle (dbo)

Die dbo.RowGenerator-Tabelle wird als schreibgeschützte Tabelle mit 255 Zeilen vorgegeben. Diese Tabelle kann nützlich sein für Abfragen, die kleine Ergebnismengen produzieren und einen Bereich numerischer Werte benötigen.

Die Tabelle RowGenerator wird von Systemprozeduren und Ansichten verwendet und darf nicht verändert werden.

Sie können auch die Systemprozedur sa_rowgenerator verwenden, um einen Bereich numerischer Werte zu generieren.

Spaltenname	Spaltentyp
row_num	SMALLINT

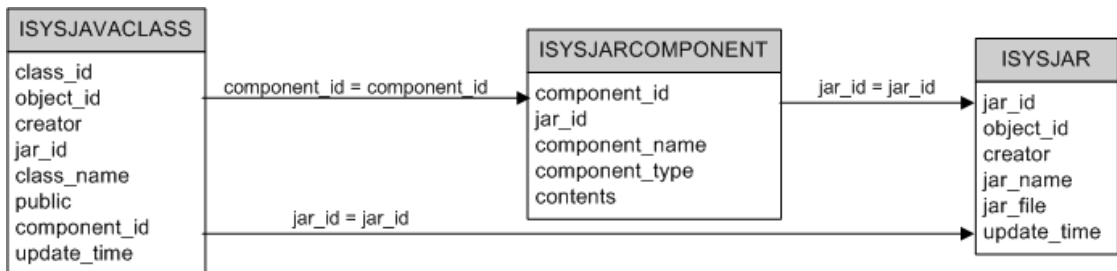
row_num Ganzzahl zwischen 1 und 255.

Siehe auch

- „sa_rowgenerator-Systemprozedur“ auf Seite 1300

Java-Systemtabellen

Die Systemtabellen, die für Java verwendet werden, sind unten aufgelistet. Fremdschlüsselbeziehungen zwischen Tabellen sind durch Pfeile angezeigt: Der Pfeil führt von der Fremdtabelle zur Primärtabelle.



MobiLink-Systemtabellen

Weitere Hinweise zu den MobiLink-Systemtabellen finden Sie unter „Systemtabellen des MobiLink-Servers“ [[MobiLink - Serveradministration](#)].

Systemtabellen von SQL Remote

Weitere Hinweise zu den SQL Remote-Systemtabellen finden Sie unter „Systemtabellen von SQL Remote“ [[SQL Remote](#)].

UltraLite-Systemtabellen

Weitere Hinweise zu den UltraLite-Systemtabellen finden Sie unter „[UltraLite-Systemtabellen](#)“
[\[UltraLite - Datenbankverwaltung\]](#).

Systemprozeduren

Einige Systemprozeduren, z.B. `sa_get_table_definition`, werden als Funktionen implementiert. Da sie aber in demselben Kontext und auf dieselbe Weise verwendet werden wie Systemprozeduren, werden sie wie die Systemprozeduren behandelt und ähnlich benannt (`sa_xxx`). Alle Systemprozeduren können als gesicherte Funktionen aktiviert und deaktiviert werden. Unter Windows Mobile können keine externen Funktionen aufgerufen werden.

Anzeigen von Details über Systemprozeduren und Funktionen

Datenbankadministratoren können auf die Definitionen für Systemprozeduren und Funktionen aus Sybase Central zugreifen.

Voraussetzungen

Es sind keine Voraussetzungen erforderlich.

Anzeigen von Details zu Systemprozeduren und Funktionen

1. Stellen Sie eine Verbindung zur Datenbank mithilfe des SQL Anywhere 16-Plug-Ins her.
2. Rechtsklicken Sie auf die Datenbank und klicken Sie auf **Eigentümerfilter konfigurieren**.
3. Klicken Sie auf **DBO** und dann auf **OK**.
4. Doppelklicken Sie im linken Fensterausschnitt auf **Prozeduren und Funktionen**.
5. Im linken Fensterausschnitt wählen Sie die Prozedur und im rechten Fensterausschnitt klicken Sie auf die Registerkarte **SQL**.

Ergebnisse

Die Prozedur- oder Funktionsdefinition erscheint auf der Registerkarte **SQL**.

Webdienst-Systemprozeduren

Folgende Systemprozeduren werden für Webdienste verwendet:

- „sa_http_header_info-Systemprozedur“
- „sa_http_php_page-Systemprozedur“
- „sa_http_php_page_interpreted-Systemprozedur“
- „sa_http_variable_info-Systemprozedur“
- „sa_set_http_header-Systemprozedur“
- „sa_set_http_option-Systemprozedur“
- „sa_set_soap_header-Systemprozedur“

Siehe auch

- „Webdienstfunktionen“ auf Seite 162
- „SQL Anywhere als HTTP-Webserver“ [*SQL Anywhere Server - Programmierung*]
- „Datenbankserveroption -xs“ [*SQL Anywhere Server - Datenbankadministration*]

Systemprozeduren für Rollen und Privilegien

Die folgenden Systemprozeduren werden mit Rollen und Privilegien verwendet:

- „sp_displayroles-Systemprozedur“ auf Seite 1363
- „sp_has_role-Systemprozedur“ auf Seite 1371
- „sp_objectpermission-Systemprozedur“ auf Seite 1379
- „sp_proc_priv-Systemprozedur“ auf Seite 1384
- „sp_sys_priv_role_info-Systemprozedur“ auf Seite 1399

MAPI- und SMTP-Systemprozeduren

SQL Anywhere umfasst Systemprozeduren zum Senden von E-Mail über den Standard "Microsoft Messaging API" (MAPI) oder den Internetstandard "Simple Mail Transfer Protocol" (SMTP). Diese Systemprozeduren sind als erweiterte Systemprozeduren implementiert: Jede Prozedur ruft eine Funktion in einer externen DLL auf.

Eigentümer dieser Prozeduren ist die dbo-Rolle. Benutzern muss das EXECUTE-Privileg für diese Systemprozeduren erteilt werden, bevor sie sie verwenden können.

Um die MAPI- bzw. SMTP-Systemprozeduren verwenden zu können, muss ein MAPI- bzw. SMTP-E-Mail-System auf dem Datenbankservercomputer verfügbar sein.

Die folgenden MAPI- und SMTP-Systemprozeduren stehen zur Verfügung:

- **xp_startmail** Startet eine Mail-Sitzung in einem angegebenen Mail-Konto, indem sie sich beim MAPI-Nachrichtensystem anmeldet.
- **xp_startsmtp** Startet eine Mail-Sitzung in einem angegebenen Mail-Konto, indem sie sich beim SMTP-Nachrichtensystem anmeldet.

- **xp_sendmail** Sendet eine Mail-Nachricht an angegebene Benutzer.
- **xp_stopmail** Schließt die MAPI-Mail-Sitzung.
- **xp_stopsmtplib** Schließt die SMTP-Mail-Sitzung.
- **xp_get_mail_error_code** Gibt Informationen zum letzten SMTP- oder MAPI-Fehler zurück.
- **xp_get_mail_error_text** Gibt den Text des letzten SMTP-Fehlers zurück.

Rückgabecodes für MAPI- und SMTP-Systemprozeduren

Die folgenden Codes können von MAPI- oder SMTP-Systemprozeduren zurückgegeben werden:

Rückgabecode	Bedeutung
-1	Unbekannter Fehler ¹
0	Erfolg
1	Ein ungültiger Parameter wurde angegeben
2	Kein Speicher mehr
3	xp_startmail oder xp_startsmtp wurde nicht aufgerufen
4	Ungültiger Hostname
5	Verbindungsfehler ¹
6	Fehler der sicheren Verbindung ¹
7	MAPI-Funktionen sind nicht verfügbar ¹

¹ Verwenden Sie die xp_get_mail_error_code-Systemprozedur, um zusätzliche Informationen zu einem Rückgabecode anzurufen.

Beispiel

Die folgende Prozedur benachrichtigt eine Gruppe von Personen, dass eine Sicherung abgeschlossen ist. Sie überprüft keine Rückgabecodes.

```
CREATE PROCEDURE notify_backup( )
BEGIN
    CALL xp_startmail( mail_user='ServerAccount',
                      mail_password='ServerPassword' );
    CALL xp_sendmail( recipient='IS Group',
                      subject='Backup',
                      "message"='Backup completed' );
    CALL xp_stopmail( );
END;
```

Die folgende Prozedur benachrichtigt eine Gruppe von Personen, dass eine Sicherung abgeschlossen ist, und umfasst eine Fehlerüberprüfung.

```
CREATE OR REPLACE PROCEDURE notify_backup_with_error_check( )
BEGIN
    DECLARE result_code INTEGER;
    DECLARE error_code INTEGER;
    DECLARE error_text LONG VARCHAR;
    SET result_code = xp_startmail( 'ServerAccount', -- mail_user
                                   'ServerPassword' -- mail_password
                                   );

    IF result_code = 0 THEN
        SET result_code = xp_sendmail( 'IS Group', --recipient
                                      'Backup', -- subject
                                      NULL, -- cc_recipient
                                      NULL, -- bcc_recipient
                                      NULL, -- query
                                      'Backup completed' -- message
                                      );
    ENDIF;
    IF result_code = 0 THEN
        SET result_code = xp_stopmail( );
    ENDIF;
    IF result_code = 0 THEN
        MESSAGE 'Backup completed message successully sent';
    ELSE
        SET error_code = xp_get_mail_error_code();
        SET error_text = xp_get_mail_error_text();
        MESSAGE 'Error: ' || result_code ||
                ' Code: ' || error_code ||
                ' Text: ' || error_text;
    ENDIF;
END;
```

Siehe auch

- „xp_startmail-Systemprozedur“ auf Seite 1431
- „xp_startsmtp-Systemprozedur“ auf Seite 1431
- „xp_sendmail-Systemprozedur“ auf Seite 1426
- „xp_stopmail-Systemprozedur“ auf Seite 1434
- „xp_stopsmtp-Systemprozedur“ auf Seite 1434
- „xp_get_mail_error_code-Systemprozedur“ auf Seite 1419
- „xp_get_mail_error_text-Systemprozedur“ auf Seite 1420

Systemprozeduren für Verzeichnisse und Dateien

Sie können Folgendes verwenden, um auf die lokale Dateistruktur des Computers zuzugreifen, aufFügt eine Gruppe zu einer Datenbank hinzu dem ein Datenbankserver läuft:

- Verzeichniszugriffsserver
- Systemprozeduren für Verzeichnisse und Dateien, z.B. die sp_create_directory-Systemprozedur

Die Systemprozeduren für Verzeichnisse und Dateien sind einfacher zu verwenden als Verzeichniszugriffsserver, aber nicht so flexibel und leistungsfähig wie Proxy-Tabellen und Server für den Verzeichniszugriff.

Verwenden Sie die folgenden Systemprozeduren, um Verzeichnisse und Dateien auf dem Computer zu bearbeiten, auf dem sich ein Server befindet:

- **sp_list_directory-Systemprozedur** Listet den Inhalt eines angegebenen Verzeichnisses auf.
- **sp_create_directory-Systemprozedur** Erstellt ein angegebenes Verzeichnis.
- **sp_copy_directory-Systemprozedur** Kopiert das angegebene Verzeichnis an einen anderen Speicherort.
- **sp_move_directory-Systemprozedur** Verschiebt ein angegebenes Verzeichnis.
- **sp_delete_directory-Systemprozedur** Löscht ein angegebenes Verzeichnis.
- **sp_copy_file-Systemprozedur** Kopiert eine angegebene Datei.
- **sp_move_file-Systemprozedur** Verschiebt eine angegebene Datei.
- **sp_delete_file-Systemprozedur** Löscht eine angegebene Datei.

Die folgenden Systemprozeduren können ebenfalls zum Lesen und Schreiben in Dateien verwendet werden:

- **xp_read_file-Systemprozedur** Liest die Datei und gibt den Inhalt der Datei als LONG BINARY-Variable zurück.
- **xp_write_file-Systemprozedur** Schreibt Daten aus einer SQL-Anweisung in eine Datei.

Siehe auch

- „sp_copy_directory-Systemprozedur“ auf Seite 1356
- „sp_copy_file-Systemprozedur“ auf Seite 1358
- „sp_create_directory-Systemprozedur“ auf Seite 1359
- „sp_delete_directory-Systemprozedur“ auf Seite 1361
- „sp_delete_file-Systemprozedur“ auf Seite 1362
- „sp_list_directory-Systemprozedur“ auf Seite 1373
- „sp_move_directory-Systemprozedur“ auf Seite 1377
- „sp_move_file-Systemprozedur“ auf Seite 1378
- „xp_read_file-Systemprozedur“ auf Seite 1424
- „xp_write_file-Systemprozedur“ auf Seite 1435
- „Verzeichniszugriffsserver“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Systemprozeduren für gesicherte Funktionen

Die folgenden Systemprozeduren werden mit gesicherten Funktionen verwendet:

- „sa_server_option-Systemprozedur“
- „sp_alter_secure_feature_key-Systemprozedur“
- „sp_create_secure_feature_key-Systemprozedur“

- „sp_drop_secure_feature_key-Systemprozedur“
- „sp_list_secure_feature_keys-Systemprozedur“
- „sp_use_secure_feature_key-Systemprozedur“

System- und Katalogprozeduren von Adaptive Server Enterprise

Adaptive Server Enterprise bietet System- und Katalogprozeduren, mit denen viele Verwaltungsfunktionen ausgeführt und Systeminformationen abgerufen werden können. Systemprozeduren sind integrierte gespeicherte Prozeduren, um Berichte von Systemtabellen zu erhalten und um Systemtabellen zu aktualisieren. Katalog-Systemprozeduren rufen Informationen von den Systemtabellen in Tabellenform ab.

SQL Anywhere hat Unterstützung für einige dieser Adaptive Server Enterprise-Prozeduren implementiert. Hinweise zur Verwendung dieser Prozeduren finden Sie in Ihrer Adaptive Server Enterprise-Dokumentation.

Systemprozeduren von Adaptive Server Enterprise

SQL Anywhere enthält einige Systemprozeduren von Adaptive Server Enterprise. Diese Prozeduren übernehmen zwar dieselben Aufgaben wie in Adaptive Server Enterprise, sind aber nicht identisch. Wenn Sie vorhandene Skripten haben, die diese Prozeduren verwenden, sollten Sie den Prozedurcode überprüfen. Wenn den Text einer gespeicherten Prozedur anzeigen möchten, führen Sie den folgenden Befehl aus.

```
sp_helptext 'dbo.procedure_name'
```

Die implementierten Systemprozeduren werden in der folgenden Tabelle beschrieben.

Name der Systemprozedur/Parameter	Beschreibung
sp_addgroup @grpname	Fügt eine Gruppe zu einer Datenbank hinzu
sp_addlogin @login_name , @passwd [, @defaultdb [, @deflanguage [, @fullname]]]	Fügt eine neue Login-ID zu einer Datenbank hinzu
sp_addmessage @message_num , @message_text [, @language]	Fügt eine benutzerdefinierte Nachricht zu ISYUSERMES- SAGE hinzu, die von PRINT- und RAISERROR-Aufrufen in gespeicherten Prozeduren verwendet werden kann
sp_addtype @typename , @phystype [, @ident_null]	Erstellt einen benutzerdefinierten Datentyp

Name der Systemprozedur/Parameter	Beschreibung
<code>sp_adduser @login_name [, @name_in_db [, @grpname]]</code>	Fügt eine neue Benutzer-ID zu einer Datenbank hinzu
<code>sp_changegroup @grpname , @name_in_db</code>	Ändert die Gruppe eines Benutzers oder fügt einen Benutzer zu einer Gruppe hinzu
<code>sp_dropgroup @grpname</code>	Löscht eine Gruppe aus einer Datenbank
<code>sp_droplogin @login_name</code>	Löscht eine Login-ID aus einer Datenbank
<code>sp_dropmessage @message_number [, @language]</code>	Löscht eine benutzerdefinierte Nachricht
<code>sp_droptype @typename</code>	Löscht einen benutzerdefinierten Datentyp
<code>sp_dropuser @name_in_db</code>	Löscht eine Benutzer-ID aus einer Datenbank
<code>sp_getmessage @message_num , @msg_var [, @language]</code>	Ruft für PRINT- und RAISERROR-Anweisungen eine gespeicherte Nachrichtenzeichenfolge von ISYSUSER-MESSAGES ab
<code>sp_helptext [@objname]</code>	Zeigt den Text einer Systemprozedur, eines Triggers oder einer Ansicht an
<code>sp_password @caller_pswd , @new_pswd [, @login_name]</code>	Fügt ein Kennwort zu einer Benutzer-ID hinzu oder ändert es

Katalogprozeduren von Adaptive Server Enterprise

SQL Anywhere implementiert eine Teilmenge der Katalogprozeduren von Adaptive Server Enterprise. Diese Prozeduren übernehmen zwar dieselben Aufgaben wie in Adaptive Server Enterprise, sind aber nicht identisch. Wenn Sie vorhandene Skripten haben, die diese Prozeduren verwenden, sollten Sie den Prozedurcode überprüfen. Wenn den Text einer gespeicherten Prozedur anzeigen möchten, führen Sie den folgenden Befehl aus.

```
sp_helptext 'dbo.procedure_name'
```

Die implementierten Katalogprozeduren werden in der folgenden Tabelle beschrieben.

Name der Katalogprozedur/Parameter	Beschreibung
<code>sp_column_privileges</code>	Nicht unterstützt
<code>sp_columns [@table_name [, @table_owner [, @table_qualifier [, @column_name]]]]</code>	Gibt die Datentypen der angegebenen Spalten zurück

Name der Katalogprozedur/Parameter	Beschreibung
<code>sp_fkeys [@pktable_name [, @pktable_owner [, @pktable_qualifier [, @fktable_name [, @fktable_owner [, @fktable_qualifier]]]]]]</code>	Gibt Fremdschlüsselinformationen zur angegebenen Tabelle zurück
<code>sp_pkeys @table_name [, @table_owner [, @table_qualifier]]</code>	Gibt Primärschlüsselinformationen zur angegebenen Tabelle zurück
<code>sp_special_columns @table_name [, @table_owner [, @table_qualifier [, @col_type]]]</code>	Gibt die optimale Gruppe von Spalten zurück, die eine Zeile in der angegebenen Tabelle eindeutig identifizieren
<code>sp_sproc_columns @sp_name [, @sp_owner [, @sp_qualifier [, @column_name]]]</code>	Gibt Informationen zu den Eingabe- und Rückgabeparametern einer gespeicherten Prozedur zurück
<code>sp_statistics [@table_name [, @table_owner [, @table_qualifier [, @index_name [, @is_unique]]]]]</code>	Gibt Informationen zu Tabellen und deren Indizes zurück
<code>sp_stored_procedures [@sp_name [, @sp_owner [, @sp_qualifier]]]</code>	Gibt Informationen zu einer oder mehreren gespeicherten Prozeduren zurück
<code>sp_tables [@table_name [, @table_owner [, @table_qualifier [, @table_type]]]]</code>	Gibt eine Liste von Objekten zurück, die in einer FROM-Klausel erscheinen können

Alphabetische Liste der Systemprozeduren

Eigentümer von Systemprozeduren ist dbo. Einige dieser Prozeduren werden nur intern vom System verwendet. In diesem Abschnitt werden nur diejenigen dokumentiert, die nicht ausschließlich für das System und die interne Verwendung bestimmt sind.

sa_ansi_standard_packages-Systemprozedur

Gibt Informationen zu den in einer SQL-Anweisung verwendeten Nicht-Kern-Erweiterungen zurück.

Syntax

```
sa_ansi_standard_packages(  
    standard  
, statement  
)
```

Argumente

- **standard** Verwenden Sie diesen LONG VARCHAR-Parameter, um den bei Kernerweiterungen zu verwendenden Standard anzugeben. Entweder SQL:1999 oder SQL:2003.

- **statement** Verwenden Sie diesen LONG VARCHAR-Parameter, um die auszuwertende SQL-Anweisung anzugeben.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
package_id	VARCHAR(10)	Der Funktionsbezeichner.
package_name	LONG VARCHAR	Der Funktionsname.

Bemerkungen

Wenn bei der Anweisung keine Nicht-Kernerweiterungen verwendet werden, ist die Ergebnismenge leer.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Der SQL-Präprozessor“ [[SQL Anywhere Server - Programmierung](#)]
- „SQLFLAGGER-Funktion [Verschiedene]“ auf Seite 390
- „sql_flagger_error_level-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sql_flagger_warning_level-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Das folgende Beispiel ruft die sa_ansi_standard_packages-Systemprozedur auf:

```
CALL sa_ansi_standard_packages( 'SQL:2003',
'SELECT *
  FROM ( SELECT o.SalesRepresentative,
              o.Region,
              SUM( s.Quantity * p.UnitPrice ) AS total_sales,
              DENSE_RANK() OVER ( PARTITION BY o.Region,
                                  GROUPING( o.SalesRepresentative )
                                  ORDER BY total_sales DESC ) AS
sales_rank
  FROM Product p, SalesOrderItems s, SalesOrders o
  WHERE p.ID = s.ProductID AND s.ID = o.ID
  GROUP BY GROUPING SETS( ( o.SalesRepresentative, o.Region ),
                          o.Region ) ) AS DT
 WHERE sales_rank <= 3
  ORDER BY Region, sales_rank');
```

Im Beispiel wird die folgende Ergebnismenge generiert:

package_id	package_name
T612	Advanced OLAP operations

package_id	package_name
T611	Elementary OLAP operations
F591	Derived tables
T431	Extended grouping capabilities

sa_audit_string-Systemprozedur

Fügt eine Zeichenfolge zum Transaktionslog hinzu.

Syntax

sa_audit_string(*string*)

Argumente

- **string** Die VARCHAR(128)-Zeichenfolge, die zum Transaktionslog hinzugefügt werden soll.

Bemerkungen

Wenn Auditing aktiviert ist, fügt diese Systemprozedur den im Transaktionslog enthaltenen Audit-Daten einen Kommentar hinzu. Die Zeichenfolge kann maximal 128 Zeichen lang sein.

Privilegien

Sie müssen das MANAGE AUDITING-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „auditing-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Audits von Datenbankaktivitäten“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Im folgenden Beispiel wird sa_audit_string verwendet, um einen Kommentar zum Transaktionslog hinzuzufügen:

```
CALL sa_audit_string( 'Auditing test' );
```

sa_certificate_info-Systemprozedur

Zeigt Informationen zum angegebenen Zertifikat an, das in der Datenbank gespeichert ist.

Syntax

sa_certificate_info(*cert_name*)

Argumente

- **cert_name** Der CHAR(128)-Zertifikatname, der in der CREATE CERTIFICATE-Anweisung verwendet wurde.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
name	CHAR(128)	Der Name des Attributs.
Wert	LONG VARCHAR	Der Wert des Attributs.

Bemerkungen

Wenn cert_name NULL ist oder kein Zertifikat in der ISYSCERTIFICATE-Tabelle diesen Namen hat, werden keine Zeilen zurückgegeben. Andernfalls werden immer die folgenden Schlüssel zurückgegeben:

Name	Wert
Name	Wert der cert_name-Spalte in ISYSCERTIFICATE.
ID	Wert der object_id-Spalte in ISYSCERTIFICATE.

Andere zurückgegebene Zeilen enthalten Schlüssel, die Attributen des Zertifikats entsprechen, z.B. Namen (Common Name), Ländercode, Ort und Organisation.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- [„CREATE CERTIFICATE-Anweisung“ auf Seite 582](#)
- [„SYSCERTIFICATE-Systemansicht“ auf Seite 1440](#)

Beispiel

Im folgenden Beispiel werden Informationen zum Zertifikat mycert zurückgegeben:

```
CALL sa_certificate_info( 'mycert' );
```

sa_char_terms-Systemprozedur

Teilt eine CHAR-Zeichenfolge in Begriffe auf und gibt jeden Begriff als Zeile mit Angabe seiner Position zurück.

Syntax

```
sa_char_terms(  
  text  
  [, config_name  
  [, owner ] ]  
)
```

Argumente

- **text** Die LONG VARCHAR-Zeichenfolge, die Sie syntaktisch analysieren.
- **config_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um das Textkonfigurationsobjekt anzugeben, das beim Verarbeiten der Zeichenfolge übernommen werden soll. Der Standardwert ist 'default_char'.
- **owner** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümer des Textkonfigurationsobjekts anzugeben. Der Standardwert ist NULL. Der aktuelle Benutzer wird angenommen, wenn der Eigentümer nicht angegeben oder auf NULL gesetzt wird.

Bemerkungen

Sie können diese Systemprozedur verwenden, um herauszufinden, wie eine Zeichenfolge interpretiert wird, wenn die Einstellungen für ein Textkonfigurationsobjekt angewendet werden. Dies kann hilfreich sein, wenn Sie wissen möchten, welche Begriffe während der Indizierung oder aus einer Abfragezeichenfolge gelöscht würden.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Volltextsuche“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Konzepte und Referenz zu Textkonfigurationsobjekten“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „sa_nchar_terms-Systemprozedur“ auf Seite 1277

Beispiel

Die folgende Anweisung gibt die Begriffe in der CHAR-Zeichenfolge "It's a work-at-home day!" zurück und verwendet dazu das standardmäßige CHAR-Textkonfigurationsobjekt, default_char:

```
CALL sa_char_terms ('It''s a work-at-home day!', 'default_char', 'sys');
```

term	position
It	1
s	2

term	position
a	3
work	4
at	5
home	6
day	7

sa_check_commit-Systemprozedur

Überprüft vor dem Festschreiben, ob Verletzungen der referenziellen Integrität vorhanden sind

Syntax

```
sa_check_commit(  
  tname  
  , keyname  
)
```

Argumente

- **tname** Ein VARCHAR(128)-Parameter, der den Namen einer Tabelle mit einer Zeile enthält, die derzeit die referenzielle Integrität verletzt
- **keyname** Ein VARCHAR(128)-Parameter, der den Namen des entsprechenden Fremdschlüssel-Indexes enthält

Bemerkungen

Wenn die Datenbankoption wait_for_commit auf ON gesetzt ist oder ein Fremdschlüssel mithilfe von CHECK ON COMMIT in der Anweisung CREATE TABLE definiert wurde, können Sie die Datenbank auch dann aktualisieren, wenn dadurch die referenzielle Integrität verletzt wird, sofern Sie diese Verletzungen vor dem Festschreiben der Änderungen beseitigen.

Sie können mithilfe der sa_check_commit-Systemprozedur überprüfen, ob es ausstehende Verletzungen der referenziellen Integrität gibt, bevor Sie versuchen, Ihre Änderungen festzuschreiben.

Die zurückgegebenen Parameter zeigen den Namen der Tabelle an, die eine Zeile enthält, die derzeit die referenzielle Integrität verletzt. Der Name des entsprechenden Fremdschlüssel-Indexes wird ebenfalls angezeigt.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „wait_for_commit-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „CREATE TABLE-Anweisung“ auf Seite 737

Beispiel

Die folgende Gruppe von Anweisungen kann über Interactive SQL ausgeführt werden. Zeilen werden aus der Departments-Tabelle in der Beispieldatenbank gelöscht und eine Verletzung der referenziellen Integrität tritt ein. Der Aufruf der sa_check_commit-Systemprozedur überprüft, welche Tabellen und Schlüssel auffällige Verletzungen aufweisen, und das Zurücksetzen macht die Änderung rückgängig.

```
SET TEMPORARY OPTION wait_for_commit='On';
DELETE FROM Departments;
CREATE OR REPLACE VARIABLE tname VARCHAR( 128 );
CREATE OR REPLACE VARIABLE keyname VARCHAR( 128 );
CALL sa_check_commit( tname, keyname );
SELECT tname, keyname;
ROLLBACK;
SET TEMPORARY OPTION wait_for_commit='Off';
```

sa_clean_database-Systemprozedur

Startet den Datenbankaufräumvorgang und setzt die maximale Dauer für seine Ausführung

Syntax

sa_clean_database([*duration*])

Argumente

- **duration** Verwenden Sie diesen optionalen UNSIGNED INTEGER-Parameter, um anzugeben, wie viele Sekunden der Aufräumvorgang dauern darf. Der Standardwert ist 0 und wird so interpretiert, dass keine Grenze für die Dauer des Aufräumvorgangs gilt.

Bemerkungen

Der Datenbankaufräumvorgang ist eine interne Aufgabe, die nach einem Standardzeitplan ausgeführt wird. Sie können diese Systemprozedur verwenden, um den Datenbankaufräumvorgang dazu zu veranlassen, sofort auszuführen, und um anzugeben, wie lange der Aufräumvorgang bei jedem Aufruf dauern kann. Wenn der Wert 0 ist, läuft der Datenbankaufräumvorgang so lange, bis alle Seiten in allen DBSpaces bereinigt sind.

Wenn Sie diese Systemprozedur verwenden, um den Datenbankaufräumvorgang während einer Datenbankvalidierung zu starten, wird der Datenbankaufräumvorgang erst nach Abschluss der Validierung ausgeführt.

Manche Datenbankaufgaben, wie die Verarbeitung von Snapshot-Isolationstransaktionen, Indexwartung und das Löschen von Zeilen, können effizienter ausgeführt werden, wenn einige Teile der Anforderung auf einen späteren Zeitpunkt verschoben werden. Diese verschiebbaren Aktivitäten betreffen üblicherweise Aufräumvorgänge, bei denen gelöschte, veraltete und andere nicht mehr benötigte Einträge von Datenbankseiten entfernt oder Datenbankseiten für einen effizienteren Zugriff reorganisiert werden.

Das Verschieben einiger dieser Aktivitäten ermöglicht nicht nur einen schnelleren Abschluss der aktuellen Anforderung, sondern auch das Ausführen des Aufräumvorgangs zu einem Zeitpunkt, an dem der Datenbankserver weniger aktiv ist. Diese nicht mehr benötigten Einträge werden so gekennzeichnet, dass sie für andere Transaktionen nicht sichtbar sind. Sie beanspruchen jedoch Platz auf der Seite und müssen früher oder später entfernt werden.

Der Datenbankaufräumvorgang führt alle verschobenen Bereinigungsaktivitäten durch. Sein Zeitplan sieht vor, dass er alle 20 Sekunden läuft. Wenn er aufgerufen wird, durchsucht der Datenbankaufräumvorgang sequenziell die DBSpaces der Datenbank und überprüft bzw. bereinigt jede entsprechende Seite, bevor er zur nächsten geht. Wenn er automatisch vom Datenbankserver aufgerufen wird, ist der Datenbankaufräumvorgang ein selbstoptimierender Prozess. Die Anzahl der Aufgaben, die der Datenbankaufräumvorgang durchführt, und die Dauer seiner Ausführung hängen von mehreren Faktoren ab, wie vom Prozentsatz von zu bereinigenden Seiten in einem DBSpace, vom aktuellen Aktivitätsumfang im Datenbankserver und von der Zeit, die der Datenbankaufräumvorgang bereits läuft. Wenn der Aufräumvorgang, nachdem er 0,5 Sekunden gelaufen ist, eine aktive Anforderung im Server feststellt, stoppt er und erstellt einen neuen Zeitplan, um zu seinem regelmäßigen Intervall auszuführen. Der Datenbankaufräumvorgang versucht, Seiten zu verarbeiten, wenn keine anderen Anforderungen im Server ausgeführt werden, und nutzt daher die Perioden der Inaktivität des Servers.

Datenbankaufräumvorgangs-Statistiken sind über vier Datenbankeigenschaften verfügbar:

- **CleanablePagesAdded** Gibt die Anzahl der Seiten zurück, die bereinigt werden müssen
- **CleanablePagesCleaned** Gibt die Anzahl der Seiten zurück, die bereits bereinigt sind
- **CleanableRowsAdded** Gibt die Anzahl der Zeilen zurück, die bereinigt werden müssen
- **CleanableRowsCleaned** Gibt die Anzahl der Zeilen zurück, die bereits bereinigt sind

Die Differenz zwischen den Werten von CleanablePagesAdded und CleanablePagesCleaned gibt an, wie viele Datenbankseiten noch bereinigt werden müssen.

Sie können die sa_clean_database-Systemprozedur verwenden, um den Datenbankaufräumvorgang so zu konfigurieren, dass er läuft, bis alle Seiten in einer Datenbank bereinigt sind, oder um eine maximale Dauer für die Ausführung des Datenbankaufräumvorgangs festzulegen.

Um das Verhalten des Datenbankaufräumvorgangs weiter anzupassen, können Sie ein Ereignis bestimmen, das den Datenbankaufräumvorgang startet, wenn die Anzahl der zu bereinigenden Seiten oder Zeilen einen angegebenen Schwellenwert überschreitet.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „CREATE EVENT-Anweisung“ auf Seite 606
- CleanablePagesAdded-Datenbankeigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]
- CleanablePagesCleaned-Datenbankeigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]
- CleanableRowsAdded-Datenbankeigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]
- CleanableRowsCleaned-Datenbankeigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Das folgende Beispiel setzt die Dauer des Datenbankaufräumvorgangs auf 10 Sekunden:

```
CALL sa_clean_database( 10 );
```

Das folgende Beispiel erstellt ein geplantes Ereignis, das täglich ausgeführt wird, um es dem Datenbankaufräumvorgang zu ermöglichen, so lange auszuführen, bis alle Seiten in der Datenbank bereinigt sind:

```
CREATE EVENT DailyDatabaseCleanup
SCHEDULE
  START TIME '6:00 pm'
  ON ( 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday' )
HANDLER
  BEGIN
    CALL sa_clean_database( );
  END;
```

Das folgende Beispiel erzwingt die Ausführung des Datenbankaufräumvorgangs, wenn 20 % oder mehr der Seiten in der Datenbank bereinigt werden müssen:

```
CREATE EVENT PeriodicCleaner
SCHEDULE
  BETWEEN '9:00 am' and '5:00 pm'
  EVERY 1 HOURS
HANDLER
  BEGIN
    DECLARE @num_db_pages INTEGER;
    DECLARE @num_dirty_pages INTEGER;

    -- Get the number of database pages
    SELECT (SUM( DB_EXTENDED_PROPERTY( 'FileSize', t.dbpace_id ) -
              DB_EXTENDED_PROPERTY( 'FreePages', t.dbpace_id ) ))
    INTO @num_db_pages
    FROM (SELECT dbpace_id FROM SYSDBSPACE) AS t;

    -- Get the number of dirty pages to be cleaned
    SELECT (DB_PROPERTY( 'CleanablePagesAdded' ) -
           DB_PROPERTY( 'CleanablePagesCleaned' ))
    INTO @num_dirty_pages;

    -- Check whether the number of dirty pages exceeds 20% of
    -- the size of the database
    IF @num_dirty_pages > @num_db_pages * 0.20 THEN
      -- Start cleaning the database for a maximum of 60 seconds
      CALL sa_clean_database( 60 );
    END IF;
  END;
```

sa_column_stats-Systemprozedur

Gibt verschiedene Statistiken über die angegebene(n) Spalte(n) zurück. Diese Statistiken hängen nicht mit den Spaltenstatistiken zusammen, die für die Verwendung durch den Optimierer geführt werden.

Syntax

```
sa_column_stats(
  [ tab_name
  [, col_name
  [, tab_owner
  [, max_rows ] ] ]
)
```

Argumente

- **tab_name** Dieser optionale CHAR(128)-Parameter gibt den Eigentümer der Tabelle an. Wenn dieser Parameter nicht angegeben ist, werden Statistiken für alle Spalten in allen Tabellen erstellt. Der Standardwert ist '%'.
- **col_name** Dieser optionale CHAR(128)-Parameter gibt die Spalten an, für die Statistiken erstellt werden sollen. Wenn dieser Parameter nicht angegeben ist, werden Statistiken für alle Spalten in der/den angegebenen Tabelle(n) erstellt. Der Standardwert ist '%'.
- **tab_owner** Dieser optionale CHAR(128)-Parameter gibt den Eigentümer der Tabelle an. Wenn dieser Parameter nicht angegeben wird, verwendet der Datenbankserver den Eigentümer der ersten Tabelle, die dem angegebenen *tab_name* entspricht. Der Standardwert ist '%'.
- **max_rows** Dieser optionale INTEGER-Parameter gibt die Anzahl der Zeilen an, die für die Berechnung herangezogen werden. Der Standardwert ist 1000. Die Angabe von 0 teilt dem Datenbankserver mit, das Ergebnis basierend auf allen Zeilen in der Tabelle zu berechnen.

Ergebnismenge

Mit der Ausnahme von *table_owner*, *table_name* und *column_name* sind bei Nicht-Zeichenfolgenspalten alle Werte in der Ergebnismenge NULL. Auch sind bei leeren Tabellen *num_rows_processed* und *num_values_compressed* 0, während alle anderen Werte NULL sind.

Spaltenname	Datentyp	Beschreibung
table_owner	CHAR(128)	Eigentümer der Tabelle
table_name	CHAR(128)	Der Tabellename.
column_name	CHAR(128)	Der Spaltenname.
num_rows_processed	INTEGER	Die Gesamtanzahl der Zeilen, die zum Berechnen der Statistiken gelesen werden
num_values_compressed	INTEGER	Die Anzahl der Werte in der Spalte, die komprimiert sind. Wenn die Spalte nicht komprimiert ist, ist der Wert "0".

Spaltenname	Datentyp	Beschreibung
avg_compression_ratio	DOUBLE	Das durchschnittliche Komprimierungsverhältnis, ausgedrückt als prozentuale Größenverminderung, von komprimierten Werten in der Spalte. Wenn die Spalte nicht komprimiert ist, ist der Wert "0".
avg_length	DOUBLE	Die durchschnittliche Länge aller Nicht-NULL-Zeichenfolgen in der Spalte
stddev_length	DOUBLE	Die Längen-Standardabweichung aller Nicht-NULL-Zeichenfolgen in der Spalte
min_length	INTEGER	Die minimale Länge von Nicht-NULL-Zeichenfolgen in der Spalte
max_length	INTEGER	Die maximale Länge von Zeichenfolgen in der Spalte
avg_uncompressed_length	DOUBLE	Die durchschnittliche Länge aller unkomprimierten Nicht-NULL-Zeichenfolgen in der Spalte
stddev_uncompressed_length	DOUBLE	Die Längen-Standardabweichung aller unkomprimierten Nicht-NULL-Zeichenfolgen in der Spalte
min_uncompressed_length	INTEGER	Die minimale Länge aller unkomprimierten Nicht-NULL-Zeichenfolgen in der Spalte
max_uncompressed_length	INTEGER	Die maximale Länge aller unkomprimierten Nicht-NULL-Zeichenfolgen in der Spalte

Bemerkungen

Der Datenbankserver bestimmt die Spalten, die dem Eigentümer-, Tabellen- und Spaltennamen entsprechen, und berechnet Statistiken für die Daten in jeder angegebenen Spalte. Standardmäßig verwendet der Datenbankserver nur die ersten 1000 Datenzeilen.

Bei avg_compression_ratio dürfen Werte nicht größer oder gleich 100 sein. Sie können jedoch kleiner als 0 sein, wenn unkomprimierbare Daten (z.B. bereits komprimierte Daten) in eine komprimierte Spalte eingefügt werden. Höhere Werte zeigen eine bessere Komprimierung an. Wenn die zurückgegebene Zahl z.B. 80 ist, ist die Größe der komprimierten Daten um 80 % geringer als die der unkomprimierten Daten.

Privilegien

Sie müssen Eigentümer der Tabelle sein, das SELECT-Privileg für die Spalten haben oder das MONITOR-Systemprivileg oder das MANAGE ANY STATISTICS-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Hinweise zur Spaltenkomprimierung“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

In diesem Beispiel verwenden Sie die sa_column_stats-Systemprozedur in einer SELECT-Anweisung, um zu bestimmen, welche Spalten in der Datenbank am meisten von der Spaltenkomprimierung profitieren:

```
SELECT * FROM sa_column_stats()
WHERE num_values_compressed > 0
ORDER BY avg_compression_ratio desc;
```

In diesem Beispiel begrenzen Sie Ihre Auswahl aus dem vorherigen Beispiel auf Tabellen, die 'bsmith' gehören:

```
SELECT * FROM sa_column_stats( tab_owner='GROUPO' )
WHERE num_values_compressed > 0
ORDER BY avg_compression_ratio desc;
```

sa_conn_activity-Systemprozedur

Gibt für jede einzelne Verbindung mit der angegebenen Datenbank auf dem Server die zuletzt vorbereitete SQL-Anweisung zurück.

Syntax

```
sa_conn_activity( [ connidparm ] )
```

Argumente

- **connidparm** Verwenden Sie diesen optionalen INTEGER-Parameter, um die Verbindungs-ID-Nummer anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
Number	INTEGER	Gibt die Verbindungs-ID (eine Nummer) für die aktuelle Verbindung zurück.
Name	VARCHAR(255)	Gibt den Namen der aktuellen Verbindung zurück. Bei temporären Verbindungsnamen wird dem Verbindungsnamen INT: vorangestellt.
Userid	VARCHAR(255)	Gibt die Benutzer-ID für die Verbindung zurück.
DBNumber	INTEGER	Gibt die ID-Nummer der Datenbank zurück.

Spaltenname	Datentyp	Beschreibung
LastReqTime	VARCHAR(255)	Gibt den Zeitpunkt zurück, an dem die letzte Anforderung für die angegebene Verbindung gestartet wurde. Diese Eigenschaft kann eine leere Zeichenfolge für interne Verbindungen wie zum Beispiel Ereignisse zurückgeben.
LastStatement	LONG VARCHAR	Gibt die zuletzt vorbereitete SQL-Anweisung für die aktuelle Verbindung zurück.

Bemerkungen

Wenn *connidparm* kleiner als Null ist, werden Informationen für die aktuelle Verbindung zurückgegeben. Wenn kein *connidparm* angegeben wird, werden Informationen zu allen Verbindungen mit allen auf dem Datenbankserver laufenden Datenbanken zurückgegeben.

Die *sa_conn_activity*-Systemprozedur gibt eine Ergebnismenge zurück, die aus der zuletzt vorbereiteten SQL-Anweisung für die Verbindung besteht. Das Aufzeichnen von Anweisungen muss auf dem Datenbankserver aktiviert sein, bevor *sa_conn_activity* aufgerufen wird. Geben Sie dazu beim Starten des Datenbankservers die Option *-zl* an oder führen Sie folgenden Prozeduraufruf aus:

```
CALL sa_server_option('RememberLastStatement','ON');
```

Diese Prozedur ist nützlich, wenn der Datenbankserver beschäftigt ist und Sie Informationen über die letzte SQL-Anweisung benötigen, die für jede Verbindung vorbereitet wurde. Diese Funktion kann als Alternative zur Anforderungsprotokollierung verwendet werden.

🔥 **Cloud-Hinweis:** Aufgrund von Isolationsregeln für Tenant-Datenbanken werden bei der Ausführung dieser Systemprozedur in der Cloud nur Informationen über die aktuelle Tenant-Datenbank zurückgegeben.

Privilegien

Keine Privilegien sind erforderlich, um diese Systemprozedur für die aktuelle Verbindungs-ID auszuführen. Um diese Systemprozedur für andere Verbindungen ausführen zu können, benötigen Sie das SERVER OPERATOR-Systemprivileg, das MONITOR-Systemprivileg oder das DROP CONNECTION-Systemprivileg.

Nebenwirkungen

Keine

Siehe auch

- „Datenbankserveroption -zl“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sa_server_option-Systemprozedur“ auf Seite 1306
- Name-Verbindungseigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]
- „Liste der Verbindungseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Im folgenden Beispiel wird die `sa_conn_activity`-Systemprozedur verwendet, um die zuletzt vorbereitete SQL-Anweisung für jede einzelne Verbindung anzuzeigen.

```
CALL sa_conn_activity( );
```

Number	Name	Userid	DBNumber	...
1,949	SQL_DBC_117acc40	DBA	0	...
1,948	setup	User1	0	...
...

sa_conn_compression_info-Systemprozedur

Fasst Komprimierungsraten zusammen

Syntax

```
sa_conn_compression_info( [ connidparm ] )
```

Argumente

- **connidparm** Verwenden Sie diesen optionalen INTEGER-Parameter, um die Verbindungs-ID-Nummer anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
Type	VAR-CHAR(20)	Gibt eine Zeichenfolge zurück, mit der festgelegt wird, ob die nachfolgende Komprimierungsstatistik sich auf eine Verbindung (Connection) oder auf alle Verbindungen mit dem Server (Server) bezieht
ConnNumber	INTEGER	Ein INTEGER-Wert, der eine Verbindungs-ID-Nummer darstellt. Gibt Null zurück, wenn der Typ Server ist.
Compression	VAR-CHAR(10)	Gibt "On" bzw. "Off" zurück, um anzuzeigen, ob die Kommunikationskomprimierung für die Verbindung aktiviert ist. Gibt NULL zurück, wenn der Typ "Server" ist, oder ON/OFF, wenn der Typ "Connection" ist.
TotalBytes	INTEGER	Gibt eine Ganzzahl zurück, die die Gesamtzahl der tatsächlich gesendeten und empfangenen Bytes repräsentiert

Spaltenname	Datentyp	Beschreibung
TotalBytesUn-Comp	INTEGER	Gibt eine Ganzzahl zurück, die die Anzahl der Bytes repräsentiert, die bei deaktivierter Komprimierung gesendet und empfangen worden wären
CompRate	NUMERIC(5,2)	Gibt einen Wert vom Typ NUMERIC(5,2) zurück, der die allgemeine Komprimierungsrate repräsentiert. Ein Wert von "0" gibt z.B. an, dass keine Komprimierung erfolgt ist. Ein Wert von "75" gibt an, dass die Daten um 75 % bzw. auf ein Viertel der ursprünglichen Größe komprimiert wurden.
CompRateSent	NUMERIC(5,2)	Gibt einen Wert vom Typ NUMERIC(5,2) zurück, der die Komprimierungsrate für an den Client gesendete Daten repräsentiert
CompRateReceived	NUMERIC(5,2)	Gibt einen Wert vom Typ NUMERIC(5,2) zurück, der die Komprimierungsrate für vom Client empfangene Daten repräsentiert.
TotalPackets	INTEGER	Gibt eine Ganzzahl zurück, die die Gesamtzahl der tatsächlich gesendeten und empfangenen Pakete repräsentiert.
TotalPacketsUn-Comp	INTEGER	Gibt eine Ganzzahl zurück, die die Anzahl der Pakete repräsentiert, die bei deaktivierter Komprimierung gesendet und empfangen worden wären.
CompPktRate	NUMERIC(5,2)	Gibt einen Wert vom Typ NUMERIC(5,2) zurück, der die allgemeine Komprimierungsrate der Pakete repräsentiert.
CompPktRateSent	NUMERIC(5,2)	Gibt einen Wert vom Typ NUMERIC(5,2) zurück, der die Komprimierungsrate für an den Client gesendete Pakete repräsentiert.
CompPktRateReceived	NUMERIC(5,2)	Gibt einen Wert vom Typ NUMERIC(5,2) zurück, der die Komprimierungsrate für vom Client empfangene Pakete repräsentiert.

Bemerkungen

Wenn *connidparm* kleiner als Null ist, wird eine Ergebnismenge zurückgegeben, die aus Komprimierungseigenschaften für die aktuelle Verbindung besteht. Wenn kein *connidparm* oder NULL angegeben wird, werden Komprimierungseigenschaften für alle Verbindungen mit allen auf dem Datenbankserver laufenden Datenbanken zurückgegeben.

Privilegien

Keine Privilegien sind erforderlich, um diese Systemprozedur für die aktuelle Verbindungs-ID auszuführen. Um diese Systemprozedur für andere Verbindungen ausführen zu können, benötigen Sie das SERVER OPERATOR-Systemprivileg, das MONITOR-Systemprivileg oder das DROP CONNECTION-Systemprivileg.

Nebenwirkungen

Keine

Siehe auch

- „Liste der Verbindungseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Im folgenden Beispiel wird die `sa_conn_compression_info`-Systemprozedur verwendet, um eine Ergebnismenge zurückzugeben, die Komprimierungseigenschaften für alle Verbindungen zum Server zusammenfasst.

```
CALL sa_conn_compression_info( );
```

Type	ConnNumber	Compression	TotalBytes	...
Connection	79	Off	7841	...
Server	(NULL)	(NULL)	2737761	...
...

sa_conn_info-Systemprozedur

Liefert Informationen über Verbindungseigenschaften

Syntax

```
sa_conn_info( [ connidparm ] )
```

Argumente

- **connidparm** Dieser optionale INTEGER-Parameter gibt die Verbindungs-ID-Nummer an. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
Number	INTEGER	Gibt die Verbindungs-ID (eine Nummer) für die aktuelle Verbindung zurück.
Name	VAR-CHAR(255)	Gibt die Verbindungs-ID (eine Nummer) für die aktuelle Verbindung zurück. Bei temporären Verbindungsnamen wird dem Verbindungsnamen INT: vorangestellt.

Spaltenname	Datentyp	Beschreibung
Userid	VAR-CHAR(255)	Gibt die Benutzer-ID für die Verbindung zurück.
DBNumber	INTEGER	Gibt die ID-Nummer der Datenbank zurück.
LastReqTime	VAR-CHAR(255)	Gibt den Zeitpunkt zurück, an dem die letzte Anforderung für die angegebene Verbindung gestartet wurde. Diese Eigenschaft kann eine leere Zeichenfolge für interne Verbindungen wie zum Beispiel Ereignisse zurückgeben.
ReqType	VAR-CHAR(255)	Gibt den Typ der letzten Anforderung zurück. Wenn eine Verbindung vom Verbindungspooling im Cache abgelegt wurde, lautet ihr ReqType-Wert CONNECT_POOL_CACHE.
CommLink	VAR-CHAR(255)	Gibt die Kommunikationsverbindung für die Verbindung zurück. Dies ist eines der von SQL Anywhere unterstützten Netzwerkprotokolle oder "local" bei einer Verbindung auf einem Computer.
NodeAddr	VAR-CHAR(255)	Gibt die Adresse für den Client in einer Client/Server-Verbindung zurück.
ClientPort	INTEGER	Gibt die TCP/IP-Portnummer des Clients oder "0" zurück, wenn die Verbindung keine TCP/IP-Verbindung ist.
ServerPort	INTEGER	Gibt die TCP/IP-Portnummer des Datenbankservers zurück, oder "0".
BlockedOn	INTEGER	Gibt Null zurück, wenn die aktuelle Verbindung nicht blockiert ist, bzw. im Fall einer Blockierung die Verbindungsnummer, auf der die Verbindung aufgrund eines Sperrenkonflikts blockiert ist.
LockRowID	UNSIGNED BIGINT	<p>Gibt den Bezeichner der gesperrten Zeile zurück.</p> <p>LockRowID ist NULL, wenn die Verbindung nicht auf eine einer Zeile zugeordnete Sperre wartet (d.h., sie wartet nicht auf eine Sperre oder sie wartet auf eine Sperre, der keine Zeilen zugeordnet sind).</p>
LockIndexID	INTEGER	<p>Gibt den Bezeichner der gesperrten Indexes zurück.</p> <p>LockIndexID ist -1, wenn die Sperre in LockTable allen Indizes für die Tabelle zugeordnet ist. LockIndexID ist NULL, wenn die Verbindung nicht auf eine einem Index zugeordnete Sperre wartet (d.h., sie wartet nicht auf eine Sperre oder sie wartet auf eine Sperre, der kein Index zugeordnet ist).</p>

Spaltenname	Datentyp	Beschreibung
LockTable	VAR-CHAR(255)	Gibt den Namen der einer Sperre zugeordneten Tabelle zurück, wenn die Verbindung derzeit auf eine Sperre wartet. Andernfalls gibt LockTable eine leere Zeichenfolge zurück.
UncommitOps	INTEGER	Gibt die Anzahl der nicht festgeschriebenen Vorgänge zurück.
ParentConnection	INTEGER	Gibt die Verbindungs-ID der Verbindung an, die eine temporäre Verbindung für die Ausführung eines Datenbankvorgangs erstellt hat (wie etwa die Durchführung einer Sicherung oder das Erstellen einer Datenbank). Bei anderen Typen von Verbindungen gibt diese Eigenschaft NULL zurück.

Bemerkungen

Wenn *connidparm* kleiner ist als Null, wird eine Ergebnismenge zurückgegeben, die aus Verbindungseigenschaften für die aktuelle Verbindung besteht. Wenn kein *connidparm* oder NULL angegeben wird, werden Verbindungseigenschaften für alle Verbindungen mit allen auf dem Datenbankserver laufenden Datenbanken zurückgegeben.

Bei einer blockierten Verbindung wird der von dieser Prozedur zurückgegebene Wert "BlockedOn" verwendet, um zu ermitteln, welche Benutzer blockiert sind und von wem sie blockiert werden. Die *sa_locks*-Systemprozedur kann verwendet werden, um die Sperren anzuzeigen, die der blockierenden Verbindung gehören.

Um weitere auf diesen Eigenschaften basierende Informationen zu erhalten, können Sie eine Abfrage ähnlich der Folgenden ausführen:

```
SELECT *, DB_NAME( DBNumber ),
        CONNECTION_PROPERTY( 'LastStatement', Number )
FROM sa_conn_info( );
```

Der Wert von LockRowID kann verwendet werden, um nach einer Sperre in der Ausgabe der *sa_locks*-Prozedur zu suchen.

Der Wert in LockIndexID kann verwendet werden, um nach einer Sperre in der Ausgabe der *sa_locks*-Prozedur zu suchen. Der Wert in LockIndexID entspricht zudem dem Primärschlüssel in der ISYSDX-Systemtabelle, die unter Verwendung der SYSIDX-Systemansicht angezeigt werden kann.

Jede Sperre hat eine zugeordnete Tabelle, daher kann der Wert von LockTable verwendet werden, um eindeutig zu ermitteln, ob eine Verbindung auf eine Sperre wartet.

🔥 **Cloud-Hinweis:** Aufgrund von Isolationsregeln für Tenant-Datenbanken werden bei der Ausführung dieser Systemprozedur in der Cloud nur Informationen über die aktuelle Tenant-Datenbank zurückgegeben.

Privilegien

Keine Privilegien sind erforderlich, um diese Systemprozedur für die aktuelle Verbindungs-ID auszuführen. Um diese Systemprozedur für andere Verbindungen ausführen zu können, benötigen Sie das

SERVER OPERATOR-Systemprivileg, das MONITOR-Systemprivileg oder das DROP CONNECTION-Systemprivileg.

Nebenwirkungen

Keine

Siehe auch

- „Liste der Verbindungseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- Name-Verbindungseigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]
- „sa_locks-Systemprozedur“ auf Seite 1250
- „SYSIDX-Systemansicht“ auf Seite 1456

Beispiele

Im folgenden Beispiel wird die sa_conn_info-Systemprozedur verwendet, um eine Ergebnismenge zurückzugeben, die Verbindungseigenschaften für alle Verbindungen zum Server zusammenfasst.

```
CALL sa_conn_info( );
```

Number	Name	Userid	DBNumber	...
79	SQL_DBC_10dcf810	DBA	0	...
46	setup	User1	0	...
...

Im folgenden Beispiel wird die sa_conn_info-Systemprozedur verwendet, um eine Ergebnismenge zurückzugeben, die zeigt, welche Verbindung eine temporäre Verbindung erstellt hat.

```
SELECT Number, Name, ParentConnection FROM sa_conn_info();
```

Verbindung 8 hat die temporäre Verbindung erstellt, von der eine DATABASE CREATE-Anweisung ausgeführt wurde.

```
Number      Name      ParentConnection
-----
1000000048  INT: CreateDB      8
9           SQL_DBC_14675af8    (NULL)
8           SQL_DBA_152d5ac0    (NULL)
```

sa_conn_list-Systemprozedur

Gibt eine Ergebnismenge zurück, die Verbindungs-IDs enthält

Syntax

```
sa_conn_list(
[ connidparm
[, dbidparm ] ]
)
```

Argumente

- **connidparm** Verwenden Sie diesen optionalen INTEGER-Parameter, um die Verbindungs-ID-Nummer anzugeben. Der Standardwert ist NULL.
- **dbidparm** Verwenden Sie diesen optionalen INTEGER-Parameter, um die Datenbank-ID-Nummer anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
Number	INTEGER	Gibt die Verbindungs-ID (eine Nummer) für die aktuelle Verbindung zurück.

Bemerkungen

Wenn *connidparm* größer als Null ist, werden Informationen für die angegebene Verbindung zurückgegeben. Wenn *connidparm* kleiner als Null ist, werden Informationen für die aktuelle Verbindung zurückgegeben. Wenn *connidparm* und *dbidparm* nicht angegeben werden oder NULL sind, werden die Verbindungs-IDs für alle Verbindungen mit allen auf dem Datenbankserver laufenden Datenbanken zurückgegeben.

Wenn *connidparm* NULL ist und *dbidparm* größer oder gleich Null ist, werden nur Verbindungs-IDs für diese Datenbank zurückgegeben. Wenn *connidparm* NULL und *dbidparm* kleiner als Null ist, werden nur Verbindungs-IDs für die aktuelle Datenbank zurückgegeben.

Privilegien

Keine Privilegien sind erforderlich, um diese Systemprozedur auszuführen und die aktuelle Verbindung zurückzugeben. Um diese Systemprozedur für andere Verbindungen ausführen zu können, benötigen Sie das SERVER OPERATOR-Systemprivileg, das MONITOR-Systemprivileg oder das DROP CONNECTION-Systemprivileg.

Nebenwirkungen

Keine

Siehe auch

- [„sa_db_list-Systemprozedur“ auf Seite 1197](#)
- [„sa_conn_options-Systemprozedur“ auf Seite 1188](#)

Beispiel

Im folgenden Beispiel wird die *sa_conn_list*-Systemprozedur verwendet, um eine Liste von Verbindungs-IDs anzuzeigen.

```
CALL sa_conn_list( );
```

Number
1,949

Number
1,948
...

sa_conn_options-Systemprozedur

Gibt Informationen zu Verbindungseigenschaften zurück, die Datenbankoptionen entsprechen

Syntax

sa_conn_options([*connidparm*])

Argumente

- **connidparm** Verwenden Sie diesen optionalen INTEGER-Parameter, um die Verbindungs-ID-Nummer anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
Number	INTEGER	Gibt die Verbindungs-ID (eine Nummer) für die aktuelle Verbindung zurück.
PropNum	INTEGER	Gibt die Nummer der Verbindungseigenschaft zurück.
OptionName	VARCHAR(255)	Gibt den Optionsnamen zurück.
OptionDescription	VARCHAR(255)	Gibt die Optionsbeschreibung zurück.
Value	LONG VARCHAR	Gibt den Optionswert zurück.

Bemerkungen

Gibt die Verbindungs-ID als Nummer (Number) zurück und gibt die Nummer der Verbindungseigenschaft (PropNum), den Optionsnamen (OptionName), die Optionsbeschreibung (OptionDescription) und den Optionswert (Value) für jede verfügbare Verbindungseigenschaft zurück, die einer Datenbankoption entspricht.

Wenn *connidparm* kleiner als Null ist, werden Optionswerte für die aktuelle Verbindung zurückgegeben. Wenn *connidparm* nicht angegeben wird oder NULL ist, werden Optionswerte für alle Verbindungen mit der aktuellen Datenbank zurückgegeben.

Privilegien

Keine Privilegien sind erforderlich, um diese Systemprozedur für die aktuelle Verbindungs-ID auszuführen. Um diese Systemprozedur für andere Verbindungen ausführen zu können, benötigen Sie das

SERVER OPERATOR-Systemprivileg, das MONITOR-Systemprivileg oder das DROP CONNECTION-Systemprivileg.

Nebenwirkungen

Keine

Siehe auch

- „sa_db_list-Systemprozedur“ auf Seite 1197
- „sa_conn_list-Systemprozedur“ auf Seite 1186
- „Liste der Verbindungseigenschaften“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankoptionen“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Im folgenden Beispiel wird die sa_conn_options-Systemprozedur verwendet, um Eigenschaftsinformationen für Verbindungseigenschaften anzuzeigen, die Datenbankoptionen entsprechen.

```
CALL sa_conn_options( );
```

Number	PropNum	OptionName	OptionDescription	Value
1,952	461	blocking	Steuert Antwort auf Sperrenkonflikte	On
1,952	462	blocking_timeout	Steuert, wie lange eine Transaktion auf den Erhalt einer Sperre wartet	0
...

sa_conn_properties-Systemprozedur

Liefert Informationen über Verbindungseigenschaften

Syntax

```
sa_conn_properties( [ connidparm ] )
```

Argumente

- **connidparm** Verwenden Sie diesen optionalen INTEGER-Parameter, um die Verbindungs-ID-Nummer anzugeben. Der Standardwert ist NULL.

Ergebnismenge


Spaltenname	Datentyp	Beschreibung
Number	INTEGER	Gibt die Verbindungs-ID (eine Nummer) für die aktuelle Verbindung zurück.

Spaltenname	Datentyp	Beschreibung
PropNum	INTEGER	Gibt die Nummer der Verbindungseigenschaft zurück.
PropName	VARCHAR(255)	Gibt den Namen der Verbindungseigenschaft zurück.
PropDescription	VARCHAR(255)	Gibt die Beschreibung der Verbindungseigenschaft zurück.
Value	LONG VARCHAR	Gibt den Wert der Verbindungseigenschaft zurück.

Bemerkungen

Gibt die Verbindungs-ID als Nummer (Number) zurück und gibt die Nummer der Verbindungseigenschaft (PropNum), den Eigenschaftsnamen (PropName), die Eigenschaftsbeschreibung (PropDescription) und den Wert (Value) für jede verfügbare Verbindungseigenschaft zurück. Für alle Verbindungseigenschaften, Datenbankoptionseinstellungen für Verbindungen und Statistiken für Verbindungen werden Werte zurückgegeben. Gültige Eigenschaften mit NULL-Werten werden ebenfalls zurückgegeben.

Wenn *connidparm* kleiner als Null ist, werden Eigenschaftswerte für die aktuelle Verbindung zurückgegeben. Wenn *connidparm* nicht angegeben wird oder NULL ist, werden Eigenschaftswerte für alle Verbindungen mit der aktuellen Datenbank zurückgegeben.

 **Cloud-Hinweis:** Aufgrund von Isolationsregeln für Tenant-Datenbanken werden bei der Ausführung dieser Systemprozedur in der Cloud nur Informationen über die aktuelle Tenant-Datenbank zurückgegeben.

Privilegien

Keine Privilegien sind erforderlich, um diese Systemprozedur für die aktuelle Verbindungs-ID auszuführen. Um diese Systemprozedur für andere Verbindungen ausführen zu können, benötigen Sie das SERVER OPERATOR-Systemprivileg, das MONITOR-Systemprivileg oder das DROP CONNECTION-Systemprivileg.

Nebenwirkungen

Keine

Siehe auch

- „sa_conn_list-Systemprozedur“ auf Seite 1186
- „sa_conn_options-Systemprozedur“ auf Seite 1188
- „Systemfunktionen“ auf Seite 165
- „Liste der Verbindungseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiele

Im folgenden Beispiel wird die sa_conn_options-Systemprozedur verwendet, um eine Ergebnismenge zurückzugeben, die Informationen über die Verbindungseigenschaften für alle Verbindungen zusammenfasst.

```
CALL sa_conn_properties( );
```

Number	PropNum	PropName	...
79	37	ClientStmtCacheHits	...
79	38	ClientStmtCacheMisses	...
...

In diesem Beispiel wird die sa_conn_properties-Systemprozedur verwendet, um eine Liste aller Verbindungen in absteigender Reihenfolge der CPU-Zeit zurückzugeben*:

```
SELECT Number AS connection_number,
       CONNECTION_PROPERTY ( 'Name', Number ) AS connection_name,
       CONNECTION_PROPERTY ( 'Userid', Number ) AS user_id,
       CAST ( Value AS NUMERIC ( 30, 2 ) ) AS approx_cpu_time
FROM sa_conn_properties( )
WHERE PropName = 'ApproximateCPUTime'
ORDER BY approx_cpu_time DESC;
```

*Beispiel mit freundlicher Genehmigung von Breck Carter, RisingRoad Professional Services (<http://www.risingroad.com>).

sa_convert_ml_progress_to_timestamp-Systemprozedur

Nur für skriptgesteuerte MobiLink-Uploads. Konvertiert den Verarbeitungsfortschrittswert für skriptgesteuerte Uploads von UNSIGNED BIGINT in TIMESTAMP.

Syntax

sa_convert_ml_progress_to_timestamp(*progress*)

Argumente

- **progress** Verwenden Sie diesen UNSIGNED BIGINT-Parameter, um den Fortschrittswert anzugeben, der in einen TIMESTAMP-Wert konvertiert werden soll.

Rückgabe

Die Funktion gibt den TIMESTAMP-Wert zurück, der durch den übergebenen Wert dargestellt wird.

Bemerkungen

Diese Funktion ist die Umkehrung von sa_convert_timestamp_to_ml_progress.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „sa_convert_timestamp_to_ml_progress-Systemprozedur“ auf Seite 1192
- „Skriptgesteuerter Upload“ [*MobiLink - Clientadministration*]

Beispiel

Im folgenden Beispiel wird ein UNSIGNED BIGINT-Wert in einen Zeitstempelwert (2009-10-20 13:36:51.199) konvertiert.

```
SELECT sa_convert_ml_progress_to_timestamp( 3465034611199 );
```

sa_convert_timestamp_to_ml_progress-Systemprozedur

Nur für skriptgesteuerte MobiLink-Uploads. Konvertiert den Verarbeitungsfortschrittswert für skriptgesteuerte Uploads von TIMESTAMP in UNSIGNED BIGINT.

Syntax

```
sa_convert_timestamp_to_ml_progress( t1 )
```

Argumente

- **t1** Verwenden Sie diesen TIMESTAMP-Parameter, um den Fortschrittswert anzugeben, der in UNSIGNED BIGINT konvertiert werden soll.

Rückgabe

Die Funktion gibt einen UNSIGNED BIGINT-Wert zurück, der den in einem Parameter übergebenen Zeitstempel darstellt.

Bemerkungen

Diese Prozedur ist die Umkehrung von sa_convert_ml_progress_to_timestamp.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „sa_convert_ml_progress_to_timestamp-Systemprozedur“ auf Seite 1191
- „Skriptgesteuerter Upload“ [*MobiLink - Clientadministration*]

Beispiele

In den folgenden Beispielen werden Zeitstempelwerte in UNSIGNED BIGINT-Werte konvertiert.

```
SELECT sa_convert_timestamp_to_ml_progress( CURRENT_TIMESTAMP );  
  
SELECT sa_convert_timestamp_to_ml_progress( '2009-10-20 13:36:51.199' );
```

sa_copy_cursor_to_temp_table-Systemprozedur

Erstellt eine temporäre Tabelle und kopiert die Ergebnismenge eines geöffneten Cursors hinein.

Syntax

```
sa_copy_cursor_to_temp_table(  
  cursor_name  
  , table_name  
  [, first_row  
  [, max_rows ] ]  
)
```

Argumente

- **cursor_name** Verwenden Sie diesen VARCHAR(256)-Parameter, um den Namen des geöffneten Cursors anzugeben.
- **table_name** Verwenden Sie diesen VARCHAR(256)-Parameter, um den Namen der temporären Tabelle anzugeben.
- **first_row** Verwenden Sie diesen BIGINT-Parameter, um die Nummer der ersten in die temporäre Tabelle zu kopierenden Zeile anzugeben. Der Standardwert ist 1.
- **max_rows** Verwenden Sie diesen BIGINT-Parameter, um die maximale Anzahl der in die temporäre Tabelle zu kopierenden Zeilen anzugeben. Der Standardwert ist 9223372036854775807 (alle Zeilen).

Bemerkungen

Angenommen, Sie haben einen Cursor von mehreren Ganzzahlspalten. sa_copy_cursor_to_temp_table erstellt eine temporäre Tabelle mithilfe einer Anweisung in folgendem Format:

```
BEGIN  
  CREATE LOCAL TEMPORARY TABLE TempTab (  
    col1 INT,  
    col2 INT,  
    ...  
    rownum bigint primary key )  
END;
```

sa_copy_cursor_to_temp_table benennt die Spalten col1, col2 usw., um zu vermeiden, dass Namen dupliziert werden oder Schwierigkeiten auftreten, wenn Cursorspalten nicht über einen klar definierten Namen verfügen (z.B. im Falle eines komplexen Ausdrucks).

Wenn die temporäre Tabelle erstellt ist, wird der Inhalt des geöffneten Cursors eingefügt. Dazu wird die durch *first_row* angegebene Zeile gesucht und die durch *max_rows* angegebene Anzahl an Zeilen wird eingefügt. Nach dem Einfügen der Inhalte in die temporäre Tabelle wird der Cursor wieder an seine ursprüngliche Position gesetzt.

Privilegien

Keine

Nebenwirkungen

Beim Kopieren aus dem Cursor werden die Zeilen anhand der Isolationseinstellungen des Cursors abgerufen. Dadurch können Sperren für Zeilen gesetzt werden und es gibt weitere mögliche Auswirkungen, die dem Abrufen aus dem Cursor entsprechen.

Wenn gleichzeitig Änderungen außerhalb der aktuellen Verbindung vorgenommen werden und der Cursor nicht durch Materialisierungs- oder Isolationseinstellungen dagegen geschützt wird, ist es möglich, dass der Cursor nach Abschluss der Prozedur auf einer anderen Zeile positioniert wird. Wenn zum Beispiel die vorherige aktuelle Zeile des Cursors gelöscht wurde, kann der Cursor in die Zeile hinter der ursprünglichen Position gesetzt werden.

Wenn während des Kopierens aus dem Cursor ein Fehler auftritt, wechselt der Cursor in einen ungültigen Status und weitere Vorgänge im Cursor schlagen mit einer Fehlermeldung fehl.

Siehe auch

- „sa_list_cursors-Systemprozedur“ auf Seite 1248
- „sa_describe_cursor-Systemprozedur“ auf Seite 1203

Beispiel

Der folgende Batch erstellt einen Cursor namens myCursor und lädt Daten aus der Tabelle "Products" hinein. Anschließend wird der Cursor geöffnet (OPEN-Anweisung). Eine DROP-Anweisung löscht myTempTable, falls sie bereits vorhanden ist. Das Aufrufen von sa_copy_cursor_to_temp_table erstellt eine temporäre Tabelle namens myTempTable und kopiert die Inhalte von myCursor hinein. Schließlich gibt eine SELECT-Anweisung die Daten zurück, die aus dem Cursor in die temporäre Tabelle kopiert wurden:

```
BEGIN
  DECLARE myCursor CURSOR FOR
    SELECT ID, Name, Description, Color, Quantity FROM Products;
  OPEN myCursor;
  DROP TABLE IF EXISTS myTempTable;
  CALL sa_copy_cursor_to_temp_table( 'myCursor', 'myTempTable' );
  CLOSE myCursor;
  SELECT * FROM myTempTable;
END
```

sa_cpu_topology-Systemprozedur

Gibt die Prozessortopologie des Computers zurück, auf dem der Datenbankserver läuft.

Syntax

sa_cpu_topology()

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
os_id	UNSIGNED INTEGER	Gibt den logischen Prozessorbezeichner zurück, der vom zugrunde liegenden Betriebssystem verwendet wird. Unter Windows stimmt dieser Wert beispielsweise mit der Prozessor-ID überein, die im Windows-Task-Manager angezeigt wird.
socket	UNSIGNED INTEGER	Gibt einen Bezeichner für einen CPU-Socket oder ein Paket zurück.
core	UNSIGNED INTEGER	Gibt einen Bezeichner für einen Prozessorkern innerhalb eines bestimmten Sockets zurück.
thread	UNSIGNED INTEGER	Gibt einen Bezeichner für einen Thread innerhalb eines bestimmten Sockets und Prozessorkerns zurück.
apic	UNSIGNED INTEGER	Gibt einen Bezeichner für den logische Prozessor innerhalb des Systems zurück. Der Rückgabewert enthält die Kodierung von Socket, Prozessorkern und Thread.
group	UNSIGNED INTEGER	Gibt unter Windows die vom Betriebssystem zugeordnete Prozessgruppennummer zurück. Unter Solaris ist der Rückgabewert der Bezeichner für den Prozessorsatz. Andernfalls ist er gleich 0.
numa_node	UNSIGNED INTEGER	Gibt unter Windows den vom Betriebssystem zugeordneten NUMA-Knoten zurück.
online	BIT	Gibt dann und nur dann 1 zurück, wenn der Prozessor online ist.
in_use	BIT	Gibt 1 zurück, wenn der durch die aktuelle Zeile beschriebene logische Prozessor in der Prozessaffinitätsmaske für den SQL Anywhere-Datenbankserverprozess aktiviert ist, und andernfalls 0.

Bemerkungen

Auf allen 32-Bit- und 64-Bit-Plattformen mit Intel x86/x64 (außer in Mac-Versionen, in denen die Prozessoraaffinität nicht gesteuert werden kann) beschreiben die von dieser Prozedur zurückgegebenen Informationen genau die zugrunde liegende Hardware. Auf allen anderen Plattformen wird für jeden logischen Prozessor im System eine Zeile zurückgegeben, aber die Spaltenwerte werden folgendermaßen festgelegt: os_id, socket und apic sind gleich, core und thread und numa_node sind Null und group ist ebenfalls Null, außer unter Solaris, wo es mit dem pset-Bezeichner übereinstimmt. In einigen VM-Implementierungen sind die Ergebnisse möglicherweise nicht genau.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Datenbankserveroption -gta “ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sa_server_option-Systemprozedur“ auf Seite 1306

Beispiel

Das folgende Beispiel gibt eine Ergebnismenge mit der Prozessortopologie des Computers zurück, auf dem der Datenbankserver läuft.

```
CALL sa_cpu_topology( );
```

sa_db_info-Systemprozedur

Liefert Informationen über Datenbankeigenschaften

Syntax

```
sa_db_info( [ dbidparm ] )
```

Argumente

- **dbidparm** Verwenden Sie diesen optionalen INTEGER-Parameter, um die Datenbank-ID-Nummer anzugeben. Der Standardwert ist NULL.


Ergebnismenge

Spaltenname	Datentyp	Beschreibung
Number	INTEGER	Gibt die Verbindungs-ID (eine Nummer) für die aktuelle Verbindung zurück.
Alias	VARCHAR(255)	Gibt den Datenbanknamen zurück.
File	VARCHAR(255)	Gibt den Dateinamen der Datenbank-Stammdatei einschließlich des Pfads zurück.
ConnCount	INTEGER	Gibt die Anzahl der Verbindungen zur Datenbank zurück. Der Eigenschaftswert enthält keine Verbindungen, die zum Auslösen von internen Vorgängen verwendet werden. Verbindungen für Ereignisse und externe Umgebungen sind hingegen enthalten.
PageSize	INTEGER	Gibt die Seitengröße der Datenbank (in Byte) zurück
LogName	VARCHAR(255)	Gibt den Dateinamen des Transaktionslogs einschließlich des Pfads zurück.

Bemerkungen

Wenn Sie eine Datenbank-ID angeben, gibt `sa_db_info` eine einzelne Zeile zurück, die 'Number', 'Alias', 'File', 'ConnCount', 'PageSize' und 'LogName' für die angegebene Datenbank enthält.

Wenn `dbidparm` größer als Null ist, werden Eigenschaften für die angegebene Datenbank zurückgegeben. Wenn `dbidparm` kleiner als Null ist, werden Eigenschaften für die aktuelle Datenbank zurückgegeben. Wenn `dbidparm` nicht angegeben wird oder NULL ist, werden Eigenschaften für alle auf dem Datenbankserver laufenden Datenbanken zurückgegeben.

 **Cloud-Hinweis:** Aufgrund von Isolationsregeln für Tenant-Datenbanken werden bei der Ausführung dieser Systemprozedur in der Cloud nur Informationen über die aktuelle Tenant-Datenbank zurückgegeben.

Privilegien

Keine Privilegien sind erforderlich, um diese Systemprozedur für die aktuelle Datenbank auszuführen. Um diese Systemprozedur für andere Datenbanken ausführen zu können, benötigen Sie das SERVER OPERATOR-Systemprivileg oder das MONITOR-Systemprivileg.

Nebenwirkungen

Keine

Siehe auch

- „`sa_db_properties`-Systemprozedur“ auf Seite 1200
- „Liste der Datenbankeigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Die folgende Anweisung gibt eine Zeile für jede Datenbank zurück, die auf dem Server läuft:

```
CALL sa_db_info( );
```

Eigenschaft	Wert
Number	0
Alias	demo
File	<i>C:\Users\Public\Documents\SQL Anywhere 16\Samples\demo.db</i>
ConnCount	3
PageSize	4096
LogName	<i>C:\Users\Public\Documents\SQL Anywhere 16\Samples\demo.log</i>

sa_db_list-Systemprozedur

Gibt eine Datenbank-ID zurück

Syntax

sa_db_list([*dbidparm*])

Argumente

- **dbidparm** Verwenden Sie diesen optionalen INTEGER-Parameter, um die Datenbank-ID-Nummer anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
Number	INTEGER	Die Datenbank-ID-Nummer.

Bemerkungen

Wenn *dbidparm* größer als Null ist, wird die ID für die angegebene Datenbank zurückgegeben. Wenn *dbidparm* kleiner als Null ist, wird die ID für die aktuelle Datenbank zurückgegeben. Wenn *dbidparm* nicht angegeben wird oder NULL ist, werden IDs für alle auf dem Datenbankserver laufenden Datenbanken zurückgegeben.

🔥 **Cloud-Hinweis:** Aufgrund von Isolationsregeln für Tenant-Datenbanken werden bei der Ausführung dieser Systemprozedur in der Cloud nur Informationen über die aktuelle Tenant-Datenbank zurückgegeben.

Privilegien

Keine Privilegien sind erforderlich, um diese Systemprozedur für die aktuelle Datenbank auszuführen. Um diese Systemprozedur für andere Datenbanken ausführen zu können, benötigen Sie das SERVER OPERATOR-Systemprivileg oder das MONITOR-Systemprivileg.

Nebenwirkungen

Keine

Siehe auch

- „sa_conn_list-Systemprozedur“ auf Seite 1186
- „sa_conn_options-Systemprozedur“ auf Seite 1188
- „DB_NAME-Funktion [System]“ auf Seite 231

Beispiel

Das folgende Beispiel gibt die Anzahl der Datenbanken zurück, die auf einem Datenbankserver ausgeführt werden.

```
SELECT count(*) FROM sa_db_list();
```

Im folgenden Beispiel wird mithilfe der sa_db_list-Systemprozedur und der DB_Name-Funktion eine Liste der auf einem Datenbankserver laufenden Datenbanken zurückgegeben.

```
SELECT DB_NAME(Number) FROM sa_db_list();
```

sa_db_option-Systemprozedur

Hebt eine Datenbankoption auf, während die Datenbank läuft.

Syntax

```
sa_db_option(  
  opt  
  , val  
)
```

Argumente

- **opt** Verwenden Sie diesen CHAR(128)-Parameter, um einen Datenbankoptionsnamen anzugeben.
- **val** Verwenden Sie diesen CHAR(128)-Parameter, um den neuen Wert für die Datenbankoption anzugeben.

Bemerkungen

Datenbankadministratoren können mit dieser Prozedur einige Datenbankoptionen vorübergehend aufheben, ohne die Datenbank neu starten zu müssen.

Die mit dieser Prozedur geänderten Optionswerte werden auf ihre Standardwerte zurückgesetzt, wenn die Datenbank heruntergefahren wird. Wenn Sie einen Optionswert bei jedem Start der Datenbank ändern möchten, geben Sie beim Starten der Datenbank die entsprechende Datenbankoption (sofern vorhanden) an.

Die folgenden Optionseinstellungen können geändert werden:

Name der Option	Werte	Standardwert	Systemprivileg	Datenbankoption
DiskSand-box	ON, OFF	OFF	SERVER OPERA-TOR	„Datenbankoption -sbx “ [SQL Anywhere Server - Datenbank-administration]

- **DiskSandbox** Wenn diese Option auf ON gesetzt ist, werden Dateivorgänge der Datenbank mit Lese- und Schreibzugriff auf das Verzeichnis beschränkt, in dem sich die Hauptdatenbankdatei befindet, einschließlich der dazugehörigen Unterverzeichnisse. Wenn Sie die sa_db_option-Systemprozedur verwenden möchten, um Sandboxing-Einstellungen zu ändern, müssen Sie den Schlüssel für die gesicherte Funktion manage_disk_sandbox angeben.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Keine.

Siehe auch

- „Datenbankserveroption -sbx “ [*SQL Anywhere Server - Datenbankadministration*]
- „disk_sandbox-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „sa_server_option-Systemprozedur“ auf Seite 1306

Beispiel

Damit das folgende Beispiel funktioniert, muss der Server mit der Option `-sk securefkey` gestartet werden.

In diesem Beispiel wird die Gruppe gesicherter Funktionen namens `SYSTEM` aktiviert, die `MANAGE_KEYS` enthält. Sie erstellt einen neuen Schlüssel für gesicherte Funktionen namens `SECURITY` mit dem Autorisierungsschlüssel `newsecuritykey`, der Groß- und Kleinschreibung berücksichtigt, und verwendet anschließend den neuen Schlüssel für gesicherte Funktionen zum Aktivieren der `DiskSandbox`-Option.

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );
CALL sp_create_secure_feature_key( 'security', 'newsecuritykey',
'manage_security' );
CALL sp_use_secure_feature_key( 'security', 'newsecuritykey' );
CALL sa_db_option( 'DiskSandbox', 'on' );
```

sa_db_properties-Systemprozedur

Liefert Informationen über Datenbankeigenschaften

Syntax

```
sa_db_properties( [ dbidparm ] )
```

Argumente

- **dbidparm** Verwenden Sie diesen optionalen `INTEGER`-Parameter, um die Datenbank-ID-Nummer anzugeben. Der Standardwert ist `NULL`.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
Number	INTEGER	Die Datenbank-ID-Nummer.
PropNum	INTEGER	Die Nummer der Datenbankeigenschaft
PropName	VARCHAR(255)	Der Name der Datenbankeigenschaft
PropDescription	VARCHAR(255)	Die Beschreibung der Datenbankeigenschaft
Wert	LONG VARCHAR	Der Wert der Datenbankeigenschaft

Bemerkungen

Wenn Sie eine Datenbank-ID angeben, gibt die `sa_db_properties`-Systemprozedur die Datenbank-ID sowie die folgenden Werte für jede verfügbare Datenbankeigenschaft zurück: `PropNum`, `PropName`, `PropDescription` und `Value`. Werte werden für alle Datenbankeigenschaften und mit der Datenbank zusammenhängenden Statistiken zurückgegeben. Gültige Eigenschaften mit NULL-Werten werden ebenfalls zurückgegeben.

Wenn `dbidparm` größer als Null ist, werden Datenbankeigenschaften für die angegebene Datenbank zurückgegeben. Wenn `dbidparm` kleiner als Null ist, werden Datenbankeigenschaften für die aktuelle Datenbank zurückgegeben. Wenn `dbidparm` nicht angegeben wird oder NULL ist, werden Datenbankeigenschaften für alle auf dem Datenbankserver laufenden Datenbanken zurückgegeben.

Privilegien

Keine Privilegien sind erforderlich, um diese Systemprozedur für die aktuelle Datenbank auszuführen. Um diese Systemprozedur für andere Datenbanken ausführen zu können, benötigen Sie das `SERVER OPERATOR`-Systemprivileg oder das `MONITOR`-Systemprivileg.

Nebenwirkungen

Keine

Siehe auch

- „[sa_db_info-Systemprozedur](#)“ auf Seite 1196
- „[Liste der Datenbankeigenschaften](#)“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Im folgenden Beispiel wird die `sa_db_properties`-Systemprozedur verwendet, um eine Ergebnismenge zurückzugeben, die Datenbankeigenschaften für alle Datenbanken zusammenfasst, wenn der Aufrufer das `SERVER OPERATOR`-Systemprivileg oder das `MONITOR`-Systemprivileg hat. Andernfalls werden Datenbankeigenschaften für die aktuelle Datenbank zurückgegeben.

```
CALL sa_db_properties( );
```

Number	PropNum	PropName	...
0	0	ConnCount	...
0	1	IdleCheck	...
0	2	IdleWrite	...
...

Im folgenden Beispiel wird die `sa_db_properties`-Systemprozedur verwendet, um eine Ergebnismenge zurückzugeben, die Datenbankeigenschaften für eine zweite Datenbank zusammenfasst.

```
CALL sa_db_properties( 1 );
```

sa_dependent_views-Systemprozedur

Gibt die Liste aller abhängigen Ansichten einer angegebenen Tabelle oder Ansicht zurück.

Syntax

```
sa_dependent_views(  
  [ tbl_name  
    [, owner_name ] ]  
)
```

Argumente

- **tbl_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Namen der Tabelle oder Ansicht anzugeben. Der Standardwert ist NULL.
- **owner_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümer von *tbl_name* anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
table_id	UNSIGNED INTEGER	Die Objekt-ID der Tabelle oder Ansicht
dep_view_id	UNSIGNED INTEGER	Die Objekt-ID der abhängigen Ansichten

Bemerkungen

Verwenden Sie diese Prozedur, um die Liste der IDs von Tabellen und deren abhängigen Ansichten abzurufen.

Es werden keine Fehler generiert, wenn keine bestehenden Tabellen den angegebenen Kriterien für Tabellen- und Eigentümernamen entsprechen. Die folgenden Bedingungen gelten ebenfalls:

- Wenn sowohl *owner* als auch *tbl_name* NULL sind, werden Informationen zu allen Tabellen zurückgegeben, die abhängige Ansichten haben.
- Wenn *tbl_name* NULL ist, aber *owner* angegeben wird, werden Informationen zu allen Tabellen zurückgegeben, die dem angegebenen Eigentümer gehören.
- Wenn *tbl_name* angegeben wird, aber *owner* NULL ist, werden Informationen zu einer der Tabellen mit dem angegebenen Namen zurückgegeben.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „SYSDEPENDENCY-Systemansicht“ auf Seite 1445
- „Ansichtenabhängigkeiten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Beispiele

In diesem Beispiel wird die sa_dependent_views-Systemprozedur verwendet, um eine Liste von IDs für die Ansichten zu erhalten, die von der SalesOrders-Tabelle abhängen. Die Prozedur gibt die table_id für 'SalesOrders' und die dep_view_id für die abhängige Ansicht 'ViewSalesOrders' zurück.

```
CALL sa_dependent_views( 'SalesOrders' );
```

In diesem Beispiel wird die sa_dependent_views-Systemprozedur in einer SELECT-Anweisung verwendet, um eine Liste der Namen von Ansichten zu erhalten, die von der SalesOrders-Tabelle abhängen. Die Prozedur gibt die Ansicht 'ViewSalesOrders' zurück.

```
SELECT t.table_name FROM SYSTAB t,
sa_dependent_views( 'SalesOrders' ) v
WHERE t.table_id = v.dep_view_id;
```

sa_describe_cursor-Systemprozedur

Beschreibt die Namens- und Typinformationen für die Spalten eines Cursors.

Syntax

```
sa_describe_cursor( cursor_name )
```

Argumente

- **cursor_name** Dieser VARCHAR(256)-Wert kennzeichnet den zu beschreibenden geöffneten Cursor.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
column_number	INTEGER	Die Ordinalposition der von dieser Zeile beschriebenen Spalte, beginnend mit 1.
name	VAR-CHAR(128)	Der Name der Spalte.
domain_id	SMALLINT	Der Datentyp der Spalte.
domain_name	VAR-CHAR(128)	Der Datentypname der Spalte.
domain_name_with_size	VAR-CHAR(160)	Der Datentypname mit Größe und Gesamtstellenzahl (wie in CREATE TABLE- und CAST-Funktionen verwendet).

Spaltenname	Datentyp	Beschreibung
width	INTEGER	Die Länge eines Zeichenfolgenparameters, die Gesamtstellenzahl eines numerischen Parameters oder die Anzahl der Byte zum Speichern eines anderen Datentyps.
scale	INTEGER	Die Anzahl der Stellen nach dem Dezimalzeichen für Spalten mit numerischem Datentyp und Null für alle anderen Datentypen.
declared_width	INTEGER	Die Länge eines Zeichenfolgenparameters, die Gesamtstellenzahl eines numerischen Parameters oder die Anzahl der Byte zum Speichern eines anderen Datentyps.
user_type_id	SMALLINT	Der benutzerdefinierte Datentyp, falls vorhanden, sonst NULL.
user_type_name	VAR-CHAR(128)	Der benutzerdefinierte Datentyp, falls vorhanden, sonst NULL.
correlation_name	VAR-CHAR(128)	Der mit dem Ausdruck verbundene Korrelationsname, falls vorhanden, sonst NULL.
base_table_id	UNSIGNED INTEGER	Die table_id, wenn der Ausdruck eine Spalte ist, sonst NULL.
base_column_id	UNSIGNED INTEGER	Die column_id, wenn der Ausdruck eine Spalte ist, sonst NULL.
base_owner_name	VAR-CHAR(128)	Der Eigentümername, wenn der Ausdruck eine Spalte ist, sonst NULL.
base_table_name	VAR-CHAR(128)	Der Tabellename, wenn der Ausdruck eine Spalte ist, sonst NULL.
base_column_name	VAR-CHAR(128)	Der Spaltenname, wenn der Ausdruck eine Spalte ist, sonst NULL.
nulls_allowed	BIT	Der Indikator, ob der Ausdruck NULL sein kann (1).
is_autoincrement	BIT	Ein Indikator, ob es sich bei dem Ausdruck um eine AUTOINCREMENT-Spalte handelt (1).
is_key_column	BIT	Ein Indikator, ob der Ausdruck Teil eines Schlüssels für die Ergebnismenge ist (1). Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Spaltenname	Datentyp	Beschreibung
is_added_key_column	BIT	Ein Indikator, ob der Ausdruck eine hinzugefügte Spaltenspalte ist (1). Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Bemerkungen

Die `sa_describe_cursor`-Systemprozedur bietet einen API-unabhängigen Mechanismus zum Abfragen der Beschreibung für die vom Cursor zurückgegebenen Spalten. Die Systemprozedur kann beim Schreiben von gespeicherten Prozeduren nützlich sein, die mit Dynamic SQL arbeiten.

Die `sa_describe_cursor`-Systemprozedur kann in einer CALL-Anweisung oder in der FROM-Klausel einer SELECT-Anweisung verwendet werden.

cursor_name muss sich auf einen geöffneten Cursor in der aktuellen Verbindung beziehen. Verwenden Sie die `sa_list_cursors`-Systemprozedur, um die Liste der geöffneten Cursor für die Verbindung abzurufen.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „`sa_list_cursors`-Systemprozedur“ auf Seite 1248
- „`sa_copy_cursor_to_temp_table`-Systemprozedur“ auf Seite 1193
- „`SYSDOMAIN`-Systemansicht“ auf Seite 1446
- „`SYSUSER`-Systemansicht“ auf Seite 1506
- „`SYSTABCOL`-Systemansicht“ auf Seite 1495
- „`SYSUSERTYPE`-Systemansicht“ auf Seite 1510
- „`SYSDOMAIN`-Systemansicht“ auf Seite 1446

Beispiel

Der folgende Batch erstellt einen Cursor namens `myCursor` in der Tabelle "Products" und öffnet ihn anschließend. Die `sa_describe_cursor`-Systemprozedur wird verwendet, um die Spalten des Cursors zu beschreiben. Die erzeugte Ergebnismenge enthält 5 Zeilen, eine für jede Spalte.

```
BEGIN
  DECLARE myCursor CURSOR FOR
    SELECT ID, Name, Description, Color, Quantity FROM Products;
  OPEN myCursor;
  CALL sa_describe_cursor( 'myCursor' );
  CLOSE myCursor;
END
```

sa_describe_query-Systemprozedur

Beschreibt die Ergebnismenge für eine Abfrage, wobei eine Zeile die jeweilige Ausgabespalte der Abfrage beschreibt

Syntax

```
sa_describe_query(  
  query  
  [, add_keys ]  
)
```

Argumente

- **query** Verwenden Sie diesen LONG VARCHAR-Parameter, um den Text der zu beschreibenden SQL-Anweisung anzugeben.
- **add_keys** Mit diesem optionalen BIT-Parameter können Sie angeben, ob ein Satz von Spalten festgelegt wird, der Zeilen in der Ergebnismenge für die zu beschreibende Abfrage eindeutig identifiziert. Der Standardwert ist "0". Beim Standardwert versucht der Datenbankserver nicht, die Spalten zu identifizieren. Im Abschnitt "Bemerkungen" weiter unten finden Sie eine ausführliche Beschreibung dieses Parameters.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
column_number	INTEGER	Die Ordinalposition der von dieser Zeile beschriebenen Spalte, beginnend mit 1.
name	VAR-CHAR(128)	Der Name der Spalte.
domain_id	SMALLINT	Der Datentyp der Spalte.
domain_name	VAR-CHAR(128)	Der Datentypname.
domain_name_with_size	VAR-CHAR(160)	Der Datentypname mit Größe und Gesamtstellenzahl (wie in CREATE TABLE- und CAST-Funktionen verwendet).
width	INTEGER	Die Länge eines Zeichenfolgenparameters, die Gesamtstellenzahl eines numerischen Parameters oder die Anzahl der Byte zum Speichern eines anderen Datentyps.
scale	INTEGER	Die Anzahl der Stellen nach dem Dezimalzeichen für Spalten mit numerischem Datentyp und Null für alle anderen Datentypen.

Spaltenname	Datentyp	Beschreibung
declared_width	INTEGER	Die Länge eines Zeichenfolgenparameters, die Gesamtstellenzahl eines numerischen Parameters oder die Anzahl der Byte zum Speichern eines anderen Datentyps.
user_type_id	SMALLINT	Die type_id eines benutzerdefinierten Datentyps, falls vorhanden, ansonsten NULL.
user_type_name	VAR-CHAR(128)	Der Name eines benutzerdefinierten Datentyps, falls vorhanden, sonst NULL.
correlation_name	VAR-CHAR(128)	Der mit dem Ausdruck verbundene Korrelationsname, falls vorhanden, sonst NULL.
base_table_id	UNSIGNED INTEGER	Die table_id, wenn der Ausdruck eine Spalte ist, sonst NULL.
base_column_id	UNSIGNED INTEGER	Die column_id, wenn der Ausdruck eine Spalte ist, sonst NULL.
base_owner_name	VAR-CHAR(128)	Der Eigentümername, wenn der Ausdruck eine Spalte ist, sonst NULL.
base_table_name	VAR-CHAR(128)	Der Tabellename, wenn der Ausdruck eine Spalte ist, sonst NULL.
base_column_name	VAR-CHAR(128)	Der Spaltenname, wenn der Ausdruck eine Spalte ist, sonst NULL.
nulls_allowed	BIT	Ein Indikator mit Wert "1", wenn der Ausdruck NULL sein kann, sonst "0"
is_autoincrement	BIT	Ein Indikator mit dem Wert "1", wenn der Ausdruck eine Autoincrement-Spalte ist, sonst "0"
is_key_column	BIT	Ein Indikator mit dem Wert "1", wenn der Ausdruck Teil eines Schlüssels für die Ergebnismenge ist, sonst "0". Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".
is_added_key_column	BIT	Ein Indikator mit dem Wert "1", wenn der Ausdruck eine zusätzliche Schlüsselspalte ist, sonst "0". Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Bemerkungen

Die `sa_describe_query`-Prozedur stellt einen API-unabhängigen Mechanismus zur Beschreibung der Namens- und Typinformationen für die Ausdrücke in der Ergebnismenge einer Abfrage zur Verfügung.

Wenn für `add_keys` "1" angegeben ist, versucht die `sa_describe_query`-Prozedur, einen Satz von Spalten unter den abgefragten Objekten zu finden, die zusammengefasst als ein Schlüssel für die eindeutige Identifizierung von Zeilen in der Ergebnismenge der zu beschreibenden Abfrage verwendet werden können. Der Schlüssel nimmt die Form einer oder mehrerer Spalten der abgefragten Objekte an und kann auch Spalten umfassen, die von der Abfrage nicht explizit referenziert werden. Wenn der Optimierer einen Schlüssel findet, werden die im Schlüssel verwendeten Spalten in den Ergebnissen mit einem `is_key_column`-Wert von "1" identifiziert. Wenn kein Schlüssel gefunden wird, wird ein Fehler zurückgegeben.

Bei jeder Spalte, die im Schlüssel enthalten ist, aber in der Abfrage nicht explizit referenziert wird, wird der `is_added_key_column`-Wert auf "1" gesetzt, um anzudeuten, dass die Spalte den Ergebnissen der Prozedur hinzugefügt wurde. Sonst ist der Wert von `is_added_key_column` "0".

Wenn Sie keinen `add_keys` oder einen Wert von "0" angeben, versucht der Optimierer nicht, einen Schlüssel für die Ergebnismenge zu finden, und die Spalten `is_key_column` und `is_added_key_column` enthalten NULL.

Die Werte `declared_width` und `width` beschreiben beide die Größe einer Spalte. Der `declared_width`-Wert beschreibt die Größe der Spalte, wie sie von der CREATE TABLE-Anweisung oder der Abfrage definiert wird, während der `width`-Wert die Größe des Felds angibt, wenn es vom Client abgerufen wird. Die Darstellung eines Typs auf dem Client kann sich von der des Datenbankservers unterscheiden. Datums- und Zeittypen beispielsweise werden in Zeichenfolgen konvertiert, wenn die `return_date_time_as_string`-Option auf ON gesetzt ist. Bei Zeichenfolgen haben mit Zeichenlängensemantik deklarierte Felder einen `declared_width`-Wert, der der CREATE TABLE-Größe entspricht, während der `width`-Wert die maximale Anzahl von Byte angibt, die zum Speichern der zurückgegebenen Zeichenfolge erforderlich sind.
Beispiel:

Deklaration	width	declared_width
CHAR(10)	10	10
CHAR(10 CHAR)	40	10
TIMESTAMP	Hängt von der Länge der Zeitstempel-Zeichenfolge ab	8
NUMERIC(10, 3)	10 (Gesamtstellenzahl)	10 (Gesamtstellenzahl)

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „EXPRTYPE-Funktion [Verschiedene]“ auf Seite 262
- „Zeichendatentypen“ auf Seite 95
- „return_date_time_as_string-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „SYSDOMAIN-Systemansicht“ auf Seite 1446
- „SYSDOMAIN-Systemansicht“ auf Seite 1446
- „SYSUSERTYPE-Systemansicht“ auf Seite 1510
- „SYSTAB-Systemansicht“ auf Seite 1492
- „SYSUSER-Systemansicht“ auf Seite 1506

Beispiele

Das folgende Beispiel beschreibt die zurückgegebenen Informationen, wenn alle Spalten in der Departments-Tabelle abgefragt werden:

```
SELECT *  
FROM sa_describe_query( 'SELECT * FROM Departments DEPT' );
```

Die Ergebnisse zeigen die Werte der Spalten is_key_column und is_added_key_column als NULL, weil der Parameter *add_keys* nicht angegeben wurde.

Das folgende Beispiel beschreibt die Informationen, die durch Abfragen der DepartmentName- und Surname-Spalten der Employees-Tabelle, die durch einen Join mit der Departments-Tabelle verknüpft ist, zurückgegeben werden.

```
SELECT *  
FROM sa_describe_query( 'SELECT DepartmentName, Surname  
FROM Employees E JOIN Departments D ON E.EmployeeID = D.DepartmentHeadId',  
add_keys = 1 );
```

Die Ergebnisse enthalten eine 1 in den Zeilen 3 und 4 der Ergebnismenge, was darauf hinweist, dass die Spalten, die zum eindeutigen Identifizieren von Zeilen in der Ergebnismenge benötigt werden, Employees.EmployeeID und Departments.DepartmentID sind. Auch ist 1 der Wert von is_added_key_column für die Zeilen 3 und 4, weil Employees.EmployeeID und Departments.DepartmentID nicht explizit in der zu beschreibenden Abfrage referenziert wurden.

sa_describe_shapefile-Systemprozedur

Beschreibt die Namen und Typen der Spalten in einer ESRI-Formdatei. Diese Systemfunktion wird zusammen mit den Funktionen für räumliche Daten verwendet.

Syntax

```
sa_describe_shapefile(  
  shp_filename  
  , srid  
  [, encoding ]  
)
```

Argumente

- **shp_filename** Ein VARCHAR(512)-Parameter, der den Speicherort der ESRI-Formdatei identifiziert. Der Dateiname muss die Erweiterung *.shp* haben und ihm muss eine *.dbf*-Datei

zugeordnet sein, die denselben Basisnamen hat und sich in demselben Verzeichnis befindet. Der Pfad ist relativ zum Datenbankserver, nicht zur Clientanwendung.

- **srid** Ein INTEGER-Parameter, der die SRID für die Geometrien in der Formdatei identifiziert. Geben Sie NULL an, um anzuzeigen, dass in der Spalte mehrere SRIDs gespeichert werden können. Wenn Sie NULL angeben, beschränkt dies allerdings die Vorgänge, die mit den Geometriewerten durchgeführt werden können.
- **encoding** Ein optionaler VARCHAR(50)-Parameter, der die beim Lesen der Formdatei zu verwendende Kodierung identifiziert. Der Standardwert ist NULL. Wenn die Kodierung NULL ist, wird der ISO-8859-1-Zeichensatz verwendet.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
column_number	INTEGER	Die Ordinalposition der von dieser Zeile beschriebenen Spalte, beginnend mit 1.
name	VARCHAR(128)	Der Name der Spalte.
domain_name_with_size	VARCHAR(160)	Der Datentypname mit Größe und Gesamtstellenzahl (wie in CREATE TABLE- und CAST-Funktionen verwendet).

Bemerkungen

Die sa_describe_shapefile-Systemprozedur wird verwendet, um Namen und Typ der Spalten in einer ESRI-Formdatei zu beschreiben. Diese Informationen können verwendet werden, um eine Tabelle zu erstellen, anhand derer Daten aus einer Formdatei mithilfe der LOAD TABLE- oder INPUT-Anweisung geladen werden. Alternativ kann diese Systemprozedur verwendet werden, um eine Formdatei durch Angeben der WITH-Klausel für OPENSTRING ... FORMAT SHAPEFILE zu lesen.

Privilegien

- Wenn die Datenbankoption -gl auf DBA gesetzt ist, müssen Sie eines der folgenden Systemprivilegien haben:
 - ALTER ANY TABLE
 - ALTER ANY TABLE
 - LOAD ANY TABLE
 - READ FILE
- Wenn die Datenbankoption -gl auf ALL gesetzt ist, sind keine Privilegien erforderlich.
- Wenn Datenbankoption -gl auf NONE gesetzt ist, müssen Sie das READ FILE-Systemprivileg haben.

Siehe auch

- „LOAD TABLE-Anweisung“ auf Seite 931
- „INPUT-Anweisung [Interactive SQL]“ auf Seite 910
- „Unterstützung von ESRI-Formdateien“ [[SQL Anywhere Server - Unterstützung für räumliche Daten](#)]

Beispiel

Im folgenden Beispiel wird eine Zeichenfolge angezeigt, die verwendet werden kann, um eine Tabelle zum Speichern von Formdatei-Daten zu erstellen:

```
BEGIN
  DECLARE create_cmd LONG VARCHAR;
  SELECT 'create table if not exists esri_load( record_number int primary
key, ' ||
        (SELECT list( name || ' ' || domain_name_with_size, ', ' ORDER BY
column_number )
FROM sa_describe_shapefile( 'c:\\esri\\tgr36069trt00.shp', 1000004326 )
WHERE column_number > 1 ) || ' )'
INTO create_cmd;
SELECT create_cmd;
EXECUTE IMMEDIATE create_cmd;
END
```

Sie können die Formdatei-Daten in die Tabelle laden, indem Sie die folgende Anweisung ausführen (vorausgesetzt, Sie haben das LOAD ANY TABLE-Systemprivileg und die Datenbankoption -gl wurde nicht auf NONE gesetzt):

```
LOAD TABLE esri_load
USING FILE 'c:\\esri\\tgr36069trt00.shp'
FORMAT SHAPEFILE;
```

sa_disable_auditing_type-Systemprozedur

Deaktiviert das Auditing bestimmter Ereignisse

Syntax

sa_disable_auditing_type(*types*)

Argumente

- **types** Verwenden Sie diesen VARCHAR(128)-Parameter, um eine kommasetrennte Zeichenfolge anzugeben, die einen oder mehrere der folgenden Werte enthält:
 - **all** Deaktiviert alle Auditing-Typen.
 - **connect** Deaktiviert das Auditing von erfolgreichen und fehlgeschlagenen Verbindungsversuchen.
 - **connectFailed** Deaktiviert das Auditing von fehlgeschlagenen Verbindungsversuchen.
 - **DDL** Deaktiviert das Auditing von DDL-Anweisungen.
 - **options** Deaktiviert das Auditing von öffentlichen Optionen.

- **permission** Deaktiviert das Auditing von Berechtigungsprüfungen, Benutzerprüfungen und SETUSER-Anweisungen.
- **permissionDenied** Deaktiviert das Auditing von fehlgeschlagenen Berechtigungs- und Benutzerprüfungen.
- **triggers** Deaktiviert das Auditing als Antwort auf Trigger-Ereignisse.

Bemerkungen

Mit der `sa_disable_auditing_type`-Systemprozedur können Sie das Auditing für eine oder mehrere Informationskategorien deaktivieren.

Wenn Sie diese Option auf 'all' setzen, wird das gesamte Auditing deaktiviert. Sie können das Auditing auch deaktivieren, indem Sie die Option `PUBLIC.auditing` auf OFF setzen.

Privilegien

Sie müssen das SET ANY SECURITY OPTION-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „`sa_enable_auditing_type`-Systemprozedur“ auf Seite 1214
- „Audits von Datenbankaktivitäten“ [*SQL Anywhere Server - Datenbankadministration*]
- „auditing-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Um das gesamte Auditing zu deaktivieren:

```
CALL sa_disable_auditing_type( 'all' );
```

sa_disk_free_space-Systemprozedur

Meldet Informationen zum verfügbaren Speicherplatz für einen DBSpace, ein Transaktionslog, einen Transaktionslog-Spiegel bzw. eine temporäre Datei.

Syntax

```
sa_disk_free_space( [ p_dbspace_name ] )
```

Argumente

- **p_dbspace_name** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Namen eines DBSpace, einer Transaktionslogdatei, einer Transaktionslog-Spiegeldatei oder einer temporären Datei anzugeben. Der Standardwert ist NULL.

Wenn ein DBSpace mit der Bezeichnung "log", "mirror" oder "temp" vorhanden ist, können Sie dem Schlüsselwort einen Unterstrich voranstellen. Für einen DBSpace namens log können Sie beispielsweise "_log" verwenden, um Informationen zur Logdatei abzurufen.

Geben Sie SYSTEM an, um Informationen über die Hauptdatenbankdatei abzurufen, TEMPORARY oder TEMP, um Informationen über die temporäre Datei abzurufen, TRANSLOG, um Informationen über das Transaktionslog abzurufen, oder TRANSLOGMIRROR, um Informationen über den Transaktionslogspiegel abzurufen.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
dbspace_name	VARCHAR(128)	Dies ist der DBSpace-Name, die Transaktionslogdatei, die Transaktionslog-Spiegeldatei oder die temporäre Datei.
free_space	UNSIGNED BIGINT	Die Anzahl von freien Byte auf dem Datenträger
total_space	UNSIGNED BIGINT	Der gesamte verfügbare Speicherplatz auf dem Laufwerk, auf dem sich der DBSpace befindet.

Bemerkungen

Wenn der *p_dbspace_name*-Parameter nicht angegeben wird oder NULL ist, enthält die Ergebnismenge eine Zeile für jeden DBSpace sowie jeweils eine Zeile für das Transaktionslog, die Transaktionslog-Spiegeldatei und die temporäre Datei, falls vorhanden. Wenn *p_dbspace_name* angegeben wird, dann wird genau eine oder keine Zeile ausgegeben (keine Zeile dann, wenn kein solcher DBSpace vorhanden ist beziehungsweise wenn "log" oder "mirror" angegeben wurde und keine Log- oder Spiegeldatei vorhanden ist).

Privilegien

Sie müssen das MANAGE ANY DBSPACE-Systemprivileg haben.

Nebenwirkungen

Keine

Beispiel

Im folgenden Beispiel wird die sa_disk_free_space-Systemprozedur verwendet, um eine Ergebnismenge zurückzugeben, die Informationen zum verfügbaren Speicherplatz enthält.

```
CALL sa_disk_free_space( );
```

dbspace_name	free_space	total_space
system	10952101888	21410402304
translog	10952101888	21410402304
temporary	10952101888	21410402304

Siehe auch

- „Vordefinierte DBSpaces“ [[SQL Anywhere Server - Datenbankadministration](#)]

sa_enable_auditing_type-Systemprozedur

Aktiviert das Auditing und gibt an, welche Ereignisse einem Auditing unterzogen werden sollen.

Syntax

sa_enable_auditing_type(*types*)

Argumente

- **types** Verwenden Sie diesen VARCHAR(128)-Parameter, um eine kommasetrennte Zeichenfolge anzugeben, die einen oder mehrere der folgenden Werte enthält:
 - **all** Aktiviert alle Auditing-Typen.
 - **connect** Aktiviert das Auditing von erfolgreichen und fehlgeschlagenen Verbindungsversuchen.
 - **connectFailed** Aktiviert das Auditing von fehlgeschlagenen Verbindungsversuchen.
 - **DDL** Aktiviert das Auditing von DDL-Anweisungen.
 - **options** Aktiviert das Auditing von öffentlichen Optionen.
 - **permission** Aktiviert das Auditing von Berechtigungsprüfungen, Benutzerprüfungen und SETUSER-Anweisungen.
 - **permissionDenied** Aktiviert das Auditing von fehlgeschlagenen Berechtigungs- und Benutzerprüfungen.
 - **triggers** Aktiviert das Auditing nach einem Trigger-Ereignis.

Bemerkungen

sa_enable_auditing_type aktiviert zusammen mit der PUBLIC.auditing-Option das Auditing bestimmter Datentypen.

Wenn Sie die PUBLIC.auditing-Option auf ON setzen und nicht angeben, welche Datentypen im Audit erfasst werden sollen, wird die Standardeinstellung (all) wirksam. In diesem Fall werden alle Arten von Audit-Daten erfasst.

Wenn Sie die PUBLIC.auditing-Option auf ON setzen und mit sa_disable_auditing_type alle Auditing-Typen deaktivieren, werden keine Audit-Daten erfasst. Wenn das Auditing wieder aktiviert werden soll, müssen Sie sa_enable_auditing_type verwenden und angeben, welche Arten von Daten einem Auditing unterzogen werden sollen.

Wenn Sie die PUBLIC.auditing-Option auf OFF setzen, werden keine Audit-Daten erfasst, unabhängig von der Einstellung für sa_enable_auditing_type.

Privilegien

Sie müssen das SET ANY SECURITY OPTION-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „sa_disable_auditing_type-Systemprozedur“ auf Seite 1211
- „Audits von Datenbankaktivitäten“ [*SQL Anywhere Server - Datenbankadministration*]
- „auditing-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

So wird nur das Auditing von öffentlichen Optionen (options) aktiviert:

```
CALL sa_enable_auditing_type( 'options' );
```

sa_eng_properties-Systemprozedur

Liefert Informationen über Eigenschaften des Datenbankservers

Syntax

```
sa_eng_properties( )
```

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
PropNum	INTEGER	Die Nummer der Datenbankserver-Eigenschaft
PropName	VARCHAR(255)	Der Name der Datenbankserver-Eigenschaft
PropDescription	VARCHAR(255)	Die Beschreibung der Datenbankserver-Eigenschaft
Value	LONG VARCHAR	Der Wert der Datenbankserver-Eigenschaft

Bemerkungen

Gibt PropNum, PropName, PropDescription und Value für jede verfügbare Servereigenschaft zurück. Werte werden für alle Datenbankserver-Eigenschaften und mit dem Datenbankserver zusammenhängenden Statistiken zurückgegeben.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Liste der Datenbankservereigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Systemfunktionen“ auf Seite 165

Beispiel

Die folgende Anweisung gibt eine Reihe von verfügbaren Servereigenschaften zurück:

```
CALL sa_eng_properties( );
```

PropNum	PropName	...
1	IdleWrite	...
2	IdleChkPt	...
...

sa_error_stack_trace-Systemprozedur

Gibt das Stack-Trace für den Fehler zurück, durch den die Fehlerbehandlungsroutine aufgerufen wurde.

Syntax

```
sa_error_stack_trace( )
```

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
StackLevel	UNSIGNED SMALLINT	Die Zeilennummer des Stacks (1 für die oberste Zeile). Die Anweisung, die den Fehler generiert hat, erhält die höchste Nummer.
UserName	CHAR(128)	Der Name des Eigentümers der Prozedur oder NULL, wenn sich die aktuelle Stufe in einem Batch befindet.
ProcName	CHAR(128)	Der Name der Prozedur, in der die Anweisung ausgeführt wurde, oder der Batch-Typ.
LineNumber	UNSIGNED INTEGER	Die Zeilennummer des Aufrufs innerhalb der Prozedur.
IsResignal	BIT	1, wenn es sich um eine RESIGNAL-Anweisung handelt, und andernfalls 0.

Bemerkungen

Jede Zeile in der Ergebnismenge stellt einen einzelnen Aufruf im Aufruf-Stack des Fehlers dar. Wenn die zusammengesetzte Anweisung nicht Teil einer Prozedur, einer Funktion, eines Triggers oder eines

Ereignisses ist, wird statt des Prozedurnamens der Batch-Typ (watcom_batch oder tsq_batch) zurückgegeben.

Diese Prozedur gibt dieselben Informationen zurück wie die ERROR_STACK_TRACE-Funktion.

Privilegien

Keine

Nebenwirkungen

Keine.

Siehe auch

- „TRY-Anweisung“ auf Seite 1093
- „BEGIN-Anweisung“ auf Seite 557
- „ERROR_LINE-Funktion [Verschiedene]“ auf Seite 244
- „ERROR_MESSAGE-Funktion [Verschiedene]“ auf Seite 245
- „ERROR_PROCEDURE-Funktion [Funktionstyp]“ auf Seite 246
- „ERROR_SQLCODE-Funktion [Verschiedene]“ auf Seite 247
- „ERROR_SQLSTATE-Funktion [Verschiedene]“ auf Seite 248
- „ERROR_STACK_TRACE-Funktion [Verschiedene]“ auf Seite 249
- „STACK_TRACE-Funktion [Verschiedene]“ auf Seite 392
- „sa_stack_trace-Systemprozedur“ auf Seite 1335
- „Verschachtelte zusammengesetzte Anweisungen und Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Beispiel

Dieses Beispiel zeigt das Aufrufen der sa_error_stack_trace-Systemprozedur:

```
CALL sa_error_stack_trace();
```

Dieses Beispiel zeigt die Ausgabe der sa_error_stack_trace-Systemprozedur mit RESIGNAL:

```
CREATE OR REPLACE PROCEDURE error_reporting_procedure()
BEGIN
    SELECT *
    FROM sa_error_stack_trace();
END;

CREATE OR REPLACE PROCEDURE proc1()
BEGIN TRY
    BEGIN TRY
        DECLARE v INTEGER = 0;
        SET v = 1 / v;
    END TRY
    BEGIN CATCH
        CALL proc2();
    END CATCH
END TRY
BEGIN CATCH
    CALL error_reporting_procedure();
END CATCH;

CREATE OR REPLACE PROCEDURE proc2()
```

```
BEGIN
    CALL proc3();
END;

CREATE OR REPLACE PROCEDURE proc3()
BEGIN
    RESIGNAL;
END;

CALL proc1();
```

Beim Ausführen des obigen Beispiels wird die folgende Ergebnismenge erzeugt:

StackLevel	UserName	ProcName	LineNumber	IsResignal
1	DBA	proc1	8	0
2	DBA	proc2	3	0
3	DBA	proc3	3	1
4	DBA	proc1	5	0

sa_event_schedules-Systemprozedur

Zeigt Zeitplaninformationen zu Ereignissen an.

Syntax

sa_event_schedules(*evt_id*)

Argumente

- **evt_id** Dieser INTEGER-Parameter nimmt die ID-Nummer eines Ereignisses auf.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
sched_name	VARCHAR(128)	Der Name des Zeitplans, nach dem das Ereignis ausgeführt wird.
sched_def	LONG VARCHAR	Die Zeitplandefinition.

Bemerkungen

Diese Prozedur gibt Informationen über ein angegebenes Ereignis zurück, einschließlich der Angaben, wann das Ereignis laut Zeitplan als Nächstes ausgeführt werden soll und wie oft es ausgeführt wird. Wenn das Ereignis keinen Zeitplan aufweist, gibt die Prozedur keine Ergebnismenge zurück.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „CREATE EVENT-Anweisung“ auf Seite 606
- „EVENT_CONDITION-Funktion [System]“ auf Seite 254
- „EVENT_CONDITION_NAME-Funktion [System]“ auf Seite 256
- „EVENT_PARAMETER-Funktion [System]“ auf Seite 256

Beispiel

Im folgenden Beispiel wird der Ereigniszeitplan für das IncrementalBackup-Ereignis abgerufen.

```
call sa_event_schedules( (SELECT event_id FROM SYSEVENT WHERE  
event_name='IncrementalBackup') );
```

sched_name	sched_def
IncrementalBackup	START TIME '01:00:00' EVERY 24 HOURS

sa_external_library_unload-Systemprozedur

Entlädt eine externe Bibliothek.

Syntax

```
sa_external_library_unload( [ lib_name ] )
```

Argumente

- **lib_name** Verwenden Sie diesen optionalen LONG VARCHAR-Parameter, um den Namen einer zu entladenden Bibliothek anzugeben. Wenn keine Bibliothek angegeben ist, werden alle nicht benutzten Bibliotheken entladen. Der Standardwert ist NULL.

Bemerkungen

Wenn eine externe Bibliothek angegeben ist, die aber benutzt oder nicht geladen ist, wird ein Fehler zurückgegeben. Wenn kein Parameter angegeben wurde, wird keine Fehlermeldung zurückgegeben.

Der Bibliotheksname muss genau mit dem Pfad sowie der Groß- und Kleinschreibung der ursprünglichen Bibliotheksspezifikation übereinstimmen. Der Bibliotheksname muss beispielsweise genau mit der Zeichenfolge übereinstimmen, die in der folgenden EXTERNAL NAME-Klausel nach dem "@" angegeben wird.

```
EXTERNAL NAME 'xp_replicate@c:\\sqlany\\samples\\sqlanywhere\\  
\\ExternalProcedures\\extproc.dll'
```

Privilegien

Sie müssen das MANAGE ANY EXTERNAL OBJECT-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „SQL Anywhere-Schnittstelle für externe Aufrufe“ [[SQL Anywhere Server - Programmierung](#)]

Beispiel

Im folgenden Beispiel wird eine externe Bibliothek namens *extproc.dll* entladen:

```
CALL sa_external_library_unload( 'extproc.dll' );
```

Im folgenden Beispiel werden alle Bibliotheken entladen, die gerade nicht benutzt werden:

```
CALL sa_external_library_unload();
```

sa_flush_cache-Systemprozedur

Leert alle Seiten für die laufende Datenbank im Datenbankserver-Cache

Syntax

```
sa_flush_cache( )
```

Bemerkungen

Datenbankadministratoren können mit dieser Prozedur den Inhalt des Datenbankserver-Caches für die laufende Datenbank leeren. Dies ist bei Performance-Messungen hilfreich, um wiederholbare Ergebnisse zu garantieren.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Keine

Beispiel

Im folgenden Beispiel werden alle Seiten für die laufende Datenbank im Datenbankserver-Cache geleert.

```
CALL sa_flush_cache( );
```

sa_flush_statistics-Systemprozedur

Speichert alle Kostenmodellstatistiken im Datenbankserver-Cache

Syntax

```
sa_flush_statistics( )
```

Bemerkungen

Verwenden Sie diese Prozedur, um aktuelle Kostenmodellstatistiken in der Datenbank, die derzeit im Cache gespeichert sind, auf der Festplatte zu speichern. Sie können diese Statistiken mit der `sa_get_histogram`-Systemprozedur oder mit dem Histogramm-Dienstprogramm (`dbhist`) abrufen. Wenn diese Prozedur ausgeführt wird, wird die Systemtabelle `ISYSCOLSTAT` aktualisiert. Unter normalen Umständen sollte diese Prozedur jedoch nicht nötig sein, da der Server automatisch die Statistiken in periodischen Abständen auf die Festplatte schreibt.

Privilegien

Sie müssen das `MANAGE ANY STATISTICS`-Systemprivileg oder das `SERVER OPERATOR`-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „`sa_get_histogram`-Systemprozedur“ auf Seite 1225
- „`SYSCOLSTAT`-Systemansicht“ auf Seite 1441
- „Histogramm-Dienstprogramm (`dbhist`)“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Im folgenden Beispiel werden alle Kostenmodellstatistiken im Datenbankserver-Cache gespeichert.

```
CALL sa_flush_statistics( );
```

sa_get_bits-Systemprozedur

Nimmt eine Bitzeichenfolge und gibt eine Zeile für jedes Bit in der Zeichenfolge zurück. Standardmäßig werden nur Zeilen mit einem Bitwert von "1" zurückgegeben.

Syntax

```
sa_get_bits(  
  bit_string  
  [, only_on_bits ]  
)
```

Argumente

- **bit_string** Verwenden Sie diesen LONG VARBIT-Parameter, um die Bitzeichenfolge anzugeben, aus der die Bits abgerufen werden sollen. Wenn der `bit_string`-Parameter NULL ist, werden keine Zeilen zurückgegeben.
- **only_on_bits** Verwenden Sie diesen optionalen BIT-Parameter, um anzugeben, ob nur Zeilen mit On-Bits (Bits mit dem Wert 1) zurückgegeben werden sollen. Geben Sie "1" (Standardwert) an, um nur Zeilen mit On-Bits zurückzugeben, und "0", um Zeilen für alle Bits in der Bitzeichenfolge zurückzugeben.

Ergebnismenge

Spalte	Datentyp	Beschreibung
bitnum	UNSIGNED INTEGER	Die Position des von dieser Zeile beschriebenen Bits. Das erste Bit in der Bitzeichenfolge z.B. hat bitnum 1.
bit_val	BIT	Der Wert des Bits an der Position bitnum. Wenn <i>only_on_bits</i> auf "1" gesetzt ist, ist dieser Wert immer "1".

Bemerkungen

Die `sa_get_bits`-Systemprozedur dekodiert eine Bitzeichenfolge und gibt eine Zeile für jedes Bit in der Bitzeichenfolge zurück, die den Wert des Bits angibt. Wenn *only_on_bits* auf "1" (Standardwert) gesetzt ist, werden nur Zeilen zurückgegeben, die On-Bits entsprechen. Eine Optimierung ermöglicht in diesem Fall eine effiziente Verarbeitung bei langen Bitzeichenfolgen mit wenigen On-Bits. Wenn *only_on_bits* auf "0" gesetzt ist, wird für jedes Bit in der Bitzeichenfolge eine Zeile zurückgegeben.

Beispiel: Die Anweisung `CALL sa_get_bits('1010')` gibt die folgende Ergebnismenge zurück, die On-Bits auf Position 1 und 3 der Bitzeichenfolge anzeigt.

bitnum	bit_val
1	1
3	1

Mit der `sa_get_bits`-Systemprozedur können Sie eine Bitzeichenfolge in eine Relation konvertieren. Diese kann verwendet werden, um eine Bitzeichenfolge mit einer Tabelle zu verknüpfen oder um eine Bitzeichenfolge als Ergebnismenge und nicht als einen einzelnen Binärwert abzurufen. Es kann effizienter sein, eine Bitzeichenfolge als Ergebnismenge abzurufen, wenn es eine große Anzahl von 0-Bits gibt, da diese nicht abgerufen werden müssen.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „[sa_split_list-Systemprozedur](#)“ auf Seite 1332
- „[SET_BIT-Funktion \[Bit-Array\]](#)“ auf Seite 379
- „[SET_BITS-Funktion \[Aggregat\]](#)“ auf Seite 381
- „[GET_BIT-Funktion \[Bit-Array\]](#)“ auf Seite 266

Beispiele

Das folgende Beispiel zeigt, wie Sie die `sa_get_bits`-Systemprozedur verwenden, um eine Reihe von Ganzzahlen als Bitzeichenfolge zu kodieren und sie anschließend für die Verwendung in einem Join zu dekodieren:

```
CREATE VARIABLE @s_depts LONG VARBIT;

SELECT SET_BITS( DepartmentID )
INTO @s_depts
FROM Departments
WHERE DepartmentName like 'S%';

SELECT *
FROM sa_get_bits( @s_depts ) B
JOIN Departments D ON B.bitnum = D.DepartmentID;
```

sa_get_dtt-Systemprozedur

Gibt Auskunft über den aktuellen Wert des DTT-Modells (Disk Transfer Time), das Teil des Kostenmodells ist.

Syntax

```
sa_get_dtt( file_id )
```

Argumente

- **file_id** Verwenden Sie diesen UNSIGNED SMALLINT-Parameter, um die Datenbankdatei-ID anzugeben.

Bemerkungen

Sie können die *file_id* aus der SYSDBSPACE-Systemansicht ermitteln.

Diese Prozedur, die für interne Diagnosezwecke gedacht ist, ruft Daten aus der Systemtabelle ISYSOPTSTAT ab.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
BandSize	UNSIGNED INTEGER	Größe, in Seiten, des Festplattenbereichs, bei dem wahlfreier Zugriff stattfindet
ReadTime	UNSIGNED INTEGER	Amortisierte Kosten, in Mikrosekunden, beim Lesen einer Seite
WriteTime	UNSIGNED INTEGER	Amortisierte Kosten, in Mikrosekunden, beim Schreiben einer Seite

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „[SYSDBSpace-Systemansicht](#)“ auf Seite 1444
- „[SYSOPTSTAT-Systemansicht](#)“ auf Seite 1468
- „[sa_get_dtt_groupreads-Systemprozedur](#)“ auf Seite 1224

Beispiel

Im folgenden Beispiel wird der aktuelle Wert des DTT-Modells (Disk Transfer Time) für den System-DBSpace gemeldet.

```
CALL sa_get_dtt( (select dbspace_id from SYSDBSpace where  
dbspace_name='system') );
```

sa_get_dtt_groupreads-Systemprozedur

Erstellt Schätzungen und Berichte zu den Kosten für Gruppen-Lesevorgänge auf dem Datenbankserver.

Syntax

```
sa_get_dtt_groupreads( dbspace_id )
```

Argumente

- **dbspace_id** Verwenden Sie diesen UNSIGNED SMALLINT-Parameter, um die Datenbankdatei-ID anzugeben.

Bemerkungen

Sie können die *dbspace_id* aus der SYSDBSpace-Systemansicht abrufen. Die von der sa_get_dtt_groupreads-Systemprozedur zurückgegebenen Schätzungen sind Teil des Kostenmodells und werden benutzt, um Gruppen-Lesevorgänge geeigneter Größe während bestimmter Vorgänge, wie z.B. Sortierungen, auszuwählen.

Diese Prozedur, die für interne Diagnosezwecke gedacht ist, ruft Daten aus der Systemtabelle ISYSOPTSTAT ab. Wenn in dieser Tabelle keine Einträge erfasst sind, werden typische Werte zurückgegeben. Wenn Sie Schätzwerte auf Ihre Hardware abstimmen möchten, führen Sie die folgende Anweisung aus:

```
ALTER DATABASE CALIBRATE GROUP READ;
```

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
GroupSize	UNSIGNED INTEGER	Größe, in Seiten, des Festplattenbereichs, bei dem wahlfreier Zugriff stattfindet
ReadTime	FLOAT	Amortisierte Kosten, in Mikrosekunden, beim Lesen einer Seite

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „SYSDBSpace-Systemansicht“ auf Seite 1444
- „SYSOPTSTAT-Systemansicht“ auf Seite 1468
- „ALTER DATABASE-Anweisung“ auf Seite 451
- „sa_get_dtt-Systemprozedur“ auf Seite 1223

Beispiel

Im folgenden Beispiel wird angegeben, wie hoch die Kosten des Ausgebens von Gruppenlesevorgängen für den System-DBSpace sind.

```
CALL sa_get_dtt_groupreads( (select dbspace_id from SYSDBSpace where
dbspace_name='system') );
```

sa_get_histogram-Systemprozedur

Ruft das Histogramm für eine Spalte ab

Syntax

```
sa_get_histogram(
  col_name
, tbl_name
[, owner_name ]
)
```

Argumente

- **col_name** Verwenden Sie diesen CHAR(128)-Parameter, um die Spalte anzugeben, aus der das Histogramm abgerufen werden soll.
- **tbl_name** Verwenden Sie diesen CHAR(128)-Parameter, um die Tabelle anzugeben, in der sich *col_name* befindet.
- **owner_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümer von *tbl_name* anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
StepNumber	SMALLINT	Nummer des Histogramm-Buckets (Reihe von Wertebereichen). Die Frequenz des ersten Buckets (StepNumber = 0) zeigt die Selektivität von NULL an

Spaltenname	Datentyp	Beschreibung
Low	CHAR(128)	Niedrigster (inklusive) Spaltenwert im Bucket
High	CHAR(128)	Höchster (exklusiver) Spaltenwert im Bucket
Frequency	DOUBLE	Selektivität von Werten im Bucket

Bemerkungen

Diese Prozedur, die für interne Diagnosezwecke gedacht ist, ruft Spaltenstatistiken aus dem Datenbankserver für die angegebenen Spalten ab. Obwohl diese Statistiken dauerhaft in der ISYSCOLSTAT-Systemtabelle gespeichert werden, werden sie bei laufendem Server im Speicher verwaltet und regelmäßig in ISYSCOLSTAT geschrieben. Daher können sich die von der sa_get_histogram-Systemprozedur zurückgegebenen Statistiken zu jedem gegebenen Zeitpunkt von jenen unterscheiden, die über ISYSCOLSTAT bezogen werden.

Sie können ISYSCOLSTAT manuell mit den aktuellsten im Speicher gehaltenen Statistiken aktualisieren, indem Sie die sa_flush_statistics-Systemprozedur verwenden, was allerdings in einer Produktionsumgebung nicht empfehlenswert ist und nur zu Diagnosezwecken angewendet werden soll.

Ein Einzelwert-Bucket wird dadurch angezeigt, dass ein Low-Wert in der Ergebnismenge mit dem entsprechenden High-Wert übereinstimmt.

Es wird empfohlen, dass Sie das Histogramm-Dienstprogramm zur Anzeige von Histogrammen verwenden.

Wenn die Selektivität eines Prädikats- gegenüber einer Zeichenfolgenspalte ermittelt werden soll, verwenden Sie die Funktionen ESTIMATE oder ESTIMATE_SOURCE. Für Zeichenfolgenspalten rufen sa_get_histogram und das Histogramm-Dienstprogramm aus der ISYSCOLSTAT-Systemtabelle keine Werte ab. Falls versucht wird, Zeichenfolgedaten abzurufen, wird ein Fehler erzeugt.

Statistiken (einschließlich Histogramme) sind möglicherweise für eine Tabelle oder materialisierte Ansicht nicht vorhanden, beispielsweise wenn Statistiken kürzlich gelöscht wurden. In diesem Fall ist die Ergebnismenge für die sa_get_histogram-Systemprozedur leer. Um Statistiken für eine Tabelle oder eine materialisierte Ansicht zu erstellen, führen Sie eine CREATE STATISTICS-Anweisung aus.

Privilegien

Sie müssen das MANAGE ANY STATISTICS-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Optimiererschätzungen und -statistiken“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Histogramm-Dienstprogramm (dbhist)“ [*SQL Anywhere Server - Datenbankadministration*]
- „SYSCOLSTAT-Systemansicht“ auf Seite 1441
- „CREATE STATISTICS-Anweisung“ auf Seite 729
- „sa_flush_statistics-Systemprozedur“ auf Seite 1220
- „ESTIMATE_SOURCE-Funktion [Verschiedene]“ auf Seite 253
- „ESTIMATE-Funktion [Verschiedene]“ auf Seite 252
- „Histogramm-Dienstprogramm (dbhist)“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Die folgende Anweisung ruft das Histogramm für die ProductID-Spalte aus der SalesOrderItems-Tabelle ab:

```
CALL sa_get_histogram( 'ProductID', 'SalesOrderItems' );
```

sa_get_ldapserver_status-Systemprozedur

Ermöglicht es Ihnen, den aktuellen Status von LDAP-Servern zu ermitteln.

Syntax

```
sa_get_ldapserver_status( )
```

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
ldsrv_id	UNSIGNED BIGINT	Eine eindeutige Nummer, die den LDAP-Server identifiziert.
ldsrv_name	CHAR(128)	Der Name des LDAP-Servers.
ldsrv_state	CHAR(9)	Der aktuelle Zustand des LDAP-Servers beim letzten Checkpoint.
ldsrv_last_state_change	TIMESTAMP	Der Zeitpunkt in Ortszeit, zu dem der Zustand zuletzt geändert wurde.

Bemerkungen

Diese Prozedur gibt eine Ergebnismenge zurück, die den aktuellen Status von LDAP-Servern zeigt.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „LDAP-Benutzerauthentifizierung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Beispiel

Im folgenden Beispiel wird die `sa_get_ldapserver_status`-Systemprozedur verwendet, um den Status aller LDAP-Server zurückzugeben.

```
CALL sa_get_ldapserver_status;
```

sa_get_request_profile-Systemprozedur

Analysiert das Anforderungslog, um die Ausführungszeiten ähnlicher Anweisungen zu ermitteln

Syntax

```
sa_get_request_profile(  
  [ filename  
  [, conn_id  
  [, first_file  
  [, num_files ] ] ] ]  
)
```

Argumente

- **filename** Verwenden Sie diesen optionalen LONG VARCHAR-Parameter, um den Dateinamen des Anforderungslogs anzugeben. Der Standardwert ist NULL.
- **conn_id** Verwenden Sie diesen optionalen UNSIGNED INTEGER-Parameter, um die Verbindungs-ID-Nummer anzugeben. Der Standardwert ist 0.
- **first_file** Verwenden Sie diesen optionalen INTEGER-Parameter, um die erste zu analysierende Anforderungs-Logdatei anzugeben. Der Standardwert ist -1.
- **num_files** Verwenden Sie diesen optionalen INTEGER-Parameter, um die Anzahl der zu analysierenden Anforderungs-Logdateien anzugeben. Standardwert ist "1".

Bemerkungen

Diese Prozedur ruft `sa_get_request_times` auf, um eine Anforderungs-Logdatei zu verarbeiten, und fasst dann die Ergebnisse in der globalen temporären Tabelle `satmp_request_profile` zusammen. Diese Tabelle enthält die Anweisungen aus dem Log sowie Angaben zur Ausführungshäufigkeit der einzelnen Anweisungen und Gesamt-, Durchschnitts- und Maximalzeiten der Ausführungen. Durch Sortieren der Tabelle auf unterschiedliche Arten können Ziele für die Performance-Optimierung ermittelt werden.

Wenn Sie keine Logdatei (*filename*) angeben, wird standardmäßig die aktuelle Logdatei verwendet, die mit der Datenbankserveroption `-zo` oder mit dem folgenden Befehl angegeben wurde:

```
sa_server_option( 'RequestLogFile', filename )
```

Wird eine Verbindungs-ID angegeben, dann wird sie benutzt, um Daten aus dem Log zu filtern, sodass nur Anforderungen für die betreffende Verbindung abgerufen werden.

Privilegien

Sie müssen die DIAGNOSTICS-Systemrolle und das MANAGE PROFILING-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „sa_get_request_times-Systemprozedur“ auf Seite 1229
- „sa_statement_text-Systemprozedur“ auf Seite 1337
- „sa_server_option-Systemprozedur“ auf Seite 1306
- „Datenbankserveroption -zo“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Der folgende Befehl ruft die Anforderungszeiten für die Anforderungen in der Datei *req.out* ab.

```
CALL sa_get_request_profile('req.out');
```

Der folgende Befehl ruft die Anforderungszeiten für die Anforderungen in den Dateien *req.out.3*, *req.out.4* und *req.out.5* ab.

```
CALL sa_get_request_profile('req.out',0,3,3);
```

sa_get_request_times-Systemprozedur

Analysiert das Anforderungslg, um die Ausführungszeiten von Anweisungen zu ermitteln

Syntax

```
sa_get_request_times( [ filename  
    [, conn_id  
    [, first_file  
    [, num_files ] ] ] ]  
)
```

Argumente

- **filename** Verwenden Sie diesen optionalen LONG VARCHAR-Parameter, um den Dateinamen des Anforderungslgs anzugeben. Der Standardwert ist NULL.
- **conn_id** Verwenden Sie diesen optionalen UNSIGNED INTEGER-Parameter, um die Verbindungs-ID-Nummer anzugeben. Der Standardwert ist 0.
- **first_file** Verwenden Sie diesen optionalen INTEGER-Parameter, um die erste zu analysierende Datei anzugeben. Der Standardwert ist -1.
- **num_files** Verwenden Sie diesen optionalen INTEGER-Parameter, um die Anzahl der zu analysierenden Anforderungs-Logdateien anzugeben. Standardwert ist "1".

Bemerkungen

Diese Prozedur liest das angegebene Anforderungslog und überträgt die Anweisungen aus dem Log sowie deren Ausführungszeiten in die globale temporäre Tabelle `satmp_request_time`.

Bei Anweisungen wie Einfügungen und Aktualisierungen ist die Ausführungszeit einfach zu ermitteln. Bei Abfragen wird die Zeit von der Vorbereitung der Anweisung bis zum Löschen eingerechnet, einschließlich Beschreiben, Öffnen des Cursors, Abrufen der Zeilen und Schließen des Cursors. Bei den meisten Abfragen ergibt dies ein genaues Bild der erforderlichen Zeit. Wenn der Cursor offen gelassen wird, während andere Ereignisse stattfinden, etwa ein Eingriff des Bedieners oder eine Verarbeitung am Client, erscheint die Zeit als hoher Wert. Das bedeutet aber nicht, dass die Abfrage kostenträchtig ist.

Diese Prozedur erkennt Hostvariablen im Anforderungslog und überträgt deren Werte in die globale temporäre Tabelle `satmp_request_hostvar`. Bei älteren Datenbanken, wo diese temporäre Tabelle nicht vorhanden ist, werden Werte von Hostvariablen ignoriert.

Wenn Sie keine Logdatei angeben, wird standardmäßig die aktuelle Logdatei verwendet, die mit der Datenbankserveroption `-zo` oder mit dem folgenden Befehl angegeben wurde:

```
call sa_server_option( 'RequestLogFile', filename )
```

Wird eine Verbindungs-ID angegeben, dann wird sie benutzt, um Daten aus dem Log zu filtern, sodass nur Anforderungen für die betreffende Verbindung abgerufen werden.

Privilegien

Sie müssen das `MANAGE PROFILING`-Systemprivileg oder das `MONITOR`-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Beispiel

Der folgende Befehl ruft die Ausführungszeiten für die Anforderungen in der Datei `req.out` ab.

```
CALL sa_get_request_times('req.out');
```

Der folgende Befehl ruft die Ausführungszeiten für die Anforderungen in den Dateien `req.out.3`, `req.out.4` und `req.out.5` ab.

```
CALL sa_get_request_times('req.out',0,3,3);
```

Siehe auch

- [„sa_get_request_profile-Systemprozedur“ auf Seite 1228](#)
- [„sa_statement_text-Systemprozedur“ auf Seite 1337](#)
- [„sa_server_option-Systemprozedur“ auf Seite 1306](#)

sa_get_server_messages-Systemprozedur [nicht mehr empfohlen]

Hiermit können Sie Konstanten im Fenster für die Datenbankservermeldungen als Ergebnismenge ausgeben.

Diese Systemprozedur wird nicht mehr empfohlen. Verwenden Sie an ihrer Stelle sa_server_messages.

Syntax

```
sa_get_server_messages( first_line )
```

Argumente

- **first_line** Verwenden Sie diesen INTEGER-Parameter, um die Zeilennummer anzugeben, ab der Servermeldungen angezeigt werden sollen.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
line_num	INTEGER	Die Zeilennummer einer Servermeldung
message_text	VARCHAR(255)	Der Servermeldungstext
message_time	TIMESTAMP	Die Uhrzeit der Meldung

Bemerkungen

Diese Prozedur verwendet einen Parameter des Typs Ganzzahl, der die Zeilennummer der ersten anzuzeigenden Zeile festlegt und eine Zeile für diese und alle folgenden Zeilen zurückgibt. Wenn die Startzeile negativ ist, beginnt die Ergebnismenge mit der ersten verfügbaren Zeile. Die Ergebnismenge enthält die Zeilennummer, den Meldungstext und die Meldungszeit.

Privilegien

Keine

Nebenwirkungen

Keine

Beispiel

Im folgenden Beispiel wird die sa_get_server_messages-Systemprozedur verwendet, um eine Ergebnismenge zurückzugeben, die den Inhalt aus dem Meldungsfenster des Datenbankservers enthält, beginnend mit Zeile 16.

```
CALL sa_get_server_messages( 16 );
```

line_num	message_text	...
16	Windows Build 2195...	...
17	2132 kB Speicher für Caching benutzt	...
...

sa_get_table_definition-Systemprozedur

Gibt eine LONG VARCHAR-Zeichenfolge mit den SQL-Anweisungen zurück, die zum Erstellen der angegebenen Tabelle sowie der dazugehörigen Indizes, Fremdschlüssel, Trigger und erteilten Privilegien erforderlich sind.

Syntax

```
sa_get_table_definition(  
    @owner  
    , @tablename  
)
```

Argumente

- **@owner** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Eigentümer von *tablename* anzugeben.
- **@tablename** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Namen der Tabelle anzugeben.

Bemerkungen

Diese Funktion gibt eine LONG VARCHAR-Zeichenfolge mit den SQL-Anweisungen zurück, die zum Erstellen der angegebenen Tabelle sowie der dazugehörigen Indizes, Fremdschlüssel, Trigger und erteilten Privilegien erforderlich sind. Wenn Sie eine neue Tabelle mit derselben Definition erstellen möchten, verwenden Sie die von dieser Funktion zurückgegebene Zeichenfolge zusammen mit der EXECUTE IMMEDIATE-Anweisung sowie den Funktionen LOCATE, SUBSTRING und REPLACE.

Privilegien

Sie müssen das SELECT ANY TABLE-Systemprivileg oder das SELECT-Privileg für die SYSUSERPERM-Kompatibilitätsansicht haben.

Nebenwirkungen

Keine

Siehe auch

- „SYSUSERPERM-Kompatibilitätsansicht (nicht mehr empfohlen)“ auf Seite 1544
- „sa_split_list-Systemprozedur“ auf Seite 1332
- „EXECUTE IMMEDIATE-Anweisung [SP]“ auf Seite 843
- „LOCATE-Funktion [Zeichenfolge]“ auf Seite 305
- „SUBSTRING-Funktion [Zeichenfolge]“ auf Seite 401
- „REPLACE-Funktion [Zeichenfolge]“ auf Seite 365

Beispiel

Die folgende Anweisung zeigt mithilfe der sa_get_table_definition-Systemprozedur die Zeichenfolge an, die die zum Erstellen der Tabelle "Departments" erforderlichen SQL-Anweisungen enthält.

```
SELECT sa_get_table_definition( 'GROUPO', 'Departments');
```

sa_get_user_status-Systemprozedur

Damit können Sie den aktuellen Status der Benutzer ermitteln.

Syntax

```
sa_get_user_status( )
```

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
user_id	UNSIGNED INTEGER	Eine eindeutige Nummer, die den Benutzer kennzeichnet.
user_name	CHAR(128)	Der Name des Benutzers.
connections	INTEGER	Die aktuelle Anzahl von Verbindungen dieses Benutzers.
failed_logins	UNSIGNED INTEGER	Die Anzahl fehlgeschlagener Login-Versuche des Benutzers.
last_login_time	TIMESTAMP	Der Zeitpunkt in Ortszeit, zu dem sich der Benutzer zuletzt angemeldet hat.
locked	TINYINT	Zeigt an, ob das Benutzerkonto gesperrt ist.
reason_locked	LONG VARCHAR	Der Grund, warum das Konto gesperrt ist.
user_dn	CHAR(1024)	Der Distinguished Name (DN) für eine Benutzer-ID, die eine Verbindung mit einem LDAP-Server herstellt.

Spaltenname	Datentyp	Beschreibung
user_dn_cached_at	TIMESTAMP	Der Zeitpunkt in Ortszeit, zu dem der DN gespeichert wurde.
password_change_state	BIT	Ein Wert, der angibt, ob eine Kennwortänderung mit Doppelkontrolle läuft. (0 entspricht "Nein", 1 entspricht "Ja".) Der Standardwert ist 0.
password_change_first_user	UNSIGNED INTEGER	Die Benutzer-ID des Benutzers, der den ersten Teil eines zweiteiligen Kennworts festgelegt hat, andernfalls NULL.
password_change_second_user	UNSIGNED INTEGER	Die Benutzer-ID des Benutzers, der den zweiten Teil eines zweiteiligen Kennworts festgelegt hat, andernfalls NULL.

Bemerkungen

Diese Prozedur gibt eine Ergebnismenge zurück, die den aktuellen Status der Benutzer zeigt. Zusätzlich zu den Basisinformationen über die Benutzer wird von dieser Prozedur eine Spalte ausgegeben, die anzeigt, ob die Benutzer gesperrt wurden, sowie eine weitere Spalte, in der der Grund für die Sperre angegeben wird. Benutzer können aus folgenden Gründen gesperrt werden: aufgrund einer Richtlinie, eines Kennwortablaufs oder wegen zu vieler fehlgeschlagener Anmeldeversuche.

Privilegien

Sie können Informationen zu sich selbst anzeigen. Dafür sind keine Privilegien erforderlich. Sie müssen das MANAGE ANY USER-Systemprivileg haben, um Informationen zu anderen Benutzern anzeigen zu können.

Nebenwirkungen

Keine

Siehe auch

- „Login-Richtlinien“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Neue Login-Richtlinien erstellen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Benutzer erstellen (Sybase Central)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Login-Richtlinien vorhandenen Benutzern zuordnen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Login-Richtlinien ändern“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Login-Richtlinien löschen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Kennwörter für die Doppelkontrolle“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Im folgenden Beispiel wird die sa_get_user_status-Systemprozedur verwendet, um den Status von Datenbankbenutzern zurückzugeben.

```
CALL sa_get_user_status;
```

sa_http_header_info-Systemprozedur

Gibt HTTP-Anforderungsheadernamen und -werte zurück

Syntax

```
sa_http_header_info( [header_parm] )
```

Argumente

- **header_parm** Verwenden Sie diesen optionalen VARCHAR(255)-Parameter, um einen HTTP-Headernamen anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
Name	VARCHAR(255)	Der HTTP-Headername
Value	LONG VARCHAR	Der HTTP-Headerwert

Bemerkungen

Die sa_http_header_info-Systemprozedur gibt Headernamen und -werte zurück. Wenn Sie den Headernamen nicht mit dem optionalen Parameter angeben, enthält die Ergebnismenge Werte für alle Header.

Die Prozedur gibt eine nicht-leere Ergebnismenge zurück, wenn sie aufgerufen wird, während sie eine HTTP-Anforderung innerhalb eines Webdienstes verarbeitet.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- [„NEXT_HTTP_HEADER-Funktion \[Webdienst\]“ auf Seite 325](#)
- [„HTTP_HEADER-Funktion \[Webdienst\]“ auf Seite 283](#)
- [„Webdienstfunktionen“ auf Seite 162](#)
- [„Webdienst-Systemprozeduren“ auf Seite 1162](#)

Beispiel

Die folgende Webdienstprozedur, die aus einem Webdienst heraus aufgerufen wird, veranschaulicht die Verwendung der sa_http_header_info-Systemprozedur.

```
CREATE OR REPLACE PROCEDURE User1.HTTPHeaderExample()  
RESULT ( html_string LONG VARCHAR )
```

```
BEGIN
  DECLARE myname VARCHAR(255);
  DECLARE myvalue LONG VARCHAR;
  DECLARE err_notfound
    EXCEPTION FOR SQLSTATE '02000';
  DECLARE curs CURSOR FOR
    SELECT Name, Value FROM sa_http_header_info();
  MESSAGE '=== HTTP Headers ===' TO CONSOLE;
  OPEN curs;
  FetchLoop: LOOP
    FETCH next curs INTO myname, myvalue;
    IF SQLSTATE = err_notfound THEN
      LEAVE FetchLoop;
    END IF;
    MESSAGE myname, '=', myvalue TO CONSOLE;
  END LOOP FetchLoop;
  CLOSE curs;
END;
```

Wenn der Webdienst, der diese Webdienstprozedur aufruft, verwendet wird, erscheint im Meldungsfenster des Datenbankservers eine ähnliche Ausgabe wie im folgenden Beispiel.

```
=== HTTP Headers ===
@QueryString=param1=value1&param2=value2&param3=value3
User-Agent=Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101
Firefox/16.0
Authorization=Basic VXNlcjE6dXNlcg==
Cache-Control=max-age=0
Connection=keep-alive
Host=schueler-t3500.sybase.com:8082
@HttpURI=/ShowHTTPHeaders?param1=value1&param2=value2&param3=value3
@HttpMethod=GET
Accept=text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
@HttpVersion=HTTP/1.1
Accept-Language=en-US,en;q=0.5
Accept-Encoding=gzip, deflate
```

sa_http_php_page-Systemprozedur

Gibt das Ergebnis der Übergabe des PHP-Codes zurück, der von einem PHP-Interpreter mit der aktuellen HTTP-Anforderung für Kontextinformationen wie Header, GET/POST-Daten, Protokollversion, URL-Anforderung, Methode etc. interpretiert werden muss.

Syntax

sa_http_php_page(*php_page*)

Argumente

- **php_page** Dieser LONG VARCHAR-Parameter enthält den gesamten PHP-Code, der interpretiert werden muss, einschließlich Start- und Endmarken (<?php und ?>).

Rückgabe

Diese Funktion gibt einen LONG BINARY-Wert zurück.

Bemerkungen

Um diese Systemprozedur zu benutzen, muss die externe PHP-Umgebung bereits installiert sein.

Der Eigentümer dieser Systemprozedur ist DBO. Zur Erhöhung der Systemsicherheit wird allerdings die sa_http_php_page-Systemprozedur als Aufrufer ausgeführt.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- [„sa_http_php_page_interpreted-Systemprozedur“ auf Seite 1237](#)
- [„Webdienstfunktionen“ auf Seite 162](#)
- [„Webdienst-Systemprozeduren“ auf Seite 1162](#)

Beispiel

Im folgenden Beispiel wird die phpinfo()-Abfrage an einen PHP-Interpreter gesendet und das HTML-Ergebnis angezeigt.

```
SELECT CAST( sa_http_php_page('<?php phpinfo(); ?>') AS LONG VARCHAR );
```

Im folgenden Beispiel wird ein PHP-Skript an einen PHP-Interpreter gesendet und das XML-Ergebnis angezeigt.

```
SELECT CAST( sa_http_php_page('<?php '||
'$conn = sasql_connect( "UID=DBA;PWD=sql" ); '||
'$result = sasql_query( $conn, "SELECT * FROM Employees" ); '||
'$sasql_result_all( $result ); '||
'$sasql_free_result( $result ); '||
'$sasql_disconnect( $conn ); '||
'?>') AS LONG VARCHAR );
```

sa_http_php_page_interpreted-Systemprozedur

Gibt das Ergebnis der Übergabe des PHP-Codes zurück, der von einem PHP-Interpreter mit den angegebenen Parametern für Kontextinformationen wie Header, GET/POST-Daten, Protokollversion, URL-Anforderung, Methode etc. interpretiert werden muss.

Syntax

```
sa_http_php_page_interpreted(
  php_page
, method
, url
, version
, headers
, request_body
)
```

Argumente

- ***php_page*** Dieser LONG VARCHAR-Parameter enthält den gesamten PHP-Code, der interpretiert werden muss, einschließlich Start- und Endmarken (<?php und ?>).
- ***method*** Dieser LONG VARCHAR-Parameter enthält die HTTP-Anforderungsmethode (z.B. GET, POST, PUT oder eine der anderen Standard-Anforderungsmethoden). Der Wert für *method* kann mit dem Wert für @HttpMethod in der aktuellen HTTP-Anforderung ermittelt werden.
- ***url*** Dieser LONG VARCHAR-Parameter enthält die komplette HTTP-Anforderungs-URL einschließlich einer eventuellen Abfragenzeichenfolge. Der Wert für *url* kann mit dem Wert für @HttpURI in der aktuellen HTTP-Anforderung ermittelt werden.
- ***version*** Dieser LONG VARCHAR-Parameter enthält die Protokollversion der HTTP-Anforderung (Beispiel: HTTP/1.1). Der Wert für *version* kann mit dem Wert für @HttpVersion in der aktuellen HTTP-Anforderung ermittelt werden.
- ***headers*** Dieser LONG BINARY-Parameter enthält die Header in der HTTP-Anforderung im Standard-HTTP-Header-Format: Field-Name: Value\r\n. Der Wert für Header kann aus der aktuellen HTTP-Anforderung mit folgender SELECT-Anweisung ermittelt werden:

```
SELECT LIST( name || ': ' || value, CHAR(13) || CHAR(10) )  
FROM sa_http_header_info();
```
- ***request_body*** Dieser LONG BINARY-Parameter enthält den Hauptteil der HTTP-Anforderung in binärer Form. Der Wert des *request_body* kann aus der aktuellen HTTP-Anforderung mit der HTTP_BODY-Funktion ermittelt werden.

Rückgabe

Diese Funktion gibt einen LONG BINARY-Wert zurück.

Bemerkungen

Um diese Systemprozedur zu benutzen, muss die externe PHP-Umgebung bereits installiert sein.

Um die Systemprozedur außerhalb von Webdienstanforderungen zu verwenden, müssen Sie eine Anforderungsinformation bereitstellen. Header, die im PHP-Code festgelegt wurden, gehen verloren.

Der Eigentümer dieser Systemprozedur ist DBO. Zur Erhöhung der Systemsicherheit wird allerdings die sa_http_php_page_interpreted-Systemprozedur als Aufrufer ausgeführt.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „HTTP_BODY-Funktion [Webdienst]“ auf Seite 280
- „Die externe PHP-Umgebung“ [*SQL Anywhere Server - Programmierung*]
- „sa_http_php_page-Systemprozedur“ auf Seite 1236
- „sa_http_header_info-Systemprozedur“ auf Seite 1235
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Beispiel

Im folgenden Beispiel wird die phpinfo()-Abfrage an einen PHP-Interpreter gesendet und das HTML-Ergebnis angezeigt.

```
BEGIN
    DECLARE headers LONG VARCHAR;

    SELECT list( name || ': ' || value, char(13) || char(10) ) INTO headers
    FROM sa_http_header_info();

    SELECT CAST( sa_http_php_page_interpreted( '<?php phpinfo(); ?>',
        http_header( '@HttpMethod' ),
        http_header( '@HttpURI' ),
        http_header( '@HttpVersion' ),
        headers,
        HTTP_BODY() ) AS LONG VARCHAR);
END;
```

sa_http_variable_info-Systemprozedur

Gibt HTTP-Variablennamen und -werte zurück

Syntax

sa_http_variable_info([variable_parm])

Argumente

- **variable_parm** Verwenden Sie diesen optionalen VARCHAR(255)-Parameter, um einen HTTP-Variablennamen anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
Name	VARCHAR(255)	Der HTTP-Variablenname
Value	LONG VARCHAR	Der HTTP-Variablenwert

Bemerkungen

Die sa_http_variable_info-Systemprozedur gibt Variablennamen und -werte zurück. Wenn Sie den Variablennamen nicht mit dem optionalen Parameter angeben, enthält die Ergebnismenge Werte für alle Variablen.

Die Prozedur gibt eine nicht-leere Ergebnismenge zurück, wenn sie aufgerufen wird, während sie eine HTTP-Anforderung innerhalb eines Webdienstes verarbeitet.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „NEXT_HTTP_VARIABLE-Funktion [Webdienst]“ auf Seite 327
- „HTTP_VARIABLE-Funktion [Webdienst]“ auf Seite 287
- „sa_http_header_info-Systemprozedur“ auf Seite 1235
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Beispiel

Die folgende Webdienstprozedur, die aus einem Webdienst heraus aufgerufen wird, veranschaulicht die Verwendung der sa_http_variable_info-Systemprozedur.

```
CREATE OR REPLACE PROCEDURE User1.HTTPVariableExample()
RESULT ( html_string LONG VARCHAR )
BEGIN
    DECLARE myname VARCHAR(255);
    DECLARE myvalue LONG VARCHAR;
    DECLARE err_notfound
        EXCEPTION FOR SQLSTATE '02000';
    DECLARE curs CURSOR FOR
        SELECT Name, Value FROM sa_http_variable_info();
    MESSAGE '=== HTTP Variables ===' TO CONSOLE;
    OPEN curs;
    FetchLoop: LOOP
        FETCH next curs INTO myname, myvalue;
        IF SQLSTATE = err_notfound THEN
            LEAVE FetchLoop;
        END IF;
        MESSAGE myname, '=', myvalue TO CONSOLE;
    END LOOP FetchLoop;
    CLOSE curs;
END;
```

Bei einer URL-Parameterliste wie ?param1=value1¶m2=value2¶m3=value3 wird die Ausgabe dieser Beispiel-Webdienstprozedur im Meldungsfenster des Datenbankservers als Liste in der Form "Parametername=Wert" angezeigt.

```
=== HTTP Variables ===
param1=value1
param3=value3
param2=value2
```

sa_index_density-Systemprozedur

Gibt Auskunft über das Ausmaß der Fragmentierung und der Schiefe (Skew) in den Datenbankindizes

Syntax

```
sa_index_density(
  [ tbl_name
  [, owner_name ] ]
)
```

Argumente

- **tbl_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Tabellennamen anzugeben. Der Standardwert ist NULL.
- **owner_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümernamen anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
TableName	CHAR(128)	Der Name einer Tabelle
TableId	UNSIGNED INTEGER	Die Tabellen-ID.
IndexName	CHAR(128)	Der Name eines Indexes
IndexId	UNSIGNED INTEGER	Die Indexkennung. Diese Spalte enthält einen der folgenden Werte: <ul style="list-style-type: none"> • 0 für Primärschlüssel • SYSFKEY.foreign_key_id für Fremdschlüssel • SYSIDX.index_id für alle anderen Indizes
IndexType	CHAR(4)	Der Indextyp. Diese Spalte enthält einen der folgenden Werte: <ul style="list-style-type: none"> • PKEY für Primärschlüssel • FKEY für Fremdschlüssel • UI für eindeutige Indizes • UC für Eindeutigkeits-Integritätsregeln • NUI für nicht-eindeutige Indizes
LeafPages	UNSIGNED INTEGER	Die Anzahl der Blattseiten
Density	DOUBLE	Eine Dezimalzahl zwischen 0 und 1, die anzeigt, wie voll die einzelnen Indexseiten sind (Durchschnittswert)

Spaltenname	Datentyp	Beschreibung
Skew	DOUBLE	Eine Zahl, die anzeigt, wie stark die Schiefe in einem Index ist. Der Wert 1 bezeichnet einen perfekt ausbalancierten Index. Größere Werte zeigen eine immer stärkere Schiefe.

Bemerkungen

Verwenden Sie die `sa_index_density`-Systemprozedur, um Informationen zum Grad der Fragmentierung und Schiefe in Indizes abzurufen. Bei Indizes mit einer großen Anzahl von Blattseiten sind höhere Dichtewerte und niedrigere Schiefenwerte wünschenswert.

Die Indexdichte zeigt als Prozentwert, wie voll durchschnittlich die Indexseiten sind. Eine Dichte von 0,7 zeigt, dass Indexseiten durchschnittlich zu 70 % mit Indexdaten gefüllt sind. Die Schiefe eines Indexes zeigt die typische Abweichung von der durchschnittlichen Dichte an. Die Größe der Schiefe ist für den Optimierer wichtig, wenn er Selektivitätsschätzungen vornimmt.

Wenn die Anzahl der Blattseiten niedrig ist, spielen die Dichte- und Schiefewerte keine Rolle. Dichte- und Schiefenwerte werden nur dann wichtig, wenn die Anzahl der Blattseiten hoch ist. Wenn die Anzahl der Blattseiten hoch ist, kann ein niedriger Dichtewert auf Fragmentierung hinweisen, ein hoher Schiefewert zeigt, dass die Indizes nicht ausgewogen sind. Beide Phänomene können an einer Verschlechterung der Performance beteiligt sein. Mit einer `REORGANIZE TABLE`-Anweisung können beide Probleme beseitigt werden.

Wenn Sie beim Aufruf dieser Prozedur keine Tabelle angeben, werden Informationen zu allen Indizes für alle Tabellen der Datenbank zurückgegeben.

Sie können auch den **Assistenten für die Anwendungsprofilerstellung** verwenden, um zu ermitteln, ob die Indexdichte und die Schiefe akzeptable Werte haben.

Privilegien

Sie müssen eines der folgenden Systemprivilegien haben:

- MONITOR
- MANAGE ANY STATISTICS
- CREATE ANY INDEX
- ALTER ANY INDEX
- DROP ANY INDEX
- CREATE ANY OBJECT
- ALTER ANY TABLE
- DROP ANY OBJECT

Nebenwirkungen

Keine

Siehe auch

- „Indexfragmentierung und -schiefe (Skew) reduzieren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „REORGANIZE TABLE-Anweisung“ auf Seite 998
- „Verwenden des Assistenten für die Anwendungsprofilerstellung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Beispiel

Im folgenden Beispiel wird die sa_index_density-Systemprozedur verwendet, um eine Ergebnismenge zurückzugeben, die die Fragmentierung und die Schiefe in allen Indizes der Datenbank enthält.

```
CALL sa_index_density( );
```

sa_index_levels-Systemprozedur

Bietet Unterstützung bei der Optimierung der Performance, indem die Anzahl der Ebenen in einem Index angezeigt wird

Syntax

```
sa_index_levels(  
  [ tbl_name  
  [, owner_name ] ]  
)
```

Argumente

- **tbl_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Tabellennamen anzugeben. Der Standardwert ist NULL.
- **owner_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümernamen anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
TableName	CHAR(128)	Der Name einer Tabelle
TableId	UNSIGNED INTEGER	Die Tabellen-ID.
IndexName	CHAR(128)	Der Name eines Indexes

Spaltenname	Datentyp	Beschreibung
IndexId	UNSIGNED INTEGER	Die Indexkennung. Diese Spalte enthält einen der folgenden Werte: <ul style="list-style-type: none"> • 0 für Primärschlüssel • SYSFKEY.foreign_key_id für Fremdschlüssel • SYSIDX.index_id für alle anderen Indizes
IndexType	CHAR(4)	Der Indextyp. Diese Spalte enthält einen der folgenden Werte: <ul style="list-style-type: none"> • PKEY für Primärschlüssel • FKEY für Fremdschlüssel • UI für eindeutige Indizes • UC für Eindeutigkeits-Integritätsregeln • NUI für nicht-eindeutige Indizes
Levels	INTEGER	Die Anzahl der Ebenen im Index

Bemerkungen

Die Anzahl der Ebenen im Indexbaum bestimmt die Anzahl der I/O-Vorgänge, die zum Zugriff auf eine Zeile mithilfe des Indexes benötigt werden. Indizes mit wenigen Ebenen sind wirkungsvoller als Indizes mit einer großen Anzahl von Ebenen.

Die Prozedur gibt eine Ergebnismenge zurück, die den Tabellennamen, die Tabellen-ID, den Indexnamen, die Index-ID, den Indextyp und die Anzahl von Ebenen im Index enthält,

Wenn keine Argumente angegeben sind, werden die Ebenen für alle Indizes in der Datenbank zurückgegeben. Wenn nur *tbl_name* angegeben wird, werden die Ebenen für alle Indizes dieser Tabelle zurückgegeben. Wenn *tbl_name* NULL ist und *owner_name* angegeben wird, werden nur die Ebenen der Indizes von Tabellen dieses Eigentümers zurückgegeben.

Privilegien

Sie müssen eines der folgenden Systemprivilegien haben:

- **MANAGE ANY STATISTICS**
- **CREATE ANY INDEX**
- **ALTER ANY INDEX**
- **DROP ANY INDEX**
- **CREATE ANY OBJECT**
- **ALTER ANY TABLE**
- **DROP ANY OBJECT**

Nebenwirkungen

Keine

Siehe auch

- „CREATE INDEX-Anweisung“ auf Seite 638
- Die richtige Auswahl der Indizes kann sich entscheidend auf die Performance auswirken [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Beispiel

Im folgenden Beispiel wird die `sa_index_levels`-Systemprozedur verwendet, um die Anzahl der Ebenen im Products-Index zurückzugeben.

```
CALL sa_index_levels( );
```

TableName	TableId	IndexName	...	Levels
Products	436	Products	...	1
...

sa_install_feature-Systemprozedur

Installiert zusätzliche Funktionen, z.B. zusätzliche räumliche Funktionen.

Syntax

```
sa_install_feature( feat_name )
```

Argumente

- **feat_name** Ein LONG VARCHAR-Parameter, der die zu installierende Funktion identifiziert. Der Standardwert ist NULL. Die unterstützten Funktionsnamen lauten:

Wert	Beschreibung
st_geometry_pre-defined_uom	Installiert vordefinierte Maßeinheiten, die in neuen Datenbanken nicht standardmäßig installiert sind.
st_geometry_pre-defined_srs	Installiert vordefinierte räumliche Bezugssysteme und Maßeinheiten, die in neuen Datenbanken nicht standardmäßig installiert sind.
st_geometry_compat_func	Installiert eine Reihe von Kompatibilitätsfunktionen. Diese Funktionen können als Alternative zu den räumlichen Methoden verwendet werden.

Funktionsnamendefinitionen finden Sie in der Datei *st_geometry_config.tgz*, die im Verzeichnis *%SQLANY16%\scripts* gespeichert ist. Wenn die Datei entfernt wird und Sie versuchen, Funktionen zu installieren, die von der Datei abhängig sind, wird ein Fehler zurückgegeben.

Bemerkungen

Sie können den Feat_Name-Wert abfragen, um zu sehen, was installiert wird. Beispielsweise gibt die folgende Abfrage die Maßeinheiten zurück, die für st_geometry_predefined_uom installiert wurden.

```
SELECT * FROM st_geometry_predefined_uom( 'CREATE' );
```

Das vorherige Beispiel enthält außerdem Parameternamen, sodass Sie Abfragen für bestimmte Werte mit einer WHERE-Klausel ausführen können. Die folgende Anweisung fragt zum Beispiel den unit_name-Parameter für die verkettete Maßeinheit ab:

```
SELECT * FROM st_geometry_predefined_uom( 'CREATE' ) WHERE unit_name='chain';
```

unit_name	unit_type	conversion_factor	...
chain	LINEAR	20.1168	...

Folgendes gibt alle Maßeinheiten zurück, die auf Fuß basieren:

```
SELECT * FROM st_geometry_predefined_uom() WHERE unit_name LIKE '%foot%';
```

Mit der folgenden Abfrage können Sie nach den räumlichen Bezugssystemen suchen, die installiert würden:

```
SELECT * FROM st_geometry_predefined_srs();
```

Die folgende Anweisung startet eine Abfrage eines räumlichen Bezugssystems nach **organization** und **organization_coordsys_id**:

```
SELECT * FROM st_geometry_predefined_srs() WHERE organization='EPSG' AND organization_coordsys_id=2295;
```

Privilegien

Für st_geometry_predefined_uom und st_geometry_predefined_srs müssen Sie das MANAGE ANY SPATIAL OBJECT-Systemprivileg haben.

Für st_geometry_compat_func müssen Sie die Systemprivilegien MANAGE ANY OBJECT PRIVILEGE, CREATE ANY PROCEDURE und SELECT ANY TABLE haben.

Für sa_install_feature müssen Sie das MANAGE ANY SPATIAL OBJECT-Systemprivileg haben.

Siehe auch

- „Räumliche Bezugssysteme (SRS) und räumliche Referenz-IDs (SRID)“ [[SQL Anywhere Server - Unterstützung für räumliche Daten](#)]
- „Maßeinheiten“ [[SQL Anywhere Server - Unterstützung für räumliche Daten](#)]
- „Funktionen der räumlichen Kompatibilität“ [[SQL Anywhere Server - Unterstützung für räumliche Daten](#)]

Beispiel

Die folgende Anweisung installiert alle vordefinierten Maßeinheiten, die in einer neuen Datenbank nicht standardmäßig installiert sind:

```
CALL sa_install_feature( 'st_geometry_predefined_uom' );
```

Die folgende Anweisung installiert räumliche Kompatibilitätsfunktionen, die als Alternative zu den räumlichen Methoden verwendet werden können:

```
CALL sa_install_feature( 'st_geometry_compat_func' );
```

sa_java_loaded_classes-Systemprozedur

Listet die Klassen auf, die derzeit vom Datenbankserver in eine Java VM geladen sind.

Syntax

```
sa_java_loaded_classes( )
```

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
class_name	VARCHAR(512)	Der Name einer Klasse, die derzeit vom Datenbankserver in eine Java VM geladen ist.

Bemerkungen

Gibt eine Ergebnismenge zurück, die alle Namen der Java-Klassen enthält, die derzeit vom Datenbankserver in eine Java VM geladen sind.

Die Prozedur kann hilfreich sein bei der Diagnose fehlender Klassen. Sie kann auch benutzt werden, um zu ermitteln, welche Klassen aus einer bestimmten jar-Datei von einer bestimmten Anwendung verwendet werden.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Java-Klassen in Datenbanken installieren“ [[SQL Anywhere Server - Programmierung](#)]

Beispiel

Im folgenden Beispiel wird die Cover-Funktion init aufgerufen, um die init-Methode der Java-Beispielklasse Invoice aufzurufen(siehe „[Praktische Einführung: Java in der Datenbank verwenden](#)“ [[SQL Anywhere Server - Programmierung](#)]). Anschließend wird sa_java_loaded_classes aufgerufen, um eine Liste aller geladenen Java-Klassen abzurufen.

```
CALL init( 'Work boots', 79.99, 'Hay fork', 37.49 );  
CALL sa_java_loaded_classes( );
```

sa_list_cursors-Systemprozedur

Gibt die Liste der von der aktuellen Verbindung verwendeten Cursor zurück.

Syntax

sa_list_cursors()

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
handle	UNSIGNED INTEGER	Ein eindeutiges Handle, das den Cursor kennzeichnet.
scope	INTEGER	Der Bereich des Aufruf-Stacks, in dem der Cursor geöffnet ist.
cursor_name	VARCHAR(128)	Der Cursorname.
is_open	BIT	Der Indikator, ob der Cursor aktuell geöffnet ist (1).
is_pinned	BIT	Die Indikator, ob der Cursor derzeit im Speicher fixiert ist (1), weil er erneut verwendet werden soll.
fetch_count	UNSIGNED BIGINT	Die Anzahl der Zeilen, die aus dem Cursor abgerufen wurden.

Bemerkungen

Die sa_list_cursors-Systemprozedur kann in einer CALL-Anweisung oder in der FROM-Klausel einer SELECT-Anweisung verwendet werden.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- [„sa_copy_cursor_to_temp_table-Systemprozedur“](#) auf Seite 1193
- [„sa_describe_cursor-Systemprozedur“](#) auf Seite 1203

Beispiel

Das folgende Beispiel gibt eine Liste der offenen Cursor für die Verbindung zurück:

```
CALL sa_list_cursors();
```

sa_load_cost_model-Systemprozedur

Ersetzt das aktuelle Kostenmodell durch das in der angegebenen Datei gespeicherte Kostenmodell

Syntax

```
sa_load_cost_model( file_name )
```

Argumente

- **file_name** Verwenden Sie diesen CHAR(1024)-Parameter, um den Namen der zu ladenden Kostenmodelldatei anzugeben.

Bemerkungen

Der Optimierer verwendet Kostenmodelle, um optimale Zugriffspläne für Abfragen zu ermitteln. Der Datenbankserver hält ein Kostenmodell für jede Datenbank vor. Das Kostenmodell für eine Datenbank kann jederzeit neu kalibriert werden, indem die CALIBRATE SERVER-Klausel der ALTER DATABASE-Anweisung verwendet wird. Sie könnten zum Beispiel das Kostenmodell neu kalibrieren, wenn Sie die Datenbank auf eine Nicht-Standard-Hardware verschieben.

Mit der sa_load_cost_model-Systemprozedur können Sie ein Kostenmodell laden, das in einer Datei (*file_name*) gespeichert wurde. Das Laden eines Kostenmodells ersetzt das aktuelle Kostenmodell für die Datenbank.

Hinweis

Die sa_unload_cost_model-Systemprozedur bezieht keine CALIBRATE PARALLEL READ-Informationen in die Datei ein, die sa_load_cost_model lädt.

Die Verwendung der sa_load_cost_model-Systemprozedur kann wiederholte, zeitraubende Neu-Kalibrierungen unnötig machen, wenn es eine große Anzahl identischer Hardware-Installationen gibt.

Die exklusive Verwendung der Datenbank ist erforderlich, wenn Sie das neue Kostenmodell laden.

Wenn Sie ein Kostenmodell laden, bedenken Sie, ob es für eine Datenbank erstellt wurde, die sich auf einer ähnlichen Hardware befindet. Das Laden eines Kostenmodells von einer Datenbank, die auf einer signifikant unterschiedlichen Hardware gespeichert ist, kann zu verminderter Performance aufgrund von ineffizienten Zugriffsplänen führen.

Kostenmodelle werden unter Verwendung der sa_unload_cost_model-Systemprozedur in einer Datei gespeichert.

Privilegien

Sie müssen die Systemprivilegien ALTER DATABASE und LOAD ANY TABLE haben.

Nebenwirkungen

Der Datenbankserver führt nach dem Laden des neuen Kostenmodells ein COMMIT aus.

Siehe auch

- „ALTER DATABASE-Anweisung“ auf Seite 451
- „sa_unload_cost_model-Systemprozedur“ auf Seite 1348
- „Fortgeschrittene Aufgaben: Abfrageoptimierung“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Beispiel

Das folgende Beispiel lädt das Kostenmodell von einer Datei namens 'costmodel8':

```
CALL sa_load_cost_model( 'costmodel8' );
```

sa_locks-Systemprozedur

Zeigt alle Sperren in der Datenbank an

Syntax

```
sa_locks(  
  [ connection  
  [, creator  
  [, table_name  
  [, max_locks ] ] ] ]  
)
```

Argumente

- **connection** Verwenden Sie diesen INTEGER-Parameter, um eine Verbindungs-ID-Nummer anzugeben. Die Prozedur gibt nur Sperreninformationen über die angegebene Verbindung zurück. Der Standardwert ist 0 (oder NULL). In diesem Fall werden Informationen über alle Verbindungen zurückgegeben.
- **creator** Verwenden Sie diesen CHAR(128)-Parameter, um eine Benutzer-ID anzugeben. Die Prozedur gibt nur Informationen zu Tabellen zurück, deren Eigentümer der angegebene Benutzer ist. Der Standardwert für den Ersteller-Parameter ist NULL. Wenn dieser Parameter auf NULL gesetzt ist, gibt sa_locks die folgenden Informationen zurück:
 - Wenn kein Tabellennamen-Parameter angegeben wird, werden Informationen über Sperren für alle Tabellen in der Datenbank zurückgegeben.
 - Wenn der Tabellennamen-Parameter angegeben ist, werden Informationen über Sperren für Tabellen mit dem angegebenen Namen zurückgegeben, die vom aktuellen Benutzer erstellt wurden.
- **table_name** Verwenden Sie diesen CHAR(128)-Parameter, um einen Tabellennamen anzugeben. Die Prozedur gibt nur Informationen über die angegebenen Tabellen zurück. Der Standardwert ist NULL. In diesem Fall werden Informationen über alle Tabellen zurückgegeben.
- **max_locks** Verwenden Sie diesen INTEGER-Parameter, um die maximale Anzahl der Sperren anzugeben, für die Informationen zurückgegeben werden sollen. Der Standardwert ist "1000". Der Wert -1 gibt an, dass alle Sperrendaten zurückgegeben werden.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
conn_name	VARCHAR(128)	Der Name der aktuellen Verbindung
conn_id	INTEGER	Die Verbindungs-ID-Nummer.
user_id	CHAR(128)	Die Benutzer-ID für die Verbindung
table_type	CHAR(6)	Der Typ der Tabelle. Dieser Typ ist BASE für eine Tabelle, GLBTMP für eine globale temporäre Tabelle oder MVIEW für eine materialisierte Ansicht.
creator	VARCHAR(128)	Eigentümer der Tabelle
table_name	VARCHAR(128)	Die Tabelle, die gesperrt ist
index_id	INTEGER	Die Index-ID oder NULL
lock_class	CHAR(8)	Die Sperrenklasse. Kann "Schema", "Zeile", "Tabelle" oder "Position" sein.
lock_duration	CHAR(11)	Die Dauer der Sperre. Kann "Transaktion", "Position" oder "Verbindung" sein
lock_type	CHAR(9)	Der Sperrentyp (hängt von der Sperrenklasse ab)
row_identifizier	UNSIGNED BIGINT	Der Bezeichner für die Zeile. Kann ein 8-Byte-Zeilenbezeichner oder NULL sein

Bemerkungen

Die Prozedur sa_locks gibt eine Ergebnismenge zurück, die Informationen über alle Sperren in der Datenbank enthält.

Der Wert der lock_type-Spalte hängt von der Sperrenklassifizierung in der lock_class-Spalte ab. Die folgenden Werte können zurückgegeben werden:

Sperrenklasse	Sperrentypen	Kommentare
Schema	Gemeinsam genutzt (gemeinsam genutzte Schemasperre)	Bei Schemasperren sind die Werte von row_identifizier- und Index-ID NULL.
	Exklusiv (exklusive Schemasperre)	

Sperrenklasse	Sperrentypen	Kommentare
Zeile	Read (Lesesperre) Absicht (Absichtssperre) LesenPS (Lesesperre) Schreiben (Schreibsperre) SchreibenKeinPS (Schreibsperre) Platzhalter (Platzhaltersperre)	<p>Zeilen-Lesesperren können kurzfristige Sperren (Scans auf Isolationsstufe 1) oder langfristige Sperren auf höheren Isolationsstufen sein. Die lock_duration-Spalte gibt an, ob die Lesesperre aufgrund der Cursorstabilität (Position) von kurzer Dauer ist, oder von langer Dauer, d.h. bis zum COMMIT/ROLLBACK (Transaktion) aufrechterhalten wird. Zeilensperren werden immer auf einer bestimmten Zeile aufrechterhalten, deren 8-Byte-Zeilenbezeichner ein 64-Bit-Ganzzahlwert in der row_identifizier-Spalte ist. Eine Platzhaltersperre ist ein Spezialfall einer Zeilensperre. Platzhaltersperren werden auf Platzhaltereinträgen gehalten, die erstellt werden, wenn sich die Prüfung der referenziellen Integrität verzögert.</p> <p>Es gibt keine eindeutige Platzhaltersperre für jeden in der Tabelle erstellten Platzhaltereintrag. Stattdessen entspricht eine Platzhaltersperre der Reihe von Platzhaltereinträgen, die für eine gegebene Tabelle von einer gegebenen Verbindung erstellt werden. Der row_identifizier-Wert ist eindeutig für die Tabelle und die Verbindung, die mit der Platzhaltersperre verknüpft sind.</p> <p>Wenn erforderlich, können Schlüssel- und Nicht-Schlüsselteile einer Zeile unabhängig voneinander gesperrt werden. Eine Verbindung kann eine Lesesperre auf den Schlüsselteil einer Zeile für den gemeinsam genutzten (Lese-)Zugriff setzen, damit andere Verbindungen weiterhin Schreibsperren für andere Nicht-Schlüsselspalten einer Zeile erhalten können. Das Aktualisieren von Nicht-Schlüsselspalten einer Zeile beeinträchtigt nicht das Einfügen und Löschen von Fremdzeilen, die auf diese Zeile verweisen.</p>
Tabelle	Gemeinsam genutzt (gemeinsam genutzte Tabellensperre) Absicht (Absicht, die Tabellensperre zu aktualisieren) Exklusiv (exklusive Tabellensperre)	<p>Siehe „Tabellensperren“ [SQL Anywhere Server - SQL-Benutzerhandbuch].</p>

Sperrenklasse	Sperrentypen	Kommentare
Position	Phantom (Phantomsperr) Einfügen (Einfügesperre)	Normalerweise wird auch eine Positionssperre auf einer bestimmten Zeile gehalten und der 64-Bit-Zeilenbezeichner dieser Zeile wird in der row_identifizier-Spalte in der Ergebnismenge angezeigt. Positionssperren können allerdings auch auf ganzen Scanvorgängen (Index oder sequenziell) aufrecht erhalten werden. In diesem Fall ist die row_identifizier-Spalte NULL.

Eine Positionssperre kann mit einem sequenziellen Table Scan oder einem Index Scan verknüpft sein. Die index_id-Spalte zeigt an, ob die Positionssperre mit einem sequenziellen Scan verbunden ist. Wenn die Positionssperre aufgrund eines sequenziellen Scans aufrecht erhalten wird, ist die index_id-Spalte NULL. Wenn die Positionssperre aufgrund eines spezifischen Index Scans aufrecht erhalten wird, wird der Indexbezeichner dieses Indexes in der index_id-Spalte aufgelistet. Der Indexbezeichner entspricht dem Primärschlüssel in der ISYSIDX-Systemtabelle, die unter Verwendung der SYSIDX-Ansicht angezeigt werden kann. Wenn die Positionssperre für Scans auf allen Indizes aufrecht erhalten wird, ist der Index-ID-Wert "-1".

Privilegien

Sie müssen das MONITOR-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Funktionsweise von Sperren“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „SYSIDX-Systemansicht“ auf Seite 1456
- „Sperrentypen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Schemasperren“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Sperren bei Einfügungen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Zeilen Sperren“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Positionssperren“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Beispiel

Sie können die folgende Abfrage ausführen, um Sperren zu identifizieren.

```
CALL sa_locks( );
```

Ein weiteres Beispiel für diese Systemprozedur und Tipps zum Vergrößern der Informationsmenge, die Sie zurückgeben können, finden Sie unter „So erhalten Sie Informationen über Sperren“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)].

sa_make_object-Systemprozedur

Gewährleistet, dass eine Instanzstruktur eines Objekts existiert, bevor eine ALTER-Anweisung durchgeführt wird

Syntax

```
sa_make_object(  
  objtype  
  , objname  
  [ , owner  
  [ , tabname ] ]  
)
```

```
objtype:  
'procedure'  
'function'  
'view'  
'trigger'  
'service'  
'event'
```

Argumente

- **objtype** Verwenden Sie diesen CHAR(30)-Parameter, um den Typ des zu erstellenden Objekts anzugeben. Wenn der Objekttyp **'trigger'** ist, gibt dieses Argument den Eigentümer der Tabelle an, für die der Trigger erstellt werden soll.
- **objname** Verwenden Sie diesen CHAR(128)-Parameter, um den Namen des zu erstellenden Objekts anzugeben.
- **owner** Verwenden Sie diesen CHAR(128)-Parameter, um den Eigentümer des zu erstellenden Objekts anzugeben. Der Standardwert ist NULL.
- **tabname** Dieser CHAR(128)-Parameter ist nur erforderlich, wenn der Objekttyp **'trigger'** ist. In diesem Fall geben Sie damit den Namen der Tabelle an, für die der Trigger erstellt werden soll. Der Standardwert ist NULL.

Bemerkungen

Diese Prozedur ist in Skripten sinnvoll, die wiederholt ausgeführt werden, um das Datenbankschema zu erstellen oder zu ändern. Ein allgemeines Problem dieser Skripten besteht darin, dass bei der ersten Ausführung eine CREATE-Anweisung ausgeführt werden muss, bei jedem weiteren Ausführen jedoch eine ALTER-Anweisung. Diese Prozedur vermeidet die Notwendigkeit des Durchsuchens der Systemansichten, um herauszufinden, ob dieses Objekt existiert.

Bei Prozeduren, Funktionen, Ansichten und Triggern können Sie nun die OR REPLACE-Klausel statt dieser Systemprozedur verwenden.

Um diese Prozedur zu verwenden, stellen Sie ihr eine ALTER-Anweisung nach, die die gesamte Objektbeschreibung beinhaltet.

Privilegien

Sie müssen die erforderlichen Privilegien wie folgt haben:

- **Prozeduren oder Funktionen, deren Eigentümer der Aufrufer ist** CREATE PROCEDURE-Systemprivileg, CREATE ANY PROCEDURE-Systemprivileg oder CREATE ANY OBJECT-Systemprivileg
- **Prozeduren oder Funktionen, deren Eigentümer andere Benutzer sind** CREATE ANY PROCEDURE-Systemprivileg oder CREATE ANY OBJECT-Systemprivileg
- **Dienste** MANAGE ANY WEB SERVICE-Systemprivileg
- **Ereignisse** MANAGE ANY EVENT-Systemprivileg oder CREATE ANY OBJECT-Systemprivileg
- **Ansichten, deren Eigentümer der Aufrufer ist** CREATE VIEW-Systemprivileg, CREATE ANY VIEW-Systemprivileg oder CREATE ANY OBJECT-Systemprivileg
- **Ansichten, deren Eigentümer andere Benutzer sind** CREATE ANY VIEW-Systemprivileg oder CREATE ANY OBJECT-Systemprivileg
- **Trigger** Wenn der Trigger für eine Tabelle gilt, deren Eigentümer Sie sind, müssen Sie entweder das CREATE ANY TRIGGER-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben.

Wenn der Trigger für eine Tabelle gilt, deren Eigentümer ein anderer Benutzer ist, müssen Sie entweder das CREATE ANY TRIGGER-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben. Außerdem müssen Sie eines der folgenden Rechte haben:

- ALTER ANY TABLE-Privileg
- ALTER ANY OBJECT-Systemprivileg
- ALTER-Berechtigung für die Tabelle, für die der Trigger erstellt wird

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „ALTER EVENT-Anweisung“ auf Seite 461
- „ALTER FUNCTION-Anweisung“ auf Seite 465
- „ALTER PROCEDURE-Anweisung“ auf Seite 483
- „ALTER SERVICE-Anweisung [HTTP-Webdienst]“ auf Seite 494
- „ALTER SERVICE-Anweisung [SOAP-Webdienst]“ auf Seite 500
- „ALTER TRIGGER-Anweisung“ auf Seite 539
- „ALTER VIEW-Anweisung“ auf Seite 543

Beispiele

Die folgenden Anweisungen stellen sicher, dass die Grundstruktur einer Prozedurdefinition erstellt wird, definieren die Prozedur und erteilen Privilegien für sie. Eine Skriptdatei, die diese Instruktionen enthält, kann mehrfach auf einer Datenbank ausgeführt werden, ohne dabei einen Fehler hervorzurufen.

```
CALL sa_make_object( 'procedure', 'myproc' );
ALTER PROCEDURE myproc( in p1 INT, in p2 CHAR(30) )
BEGIN
    // ...
END;
GRANT EXECUTE ON myproc TO public;
```

Im folgenden Beispiel wird die sa_make_object-Systemprozedur verwendet, um die Grundstruktur eines Webdienstes hinzuzufügen.

```
CALL sa_make_object( 'service', 'my_web_service' );
```

sa_materialized_view_can_be_immediate-Systemprozedur

Gibt zurück, ob die angegebene materialisierte Ansicht als Sofortansicht definiert werden kann.

Syntax

```
sa_materialized_view_can_be_immediate(
    view_name
    , owner_name
)
```

Argumente

- **view_name** Verwenden Sie diesen CHAR(128)-Parameter, um den Namen der materialisierten Ansicht anzugeben. Wenn view_name NULL ist, wird ein Fehler zurückgegeben.
- **owner_name** Verwenden Sie diesen CHAR(128)-Parameter, um den Eigentümer der materialisierten Ansicht anzugeben. Wenn owner_name NULL ist, wird die aktuelle Benutzer-ID verwendet.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
SQLStateVal	CHAR(6)	Der zurückgegebene SQLSTATE.
ErrorMessage	LONG VARCHAR	Die Fehlermeldung, die dem SQLSTATE entspricht.

Bemerkungen

Ob eine angegebene manuelle Ansicht in eine Sofortansicht geändert werden kann, unterliegt gewissen Einschränkungen. Verwenden Sie diese Systemprozedur, um festzulegen, ob diese Änderung zulässig ist.

Jede Zeile in der Ergebnismenge entspricht einem bestimmten, für eine Ansicht zurückgegebenen SQLSTATE. Daher gilt: Wenn die Definition der materialisierten Ansicht mehr als eine Einschränkung verletzt, enthalten die Ergebnisse mehrere Zeilen für die Ansicht.

Sie können die Ausgabe dieser Systemprozedur mit der Ausgabe der sa_materialized_view_info-Systemprozedur kombinieren, um Informationen über den Status einer Ansicht und ihre Eignung als Sofortansicht zu erhalten.

Privilegien

Sie müssen entweder Eigentümer der materialisierten Ansicht sein oder das ALTER ANY MATERIALIZED VIEW-Systemprivileg oder das ALTER ANY OBJECT-Systemprivileg haben.

Siehe auch

- „Fortgeschrittene Aufgaben: Aktualisierungstyp für eine materialisierte Ansicht ändern“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- Einschränkungen beim Ändern des Typs einer materialisierten Ansicht von "Manuell" auf "Sofort" [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „sa_materialized_view_info-Systemprozedur“ auf Seite 1258

Nebenwirkungen

Alle Metadaten für die angegebene materialisierte Ansicht sowie alle Abhängigkeiten werden in den Servercache geladen.

Beispiel

Führen Sie die folgenden Anweisungen aus, um eine manuelle Ansicht namens view10 zu erstellen und aktualisieren Sie sie.

```
CREATE MATERIALIZED VIEW view10
AS (SELECT C.ID, C.Surname, sum(P.UnitPrice) as revenue, C.CompanyName,
SO.OrderDate
FROM Customers C, SalesOrders SO, SalesOrderItems SOI, Products P
WHERE C.ID = SO.CustomerID
AND SO.ID = SOI.ID
AND P.ID = SOI.ProductID
GROUP BY C.ID, C.Surname, C.CompanyName, SO.OrderDate);

REFRESH MATERIALIZED VIEW view10;
```

Benutzen Sie die folgende Abfrage, um die Ursache zu ermitteln, warum view10 nicht in eine Sofortansicht verwandelt werden kann:

```
SELECT SQLStateVal AS "SQLstate", ErrorMessage AS Description
FROM sa_materialized_view_can_be_immediate( 'view10', NULL )
ORDER BY SQLSTATE;
```

SQLstate	Description
42WC3	Siehe „Die materialisierte Ansicht %1 kann nicht in eine Sofortansicht geändert werden, da sie bereits initialisiert wurde“ [<i>Fehlermeldungen</i>].
42WCA	Siehe „Die materialisierte Ansicht %1 kann nicht in eine Sofortansicht geändert werden, da sie keinen eindeutigen Index für nicht-nullwertfähige Spalten enthält“ [<i>Fehlermeldungen</i>].
42WC6	Siehe „Die materialisierte Ansicht kann nicht in eine Sofortansicht geändert werden, da COUNT(*) Teil der SELECT-Liste sein muss“ [<i>Fehlermeldungen</i>].

SQLstate	Description
42WC7	Siehe „Die materialisierte Ansicht kann nicht in eine Sofortansicht geändert werden, da sie keinen eindeutigen Index für nicht nullwertfähige Nicht-Aggregatspalten enthält“ [<i>Fehlermeldungen</i>].

sa_materialized_view_info-Systemprozedur

Gibt Informationen über die angegebenen materialisierten Ansichten zurück

Syntax

```
sa_materialized_view_info(  
  [ view_name  
  [, owner_name ] ]  
)
```

Argumente

- **view_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Namen der materialisierten Ansicht anzugeben, für die Informationen zurückgegeben werden sollen. Der Standardwert ist NULL.
- **owner_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümer der materialisierten Ansicht anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
OwnerName	CHAR(128)	Der Eigentümer der Ansicht.
ViewName	CHAR(128)	Der Name der Ansicht
Status	CHAR(1)	Statusinformationen über die Ansicht. Die möglichen Werte sind: <ul style="list-style-type: none">• D deaktiviert (disabled)• E aktiviert (enabled)

Spaltenname	Datentyp	Beschreibung
DataStatus	CHAR(1)	<p>Statusinformationen über Daten in der Ansicht. Die möglichen Werte sind:</p> <ul style="list-style-type: none"> • E Ein Fehler ist während der letzten Aktualisierung aufgetreten. Die Ansicht ist aktiviert, aber nicht initialisiert. • F Die Basistabellen wurden seit der letzten Aktualisierung nicht geändert und die Ansicht wird als aktuell angesehen. Die Ansicht ist aktiviert und initialisiert. • N Die Ansicht ist nicht initialisiert. Dies tritt auf, wenn eine der folgenden Situationen zutrifft: <ul style="list-style-type: none"> ○ Die Ansicht wurde seit ihrer Erstellung nicht aktualisiert. ○ Die Daten wurden in der Ansicht gekürzt. ○ Die Ansicht ist deaktiviert. • S Eine Basistabelle wurde seit der letzten Aktualisierung geändert und die Ansicht wird als veraltet eingestuft. Die Ansicht ist aktiviert und initialisiert.
ViewLastRefreshed	TIME-STAMP	Der Zeitpunkt in Ortszeit, zu dem die Ansicht zuletzt aktualisiert wurde. Wenn der Wert von ViewLastRefreshed gleich NULL ist, wird die Ansicht deinitialisiert.
DataLastModified	TIME-STAMP	<p>Bei einer veralteten Ansicht der letzte Zeitpunkt in Ortszeit, zu dem die zugrunde liegenden Daten geändert wurden.</p> <p>Der Wert ist NULL für Ansichten, die nicht initialisiert wurden, oder für Ansichten, die nicht als veraltet eingestuft werden.</p>

Spaltenname	Datentyp	Beschreibung
AvailForOptimization	CHAR(1)	<p>Informationen über die Verfügbarkeit der Ansicht für die Verwendung durch den Optimierer. Die möglichen Werte sind:</p> <ul style="list-style-type: none"> • D Die Verwendung durch den Optimierer ist deaktiviert. Der Eigentümer der Ansicht lässt nicht zu, dass die Ansicht vom Optimierer benutzt wird. • I Die Ansicht kann vom Optimierer aus internen Gründen nicht benutzt werden, zum Beispiel weil ihre Definition die vorgegebenen Bedingungen nicht erfüllt. Der Eigentümer hat aber die Verwendung durch den Optimierer nicht explizit verboten. • N Die Ansicht enthält keine Daten, da keine Aktualisierung durchgeführt wurde oder sie fehlgeschlagen ist. Die Ansicht kann vom Optimierer verwendet werden, aber sie ist nicht initialisiert. • O Ein mit der aktuellen Verbindung nicht kompatibler Optionswert ist vorhanden. Die Ansicht kann vom Optimierer verwendet werden und ihre Definition erfüllt alle Bedingungen, aber die aktuellen Optionseinstellungen sind mit den Optionseinstellungen nicht kompatibel, die für die Erstellung der Ansicht verwendet wurden. • J Die Ansicht kann vom Optimierer verwendet werden. Der Eigentümer der Ansicht hat erlaubt, dass die Ansicht vom Optimierer verwendet wird und die Ansichtsdefinition erfüllt alle Bedingungen für die Verwendung durch den Optimierer.
RefreshType	CHAR(1)	<p>Der Aktualisierungstyp für die Ansicht. Die möglichen Werte sind:</p> <ul style="list-style-type: none"> • I Die Ansicht ist eine Sofortansicht. Sofortansichten werden sofort aktualisiert, wenn Änderungen an Daten in der Basistabelle die Daten in der materialisierten Ansicht betreffen. • M Die Ansicht ist eine manuelle Ansicht. Manuelle Ansichten werden manuell aktualisiert, z.B. mit der REFRESH MATERIALIZED VIEW-Anweisung oder der sa_refresh_materialized_views-Systemprozedur.

Bemerkungen

Wenn weder *view_name* noch *owner_name* angegeben wurde oder beide NULL sind, werden Informationen zu allen materialisierten Ansichten in der Datenbank zurückgegeben.

Wenn der *owner_name* nicht angegeben wird oder NULL ist, werden Informationen zu allen materialisierten Ansichten namens *view_name* zurückgegeben.

Diese Prozedur kann nützlich sein, um die Liste der materialisierten Ansichten zu erhalten, die nie vom Optimierer in Betracht gezogen werden, weil es ein Problem bei der Ansichtsdefinition gibt. Der Wert von AvailForOptimization ist bei diesen materialisierten Ansichten **I**.

Die folgende Tabelle zeigt, wie die AvailForOptimization-Eigenschaft festgelegt wird. Beginnend mit der linken Spalte lesen Sie quer über die Zeile, um die Bedingungen zu ermitteln, die vorliegen müssen, damit als Ergebnis der Wert gezeigt wird, der in der AvailForOptimization-Spalte steht.

Erlaubt der Benutzer die Verwendung der Ansicht in der Optimierung?	Erfüllt die Ansichtsdefinition alle Bedingungen für die Verwendung?	Entsprechen die Verbindungsoptionen denjenigen, die für die Verwendung der Ansicht gefordert werden?	Ist die Ansicht initialisiert?	AvailForOptimization-Wert
Ja	Ja	Ja	Ja	J
Nein	k.A.	k.A.	k.A.	D
Ja	Nein	k.A.	Ja	I
Ja	k.A.	k.A.	Nein	N
Ja	Ja	Nein	Ja	O

Eine initialisierte materialisierte Ansicht kann leer sein. Dies kommt vor, wenn in den Basistabellen keine Daten vorhanden sind, die zur Definition der materialisierten Ansicht passen. Eine leere Ansicht wird nicht als identisch mit einer nicht initialisierten materialisierten Ansicht angesehen, die ebenfalls keine Daten hat. Mit dem Wert der Eigenschaft ViewLastRefreshed können Sie ermitteln, ob die Ansicht nicht initialisiert (NULL) oder wegen nicht passender Daten in der Basistabelle leer ist (Nicht-Null).

Privilegien

Keine

Nebenwirkungen

Alle Metadaten für die angegebenen materialisierten Ansichten sowie alle Abhängigkeiten werden in den Cache des Datenbankservers geladen.

Siehe auch

- „Materialisierte Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Fortgeschrittene Aufgaben: Aktualisierungstyp für eine materialisierte Ansicht ändern“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- Einschränkungen beim Ändern des Typs einer materialisierten Ansicht von "Manuell" auf "Sofort" [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „sa_materialized_view_can_be_immediate-Systemprozedur“ auf Seite 1256
- „Ermitteln, welche materialisierten Ansichten vom Optimierer in Betracht gezogen wurden“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Einschränkungen für materialisierte Ansichten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Einstellen des Aktualisierungstyps auf "Manuell" oder "Sofort"“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Beispiel

Die folgende Anweisung gibt Informationen über alle materialisierten Ansichten in der Datenbank zurück:

```
SELECT *  
FROM sa_materialized_view_info();
```

Die Ergebnisse der sa_materialized_view_info-Systemprozedur können mit den Ergebnissen der sa_materialized_view_can_be_immediate-Systemprozedur kombiniert werden, um Statusinformationen darüber zurückzugeben, ob die Ansicht als Sofortansicht geeignet ist. Führen Sie die nachstehenden Anweisungen aus, um materialisierte Ansichten zu erstellen, die für dieses Beispiel geprüft werden:

```
CREATE MATERIALIZED VIEW view0 AS (  
    SELECT ID, Name, Description, Size  
    FROM Products  
    WHERE Quantity > 0 );  
  
CREATE UNIQUE INDEX u_view0  
ON view0( ID );  
  
ALTER MATERIALIZED VIEW view0  
IMMEDIATE REFRESH;  
  
CREATE MATERIALIZED VIEW view00 AS (  
    SELECT ID, Name, Description, Size  
    FROM Products  
    WHERE Quantity <= 0 );  
  
CREATE UNIQUE INDEX u_view00  
ON view00( ID );  
  
CREATE MATERIALIZED VIEW view1 AS (  
    SELECT ID, Name, Description, Size  
    FROM Products  
    WHERE Quantity = 0 );  
  
ALTER MATERIALIZED VIEW view1  
DISABLE;  
  
CREATE MATERIALIZED VIEW view100  
AS (SELECT C.ID, C.Surname, sum(P.UnitPrice) as revenue, C.CompanyName,  
    SO.OrderDate  
    FROM Customers C, SalesOrders SO, SalesOrderItems SOI, Products P
```

```

WHERE C.ID = SO.CustomerID
AND SO.ID = SOI.ID
AND P.ID = SOI.ProductID
GROUP BY C.ID, C.Surname, C.CompanyName, SO.OrderDate);

```

```

REFRESH MATERIALIZED VIEW view100;

```

Führen Sie die folgende Anweisung aus, um den Status und die Verwendbarkeitsinformationen für die Ansichten zurückzugeben, deren Eigentümer Sie sind:

```

SELECT ViewName, Status, ViewLastRefreshed, AvailForOptimization,
RefreshType, CanBeImmediate
FROM sa_materialized_view_info() AS V,
     LATERAL( SELECT LIST( ErrorMessage, ' ' )
               FROM sa_materialized_view_can_be_immediate( V.ViewName,
V.OwnerName ) ) AS I( CanBeImmediate )
WHERE OwnerName = USER_NAME();

```

ViewName	Status	ViewLastRefreshed	AvailForOptimization	RefreshType	CanBelmmmediate
view0	E	(NULL)	N	I	
view00	E	(NULL)	N	M	
view1	D	(NULL)	N	M	Cannot use view 'view1' because it has been disabled

ViewName	Status	ViewLastRefreshed	AvailForOptimization	RefreshType	CanBelmmediate
view100	E	2008-02-12 16:47:00.000	Y	M	The materi- alized view view100 can- not be chan- ged to imme- diate becau- se it has already been initialized. The materi- alized view view100 can- not be chan- ged to imme- diate becau- se it does not have a unique index on non-nul- lable columns. The materialized view cannot be changed to immediate because COUNT(*) is required to be part of the SELECT list. The materialized view cannot be changed to immediate because it does not have a unique index on non- aggregate non-nullable columns.

Aus den Ergebnissen können Sie folgende Informationen ablesen:

- view0 wurde nie aktualisiert und ist eine Sofortansicht.
- view00 wurde nie aktualisiert und ist eine manuelle Ansicht.
- view1 ist deaktiviert.
- view100 ist eine manuelle Ansicht, die zuletzt am 2008-02-12 um 16:47:00.000 aktualisiert wurde.
- view00 kann in eine Sofortansicht umgewandelt werden, weil in der Spalte CanBeImmediate keine Fehlermeldungen stehen.
- view1 und view100 können aus den Gründen, die in der CanBeImmediate-Spalte stehen, nicht in Sofortansichten umgewandelt werden.

sa_migrate-Systemprozedur

Diese Systemprozedur migriert eine Gruppe entfernter Tabellen in eine SQL Anywhere-Datenbank.

Syntax

```
sa_migrate(  
  base_table_owner  
  , server_name  
  [, table_name  
  [, owner_name  
  [, database_name  
  [, migrate_data  
  [, drop_proxy_tables  
  [, migrate_fkeys ]]]]]  
)
```

Argumente

- **base_table_owner** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Benutzer anzugeben, dem die migrierten Tabellen in der SQL Anywhere-Zioldatenbank gehören. Erstellen Sie diesen Benutzer mit der Anweisung GRANT CONNECT. Für diesen Parameter muss ein Wert angegeben werden.
- **server_name** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Namen des entfernten Servers anzugeben, der für die Verbindung zur entfernten Datenbank verwendet wird. Benutzen Sie die Anweisung CREATE SERVER, um diesen Server zu erstellen. Für diesen Parameter muss ein Wert angegeben werden.
- **table_name** Wenn Sie eine Einzeltabelle migrieren möchten, geben Sie mit diesem optionalen VARCHAR(128)-Parameter den Tabellennamen an. Andernfalls sollten Sie diesen Parameter auf NULL (Standardeinstellung) setzen. Geben Sie nicht sowohl für den Parameter table_name als auch für den Parameter owner_name NULL an.
- **owner_name** Wenn Sie nur Tabellen migrieren, die einem Eigentümer gehören, geben Sie mit diesem optionalen VARCHAR(128)-Parameter den Eigentümernamen an. Andernfalls sollten Sie diesen Parameter auf NULL (Standardeinstellung) setzen. Geben Sie nicht sowohl für den Parameter table_name als auch für den Parameter owner_name NULL an.

- **database_name** Verwenden Sie diesen optionalen VARCHAR(128)-Parameter, um den Namen der entfernten Datenbank anzugeben. Sie müssen den Datenbanknamen angeben, wenn Sie Tabellen nur aus einer Datenbank auf dem Fremdserver migrieren möchten. Andernfalls setzen Sie diesen Parameter auf NULL (Standardeinstellung).
- **migrate_data** Mit diesem optionalen BIT-Parameter können Sie angeben, ob die Daten in den entfernten Tabellen migriert werden sollen. Dieser Parameterwert kann "0" (Daten nicht migrieren) oder "1" (Daten migrieren) sein. Standardmäßig werden Daten migriert (1).
- **drop_proxy_tables** Mit diesem optionalen BIT-Parameter können Sie angeben, ob die für den Migrationsprozess erstellten Proxy-Tabellen nach Abschluss der Migration gelöscht werden sollen. Dieser Parameterwert kann "0" (Proxytabellen werden nicht gelöscht) oder "1" (Proxytabellen werden gelöscht) sein. Standardmäßig werden die Proxy-Tabellen gelöscht (1).
- **migrate_fkeys** Mit diesem optionalen BIT-Parameter können Sie angeben, ob Fremdschlüssel-Zuordnungen migriert werden sollen. Dieser Parameterwert kann "0" sein (Fremdschlüssel-Zuordnungen nicht migrieren), oder "1" (Fremdschlüssel-Zuordnungen migrieren). Standardmäßig werden die Fremdschlüsselzuordnungen migriert (1).

Bemerkungen

Mit dieser Prozedur können Sie Tabellen aus einer entfernten Oracle-, IBM DB2-, SQL Server-, Adaptive Server Enterprise- oder SQL Anywhere-Datenbank nach SQL Anywhere migrieren. Die Prozedur ermöglicht Ihnen in einem Schritt die Migration einer Gruppe entfernter Tabellen inklusive deren Fremdschlüsselzuordnung. Die sa_migrate-Systemprozedur ruft die folgenden Systemprozeduren auf:

- sa_migrate_create_remote_table_list
- sa_migrate_create_tables
- sa_migrate_data
- sa_migrate_create_remote_fks_list
- sa_migrate_create_fks
- sa_migrate_drop_proxy_tables

Sie können diese Systemprozeduren anstelle von sa_migrate verwenden, wenn Sie mehr Flexibilität benötigen. Beispielsweise ist es bei Verwendung von sa_migrate nicht möglich, Fremdschlüsselbeziehungen bei der Migration von Tabellen beizubehalten, wenn diese Fremdschlüsselbeziehungen unterschiedlichen Benutzern gehören.

Vor der Migration jeglicher Tabellen müssen Sie erst einen entfernten Server mit der Anweisung CREATE SERVER erstellen und sich mit der entfernten Datenbank verbinden. Sie müssen eventuell auch ein externes Login für die entfernte Datenbank mit der Anweisung CREATE EXTERNLOGIN erstellen.

Sie können alle Tabellen aus der entfernten Datenbank in eine SQL Anywhere-Datenbank migrieren, indem Sie nur die Parameter *base_table_owner* und *server_name* angeben. Bei der Migration mit nur diesen beiden angegebenen Parametern werden jedoch alle migrierten Tabellen in der SQL Anywhere-Datenbank nur einem Eigentümer zugeordnet. Wenn die Tabellen in der entfernten Datenbank unterschiedliche Eigentümer haben und Sie die Tabellen in der SQL Anywhere-Datenbank unterschiedlichen Eigentümern zuordnen wollen, müssen Sie die Tabellen für jeden Eigentümer einzeln migrieren. Verwenden Sie hierzu bei jedem Aufruf der Prozedur sa_migrate die Parameter *base_table_owner* und *owner_name*.

Vorsicht

Geben Sie nicht sowohl für den Parameter *table_name* als auch für den Parameter *owner_name* NULL an. Wenn Sie sowohl für *table_name* als auch für *owner_name* den Wert NULL angeben, werden alle Tabellen in der Datenbank migriert, auch die Systemtabellen. Außerdem werden Tabellen, die den gleichen Namen in der entfernten Datenbank haben, aber unterschiedlichen Eigentümern gehören, alle einem Eigentümer in der Zieldatenbank zugeordnet. Es wird empfohlen, dass Sie nur einem Eigentümer zugeordnete Tabellen auf einmal migrieren.

Privilegien

Sie müssen die folgenden Systemprivilegien haben:

- CREATE TABLE oder CREATE ANY TABLE (sofern Sie nicht Eigentümer der Basistabelle sind)
- SELECT ANY TABLE (sofern Sie nicht Eigentümer der Basistabelle sind)
- INSERT ANY TABLE (sofern Sie nicht Eigentümer der Basistabelle sind)
- ALTER ANY TABLE (sofern Sie nicht Eigentümer der Basistabelle sind)
- CREATE ANY INDEX (sofern Sie nicht Eigentümer der Basistabelle sind)
- DROP ANY TABLE (sofern Sie nicht Eigentümer der Basistabelle sind)

Nebenwirkungen

Keine

Siehe auch

- „Datenbankmigration nach SQL Anywhere“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „CREATE EXTERNLOGIN-Anweisung“ auf Seite 616
- „CREATE SERVER-Anweisung“ auf Seite 701
- „GRANT-Anweisung“ auf Seite 881
- „sa_migrate_create_remote_table_list-Systemprozedur“ auf Seite 1270
- „sa_migrate_create_tables-Systemprozedur“ auf Seite 1272
- „sa_migrate_data-Systemprozedur“ auf Seite 1273
- „sa_migrate_create_remote_fks_list-Systemprozedur“ auf Seite 1269
- „sa_migrate_create_fks-Systemprozedur“ auf Seite 1268
- „sa_migrate_drop_proxy_tables-Systemprozedur“ auf Seite 1274

Beispiele

Die folgende Anweisung migriert alle Tabellen, die dem Benutzer DBA gehören, aus der entfernten Datenbank, einschließlich Fremdschlüsselzuordnungen. Außerdem migriert sie die Daten aus den entfernten Tabellen und löscht nach Abschluss der Migration die Proxy-Tabellen. In diesem Beispiel gehören alle migrierten Tabellen in der SQL Anywhere-Zieldatenbank dem Benutzer LocalUser.

```
CALL sa_migrate( 'LocalUser', 'RemoteSA', NULL, 'DBA', NULL, 1, 1, 1 );
```

Die folgende Anweisung migriert alle Tabellen, die dem Benutzer DBA gehören, aus der entfernten Datenbank. In der SQL Anywhere-Zieldatenbank gehören diese Tabellen dem Benutzer LocalUser.

Während der Migration erstellte Proxy-Tabellen werden nach Abschluss des Vorgangs nicht gelöscht und gehören LocalUser.

```
CALL sa_migrate( 'LocalUser', 'RemoteSA', NULL, 'DBA', NULL, 1, 0, 1 );
```

sa_migrate_create_fks-Systemprozedur

Erstellt Fremdschlüssel für alle Tabellen, die in der Tabelle dbo.migrate_remote_fks_list aufgelistet sind

Syntax

```
sa_migrate_create_fks( i_table_owner )
```

Argumente

- ***i_table_owner*** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Benutzer anzugeben, dem in der SQL Anywhere-Zieldatenbank die migrierten Fremdschlüssel gehören. Wenn Sie Tabellen migrieren möchten, deren Eigentümer verschiedene Benutzer sind, führen Sie diese Prozedur für jeden Benutzer aus, dessen Tabellen für die Migration vorgesehen sind. Der *i_table_owner* wird mit der Anweisung GRANT CONNECT erstellt. Für diesen Parameter muss ein Wert angegeben werden.

Bemerkungen

Diese Prozedur erstellt Fremdschlüssel für jede Tabelle, die in der Tabelle dbo.migrate_remote_fks_list aufgelistet ist. Durch das Argument *i_table_owner* wird der Benutzer definiert, dem die Fremdschlüssel in der Zieldatenbank gehören.

Sollten nicht alle Tabellen in der SQL Anywhere-Zieldatenbank dem gleichen Benutzer gehören, müssen Sie diese Prozedur für jeden Benutzer, der die Tabellen für die Migration der Fremdschlüssel besitzt, erneut ausführen.

Hinweis

Diese Systemprozedur wird zusammen mit einigen anderen Migrations-Systemprozeduren verwendet, die in der unten aufgelisteten Reihenfolge ausgeführt werden müssen:

1. sa_migrate_create_remote_table_list
2. sa_migrate_create_tables
3. sa_migrate_data
4. sa_migrate_create_remote_fks_list
5. sa_migrate_create_fks
6. sa_migrate_drop_proxy_tables

Alternativ können Sie mithilfe der sa_migrate-Systemprozedur alle Tabellen in einem Schritt migrieren.

Privilegien

Sie müssen die folgenden Systemprivilegien haben:

- ALTER ANY TABLE
- CREATE ANY INDEX

Nebenwirkungen

Keine

Siehe auch

- „Datenbankmigration nach SQL Anywhere“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „GRANT-Anweisung“ auf Seite 881
- „sa_migrate-Systemprozedur“ auf Seite 1265
- „sa_migrate_create_remote_table_list-Systemprozedur“ auf Seite 1270
- „sa_migrate_create_tables-Systemprozedur“ auf Seite 1272
- „sa_migrate_data-Systemprozedur“ auf Seite 1273
- „sa_migrate_create_remote_fks_list-Systemprozedur“ auf Seite 1269
- „sa_migrate_drop_proxy_tables-Systemprozedur“ auf Seite 1274

Beispiel

Die erste Anweisung erstellt eine Liste von Fremdschlüsseln für die Tabellen, die in der dbo.migrate_remote_table_list-Tabelle aufgelistet sind. Die zweite Anweisung erstellt Fremdschlüssel auf Basis der dbo.migrate_remote_fks_list-Tabelle. Die Fremdschlüssel gehören zu Tabellen, deren Eigentümer der Benutzer LocalUser in der lokalen SQL Anywhere-Datenbank ist.

```
CALL sa_migrate_create_remote_fks_list( 'RemoteSA' );  
CALL sa_migrate_create_fks( 'LocalUser' );
```

sa_migrate_create_remote_fks_list-Systemprozedur

Füllt die Tabelle dbo.migrate_remote_fks_list

Syntax

```
sa_migrate_create_remote_fks_list( server_name )
```

Argumente

- **server_name** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Namen des entfernten Servers anzugeben, der für die Verbindung zur entfernten Datenbank verwendet wird. Verwenden Sie die Anweisung CREATE SERVER, um den entfernten Server zu erstellen. Für diesen Parameter muss ein Wert angegeben werden.

Bemerkungen

Diese Prozedur füllt die Tabelle dbo.migrate_remote_fks_list mit einer Liste der Fremdschlüssel, die von der entfernten Datenbank migriert werden können. Sie können Zeilen aus dieser Tabelle löschen und damit die entsprechenden Fremdschlüssel von der Migration ausschließen.

Alternativ können Sie mithilfe der sa_migrate-Systemprozedur alle Tabellen in einem Schritt migrieren.

Diese Systemprozedur wird zusammen mit einigen anderen Migrations-Systemprozeduren verwendet. Der Hinweis im Abschnitt "Bemerkungen" zur sa_migrate_create_fks-Systemprozedur enthält eine Liste von Migrationsprozeduren in der Reihenfolge, in der Sie sie ausführen müssen.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „CREATE SERVER-Anweisung“ auf Seite 701
- „Datenbankmigration nach SQL Anywhere“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „sa_migrate-Systemprozedur“ auf Seite 1265
- „sa_migrate_create_remote_table_list-Systemprozedur“ auf Seite 1270
- „sa_migrate_create_tables-Systemprozedur“ auf Seite 1272
- „sa_migrate_data-Systemprozedur“ auf Seite 1273
- „sa_migrate_create_fks-Systemprozedur“ auf Seite 1268
- „sa_migrate_drop_proxy_tables-Systemprozedur“ auf Seite 1274

Beispiel

Die folgende Anweisung erstellt eine Liste von Fremdschlüsseln für die Tabellen, die in der dbo.migrate_remote_table_list-Tabelle aufgelistet sind.

```
CALL sa_migrate_create_remote_fks_list( 'RemoteSA' );
```

sa_migrate_create_remote_table_list-Systemprozedur

Füllt die Tabelle dbo.migrate_remote_table_list

Syntax

```
sa_migrate_create_remote_table_list(  
  i_server_name  
  [, i_table_name  
  [, i_owner_name  
  [, i_database_name ] ] ]  
)
```

Argumente

- **i_server_name** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Namen des entfernten Servers anzugeben, der für die Verbindung zur entfernten Datenbank verwendet wird. Verwenden Sie die Anweisung CREATE SERVER, um den entfernten Server zu erstellen. Für diesen Parameter muss ein Wert angegeben werden.
- **i_table_name** Verwenden Sie diesen VARCHAR(128)-Parameter, um die Namen der Tabellen anzugeben, die Sie migrieren wollen, oder NULL, um alle Tabellen zu migrieren. Der Standardwert

ist NULL. Geben Sie nicht sowohl für den Parameter *i_table_name* als auch für den Parameter *i_owner_name* NULL an.

- ***i_owner_name*** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Benutzer anzugeben, dem die zu migrierenden Tabellen in der entfernten Datenbank gehören, oder NULL, um alle Tabellen zu migrieren. Der Standardwert ist NULL. Geben Sie nicht sowohl für den Parameter *i_table_name* als auch für den Parameter *i_owner_name* NULL an.
- ***i_database_name*** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Namen der entfernten Datenbank anzugeben, von der Sie Tabellen migrieren wollen. Der Standardwert ist NULL. Für die Tabellenmigration von Adaptive Server Enterprise und Microsoft SQL Server-Datenbanken müssen Sie einen Datenbanknamen angeben.

Bemerkungen

Diese Prozedur füllt die Tabelle `dbo.migrate_remote_table_list` mit einer Liste der Tabellen, die von der entfernten Datenbank migriert werden können. Sie können Zeilen aus dieser Tabelle löschen und damit die entsprechenden entfernten Tabellen von der Migration ausschließen.

Wenn Sie verhindern möchten, dass alle migrierten Tabellen demselben Eigentümer in der SQL Anywhere-Zieldatenbank gehören, müssen Sie diese Prozedur für jeden Benutzer, dessen Tabellen Sie migrieren wollen, erneut ausführen.

Alternativ können Sie mithilfe der `sa_migrate`-Systemprozedur alle Tabellen in einem Schritt migrieren.

Vorsicht

Geben Sie nicht sowohl für den Parameter *i_table_name* als auch für den Parameter *i_owner_name* NULL an. Wenn Sie sowohl für *i_table_name* als auch für *i_owner_name* den Wert NULL angeben, werden alle Tabellen in der Datenbank migriert, auch die Systemtabellen. Außerdem werden Tabellen, die den gleichen Namen in der entfernten Datenbank haben, aber unterschiedlichen Eigentümern gehören, alle einem Eigentümer in der Zieldatenbank zugeordnet. Es wird empfohlen, dass Sie nur einem Eigentümer zugeordnete Tabellen auf einmal migrieren.

Diese Systemprozedur wird zusammen mit einigen anderen Migrations-Systemprozeduren verwendet. Der Hinweis im Abschnitt "Bemerkungen" zur `sa_migrate_create_fks`-Systemprozedur enthält eine Liste von Migrationsprozeduren in der Reihenfolge, in der Sie sie ausführen müssen.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „CREATE SERVER-Anweisung“ auf Seite 701
- „Datenbankmigration nach SQL Anywhere“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „sa_migrate-Systemprozedur“ auf Seite 1265
- „sa_migrate_create_tables-Systemprozedur“ auf Seite 1272
- „sa_migrate_data-Systemprozedur“ auf Seite 1273
- „sa_migrate_create_remote_fks_list-Systemprozedur“ auf Seite 1269
- „sa_migrate_create_fks-Systemprozedur“ auf Seite 1268
- „sa_migrate_drop_proxy_tables-Systemprozedur“ auf Seite 1274

Beispiel

Die folgende Anweisung erstellt eine Liste von Tabellen, deren Eigentümer der Benutzer DBA in der entfernten Datenbank ist.

```
CALL sa_migrate_create_remote_table_list( 'RemoteSA', NULL, 'DBA', NULL );
```

sa_migrate_create_tables-Systemprozedur

Erstellt eine Proxy- und Basistabelle für alle Tabellen, die in der Tabelle `dbo.migrate_remote_table_list` aufgelistet sind

Syntax

```
sa_migrate_create_tables( i_table_owner )
```

Argumente

- ***i_table_owner*** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Benutzer anzugeben, dem die migrierten Tabellen in der SQL Anywhere-Zieldatenbank gehören. Dieser Benutzer wird mit der Anweisung GRANT CONNECT erstellt. Für diesen Parameter muss ein Wert angegeben werden.

Bemerkungen

Diese Prozedur erstellt eine Basis- und eine Proxytabelle für alle Tabellen, die in der Tabelle `dbo.migrate_remote_table_list` aufgelistet sind (wird durch die Prozedur `sa_migrate_create_remote_table_list` erstellt). Diese Proxy- und Basistabellen gehören dem Benutzer, der mit dem Argument *i_table_owner* angegeben wird. Weiterhin erstellt diese Prozedur die gleichen Primärschlüssel- und andere Indizes für die neue Tabelle, die den Indizes in der entfernten Tabelle der entfernten Datenbank entsprechen.

Wenn Sie nicht möchten, dass alle migrierten Tabellen einem einzelnen Benutzer der SQL Anywhere-Zieldatenbank gehören, müssen Sie die Prozeduren `sa_migrate_create_remote_table_list` und `sa_migrate_create_tables` für jeden einzelnen Benutzer ausführen, dem migrierte Tabellen gehören sollen.

Alternativ können Sie mithilfe der `sa_migrate`-Systemprozedur alle Tabellen in einem Schritt migrieren.

Diese Systemprozedur wird zusammen mit einigen anderen Migrations-Systemprozeduren verwendet. Der Hinweis im Abschnitt "Bemerkungen" zur `sa_migrate_create_fks`-Systemprozedur enthält eine Liste von Migrationsprozeduren in der Reihenfolge, in der Sie sie ausführen müssen.

Privilegien

Sie müssen das CREATE ANY TABLE-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Datenbankmigration nach SQL Anywhere“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „sa_migrate-Systemprozedur“ auf Seite 1265
- „sa_migrate_create_remote_table_list-Systemprozedur“ auf Seite 1270
- „sa_migrate_data-Systemprozedur“ auf Seite 1273
- „sa_migrate_create_remote_fks_list-Systemprozedur“ auf Seite 1269
- „sa_migrate_create_fks-Systemprozedur“ auf Seite 1268
- „sa_migrate_drop_proxy_tables-Systemprozedur“ auf Seite 1274

Beispiel

Die erste Anweisung erstellt eine Liste von Tabellen, deren Eigentümer der Benutzer DBA in der entfernten Datenbank ist. Die zweite Anweisung erstellt anhand dieser Liste Basistabellen und Proxy-Tabellen in der SQL Anywhere-Zieldatenbank. Eigentümer dieser Tabellen ist der Benutzer LocalUser.

```
CALL sa_migrate_create_remote_table_list( 'RemoteSA', NULL, 'DBA', NULL );  
CALL sa_migrate_create_tables( 'LocalUser' );
```

sa_migrate_data-Systemprozedur

Diese Prozedur migriert Daten von Tabellen der entfernten Datenbank in die SQL Anywhere-Zieldatenbank.

Syntax

```
sa_migrate_data( i_table_owner )
```

Argumente

- ***i_table_owner*** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Benutzer anzugeben, dem die migrierten Tabellen in der SQL Anywhere-Zieldatenbank gehören. Dieser Benutzer wird mit der Anweisung GRANT CONNECT erstellt. Für diesen Parameter muss ein Wert angegeben werden.

Bemerkungen

Diese Prozedur migriert Daten aus der entfernten Datenbank in die SQL Anywhere-Zieldatenbank, und zwar für alle Tabellen, die dem im Argument *i_table_owner* angegebenen Benutzer gehören.

Sollten nicht alle Tabellen in der SQL Anywhere-Zieldatenbank dem gleichen Benutzer gehören, müssen Sie diese Prozedur für jeden Benutzer, der die Tabellen für die zu migrierenden Daten besitzt, erneut ausführen.

Alternativ können Sie mithilfe der sa_migrate-Systemprozedur alle Tabellen in einem Schritt migrieren.

Diese Systemprozedur wird zusammen mit einigen anderen Migrations-Systemprozeduren verwendet. Der Hinweis im Abschnitt "Bemerkungen" zur `sa_migrate_create_fks`-Systemprozedur enthält eine Liste von Migrationsprozeduren in der Reihenfolge, in der Sie sie ausführen müssen.

Privilegien

Sie müssen die folgenden Systemprivilegien haben:

- `SELECT ANY TABLE`
- `INSERT ANY TABLE`

Nebenwirkungen

Keine

Siehe auch

- „Datenbankmigration nach SQL Anywhere“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „`sa_migrate`-Systemprozedur“ auf Seite 1265
- „`sa_migrate_create_remote_table_list`-Systemprozedur“ auf Seite 1270
- „`sa_migrate_create_tables`-Systemprozedur“ auf Seite 1272
- „`sa_migrate_create_remote_fks_list`-Systemprozedur“ auf Seite 1269
- „`sa_migrate_create_fks`-Systemprozedur“ auf Seite 1268
- „`sa_migrate_drop_proxy_tables`-Systemprozedur“ auf Seite 1274

Beispiel

Die folgende Anweisung migriert Daten in die SQL Anywhere-Zieldatenbank für Tabellen, die dem Benutzer `LocalUser` gehören.

```
CALL sa_migrate_data( 'LocalUser' );
```

sa_migrate_drop_proxy_tables-Systemprozedur

Löscht die für Migrationszwecke erstellten Proxytabellen

Syntax

```
sa_migrate_drop_proxy_tables( i_table_owner )
```

Argumente

- ***i_table_owner*** Verwenden Sie diesen `VARCHAR(128)`-Parameter, um den Benutzer anzugeben, dem die Proxytabellen in der SQL Anywhere-Zieldatenbank gehören. Dieser Benutzer wird mit der Anweisung `GRANT CONNECT` erstellt. Für diesen Parameter muss ein Wert angegeben werden.

Bemerkungen

Diese Prozedur löscht die für Migrationszwecke erstellten Proxytabellen. Der Benutzer, dem die Proxytabellen gehören, wird mit dem Argument *i_table_owner* angegeben.

Wenn nicht alle migrierten Tabellen demselben Eigentümer in der SQL Anywhere-Zieldatenbank gehören, müssen Sie diese Prozedur für jeden Benutzer erneut ausführen, um alle Proxytabellen löschen zu können.

Alternativ können Sie mithilfe der sa_migrate-Systemprozedur alle Tabellen in einem Schritt migrieren.

Diese Systemprozedur wird zusammen mit einigen anderen Migrations-Systemprozeduren verwendet. Der Hinweis im Abschnitt "Bemerkungen" zur sa_migrate_create_fks-Systemprozedur enthält eine Liste von Migrationsprozeduren in der Reihenfolge, in der Sie sie ausführen müssen.

Privilegien

Sie müssen das DROP ANY TABLE-Privileg auf Objektebene haben.

Nebenwirkungen

Keine

Siehe auch

- „Datenbankmigration nach SQL Anywhere“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „sa_migrate-Systemprozedur“ auf Seite 1265
- „sa_migrate_create_remote_table_list-Systemprozedur“ auf Seite 1270
- „sa_migrate_create_tables-Systemprozedur“ auf Seite 1272
- „sa_migrate_data-Systemprozedur“ auf Seite 1273
- „sa_migrate_create_remote_fks_list-Systemprozedur“ auf Seite 1269
- „sa_migrate_create_fks-Systemprozedur“ auf Seite 1268

Beispiel

Die folgende Anweisung löscht die Proxy-Tabellen aus der SQL Anywhere-Zieldatenbank, die dem Benutzer LocalUser gehören.

```
CALL sa_migrate_drop_proxy_tables( 'LocalUser' );
```

sa_mirror_server_status-Systemprozedur

Gibt den Verbindungsstatus des aktuellen Server und alle Server zurück, die direkt oder indirekt Logseiten vom aktuellen Server empfangen. Auf dem Primärserver gibt die Prozedur den Status aller verbundenen Server zurück.

Syntax

```
sa_mirror_server_status( )
```

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
server_name	CHAR(128)	Der Name des Servers.

Spaltenname	Datentyp	Beschreibung
state	CHAR(20)	<p>Der Verbindungsstatus des Servers. Dies kann einer der folgenden Werte sein:</p> <ul style="list-style-type: none"> • connected • disconnected <p>Unter bestimmten Umständen lautet der Status eines nicht verbundenen Servers "connected". In einem solchen Fall liegt der durch last_updated angegebene Zeitpunkt in der Vergangenheit.</p>
last_updated	TIMESTAMP WITH TIME ZONE	Der Zeitpunkt, zu dem der Serverstatus zuletzt aktualisiert wurde.
load_current	DOUBLE	Der Umfang der Arbeit, die der Datenbankserver zurzeit ausführt.
load_last_1_min	DOUBLE	Der Umfang der Arbeit, die der Datenbankserver in der letzten Minute ausgeführt hat.
load_last_5_mins	DOUBLE	Der Umfang der Arbeit, die der Datenbankserver in den letzten 5 Minuten ausgeführt hat.
load_last_10_mins	DOUBLE	Der Umfang der Arbeit, die der Datenbankserver in den letzten 10 Minuten ausgeführt hat.
num_connections	UNSIGNED INTEGER	Die Anzahl der Verbindungen zum Datenbankserver.
num_processors	UNSIGNED INTEGER	Die Anzahl der Prozessoren im Datenbankserver.
log_written	UNSIGNED BIGINT	Die letzten auf die Festplatte geschriebenen Transaktionslogpositionen, basierend auf der letzten vom Server empfangenen Aktualisierung.
log_applied	UNSIGNED BIGINT	Der letzte angewendete Vorgang aus dem Transaktionslog, basierend auf der letzten vom Server empfangenen Aktualisierung. Dieser Wert ist derselbe wie der Wert der CurrentRedoPos-Eigenschaft.

Bemerkungen

Jeder Server aktualisiert alle 5 Sekunden seinen Status und den der Kopieknoten. Auf Spiegelservers gibt die Prozedur den Status von Kopieknoten zurück, die Logseiten vom Spiegelserver empfangen, aber nicht den Status der Primärserver. Die Spalten mit dem Präfix **load** stehen für eine berechnete Last auf dem

Datenbankserver. Der zurückgegebene Wert steht für die Datenbankserverlast und nicht für die Last aus anderen Prozessen. Höhere Lastwerte zeigen an, dass der Datenbankserver mehr Arbeit auszuführen hat.

Die Zeit in der last_updated-Spalte ist diejenige des Servers in der server_name-Spalte. Der Zustand ist für die in der last_updated-Spalte zurückgegebene Zeit korrekt.

Wenn der Verbindungsparameter NodeType angegeben ist, verwendet der Datenbankserver Lastinformationen dazu, Verbindungen umzuleiten. Der Datenbankserver wählt den Spiegelserver mit der niedrigsten Last. Wenn die Last auf allen Servern gleich ist, wird der Server mit den wenigsten Verbindungen verwendet.

Hinweis

Bei Datenbankservern in der Cloud verwendet der NodeType-Verbindungsparameter keine Ladeinformationen, um Verbindungen umzuleiten.

Wenn ein Kopieknoten ungefähr zur gleichen Zeit heruntergefahren wurde wie sein übergeordneter Knoten, meldet die Prozedur möglicherweise noch für mehrere Minuten, dass der Kopieknoten verbunden ist. Die last_updated-Spalte bleibt jedoch unverändert, womit angezeigt wird, dass der Kopieknoten keine aktualisierten Statusmeldungen gesendet hat und die Verbindung wahrscheinlich getrennt ist.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- [CurrentRedoPos-Datenbankeigenschaft \[SQL Anywhere Server - Datenbankadministration\]](#)
- [„Verbindungsparameter NodeType \(NODE\)“ \[SQL Anywhere Server - Datenbankadministration\]](#)
- [„Datenbankspiegelung“ \[SQL Anywhere Server - Datenbankadministration\]](#)
- [„CREATE MIRROR SERVER-Anweisung“ auf Seite 656](#)
- [„ALTER MIRROR SERVER-Anweisung“ auf Seite 480](#)
- [„DROP MIRROR SERVER-Anweisung“ auf Seite 815](#)

Beispiel

Im folgenden Beispiel wird der Verbindungsstatus für den aktuellen Server zurückgegeben und für alle Server, die bei aktivierter Spiegelung direkt oder indirekt Logseiten vom aktuellen Server empfangen.

```
CALL sa_mirror_server_status( );
```

sa_nchar_terms-Systemprozedur

Teilt eine NCHAR-Zeichenfolge in Begriffe auf und gibt jeden Begriff als Zeile mit Angabe seiner Position zurück.

Syntax

```
sa_nchar_terms(  
  text  
  [, config_name  
  [, owner ] ]  
)
```

Argumente

- **text** Die LONG NVARCHAR-Zeichenfolge, die Sie syntaktisch analysieren.
- **config_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um das Textkonfigurationsobjekt anzugeben, das beim Verarbeiten der Zeichenfolge übernommen werden soll. Der Standardwert ist 'default_nchar'.
- **owner** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümer des Textkonfigurationsobjekts anzugeben. Der Standardwert ist NULL. Der aktuelle Benutzer wird angenommen, wenn der Eigentümer nicht angegeben oder auf NULL gesetzt wird.

Bemerkungen

Sie können diese Systemprozedur verwenden, um herauszufinden, wie eine Zeichenfolge interpretiert wird, wenn die Einstellungen für ein Textkonfigurationsobjekt angewendet werden. Dies kann hilfreich sein, wenn Sie wissen möchten, welche Begriffe während der Indizierung oder aus einer Abfragezeichenfolge gelöscht würden.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Volltextsuche“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Konzepte und Referenz zu Textkonfigurationsobjekten“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „sa_char_terms-Systemprozedur“ auf Seite 1171

Beispiel

Die folgende Anweisung gibt die Begriffe in der NCHAR-Zeichenfolge "It's a work-at-home day!" zurück und verwendet dazu das standardmäßige NCHAR-Textkonfigurationsobjekt, default_nchar:

```
CALL sa_nchar_terms (N'It's a work-at-home day!', 'default_nchar', 'sys');
```

term	position
It	1
s	2

term	position
a	3
work	4
at	5
home	6
day	7

sa_performance_diagnostics-Systemprozedur

Gibt eine Zusammenfassung der Anforderungszeitinformationen für alle Verbindungen zurück, wenn der Datenbankserver die Protokollierung der Anforderungszeiten aktiviert hat.

Syntax

`sa_performance_diagnostics()`

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
Number	INTEGER	Gibt die Verbindungs-ID (eine Nummer) für die aktuelle Verbindung zurück.

Spaltenname	Datentyp	Beschreibung
Name	VAR-CHAR(255)	<p>Gibt den Namen der aktuellen Verbindung zurück.</p> <p>Sie können einen Verbindungsnamen mit dem ConnectionName (CON)-Verbindungsparameter angeben. Siehe „Verbindungsparameter ConnectionName (CON)“ [<i>SQL Anywhere Server - Datenbankadministration</i>].</p> <p>Die folgenden Namen werden für temporäre Verbindungen verwendet, die vom Datenbankserver erstellt werden:</p> <ul style="list-style-type: none">• INT:ApplyRecovery• INT:BackupDB• INT:Checkpoint• INT:Cleaner• INT:CloseDB• INT>CreateDB• INT>CreateMirror• INT:DelayedCommit• INT:DiagRcvr• INT:DropDB• INT:EncryptDB• INT:Exchange• INT:FlushMirrorLog• INT:FlushStats• INT:HTTPReq• INT:PromoteMirror• INT:PurgeSnapshot• INT:ReconnectMirror• INT:RecoverMirror• INT:RedoCheckpoint• INT:RefreshIndex• INT:ReloadTrigger• INT:RenameMirror• INT:RestoreDB• INT:StartDB• INT:VSS <p>Siehe „Temporäre Verbindungen“ [<i>SQL Anywhere Server - Datenbankadministration</i>].</p>
Userid	VAR-CHAR(255)	Gibt die Benutzer-ID für die Verbindung zurück.
DBNumber	INTEGER	Gibt die ID-Nummer der Datenbank zurück.

Spaltenname	Datentyp	Beschreibung
LoginTime	TIME-STAMP	Gibt das Datum und die Uhrzeit zurück, zu der die Verbindung hergestellt wurde.
TransactionStartTime	TIME-STAMP	Gibt eine Zeichenfolge mit der Uhrzeit zurück, zu der die Datenbank erstmals nach einem COMMIT oder ROLLBACK geändert wurde, oder eine leere Zeichenfolge, wenn seit dem letzten COMMIT oder ROLLBACK keine Änderungen in der Datenbank durchgeführt wurden.
LastReqTime	TIME-STAMP	Gibt den Zeitpunkt zurück, an dem die letzte Anforderung für die angegebene Verbindung gestartet wurde. Diese Eigenschaft kann eine leere Zeichenfolge für interne Verbindungen wie zum Beispiel Ereignisse zurückgeben.
ReqType	VAR-CHAR(255)	Gibt den Typ der letzten Anforderung zurück. Wenn eine Verbindung vom Verbindungspooling im Cache abgelegt wurde, lautet ihr ReqType-Wert CONNECT_POOL_CACHE.
ReqStatus	VAR-CHAR(255)	<p>Gibt den Status der Anforderung zurück. Dies kann einer der folgenden Werte sein:</p> <ul style="list-style-type: none"> • Idle Die Verbindung verarbeitet derzeit keine Anforderung. • Unscheduled* Die Verbindung hat Arbeit und wartet auf einen verfügbaren Datenbankserver-Worker. • BlockedIO* Die Verbindung ist beim Warten auf einen I/O-Vorgang blockiert. • BlockedContention* Die Verbindung ist blockiert, weil sie auf den Zugriff auf gemeinsam genutzte Datenstrukturen des Datenbankservers wartet. • BlockedLock Die Verbindung ist beim Warten auf eine gesperrtes Objekt blockiert. • Executing Die Verbindung führt eine Anforderung aus. <p>Die mit einem Stern (*) markierten Werte werden nur zurückgegeben, wenn die Protokollierung der Anforderungszeitinformationen beim Datenbankserver unter Verwendung der Serveroption -zt aktiviert ist. Wenn keine Anforderungszeitinformationen protokolliert werden (Standardeinstellung), werden die Werte als "Executing" gemeldet.</p> <p>Siehe „Datenbankserveroption -zt“ [SQL Anywhere Server - Datenbankadministration].</p>

Spaltenname	Datentyp	Beschreibung
ReqTimeUn-scheduled	DOUBLE	Gibt die Menge der Wartezeit ohne Zeitplanung zurück, oder NULL, wenn die Option -zt nicht angegeben wurde. Siehe „Datenbankserveroption -zt“ [SQL Anywhere Server - Datenbankadministration] .
ReqTimeActive	DOUBLE	Gibt die bei der Verarbeitung von Anforderungen verstrichene Zeit in Sekunden zurück oder NULL, wenn die Option -zt nicht angegeben wurde. Siehe „Datenbankserveroption -zt“ [SQL Anywhere Server - Datenbankadministration] .
ReqTimeBlockIO	DOUBLE	Gibt die beim Warten auf den Abschluss von I/O-Vorgängen verstrichene Zeit in Sekunden zurück oder NULL, wenn die Option -zt nicht angegeben wurde. Siehe „Datenbankserveroption -zt“ [SQL Anywhere Server - Datenbankadministration] .
ReqTimeBlockLock	DOUBLE	Gibt die beim Warten auf eine Sperre verstrichene Zeit in Sekunden zurück oder NULL, wenn die Option -zt nicht angegeben wurde. Siehe „Datenbankserveroption -zt“ [SQL Anywhere Server - Datenbankadministration] .
ReqTimeBlockContention	DOUBLE	Gibt die beim Warten auf einen atomaren Zugriff verstrichene Zeit in Sekunden zurück oder NULL, wenn die RequestTiming-Servereigenschaft auf "Off" gesetzt ist. Siehe „Datenbankserveroption -zt“ [SQL Anywhere Server - Datenbankadministration] .
ReqCountUn-scheduled	INTEGER	Gibt die Häufigkeit zurück, mit der die Verbindung auf einen Zeitplan gewartet hat, oder NULL, wenn die Option -zt nicht angegeben wurde. Siehe „Datenbankserveroption -zt“ [SQL Anywhere Server - Datenbankadministration] .
ReqCountActive	INTEGER	Gibt die Anzahl der verarbeiteten Anforderungen zurück, oder NULL, wenn die Servereigenschaft RequestTiming auf "Off" gesetzt ist. Siehe „Datenbankserveroption -zt“ [SQL Anywhere Server - Datenbankadministration] .
ReqCountBlockIO	INTEGER	Gibt die Häufigkeit zurück, mit der die Verbindung auf den Abschluss von I/O-Vorgängen gewartet hat, oder NULL, wenn die Option -zt nicht angegeben wurde. Siehe „Datenbankserveroption -zt“ [SQL Anywhere Server - Datenbankadministration] .
ReqCountBlockLock	INTEGER	Gibt die Häufigkeit zurück, mit der die Verbindung auf eine Sperre gewartet hat, oder NULL, wenn die Option -zt nicht angegeben wurde. Siehe „Datenbankserveroption -zt“ [SQL Anywhere Server - Datenbankadministration] .

Spaltenname	Datentyp	Beschreibung
ReqCountBlockContention	INTEGER	Gibt die Häufigkeit zurück, mit der die Verbindung auf einen unteilbaren Zugriff gewartet hat, oder NULL, wenn die Option -zt nicht angegeben wurde. Siehe „ Datenbankserveroption -zt “ [SQL Anywhere Server - Datenbankadministration].
LastIdle	INTEGER	Gibt die Anzahl der Computertakte zwischen Anforderungen zurück.
BlockedOn	INTEGER	Gibt Null zurück, wenn die aktuelle Verbindung nicht blockiert ist, bzw. im Fall einer Blockierung die Verbindungsnummer, auf der die Verbindung aufgrund eines Sperrenkonflikts blockiert ist.
UncommitOp	INTEGER	Gibt die Anzahl der nicht festgeschriebenen Vorgänge zurück.
CurrentProcedure	VAR-CHAR(255)	Gibt den Namen der Prozedur zurück, die eine Verbindung derzeit ausführt. Wenn die Verbindung verschachtelte Prozeduraufrufe ausführt, ist der Name derjenige der aktuellen Prozedur. Wenn keine Prozedur ausgeführt wird, wird eine leere Zeichenfolge zurückgegeben.
EventName	VAR-CHAR(255)	Gibt den Namen des zugeordneten Ereignisses zurück, wenn die Verbindung einen Event-Handler ausführt. Sonst wird eine leere Zeichenfolge zurückgegeben.
CurrentLineNumber	INTEGER	Gibt die aktuelle Zeilennummer der Prozedur oder Kombinationsanweisung zurück, die eine Verbindung ausführt. Die Prozedur kann unter Verwendung der Eigenschaft CurrentProcedure identifiziert werden. Wenn die Zeile ein Bestandteil einer Kombinationsanweisung vom Client ist, wird eine leere Zeichenfolge zurückgegeben.

Spaltenname	Datentyp	Beschreibung
LastStatement	LONG VAR-CHAR	<p>Gibt die zuletzt vorbereitete SQL-Anweisung für die aktuelle Verbindung zurück.</p> <p>Der LastStatement-Wert wird gesetzt, wenn eine Anweisung vorbereitet wird, und gelöscht, wenn eine Anweisung gelöscht wird. Bei jeder Verbindung ist jeweils nur eine Anweisungszeichenfolge verfügbar.</p> <p>Wenn sa_conn_activity einen nicht-leeren Wert für eine Verbindung meldet, ist dies höchstwahrscheinlich die Anweisung, die die Verbindung derzeit ausführt. Wenn die Anweisung abgeschlossen wäre, wäre sie wahrscheinlich gelöscht und der Eigenschaftswert bereinigt worden. Wenn eine Anwendung mehrere Anweisungen vorbereitet und ihre Anweisungs-Handles beibehält, spiegelt der LastStatement-Wert nicht wieder, was eine Verbindung derzeit ausführt.</p> <p>Wenn das clientseitige Caching von Anweisungen aktiviert ist und eine zwischengespeicherte Anweisung wieder verwendet wird, gibt diese Eigenschaft eine leere Zeichenfolge zurück.</p>
LastPlanText	LONG VAR-CHAR	<p>Gibt den ausführlichen Textplan der letzten auf der Verbindung ausgeführten Abfrage zurück. Sie steuern die Verfügbarkeit des letzten Plans, indem Sie die Option RememberLastPlan der sa_server_option-Systemprozedur setzen oder die Serveroption -zp verwenden. Siehe „Datenbankserveroption -zp“ [SQL Anywhere Server - Datenbankadministration].</p>
AppInfo	LONG VAR-CHAR	<p>Gibt Informationen über den Client zurück, der die Verbindung hergestellt hat. Bei HTTP-Verbindungen umfasst dies auch Angaben zum Browser. Bei Verbindungen, die ältere Versionen von jConnect oder Sybase Open Client verwenden, können die Informationen unvollständig sein.</p> <p>Der API-Wert kann DBLIB, ODBC, OLEDB, ADO.NET, iAnywhereJDBC, PHP, PerlDBD oder DBEXPRESS sein.</p> <p>Weitere Informationen zu Werten, die bei anderen Verbindungstypen zurückgegeben werden, finden Sie unter „Verbindungsparameter AppInfo (APP)“ [SQL Anywhere Server - Datenbankadministration].</p>
LockCount	INTEGER	<p>Gibt die Anzahl der von der Verbindung gehaltenen Sperren zurück.</p>

Spaltenname	Datentyp	Beschreibung
SnapshotCount	INTEGER	Gibt die Anzahl der der Verbindung zugeordneten Snapshots zurück.

Bemerkungen

Die sa_performance_diagnostics-Systemprozedur gibt eine Ergebnismenge zurück, die aus einer Reihe von Anforderungszeiteigenschaften und Statistiken besteht, wenn der Server aufgefordert wurde, die Informationen zu sammeln. Das Aufzeichnen von Anforderungszeitinformationen muss auf dem Datenbankserver aktiviert sein, bevor sa_performance_diagnostics aufgerufen wird. Um dies zu erreichen, legen Sie die Option -zt fest, wenn Sie den Datenbankserver starten, oder führen Sie folgenden Prozeduraufruf aus:

```
CALL sa_server_option( 'RequestTiming','ON' );
```

Privilegien

Sie müssen das MONITOR-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Datenbankserveroption -zt“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sa_performance_statistics-Systemprozedur“ auf Seite 1285
- „sa_server_option-Systemprozedur“ auf Seite 1306

Beispiele

Sie können die folgende Abfrage ausführen, um Verbindungen zu identifizieren, die lange auf den Abschluss von Datenbankserver-Anforderungen gewartet haben.

```
SELECT Number, Name,
       CAST( DATEDIFF( second, LoginTime, CURRENT TIMESTAMP ) AS DOUBLE ) AS
T,
       IF T <> 0 THEN (ReqTimeActive / T) ELSE NULL ENDIF AS PercentActive
FROM   sa_performance_diagnostics()
WHERE  T > 0 AND PercentActive > 10.0
ORDER BY PercentActive DESC;
```

So finden Sie alle derzeit ausführenden Anforderungen, die seit mehr als 60 Sekunden ausgeführt wurden:

```
SELECT Number, Name,
       CAST( DATEDIFF( second, LastReqTime, CURRENT TIMESTAMP ) AS DOUBLE )
AS ReqTime
FROM   sa_performance_diagnostics()
WHERE  ReqStatus <> 'IDLE' AND ReqTime > 60.0
ORDER BY ReqTime DESC;
```

sa_performance_statistics-Systemprozedur

Gibt Performancestatistiken für den Server sowie für Datenbanken und Verbindungen zurück.

Syntax

`sa_performance_statistics()`

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
DBNumber	INTEGER	Gibt die ID-Nummer der Datenbank zurück. Gibt NULL zurück, wenn der Eigenschaftstyp "Server" ist.
ConnNumber	INTEGER	Gibt die Verbindungs-ID (eine Nummer) für die aktuelle Verbindung zurück. Gibt NULL zurück, wenn der Typ "Server" oder "Datenbank" ist.
PropNum	INTEGER	Gibt die Eigenschaftsnummer zurück.
PropName	VARCHAR(255)	Gibt den Eigenschaftsnamen zurück.
Wert	INTEGER	Gibt den Eigenschaftswert zurück.

Bemerkungen

Die `sa_performance_statistics`-Systemprozedur gibt eine Ergebnismenge zurück, die aus Performancestatistiken besteht. Die Ergebnisse sind eine Teilmenge der Ergebnisse, die Sie mithilfe der Funktionen `PROPERTY`, `DB_PROPERTY` und `CONNECTION_PROPERTY` zurückgeben können.

Privilegien

Sie müssen das `MONITOR`-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- [„sa_performance_diagnostics-Systemprozedur“ auf Seite 1279](#)
- [„sa_server_option-Systemprozedur“ auf Seite 1306](#)

Beispiel

Das folgende Beispiel entlädt alle Performance-Statistiken in eine Textdatei namens *dump_stats.txt*:

```
UNLOAD
  SELECT CURRENT_TIMESTAMP, *
  FROM sa_performance_statistics()
  TO 'dump_stats.txt'
  APPEND ON;
```

sa_post_login_procedure-Systemprozedur

Ermittelt, ob das Kennwort eines Benutzers vor dem Ablauf steht.

Syntax

```
sa_post_login_procedure( )
```

Argumente

Keine

Ergebnismenge

Die sa_post_login_procedure-Systemprozedur gibt folgende Informationen zurück:

Spaltenname	Datentyp	Beschreibung
message_text	VARCHAR(255)	Wenn message_action gleich 1 ist, gibt message_text die anzuzeigende Meldung zurück. Wenn message_action gleich 0 ist, ist message_text NULL.
message_action	INTEGER	Ob das Kennwort vor dem Ablauf steht (1=ja, 0=nein).

Bemerkungen

Die sa_post_login_procedure-Systemprozedur ist die Standardeinstellung für die post_login_procedure-Datenbankoption.

sa_post_login_procedure benutzt die Login-Richtlinien-Optionswerte des Benutzers für password_life_time und password_grace_time sowie das aktuelle Datum und die Uhrzeit, um zu ermitteln, ob das Kennwort eines Benutzers vor dem Ablauf steht. Wenn dies der Fall ist, wird die Meldung, die dem Benutzer übermittelt werden soll, in der Ergebnismenge zurückgegeben.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „post_login_procedure-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Login-Richtlinien“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Im folgenden Beispiel wird sa_post_login_procedure verwendet, um zu ermitteln, ob das Kennwort des aktuellen Benutzers vor dem Ablauf steht:

```
CALL sa_post_login_procedure( );
```

sa_procedure_profile-Systemprozedur

Gibt Auskunft über die Ausführungszeit einzelner Zeilen innerhalb von Prozeduren, Funktionen, Ereignissen und Triggern, die in einer Datenbank ausgeführt wurden.

Syntax

```
sa_procedure_profile(  
  [ filename  
  [, save_to_file ] ]  
)
```

Argumente

- **filename** Verwenden Sie diesen optionalen LONG VARCHAR-Parameter, um die Datei anzugeben, in der die Profilerstellungsinformationen gespeichert werden sollen bzw. aus der sie geladen werden sollen. Der Standardwert ist NULL. Im Abschnitt "Bemerkungen" weiter unten finden Sie weitere Hinweise zum Speichern und Laden von Profilerstellungsinformationen.
- **save_to_file** Mit diesem optionalen INTEGER-Parameter können Sie angeben, ob die Profilerstellungsinformationen in einer Datei gespeichert oder aus einer zuvor gespeicherten Datei geladen werden sollen. Der Standardwert ist 0.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
object_type	CHAR(1)	Der Typ des Objekts. Im Abschnitt "Bemerkungen" weiter unten finden Sie eine Liste der möglichen Objekttypen.
object_name	CHAR(128)	Der Name der gespeicherten Prozedur, der Funktion, des Ereignisses bzw. des Triggers. Wenn object_type mit C oder D definiert wird, ist dies der Name des Fremdschlüssels, für den der Systemtrigger festgelegt wurde.
owner_name	CHAR(128)	Der Eigentümer des Objekts
table_name	CHAR(128)	Die einem Trigger zugeordnete Tabelle (der Wert ist NULL bei anderen Objekttypen)
line_num	UNSIGNED INTEGER	Die Nummer der Zeile innerhalb der Prozedur
executions	UNSIGNED INTEGER	Gibt an, wie häufig die Zeile ausgeführt wurde
millisecs	UNSIGNED INTEGER	Die Zeit zum Ausführen der Zeile in Millisekunden
percentage	DOUBLE	Der Prozentsatz der Gesamtausführungszeit, die für die betreffende Zeile erforderlich ist

Spaltenname	Datentyp	Beschreibung
foreign_owner	CHAR(128)	Der Datenbankbenutzer, dem die entfernte Tabelle für einen Systemtrigger gehört
foreign_table	CHAR(128)	Der Name der entfernten Tabelle für einen Systemtrigger

Bemerkungen

Diese Prozedur liefert dieselben Informationen wie die Registerkarte **Profil** in Sybase Central.

Sie können diese Prozedur für Folgendes verwenden:

- **Detaillierte Profilerstellungsinformationen zurückgeben** Um das zu tun, rufen Sie einfach die Prozedur ohne Angabe von Argumenten auf.
- **Detaillierte Profilerstellungsinformationen in einer Datei speichern** Um das zu tun, müssen Sie das *filename*-Argument verwenden und "1" für das *save_to_file*-Argument angeben.
- **Detaillierte Profilerstellungsinformationen von einer vorher gespeicherten Datei laden** Um das zu tun, müssen Sie das *filename*-Argument verwenden und "0" für das *save_to_file*-Argument angeben. Wenn Sie die Prozedur auf diese Art verwenden, muss die zu ladende Datei von derselben Datenbank wie jene erstellt worden sein, von der aus Sie die Prozedur ausführen, da sonst die Ergebnisse möglicherweise unbrauchbar sind.

Da die Ergebnismenge Angaben über die Ausführungszeiten für einzelne Zeilen innerhalb von Prozeduren, Triggern, Funktionen und Ereignissen sowie den Prozentsatz der gesamten Prozedurausführungszeit enthält, den diese Zeilen verwenden, können Sie diese Profilerstellungsinformationen verwenden, um langsamere Prozeduren zu optimieren, die möglicherweise die Datenbank-Performance beeinträchtigen.

Damit Sie ein Profil Ihrer Datenbank erstellen können, müssen Sie die Profilerstellung aktivieren.

Die Spalte *object_type* der Ergebnismenge kann folgende Werte enthalten:

- **P** Gespeicherte Prozedur
- **F** Funktion
- **E** Ereignis
- **T** Trigger
- **C** ON UPDATE-Systemtrigger
- **D** ON DELETE-Systemtrigger

Wenn Sie zusammengefasste Informationen anstelle von zeilenweisen Details für jede Ausführung haben wollen, verwenden Sie stattdessen die *sa_procedure_profile_summary*-Prozedur.

Privilegien

Sie müssen das MONITOR-Systemprivileg oder das MANAGE PROFILING-Systemprivileg haben.

Außerdem müssen Sie die folgenden Privilegien haben.

- SELECT ANY TABLE (wenn *filename* nicht NULL ist und *save_to_file* 1 ist)
- LOAD ANY TABLE (wenn *filename* nicht NULL ist und *save_to_file* 0 ist)

Nebenwirkungen

Keine

Siehe auch

- „sa_server_option-Systemprozedur“ auf Seite 1306
- „sa_procedure_profile_summary-Systemprozedur“ auf Seite 1290
- „Prozedurprofilerstellung aktivieren“ [SQL Anywhere Server - SQL-Benutzerhandbuch]

Beispiel

Die folgende Anweisung gibt die Ausführungszeit für jede Zeile von allen Prozeduren, Funktionen, Ereignissen oder Triggern zurück, die in der Datenbank ausgeführt wurden:

```
CALL sa_procedure_profile( );
```

Die folgende Anweisung gibt dieselben detaillierten Profilerstellungsinformationen wie das obenstehende Beispiel zurück und speichert sie in einer Datei namens *detailedinfo.txt*:

```
CALL sa_procedure_profile( 'detailedinfo.txt', 1 );
```

Eine der folgenden Anweisungen kann verwendet werden, um detaillierte Informationen zur Prozedurprofilerstellung aus einer Datei namens *detailedinfo.txt* zu laden:

```
CALL sa_procedure_profile( 'detailedinfo.txt', 0 );
```

```
CALL sa_procedure_profile( 'detailedinfo.txt' );
```

sa_procedure_profile_summary-Systemprozedur

Gibt Auskunft über die Ausführungszeit aller Prozeduren, Funktionen, Ereignisse und Trigger, die in einer Datenbank ausgeführt wurden. Diese Prozedur liefert dieselben Informationen für diese Objekte wie die Registerkarte **Profil** in Sybase Central.

Syntax

```
sa_procedure_profile_summary(  
    [ filename  
    [, save_to_file ] ]  
)
```

Argumente

- **filename** Verwenden Sie diesen optionalen LONG VARCHAR-Parameter, um die Datei anzugeben, in der die Profilerstellungsinformationen gespeichert bzw. aus der sie geladen werden

sollen. Der Standardwert ist NULL. Im Abschnitt "Bemerkungen" weiter unten finden Sie weitere Hinweise zum Speichern und Laden von Profilerstellungsinformationen.

- **save_to_file** Mit diesem optionalen INTEGER-Parameter können Sie angeben, ob die Zusammenfassung in einer Datei gespeichert oder aus einer zuvor gespeicherten Datei geladen werden soll. Der Standardwert ist 0.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
object_type	CHAR(1)	Der Typ des Objekts. Im Abschnitt "Bemerkungen" weiter unten finden Sie eine Liste der möglichen Objekttypen.
object_name	CHAR(128)	Der Name der gespeicherten Prozedur, der Funktion, des Ereignisses bzw. des Triggers.
owner_name	CHAR(128)	Der Eigentümer des Objekts
table_name	CHAR(128)	Die einem Trigger zugeordnete Tabelle (der Wert ist NULL bei anderen Objekttypen)
executions	UNSIGNED INTEGER	Gibt an, wie häufig jede einzelne Prozedur ausgeführt wurde
millisecs	UNSIGNED INTEGER	Die Zeit zum Ausführen der Prozedur in Millisekunden
foreign_owner	CHAR(128)	Der Datenbankbenutzer, dem die entfernte Tabelle für einen Systemtrigger gehört
foreign_table	CHAR(128)	Der Name der entfernten Tabelle für einen Systemtrigger

Bemerkungen

Sie können diese Prozedur für Folgendes verwenden:

- **Aktuelle Zusammenfassungsinformationen zurückgeben** Um das zu tun, rufen Sie einfach die Prozedur ohne Angabe von Argumenten auf.
- **Aktuelle Zusammenfassungsinformationen in einer Datei speichern** Um das zu tun, müssen Sie das *filename*-Argument verwenden und "1" für das *save_to_file*-Argument angeben.
- **Gespeicherte Zusammenfassungsinformationen aus einer Datei laden** Um das zu tun, müssen Sie das *filename*-Argument verwenden und "0" für das *save_to_file*-Argument angeben. Wenn Sie die Prozedur auf diese Art verwenden, muss die zu ladende Datei von derselben Datenbank wie jene erstellt worden sein, von der aus Sie die Prozedur ausführen, da sonst die Ergebnisse möglicherweise unbrauchbar sind.

Da die Prozedur Informationen über Verwendungshäufigkeit und -effizienz von gespeicherten Prozeduren, Funktionen, Ereignissen und Triggern zurückgibt, können Sie diese Informationen benutzen, um langsamere Prozeduren zu optimieren und die Datenbank-Performance zu verbessern.

Damit Sie ein Profil Ihrer Datenbank erstellen können, müssen Sie die Profilerstellung aktivieren.

Die Spalte `object_type` der Ergebnismenge kann folgende Werte enthalten:

- **P** Gespeicherte Prozedur
- **F** Funktion
- **E** Ereignis
- **T** Trigger
- **S** Systemtrigger
- **C** ON UPDATE-Systemtrigger
- **D** ON DELETE-Systemtrigger

Wenn Sie zeilenweise Details für jede Ausführung anstelle von Zusammenfassungsinformationen haben wollen, verwenden Sie stattdessen die `sa_procedure_profile`-Prozedur.

Privilegien

Sie müssen das MONITOR-Systemprivileg oder das MANAGE PROFILING-Systemprivileg haben.

Außerdem müssen Sie die folgenden Privilegien haben.

- SELECT ANY TABLE (wenn *filename* nicht NULL ist und *save_to_file* 1 ist)
- LOAD ANY TABLE (wenn *filename* nicht NULL ist und *save_to_file* 0 ist)

Nebenwirkungen

Keine

Siehe auch

- „Prozedurprofilerstellung aktivieren“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „sa_server_option-Systemprozedur“ auf Seite 1306
- „sa_procedure_profile-Systemprozedur“ auf Seite 1288

Beispiel

Die folgende Anweisung gibt die Ausführungszeit für alle Prozeduren, Funktionen, Ereignisse oder Trigger zurück, die in der Datenbank ausgeführt wurden:

```
CALL sa_procedure_profile_summary( );
```

Die folgende Anweisung gibt dieselbe Zusammenfassung wie das obenstehende Beispiel zurück und speichert sie in einer Datei namens *summaryinfo.txt*:

```
CALL sa_procedure_profile_summary( 'summaryinfo.txt', 1 );
```

Eine der folgenden Anweisungen kann verwendet werden, um eine gespeicherte Zusammenfassung aus einer Datei namens *summaryinfo.txt* zu laden:

```
CALL sa_procedure_profile_summary( 'summaryinfo'.txt, 0 );
```

```
CALL sa_procedure_profile_summary( 'summaryinfo.txt' );
```

sa_recompile_views-Systemprozedur

Lokalisiert im Katalog gespeicherte Ansichtsdefinitionen, die keine Spaltendefinitionen haben, und bewirkt, dass die Spaltendefinitionen erstellt werden

Syntax

```
sa_recompile_views( [ ignore_errors ] )
```

Argumente

- **ignore_errors** Verwenden Sie diesen optionalen INTEGER-Parameter um anzugeben, ob Fehler während der Neukompilation zurückgegeben werden. Wenn Sie 0 angeben, wird für jede Ansicht, bei der eine Spaltendefinition fehlgeschlagen ist, ein Fehler zurückgegeben. Wenn Sie 1 bzw. einen beliebigen Wert außer 0 angeben, werden keine Fehler zurückgegeben. Der Standardwert ist 0.

Bemerkungen

Diese Prozedur wird verwendet, um Ansichten im Katalog zu lokalisieren, die keine Spaltendefinitionen haben, und um eine ALTER VIEW-Anweisung mit der RECOMPILE-Klausel auszuführen, wodurch die Spaltendefinitionen erstellt werden. Die Prozedur führt dies bei jeder Ansicht durch, die keine Spaltendefinition hat, bis es keine mehr gibt, die kompiliert werden muss, bzw. bis nur mehr Spaltendefinitionen verbleiben, die nicht erstellt werden können. Wenn die Prozedur nicht in der Lage ist, eine Ansicht neu zu kompilieren, wird ein Fehler ausgegeben. Die Fehlerausgabe kann unterdrückt werden, indem Sie für diese Prozedur einen Nichtnull-Parameter festlegen.

Vorsicht

Die sa_recompile_views-Systemprozedur darf nur in einem *reload.sql*-Skript aufgerufen werden. Diese Prozedur wird vom Dienstprogramm zum Entladen (dbunload) verwendet und darf nicht explizit verwendet werden.

Die sa_recompile_views-Systemprozedur versucht nicht, materialisierte Ansichten oder mit "Deaktiviert" markierte Ansichten zu rekompilieren.

Privilegien

Sie müssen das ALTER ANY VIEW-Systemprivileg haben.

Nebenwirkungen

Bei jeder regulären Ansicht, die nicht den Status VALID hat, wird eine ALTER VIEW owner.viewname ENABLE-Anweisung ausgeführt, die ein automatisches Festschreiben bewirkt.

Siehe auch

- „Status für reguläre Ansichten“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „force_view_creation-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „ALTER VIEW-Anweisung“ auf Seite 543

Beispiel

Im folgenden Beispiel aus einem *reload.sql*-Skript wird die `sa_recompile_views`-Systemprozedur verwendet, um im Katalog gespeicherte Ansichtsdefinitionen zu finden, die keine Spaltendefinitionen enthalten, und die Erstellung der Spaltendefinitionen zu starten. Fehler werden ignoriert.

```
CALL sa_recompile_views( 1 );
```

sa_refresh_materialized_views-Systemprozedur

Initialisiert alle materialisierten Ansichten, die in einem nicht initialisierten Status sind

Syntax

```
sa_refresh_materialized_views( [ ignore_errors ] )
```

Argumente

- **ignore_errors** Verwenden Sie diesen optionalen INTEGER-Parameter um anzugeben, ob Fehler während der Neukompilation zurückgegeben werden. Wenn Sie 0 angeben, wird für jede Ansicht, bei der eine Spaltendefinition fehlgeschlagen ist, ein Fehler zurückgegeben. Wenn Sie 1 bzw. einen beliebigen Wert außer 0 angeben, werden keine Fehler zurückgegeben. Der Standardwert ist 0.

Bemerkungen

Eine materialisierte Ansicht kann sich in einem nicht initialisiertem Status befinden, weil sie gerade erstellt oder gerade neu aktiviert wurde bzw. weil der letzte Versuch, sie zu initialisieren oder aktualisieren, fehlgeschlagen ist. Die `sa_refresh_materialized_views`-Systemprozedur durchsucht die Datenbank nach solchen Ansichten und versucht, sie zu initialisieren. Wenn die Prozedur beim Initialisieren von materialisierten Ansichten auf einen Fehler stößt, fährt sie fort, die verbliebenen materialisierten Ansichten zu verarbeiten.

Sie können auch die `REFRESH MATERIALIZED VIEW`-Anweisung verwenden, um materialisierte Ansichten zu aktualisieren.

Privilegien

Sie müssen das `ALTER ANY MATERIALIZED VIEW`-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „`REFRESH MATERIALIZED VIEW`-Anweisung“ auf Seite 987
- „Materialisierten Ansichten manuell aktualisieren“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Beispiel

Im folgenden Beispiel wird die `sa_refresh_materialized_views`-Systemprozedur verwendet, um alle materialisierten Ansichten zu initialisieren, die sich in einem nicht initialisierten Zustand befinden. Fehler werden ignoriert.

```
CALL sa_refresh_materialized_views( 1 );
```

sa_refresh_text_indexes-Systemprozedur

Aktualisiert alle Textindizes, die als MANUAL REFRESH oder AUTO REFRESH definiert sind.

Syntax

```
sa_refresh_text_indexes( )
```

Bemerkungen

Die `sa_refresh_text_indexes`-Systemprozedur aktualisiert alle Textindizes, die als MANUAL REFRESH oder AUTO REFRESH definiert wurden. Sie aktualisiert keine Textindizes, die als IMMEDIATE REFRESH definiert wurden (Standardwert), weil Änderungen an diesen Indizes durchgeführt werden, sobald sich die Daten in der Basistabelle ändern.

Privilegien

Sie müssen das CREATE ANY INDEX-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Volltextsuche“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Konzepte und Referenz zu Textkonfigurationsobjekten“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „DROP TEXT INDEX-Anweisung“ auf Seite 834
- „REFRESH TEXT INDEX-Anweisung“ auf Seite 990
- „TRUNCATE-Anweisung“ auf Seite 1089
- „SYSTEXTIDX-Systemansicht“ auf Seite 1501
- „sa_text_index_stats-Systemprozedur“ auf Seite 1342
- „sa_text_index_vocab-Systemprozedur“ auf Seite 1343

Beispiel

Die folgende Anweisung aktualisiert alle MANUAL- und AUTO REFRESH -Textindizes in der Datenbank:

```
CALL sa_refresh_text_indexes();
```

sa_remove_tracing_data-Systemprozedur

Löscht aus den Diagnoseprotokollierungstabellen permanent alle Datensätze, die sich auf die angegebene Protokollierungssitzungs-ID beziehen

Syntax

```
sa_remove_tracing_data( log_session_id )
```

Argumente

- **log_session_id** Verwenden Sie diesen UNSIGNED INTEGER-Parameter, um die ID der Protokollierungssitzung anzugeben, aus der Daten entfernt werden sollen.

Bemerkungen

Wenn es keine Datensätze für die angegebene *log_session_id* gibt, hat die Prozedur keine Auswirkung. Die Prozedur hat keine Rückgabewerte.

Privilegien

Sie müssen die DIAGNOSTICS-Systemrolle und das MANAGE PROFILING-Systemprivileg haben.

Nebenwirkungen

Verursacht beim Beenden einen Festschreibevorgang, auch wenn keine Datensätze für die angegebene *log_session_id* gefunden wurden.

Siehe auch

- „Diagnoseprotokollierungstabellen“ auf Seite 1144

Beispiel

In diesem Beispiel werden alle Datensätze, die zur Protokollierungssitzungs-ID 1 gehören, dauerhaft aus den Diagnoseprotokollierungstabellen gelöscht.

```
CALL sa_remove_tracing_data( 1 );
```

sa_report_deadlocks-Systemprozedur

Ruft Informationen über Deadlocks aus einem internen Puffer ab, der vom Datenbankserver erstellt wird

Syntax

```
sa_report_deadlocks( )
```

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
snapshotId	BIGINT	Die Deadlock-Instanz (alle Zeilen, die ein bestimmter Deadlock betreffen, haben dieselbe Kennung)

Spaltenname	Datentyp	Beschreibung
snapshotAt	TIMESTAMP	Die Uhrzeit, wann der Deadlock auftrat
waiter	INTEGER	Der Verbindungs-Handle der wartenden Verbindung
who	VAR-CHAR(128)	Die Benutzer-ID, die der wartenden Verbindung zugeordnet ist
what	LONG VAR-CHAR	Der Befehl, der von der wartenden Verbindung ausgeführt wird Diese Informationen sind nur verfügbar, wenn Sie das Erfassen der zuletzt vorbereiteten SQL-Anweisung aktiviert haben. Dazu müssen Sie entweder in der Datenbankserver-Befehlszeile die Option -zl angeben oder diese Funktion mithilfe der sa_server_option-Systemprozedur aktiviert haben.
object_id	UNSIGNED BIGINT	Die Objekt-ID der Tabelle, die die Zeile enthält
record_id	BIGINT	Die Zeilen-ID der zugeordneten Zeile
owner	INTEGER	Der Verbindungs-Handle der Verbindung, der die Sperre gehört, an der gewartet wird
is_victim	BIT	Identifiziert die zurückgesetzte Transaktion
rollback_operation_count	UNSIGNED INTEGER	Die Anzahl der nicht festgeschriebenen Vorgänge, die verloren gehen können, wenn die Transaktion zurückgesetzt wird

Bemerkungen

Wenn die log_deadlocks-Option auf ON eingestellt ist, protokolliert der Datenbankserver in einem internen Puffer Informationen über Deadlocks. Sie können die Informationen im Log anzeigen, indem Sie die sa_report_deadlocks-Systemprozedur ausführen.

Privilegien

Sie müssen das MONITOR-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Systemereignisse“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „log_deadlocks-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sa_server_option-Systemprozedur“ auf Seite 1306
- „Wie Sie ermitteln, was in einem Deadlock blockiert ist“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Datenbankserveroption -z1“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sa_server_option-Systemprozedur“ auf Seite 1306

Beispiele

Sie können die folgende Abfrage ausführen, um Deadlocks zu ermitteln.

```
CALL sa_report_deadlocks( );
```

sa_reserved_words-Systemprozedur

Gibt eine Liste reservierter Wörter zurück. Viele der in SQL-Anweisungen erscheinenden Schlüsselwörter sind reservierte Wörter.

Syntax

```
sa_reserved_words( )
```

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
reserved_word	CHAR(128)	Ein reserviertes Wort.

Bemerkungen

Die Prozedur übernimmt keine Parameter und gibt ein Wort pro Zeile zurück. Die Liste der reservierten Wörter basiert auf der Version des Datenbankservers, der die Abfrage ausführt, nicht auf der Version der Software, die für die Erstellung der Datenbankdatei verwendet wurde.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Reservierte Wörter“ auf Seite 1

Beispiel

Die folgende Anweisung gibt eine Liste von für reservierten Wörtern zurück:

```
SELECT * FROM sa_reserved_words( );
```

sa_reset_identity-Systemprozedur

Ermöglicht, dass der nächste Identity-Wert für eine Tabelle eingestellt wird. Verwenden Sie diese Prozedur, wenn der AUTOINCREMENT-Wert für die nächste einzufügende Zeile automatisch um 1 erhöht werden soll.

Syntax

```
sa_reset_identity(  
tbl_name  
[, owner_name  
[, new_identity ] ]  
)
```

Argumente

- **tbl_name** Verwenden Sie diesen CHAR(128)-Parameter, um die Tabelle anzugeben, bei der Sie den ID-Wert zurücksetzen wollen. Wenn kein Eigentümer angegeben ist, muss *tbl_name* eine Tabelle in der Datenbank eindeutig identifizieren.
- **owner_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümer der Tabelle anzugeben, für die Sie den Identity-Wert zurücksetzen möchten. Der Standardwert ist NULL. Wenn *owner_name* nicht angegeben wurde, verwenden Sie für das dritte Argument einen benannten Parameterwert. Beispiel:

```
CALL sa_reset_identity( 'Employees', new_identity=100 );
```

- **new_identity** Verwenden Sie diesen optionalen BIGINT-Parameter, um den Wert anzugeben, mit dem aus das Autoincrement beginnen soll. Der Standardwert ist NULL.

Bemerkungen

Der nächste Wert, der für eine in die Tabelle eingefügte Zeile generiert wird, ist *new_identity* + 1.

Es wird nicht überprüft, ob *new_identity* + 1 mit einer bestehenden Zeile in der Tabelle in Konflikt steht. Wenn Sie zum Beispiel für *new_identity* den Wert 100 angeben, erhält die nächste eingefügte Zeile den Wert 101. Wenn 101 jedoch bereits existiert, schlägt die Zeileneinfügung fehl.

Die sa_reset_identity-Systemprozedur kann nicht auf eine Tabelle angewendet werden, die keine Spalten mit der Standardeinstellung AUTOINCREMENT oder GLOBAL AUTOINCREMENT aufweist.

Privilegien

Sie müssen Eigentümer der Tabelle sein oder das ALTER ANY TABLE-Systemprivileg haben.

Nebenwirkungen

Bewirkt, dass ein Checkpoint gesetzt wird, nachdem der Wert aktualisiert wurde

Siehe auch

- „Der Standardwert "Autoincrement"“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Der Standardwert GLOBAL AUTOINCREMENT“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Beispiel

Die folgende Anweisung setzt den nächsten Identity-Wert auf "101":

```
CALL sa_reset_identity( 'Employees', 'GROUPO', 100 );
```

sa_rowgenerator-Systemprozedur

Gibt eine Ergebnismenge mit Zeilen zwischen einem angegebenen Start- und Endwert zurück

Syntax

```
sa_rowgenerator(  
  [ rstart  
  [, rend  
  [, rstep ] ] ]  
)
```

Argumente

- **rstart** Verwenden Sie diesen optionalen INTEGER-Parameter, um den Startwert anzugeben. Standardwert ist "0".
- **rend** Verwenden Sie diesen optionalen INTEGER-Parameter, um den Endwert anzugeben, der größer oder gleich *rstart* sein muss. Der Standardwert ist "100".
- **rstep** Verwenden Sie diesen optionalen INTEGER-Parameter, um das Inkrement anzugeben, mit dem die Sequenzwerte erhöht werden. Der Standardwert ist "1".

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
row_num	INTEGER	Sequenznummer

Bemerkungen

Die sa_rowgenerator-Prozedur kann in der FROM-Klausel einer Abfrage verwendet werden, um eine Sequenz von Zahlen zu generieren. Diese Prozedur ist eine Alternative zur Verwendung der RowGenerator-Systemtabelle. Sie können sa_rowgenerator beispielsweise für folgende Aufgaben verwenden:

- Test-Daten für eine bekannte Anzahl von Zeilen in einer Ergebnismenge generieren
- Eine Ergebnismenge mit Zeilen für Werte in jedem Bereich generieren. Sie können z.B. eine Zeile für jeden Tag des Monats oder Bereiche von Postleitzahlen generieren.
- Eine Abfrage generieren, die eine angegebene Anzahl von Zeilen in der Ergebnismenge hat. Das kann beim Testen der Performance von Abfragen nützlich sein.

Es werden keine Zeilen zurückgegeben, wenn Sie nicht den richtigen Start- und Endwert angeben und einen positiven Schrittwert ungleich Null wählen.

Sie können das Verhalten der RowGenerator-Tabelle mit der folgenden Anweisung emulieren:

```
SELECT row_num FROM sa_rowgenerator( 1, 255 );
```

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „RowGenerator-Tabelle (dbo)“ auf Seite 1159
- „Nullwert in Aggregatfunktion eliminiert“ [*Fehlermeldungen*]

Beispiel

Die folgende Abfrage gibt eine Ergebnismenge zurück, die für jeden Tag des laufenden Monats eine Zeile enthält.

```
SELECT DATEADD( day, row_num-1,
               YMD( DATEPART( year, CURRENT DATE ),
                     DATEPART( month, CURRENT DATE ), 1 ) )
       AS day_of_month
FROM sa_rowgenerator( 1, 31, 1 )
WHERE DATEPART( month, day_of_month ) = DATEPART( month, CURRENT DATE )
ORDER BY row_num;
```

Die folgende Abfrage zeigt, wie viele Mitarbeiter in Postleitzahlenbereichen (0-9999), (10000-19999), ..., (90000-99999) leben. In einigen Bereichen sind keine Mitarbeiter vorhanden, sodass eine Warnung erscheint.

Die sa_rowgenerator-Prozedur kann verwendet werden, um diese Bereiche zu generieren, auch wenn es keine Mitarbeiter gibt, die eine Postleitzahl in diesem Bereich haben.

```
SELECT row_num AS r1, row_num+9999 AS r2, COUNT( PostalCode ) AS
       zips_in_range
FROM sa_rowgenerator( 0, 99999, 10000 ) D LEFT JOIN Employees
       ON PostalCode BETWEEN r1 AND r2
GROUP BY r1, r2
ORDER BY 1;
```

Das folgende Beispiel generiert 10 Zeilen von Daten und fügt sie in die NewEmployees-Tabelle ein:

```
INSERT INTO NewEmployees ( ID, Salary, Name )
SELECT row_num, CAST( RAND() * 1000 AS INTEGER ), 'Mary'
FROM sa_rowgenerator( 1, 10 );
```

Im folgenden Beispiel wird die sa_rowgenerator-Systemprozedur verwendet, um eine Ansicht zu erstellen, die alle Ganzzahlen enthält. Der Wert "2147483647" in diesem Beispiel stellt die höchste Ganzzahl mit Vorzeichen dar, die unterstützt wird.

```
CREATE VIEW Integers AS
SELECT row_num AS n
FROM sa_rowgenerator( 0, 2147483647, 1 );
```

In diesem Beispiel wird die `sa_rowgenerator`-Systemprozedur verwendet, um eine Ansicht zu erstellen, die Datumsangaben von 0001-01-01 bis 9999-12-31 enthält. Der Wert "3652058" in diesem Beispiel stellt die Anzahl der Tage zwischen 0001-01-01 und 9999-12-31 dar, den frühesten und spätesten Datumsangaben, die unterstützt werden.

```
CREATE VIEW Dates AS
SELECT DATEADD( day, row_num, '0001-01-01' ) AS d
FROM sa_rowgenerator( 0, 3652058, 1 );
```

Die folgende Abfrage gibt alle Jahre zwischen 1900 und 2058 zurück, die 54 Wochen haben.

```
SELECT DATEADD ( day, row_num, '1900-01-01' ) AS d, DATEPART ( week, d ) w
FROM sa_rowgenerator ( 0, 63919, 1 )
WHERE w = 54;
```

sa_save_trace_data-Systemprozedur

Speichert Protokollierungsdaten in Basistabellen

Syntax

```
sa_save_trace_data( )
```

Bemerkungen

Während eine Protokollierungssitzung ausgeführt wird, werden Diagnosedaten in temporären Versionen der Diagnoseprotokollierungstabellen gespeichert. Wenn Sie eine Protokollierungssitzung stoppen, geben Sie an, ob Sie die Protokollierungsdaten permanent in den Basistabellen für die Diagnoseprotokollierung speichern wollen. Wenn Sie keine Datenspeicherung wählen, können Sie die Daten dennoch speichern, nachdem die Sitzung gestoppt wurde, indem Sie die `sa_save_trace_data`-Systemprozedur verwenden.

Die `sa_save_trace_data`-Systemprozedur gibt einen Fehler zurück, wenn die Protokollierung noch läuft. Sie müssen die Protokollierung stoppen, um diese Systemprozedur verwenden zu können.

Die `sa_save_trace_data`-Systemprozedur kann sogar verwendet werden, wenn der Benutzer beim Stoppen der Protokollierung `WITHOUT SAVING` angegeben hat. Überdies muss die Prozedur von der Protokollierungsdatenbank aus aufgerufen werden.

Privilegien

Sie müssen das `MANAGE PROFILING`-Systemprivileg haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Erstellen einer Diagnoseprotokollierungssitzung (Sybase Central)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Diagnoseprotokollierungstabellen“ auf Seite 1144

Beispiel

In diesem Beispiel werden Protokollierungsdaten in den Diagnoseprotokollierungstabellen gespeichert.

```
CALL sa_save_trace_data( );
```

sa_send_udp-Systemfunktion

Sendet ein UDP-Paket an die angegebene Adresse

Syntax

```
sa_send_udp(  
  destAddress  
  , destPort  
  , msg  
)
```

Argumente

- **destAddress** Verwenden Sie diesen CHAR(254)-Parameter, um den Hostnamen oder die IP-Nummer anzugeben.
- **destPort** Verwenden Sie diesen UNSIGNED SMALLINT-Parameter, um die zu verwendende Portnummer anzugeben.
- **msg** Verwenden Sie diesen LONG BINARY-Parameter, um die Meldung anzugeben, die an die angegebene Adresse gesendet werden soll. Wenn dieser Wert eine Zeichenfolge ist, muss er in Apostrophe eingeschlossen werden.

Rückgabe

Diese Funktion gibt einen INTEGER-Statuscode zurück.

Bemerkungen

Diese Prozedur sendet ein einzelnes UDP-Paket an die angegebene Adresse. Die Funktion gibt 0 zurück, wenn die Nachricht erfolgreich gesendet wird, und einen Fehlercode, wenn ein Fehler auftritt. Der Fehlercode ist einer der folgenden:

- -1, wenn die Meldung zu umfangreich ist, um über einen UDP-Socket gesendet zu werden (wie vom Betriebssystem festgelegt), oder wenn es ein Problem mit der Zieladresse gibt
- Der Winsock/Posix-Fehlercode, der vom Betriebssystem zurückgegeben wird

Wenn der *msg*-Parameter binäre Daten enthält oder komplexer ist als eine Zeichenfolge, empfiehlt es sich, eine BINARY-Variable zu verwenden. Beispiel:

```
CREATE VARIABLE v LONG BINARY;  
SET v='This is a UDP message';  
SELECT sa_send_udp( '10.25.99.124', 1234, v );  
DROP VARIABLE v;
```

Diese Funktion kann zusammen mit der serverinitiierten MobiLink-Synchronisation verwendet werden, um das MobiLink Listener-Dienstprogramm (*dblsn.exe*) zu aktivieren. Wenn Sie den MobiLink Listener mithilfe der *sa_send_udp*-Funktion benachrichtigen möchten, sollten Sie an das UDP-Paket eine 1 anhängen. Diese Zahl ist eine Protokollnummer für die serverinitiierte Synchronisation. In zukünftigen

Versionen von MobiLink führen neue Protokollversionen möglicherweise dazu, dass sich der MobiLink-Listener anders verhält.

Privilegien

Sie müssen das `MANAGE ANY WEB SERVICE`-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Senden einer Push-Benachrichtigung mithilfe der `sa_send_udp`-Systemprozedur“ [*MobiLink - Serverinitiierte Synchronisation*]

Beispiel

Das folgende Beispiel sendet die Meldung "This is a test" an die IP-Adresse 10.25.99.196 an Port 2345:

```
SELECT sa_send_udp( '10.25.99.196', 2345, 'This is a test' );
```

sa_server_messages-Systemprozedur

Hiermit können Sie Meldungen im Fenster für die Datenbankservermeldungen als Ergebnismenge ausgeben.

Syntax

```
sa_server_messages(  
  [ first_msg  
  [, num_msgs ] ]  
)
```

Argumente

- **first_msg** Verwenden Sie diesen optionalen UNSIGNED BIGINT-Parameter zur Angabe der ID der ersten oder letzten Meldung je nach dem Vorzeichen des Parameters `num_msg`. Der Standardwert ist NULL, was bedeutet, dass die Suche am Beginn der Liste startet, wenn `num_msgs` NULL oder nicht negativ ist. Die Suche beginnt nach dem Ende der Liste, wenn `num_msgs` negativ ist.
- **num_msgs** Verwenden Sie diesen optionalen BIGINT-Parameter, um die Anzahl der zurückzugebenden Meldungen anzugeben. Das Vorzeichen gibt an, ob die Anforderung für Meldungen ist, die mit `first_msg` beginnen, oder bei `first_msg` enden. Der Standardwert ist NULL, das heißt, dass alle Meldungen beginnend mit der `first_msg` bis zum Ende der Liste zurückgegeben werden.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
msg_id	UNSIGNED BIGINT	Eindeutige Meldungs-ID. Meldungs-ID beginnt bei 0.

Spaltenname	Datentyp	Beschreibung
msg_text	LONG VAR-CHAR	Text der Meldung.
msg_time	TIMESTAMP	Uhrzeit, zu der die Meldung ausgegeben wurde.
msg_severity	VAR-CHAR(255)	Schweregrad der Meldung. Diese Spalte enthält einen der folgenden Werte: <ul style="list-style-type: none"> • INFO Informationsmeldung. • WARN Warnung. • ERR Fehler.
msg_category	VAR-CHAR(255)	Die Kategorie der Meldung. Diese Spalte enthält einen der folgenden Werte: <ul style="list-style-type: none"> • STARTUP/SHUTDOWN Meldungen in Verbindung mit dem Start oder dem Herunterfahren des Datenbankservers oder der Datenbank. • CHKPT Meldungen im Zusammenhang mit Checkpoints. • MSG Meldungen, die mit den Anweisungen MESSAGE oder PRINT generiert werden. • DBA_MSG Mit der MESSAGE-Anweisung generierte Meldungen, für das SERVER OPERATOR-Systemprivileg erforderlich gewesen wäre, z.B. Meldungen, die an das Ereignisprotokoll gesendet werden. • CONN Meldungen über die Verbindungen zum Datenbankserver. • OTHER Alle anderen Meldungen.
msg_database	VAR-CHAR(255)	Der der Meldung zugeordnete Datenbankname, wenn sich die Meldung auf eine bestimmte Datenbank bezieht. Sonst ist der Wert NULL.

Bemerkungen

Beim Senden neuer Meldungen an die Konsole werden alte Meldungen mit derselben Kategorie oder demselben Schweregrad gelöscht, wenn die Anzahl der Meldungen den Wert der Eigenschaft MessageCategoryLimit überschreitet. Als Ergebnis könnte es zu Lücken in der Ergebnismenge kommen und zwei aufeinanderfolgende Zeilen haben eventuell keine aufeinanderfolgenden Meldungs-IDs.

Die STARTUP/SHUTDOWN-Nachrichtenkategorie zeigt keine Nachrichten über das Herunterfahren für die Server. Nachrichten über das Herunterfahren werden nur gezeigt, wenn mehrere Datenbanken auf einem Server laufen und einer oder mehrere heruntergefahren werden.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- [MessageCategoryLimit-Servereigenschaft \[SQL Anywhere Server - Datenbankadministration\]](#)

Beispiel

Der folgende Befehl ruft 100 Meldungen beginnend mit der Meldungs-ID 3 ab:

```
CALL sa_server_messages( 3, 100 );
```

Der folgende Befehl ruft 500 Meldungen bis zur Meldung 4032 ab:

```
CALL sa_server_messages( 4032, -500 );
```

Die folgenden Befehle rufen alle Meldungen ab, die mit Meldungs-ID 3 beginnen:

```
CALL sa_server_messages( 3, NULL );
```

```
CALL sa_server_messages( 3 );
```

Der folgende Befehl ruft die ersten 100 Meldungen in der Liste ab:

```
CALL sa_server_messages( NULL, 100 );
```

Der folgende Befehl ruft die letzten 100 Meldungen in der Liste ab:

```
CALL sa_server_messages( NULL, -100 );
```

Die folgenden Befehle rufen alle Meldungen in der Liste ab:

```
CALL sa_server_messages( NULL, NULL );
```

```
CALL sa_server_messages( );
```

sa_server_option-Systemprozedur

Überschreibt eine Serveroption während der Server läuft

Syntax

```
sa_server_option(  
  opt  
  , val  
)
```

Argumente

- **opt** Verwenden Sie diesen CHAR(128)-Parameter, um einen Serveroptionsnamen anzugeben.
- **val** Verwenden Sie diesen CHAR(128)-Parameter, um den neuen Wert für die Serveroption anzugeben.

Bemerkungen

Datenbankadministratoren können mit dieser Prozedur einige Optionen des Datenbankservers vorübergehend neu definieren, ohne den Datenbankserver neu starten zu müssen.

Die mit dieser Prozedur geänderten Optionswerte werden auf ihre Standardwerte zurückgesetzt, wenn der Datenbankserver heruntergefahren wird. Wenn Sie einen Optionswert bei jedem Start des Datenbankservers ändern möchten, geben Sie beim Starten des Datenbankservers die entsprechende Datenbankserveroption (sofern vorhanden) an.

Die folgenden Optionseinstellungen können geändert werden. Standardwerte werden in Fettdruck gezeigt:

Name der Option	Werte	Zusätzliche Informationen
AutoMultiProgrammingLevel	YES, NO	<p>Wenn diese Option auf YES gesetzt ist, passt der Datenbankserver automatisch seine Multiprogramming-Stufe an. Diese steuert die maximale Anzahl an Aufgaben, die gleichzeitig aktiv sein können. Wenn Sie die manuelle Steuerung der Multiprogramming-Stufe wählen, indem Sie diese Option auf NO setzen, können Sie trotzdem Anfangs-, Mindest- und Höchstwert für die Multiprogramming-Stufe festlegen.</p> <p>Siehe „Datenbankserveroption -gna“ [SQL Anywhere Server - Datenbankadministration] und „Datenbankserverkonfiguration der Multiprogramming-Stufe“ [SQL Anywhere Server - Datenbankadministration].</p>
AutoMultiProgrammingLevelStatistics	YES, NO	<p>Wenn diese Option auf YES gesetzt ist, erscheinen im Datenbankserver-Meldungslog Statistiken zu automatischen Anpassungen der Multiprogramming-Stufe.</p> <p>Siehe „Datenbankserveroption -gns“ [SQL Anywhere Server - Datenbankadministration] und AutoMultiProgrammingLevelStatistics-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>
CacheSizingStatistics	YES, NO	<p>Wenn diese Option auf YES gesetzt ist, werden Cacheinformationen im Meldungsfenster des Datenbankservers angezeigt, sobald sich die Cachegröße ändert.</p> <p>Siehe „Datenbankserveroption -cs“ [SQL Anywhere Server - Datenbankadministration] und CacheSizingStatistics-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>

Name der Option	Werte	Zusätzliche Informationen
CollectStatistics	YES, NO	<p>Wenn diese Option auf YES gesetzt ist, sammelt der Datenbankserver Statistiken des Systemmonitors.</p> <p>Siehe „Datenbankserveroption -k“ [<i>SQL Anywhere Server - Datenbankadministration</i>] und CollectStatistics-Servereigenschaft [<i>SQL Anywhere Server - Datenbankadministration</i>].</p>
ConnsDisabled	YES, NO	<p>Wenn diese Option auf YES gesetzt ist, sind keine anderen Verbindungen zu Datenbanken auf dem Datenbankserver zugelassen.</p> <p>Siehe ConnsDisabled-Servereigenschaft [<i>SQL Anywhere Server - Datenbankadministration</i>].</p>
ConnsDisabledForDB	YES, NO	<p>Wenn diese Option auf YES gesetzt ist, sind keine anderen Verbindungen zu der aktuellen Datenbank zugelassen.</p>
ConsoleLogFile	<i>filename</i>	<p>Der Name der Datei, die zum Speichern von Protokollierungsinformationen des Datenbankservers verwendet wird. Die Angabe einer leeren Zeichenfolge stoppt die Protokollierung in die Datei. Alle Backslash-Zeichen im Pfad müssen verdoppelt werden, weil dieser Wert eine SQL-Zeichenfolge ist.</p> <p>Siehe „Datenbankserveroption -o“ [<i>SQL Anywhere Server - Datenbankadministration</i>] und ConsoleLogFile-Servereigenschaft [<i>SQL Anywhere Server - Datenbankadministration</i>].</p>
ConsoleLogMaxSize	<i>file-size</i> , in Byte	<p>Die in Byte angegebene maximale Größe der Datei, die zum Protokollieren von Protokollierungsinformationen des Datenbankservers verwendet wird. Wenn die Meldungslogdatei des Datenbankservers die durch diese Eigenschaft oder die Serveroption -on angegebene Größe erreicht, wird die Datei umbenannt, wobei die Erweiterung <i>.old</i> angehängt wird, was die Datei desselben Namens (falls vorhanden) ersetzt. Die Meldungslogdatei des Datenbankservers wird dann neu gestartet.</p> <p>Siehe „Datenbankserveroption -on“ [<i>SQL Anywhere Server - Datenbankadministration</i>] und ConsoleLogMaxSize-Servereigenschaft [<i>SQL Anywhere Server - Datenbankadministration</i>].</p>
CurrentMultiProgrammingLevel	Ganzzahl. Der Standardwert ist 20 .	<p>Setzt die Multiprogramming-Stufe des Datenbankservers.</p> <p>Siehe „Datenbankserveroption -gn“ [<i>SQL Anywhere Server - Datenbankadministration</i>] und „Datenbankserverkonfiguration der Multiprogramming-Stufe“ [<i>SQL Anywhere Server - Datenbankadministration</i>].</p>

Name der Option	Werte	Zusätzliche Informationen
DatabaseCleaner	ON, OFF	<p>Ändern Sie die Einstellung für diese Option nur, wenn Sie dazu vom Technischen Kundendienst von aufgefordert werden.</p> <p>Siehe DatabaseCleaner-Datenbankeigenschaft [SQL Anywhere Server - Datenbankadministration].</p>
Deadlock-Logging	ON, OFF, RESET, CLEAR	<p>Steuert die Deadlock-Protokollierung. Der Wert deadlock_logging wird ebenfalls unterstützt. Optionen zur Deadlock-Protokollierung können auch im Fenster Datenbankeigenschaften in Sybase Central konfiguriert werden. Folgende Werte werden unterstützt:</p> <ul style="list-style-type: none"> • ON Aktiviert die Deadlock-Protokollierung. • OFF Deaktiviert die Deadlock-Protokollierung und lässt die Deadlock-Daten zur Einsicht verfügbar • RESET Löscht die protokollierten Deadlock-Daten, falls welche vorhanden sind, und aktiviert dann die Deadlock-Protokollierung. • CLEAR Löscht die protokollierten Deadlock-Daten, falls welche vorhanden sind, und deaktiviert dann die Deadlock-Protokollierung. <p>Wenn die Deadlock-Protokollierung aktiviert ist, können Sie mithilfe der sa_report_deadlocks-Systemprozedur Deadlock-Informationen aus der Datenbank abrufen.</p> <p>Siehe „log_deadlocks-Option“ [SQL Anywhere Server - Datenbankadministration].</p>
DebuggingInformation	YES, NO	<p>Zeigt Diagnosemeldungen und andere Meldungen zum Zweck der Fehlerbehandlung an. Die Meldungen werden im Fenster der Datenbankservermeldungen angezeigt.</p> <p>Siehe „Datenbankserveroption -z“ [SQL Anywhere Server - Datenbankadministration] und DebuggingInformation-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>

Name der Option	Werte	Zusätzliche Informationen
DiskSand-box	ON, OFF	<p>Legt die Sandboxing-Standardeinstellungen für alle auf dem Datenbankserver gestarteten Datenbanken fest, die keine expliziten Sandboxing-Einstellungen haben. Wenn Sie die Sandboxing-Einstellungen mithilfe der sa_server_option-Systemprozedur ändern, hat dies keine Auswirkungen auf Datenbanken, die bereits auf dem Datenbankserver laufen. Wenn Sie die sa_server_option-Systemprozedur verwenden möchten, um Sandboxing-Einstellungen zu ändern, müssen Sie den Schlüssel für die gesicherte Funktion manage_disk_sandbox angeben.</p> <p>Siehe „Datenbankserveroption -sbx“ [SQL Anywhere Server - Datenbankadministration].</p>
Drop-BadStatistics	YES , NO	Statistiken, die fehlerhafte Schätzungen zurückgeben, können von der automatischen Statistikverwaltung aus der Datenbank gelöscht werden.
DropUnusedStatistics	YES , NO	Statistiken, die an 90 aufeinander folgenden Tagen nicht verwendet wurden, können von der automatischen Statistikverwaltung aus der Datenbank gelöscht werden.
IdleTimeout	Ganzzahl in Minuten. Der Standardwert ist 240 .	<p>Trennt TCP/IP-Verbindungen, die seit der angegebenen Anzahl von Minuten keine Anforderung übermittelt haben. Das verhindert, dass inaktive Verbindungen Sperren endlos aufrecht erhalten.</p> <p>Siehe „Datenbankserveroption -ti“ [SQL Anywhere Server - Datenbankadministration] und IdleTimeout-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>
IPAddressMonitorPeriod	Ganzzahl in Sekunden. Der Standardwert ist 120 für tragbare Geräte, andernfalls 0 .	<p>Legt die Zeit zum Überprüfen auf neue IP-Adressen in Sekunden fest. Der Mindestwert ist 10, der Standardwert 0. Für tragbare Geräte ist der Standardwert 120 Sekunden.</p> <p>Siehe „Datenbankserveroption -xm“ [SQL Anywhere Server - Datenbankadministration] und IPAddressMonitorPeriod-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>

Name der Option	Werte	Zusätzliche Informationen
Liveness-Timeout	Ganzzahl in Sekunden. Der Standardwert ist 120 .	<p>Um zu bestätigen, dass eine Verbindung besteht, wird in regelmäßigen Abständen ein Verfügbarkeitspaket über ein Client/Server-TCP/IP-Netzwerk gesendet. Wenn der Netzwerkserver eine LivenessTimeout-Periode lang läuft, ohne ein Verfügbarkeitspaket zu finden, wird die Kommunikation beendet.</p> <p>Siehe „Datenbankserveroption -tl“ [<i>SQL Anywhere Server - Datenbankadministration</i>] und LivenessTimeout-Servereigenschaft [<i>SQL Anywhere Server - Datenbankadministration</i>].</p>
MaxMulti-ProgrammingLevel	Ganzzahl. Der Standardwert ist das Vierfache des Werts für CurrentMultiProgrammingLevel.	<p>Legt die maximale Anzahl von Aufgaben fest, die der Datenbankserver gleichzeitig ausführen kann.</p> <p>Siehe „Datenbankserveroption -gnh“ [<i>SQL Anywhere Server - Datenbankadministration</i>] und „Datenbankserverkonfiguration der Multiprogramming-Stufe“ [<i>SQL Anywhere Server - Datenbankadministration</i>].</p>
Message-Category-Limit	Ganzzahl. Der Standardwert ist 400 .	<p>Definiert die Mindestanzahl von Meldungen der einzelnen Schweregrade und Kategorien, die mithilfe der sa_server_messages-Systemprozedur abgerufen werden können.</p> <p>Siehe MessageCategoryLimit-Servereigenschaft [<i>SQL Anywhere Server - Datenbankadministration</i>].</p>
MinMulti-ProgrammingLevel	Ganzzahl. Der Standardwert ist das Minimum aus dem Wert der Serveroption -gtc und der Anzahl der logischen CPUs auf dem Computer.	<p>Legt die minimale Anzahl von Aufgaben fest, die der Datenbankserver gleichzeitig ausführen kann.</p> <p>Siehe „Datenbankserveroption -gnl“ [<i>SQL Anywhere Server - Datenbankadministration</i>] und „Datenbankserverkonfiguration der Multiprogramming-Stufe“ [<i>SQL Anywhere Server - Datenbankadministration</i>].</p>

Name der Option	Werte	Zusätzliche Informationen
Option-WatchAction	MESSAGE, ERROR	<p>Legt die Aktion fest, die der Datenbankserver ausführen soll, wenn versucht wird, eine Option in der Liste zu setzen. Die unterstützten Werte sind MESSAGE (MELDUNG) und ERROR (FEHLER). Wenn OptionWatchAction auf MESSAGE eingestellt ist und eine von OptionWatchList festgelegte Option eingestellt wird, erscheint eine Meldung im Fenster der Datenbankservermeldungen, die darauf hinweist, dass die Option auf der Optionen-Überwachungsliste steht.</p> <p>Wenn OptionWatchAction auf ERROR eingestellt ist, wird ein Fehler ausgegeben, der angibt, dass die Option nicht eingestellt werden kann, weil sie auf der Optionen-Überwachungsliste steht.</p> <p>Sie können die aktuelle Einstellung dieser Eigenschaften mit der folgenden Abfrage ermitteln:</p> <pre>SELECT DB_PROPERTY('OptionWatchAction');</pre> <p>Siehe „Optionseinstellungen überwachen“ [SQL Anywhere Server - Datenbankadministration] und OptionWatchList-Datenbankeigenschaft [SQL Anywhere Server - Datenbankadministration].</p>
Option-WatchList	Kommagetrennte Liste von Datenbankoptionen.	<p>Legt eine kommagetrennte Liste der Datenbankoptionen fest, über die Sie informiert werden möchten, bzw. bei denen der Datenbankserver eine Fehlermeldung zurückgeben soll, wenn sie eingestellt werden. Die Zeichenfolgelänge ist auf 128 Byte begrenzt. Standardmäßig ist dies eine leere Zeichenfolge. Beispiel: Der folgende Befehl fügt die Optionen automatic_timestamp, float_as_double und tsql_hex_constant der Liste der überwachten Optionen hinzu:</p> <pre>CALL sa_server_option('OptionWatchList','automatic_timestamp, float_as_double,tsql_hex_constant');</pre> <p>Sie können die aktuelle Einstellung dieser Eigenschaften mit der folgenden Abfrage ermitteln:</p> <pre>SELECT DB_PROPERTY('OptionWatchList');</pre> <p>Siehe „Optionseinstellungen überwachen“ [SQL Anywhere Server - Datenbankadministration] und OptionWatchAction-Datenbankeigenschaft [SQL Anywhere Server - Datenbankadministration].</p>
Procedure-Profiling	YES, NO, RESET, CLEAR	Siehe ProcedureProfiling-Datenbankeigenschaft [SQL Anywhere Server - Datenbankadministration].

Name der Option	Werte	Zusätzliche Informationen
ProfileFilterConn	<i>connection-id</i>	<p>Fordert den Datenbankserver auf, Profilerstellungsinformationen für eine bestimmte Verbindungs-ID aufzuzeichnen, ohne dass andere Verbindungen an der Verwendung der Datenbank gehindert werden. Wenn das Filtern von Verbindungen aktiviert ist, ist der Wert, der für <code>SELECT PROPERTY('ProfileFilterConn')</code> zurückgegeben wird, die Verbindungs-ID der überwachten Verbindung. Wenn keine ID angegeben wurde oder wenn das Filtern von Verbindungen deaktiviert ist, ist der zurückgegebene Wert -1.</p> <p>Siehe ProfileFilterConn-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>
ProcessorAffinity	Kommagetrennte Liste von Prozessornummern bzw. Bereichen. Der Standardwert lautet, dass alle Prozessoren verwendet werden oder die Einstellung der Option -gta.	<p>Weist den Datenbankserver an, welche logischen Prozessoren unter Windows oder Linux verwendet werden sollen. Geben Sie eine kommagetrennte Liste von Prozessornummern bzw. Bereichen an. Wenn der niedrigere Endpunkt eines Bereichs nicht angegeben ist, wird angenommen, dass er null ist. Wenn der obere Endpunkt des Bereichs nicht angegeben ist, wird angenommen, dass er die höchste CPU-Anzahl ist, die vom Betriebssystem zugelassen wird. Die von der <code>sa_cpu_topology</code>-Systemprozedur zurückgegebene <code>in_use</code>-Spalte enthält die aktuelle Prozessoraaffinität des Datenbankservers und gibt an, ob der Datenbankserver einen Prozessor nutzt. Alternativ dazu können Sie den Wert der <code>ProcessorAffinity</code>-Datenbankservereigenschaft abfragen.</p> <p>Der Datenbankserver verwendet möglicherweise nicht alle angegebenen logischen Prozessoren in folgenden Fällen:</p> <ul style="list-style-type: none"> • Wenn mindestens einer der angegebenen logischen Prozessoren nicht vorhanden oder offline ist. • Wenn die Lizenz dies nicht zulässt. <p>Wenn Sie eine ungültige Prozessor-ID angeben, gibt <code>sa_server_option</code> einen Fehler zurück.</p> <p>Siehe „Datenbankserveroption -gta“ [SQL Anywhere Server - Datenbankadministration] und „<code>sa_cpu_topology</code>-Systemprozedur“ auf Seite 1194.</p>
ProfileFilterUser	<i>user-id</i>	<p>Weist den Datenbankserver an, Profilerstellungsinformationen für eine bestimmte Benutzer-ID aufzuzeichnen</p> <p>Siehe ProfileFilterUser-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>

Name der Option	Werte	Zusätzliche Informationen
Quitting-Time	Gültiges Datum und gültige Uhrzeit	Weist den Datenbankserver an, zu einer bestimmten Zeit herunterzufahren Siehe „Datenbankserveroption -tq“ [<i>SQL Anywhere Server - Datenbankadministration</i>] und QuittingTime-Servereigenschaft [<i>SQL Anywhere Server - Datenbankadministration</i>].
RememberLastPlan	YES, NO	Teilt dem Datenbankserver mit, den ausführlichen Textplan der letzten auf der Verbindung ausgeführten Abfrage aufzuzeichnen. Diese Einstellung wird auch von der Serveroption -zp gesteuert. Wenn RememberLastPlan aktiviert ist, können Sie die Textdarstellung des Plans für die letzte auf der Verbindung ausgeführte Abfrage anzeigen, indem der Wert der LastPlanText-Verbindungseigenschaft abfragen: <pre>SELECT CONNECTION_PROPERTY('LastPlanText');</pre> Siehe „Datenbankserveroption -zp“ [<i>SQL Anywhere Server - Datenbankadministration</i>] und RememberLastPlan-Servereigenschaft [<i>SQL Anywhere Server - Datenbankadministration</i>].

Name der Option	Werte	Zusätzliche Informationen
RememberLastStatement	YES, NO	<p>Weist den Datenbankserver an, die zuletzt vorbereitete SQL-Anweisung für jede einzelne auf dem Server laufende Datenbank aufzuzeichnen. Bei Aufrufen von gespeicherten Prozeduren erscheint nur der äußere Prozeduraufruf, nicht die Anweisungen innerhalb der Prozedur.</p> <p>Wenn RememberLastStatement aktiviert ist, erhalten Sie den aktuellen Wert von LastStatement für eine Verbindung, indem Sie den Wert der Verbindungseigenschaft LastStatement abfragen:</p> <pre>SELECT CONNECTION_PROPERTY('LastStatement');</pre> <p>Wenn das clientseitige Caching von Anweisungen aktiviert ist und eine zwischengespeicherte Anweisung wieder verwendet wird, gibt diese Eigenschaft eine leere Zeichenfolge zurück.</p> <p>Wenn RememberLastStatement aktiviert ist, gibt die folgende Anweisung die zuletzt vorbereitete Anweisung für die angegebene Verbindung zurück:</p> <pre>SELECT CONNECTION_PROPERTY('LastStatement', connection-id);</pre> <p>Die sa_conn_activity-Systemprozedur gibt dieselben Informationen für alle Verbindungen zurück.</p> <div style="border: 1px solid black; padding: 5px;"> <p>Vorsicht</p> <p>Wenn -zl angegeben oder die RememberLastStatement-Servereinstellung eingeschaltet ist, kann jeder Benutzer die sa_conn_activity-Systemprozedur aufrufen oder den Wert der LastStatement-Verbindungseigenschaft erhalten, um die zuletzt vorbereitete SQL-Anweisung eines jeden anderen Benutzers herauszufinden. Verwenden Sie diese Option mit Vorsicht und deaktivieren Sie sie, wenn sie nicht benötigt wird.</p> </div> <p>Siehe „Datenbankserveroption -zl“ [SQL Anywhere Server - Datenbankadministration] und RememberLastStatement-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>

Name der Option	Werte	Zusätzliche Informationen
Request-FilterConn	<i>connection-id</i> , -1	<p>Filtert die Informationen der Anforderungsprotokollierung, damit nur die Informationen für eine bestimmte Verbindung protokolliert werden. Durch diese Filterung können Sie die Größe der Anforderungs-Logdatei verringern, wenn Sie einen Datenbankserver mit vielen aktiven Verbindungen oder mehreren Datenbanken überwachen. Die Verbindungs-ID erhalten Sie, indem Sie den folgenden Befehl ausführen:</p> <pre>CALL sa_conn_info();</pre> <p>Wenn Sie eine bestimmte Verbindung protokollieren möchten, führen Sie nach Abruf der Verbindungs-ID die folgende Anweisung aus:</p> <pre>CALL sa_server_option('RequestFilterConn', connection-id);</pre> <p>Das Filtern bleibt aktiviert, bis es explizit zurückgesetzt bzw. der Datenbankserver heruntergefahren wird. Um das Filtern zurückzusetzen, führen Sie folgende Anweisung aus:</p> <pre>CALL sa_server_option('RequestFilterConn', -1);</pre> <p>Siehe RequestFilterConn-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>
Request-FilterDB	<i>database-id</i> , -1	<p>Filtert die Informationen der Anforderungsprotokollierung, damit nur die Informationen für eine bestimmte Datenbank protokolliert werden. Dadurch können Sie den Umfang der Anforderungs-Logdatei verringern, wenn Sie einen Server mit mehreren Datenbanken überwachen. Sie können die Datenbank-ID beziehen, indem Sie die folgende Anweisung ausführen, während Sie mit der gewünschten Datenbank verbunden sind:</p> <pre>SELECT CONNECTION_PROPERTY('DBNumber');</pre> <p>Wenn Sie nur Informationen für eine bestimmte Datenbank protokollieren möchten, führen Sie die folgende Anweisung aus:</p> <pre>CALL sa_server_option('RequestFilterDB', database-id);</pre> <p>Das Filtern bleibt aktiviert, bis es explizit zurückgesetzt bzw. der Datenbankserver heruntergefahren wird. Um das Filtern zurückzusetzen, führen Sie folgende Anweisung aus:</p> <pre>CALL sa_server_option('RequestFilterDB', -1);</pre> <p>Siehe RequestFilterDB-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>

Name der Option	Werte	Zusätzliche Informationen
Request-LogFile	<i>filename</i>	<p>Der Name der Datei, die zum Aufzeichnen von Anforderungs-informationen verwendet wird. Das Angeben einer leeren Zeichenfolge stoppt die Protokollierung in die Anforderungs-Logdatei. Wenn die Anforderungsprotokollierung aktiviert ist, aber die Anforderungs-Logdatei nicht angegeben oder auf eine leere Zeichenfolge gesetzt wurde, protokolliert der Server Anforderungen im Meldungsfenster des Datenbank-servers. Alle Backslash-Zeichen im Pfad müssen verdoppelt werden, weil dieser Wert eine SQL-Zeichenfolge ist.</p> <p>Wenn das clientseitige Caching von Anweisungen aktiviert ist, muss die <code>max_client_statements_cached</code>-Option auf 0 gesetzt werden, damit das clientseitige Caching von Anweisungen beim Erfassen des Anforderungs-logs deaktiviert wird, sofern das Log mit dem Perl-Skript <i>tracetime.pl</i> analysiert werden soll.</p> <p>Siehe „Datenbankserveroption -zo“ [SQL Anywhere Server - Datenbank-administration] und RequestLogFile-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>

Name der Option	Werte	Zusätzliche Informationen
Request-Logging	SQL, HOST-VARS, PLAN, PROCEDURES, TRIGGERS, OTHER, BLOCKS, REPLACE, ALL, YES, NONE, NO	<p>Mit diesem Aufruf wird die Protokollierung einzelner SQL-Anweisungen aktiviert, die zum Datenbankserver gesendet werden, um bei der Fehlerbehandlung zusammen mit den Datenbankserveroptionen -zr und -zo verwendet zu werden. Werte können Kombinationen des Folgenden, getrennt durch ein Pluszeichen (+) oder ein Komma, sein:</p> <ul style="list-style-type: none"> • PLAN Aktiviert die Protokollierung von Ausführungsplänen (Kurzform). Wenn die Protokollierung von Prozeduren (PROCEDURES) aktiviert ist, werden Ausführungspläne für Prozeduren ebenfalls erfasst. • HOSTVARS Aktiviert die Protokollierung von Hostvariablenwerten. Wenn Sie HOSTVARS angeben, werden die für SQL aufgelisteten Informationen ebenfalls protokolliert. • PROCEDURES Aktiviert die Protokollierung von innerhalb von Prozeduren ausgeführten Anweisungen. • TRIGGERS Aktiviert die Protokollierung von innerhalb von Triggern ausgeführten Anweisungen. • OTHER Aktiviert die Protokollierung von zusätzlichen, nicht in SQL enthaltenen Anforderungstypen, wie z.B. FETCH und PRE-FETCH. Wenn Sie allerdings OTHER, aber nicht SQL angeben, ist dies äquivalent mit der Angabe von SQL+OTHER. Das Einbeziehen von OTHER kann zu einem schnellen Anwachsen der Logdatei führen und sich negativ auf die Server-Performance auswirken. • BLOCKS Aktiviert die Protokollierung von Details, die anzeigen, wenn eine Verbindung von einer anderen Verbindung blockiert bzw. deblockiert ist. • REPLACE Ersetzt bei Beginn der Protokollierung das vorhandene Anforderungslog durch ein neues (leeres) mit demselben Namen. Ansonsten wird das vorhandene Anforderungslog geöffnet und die neuen Einträge werden an das Ende der Datei angehängt. • ALL Protokolliert die gesamten unterstützten Informationen. Dieser Wert ist äquivalent mit der Angabe von SQL+PLAN+HOSTVARS+PROCEDURES+TRIGGERS+OTHER+BLOCKS. Diese Einstellung kann zu einem schnellen Anwachsen der Logdatei führen und sich negativ auf die Server-Performance auswirken. • NO oder NONE Deaktiviert die Protokollierung in das Anforderungslog.

Name der Option	Werte	Zusätzliche Informationen
		<p>Sie können die aktuelle Einstellung dieser Eigenschaften mit der folgenden Abfrage ermitteln:</p> <pre>SELECT PROPERTY('RequestLogging');</pre> <p>Weitere Hinweise finden Sie unter „Liste der Datenbankservereigenschaften“ [SQL Anywhere Server - Datenbankadministration].</p> <p>Siehe „Datenbankserveroption -zr“ [SQL Anywhere Server - Datenbankadministration] und RequestLogging-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>
Request-LogMax-Size	<i>file-size</i> , in Byte	<p>Die Maximalgröße der Datei, die zum Erfassen von Anforderungsprotokollierungsdaten verwendet wird, in Byte. Wenn Sie 0 angeben, gibt es keine maximale Dateigröße für die Anforderungsprotokollierungsdatei und die Datei wird niemals umbenannt. Dieser Wert ist die Standardeinstellung.</p> <p>Wenn die Anforderungs-Logdatei die Größe erreicht, die entweder durch die sa_server_option-Systemprozedur oder durch die Serveroption -zs festgelegt wurde, wird an den Dateinamen die Erweiterung <i>.old</i> angehängt. (Falls eine Datei desselben Namens vorhanden ist, wird sie durch die neue Datei ersetzt.) Anschließend wird die Anforderungs-Logdatei neu gestartet.</p> <p>Siehe „Datenbankserveroption -zs“ [SQL Anywhere Server - Datenbankadministration] und RequestLogMaxSize-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>
Request-LogNum-Files	Ganzzahl	<p>Die Anzahl der aufzubewahrenden Kopien der Anforderungs-Logdatei.</p> <p>Wenn die Protokollierung von Anforderungen über längere Zeit aktiviert ist, kann die Logdatei umfangreich werden. Mit der Option -zn können Sie die Anzahl der Kopien der Anforderungs-Logdatei angeben, die aufbewahrt werden sollen.</p> <p>Siehe „Datenbankserveroption -zn“ [SQL Anywhere Server - Datenbankadministration] und RequestLogNumFiles-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>

Name der Option	Werte	Zusätzliche Informationen
Request-Timing	YES, NO	<p>Weist den Datenbankserver an, Timinginformationen für jede neue Verbindung zu verwalten. Diese Funktion ist standardmäßig deaktiviert. Wenn sie aktiviert ist, verwaltet der Datenbankserver für alle neuen Verbindungen kumulative Zeitgeber, die anzeigen, für wie lange die Verbindung auf dem Server in den verschiedenen Zuständen war. Die Änderung ist nur für neue Verbindungen wirksam und bleibt für die Dauer der jeweiligen Verbindung bestehen.</p> <p>Sie können die sa_performance_diagnostics-Systemprozedur verwenden, um eine Zusammenfassung dieser Zeitablaufinformationen zu erhalten, oder einzelne Werte abrufen, indem Sie die folgenden Verbindungseigenschaften untersuchen:</p> <ul style="list-style-type: none"> • ReqCountUnscheduled • ReqTimeUnscheduled • ReqCountActive • ReqTimeActive • ReqCountBlockIO • ReqTimeBlockIO • ReqCountBlockLock • ReqTimeBlockLock • ReqCountBlockContention • ReqTimeBlockContention <p>Wenn die Servereigenschaft RequestTiming aktiviert ist, gibt es einen geringen Overhead bei jeder Anforderung, um die zusätzlichen Zähler aufrechtzuerhalten.</p> <p>Siehe „Datenbankserveroption -zt“ [SQL Anywhere Server - Datenbankadministration] und RequestTiming-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>

Name der Option	Werte	Zusätzliche Informationen
Secure-Features	<i>feature-list</i>	<p>Ermöglicht es Ihnen, gesicherte Funktionen für einen Datenbankserver zu verwalten, der bereits läuft. <i>feature-list</i> ist eine durch Komma getrennte Liste von Funktionsnamen oder Funktionsgruppen. Durch das Hinzufügen einer Funktion in der Liste begrenzen Sie deren Verfügbarkeit. Um Elemente aus der Liste der sicheren Funktionen zu entfernen, setzen Sie ein Minuszeichen (-) vor den sicheren Funktionsnamen.</p> <p>Wenn Sie <code>sa_server_option('SecureFeatures',...)</code> aufrufen möchten, muss für die betreffende Verbindung die gesicherte Funktion <code>ManageFeatures</code> aktiviert sein. Der mit <code>-sf</code> angegebene Schlüssel (der Systemschlüssel für gesicherte Funktionen) aktiviert <code>ManageFeatures</code> sowie alle anderen Funktionen. Wenn Sie also den Systemschlüssel für gesicherte Funktionen verwendet haben, hat eine Änderung an <code>SecureFeatures</code> keine Auswirkungen auf die Verbindung. Wenn Sie jedoch einen anderen Schlüssel verwendet haben (z.B. einen Schlüssel, der mithilfe der <code>create_secure_feature_key</code>-Systemprozedur erstellt wurde), wirkt sich die Änderung möglicherweise unmittelbar auf die Verbindung aus, je nachdem, welche anderen Funktionen im Schlüssel enthalten sind.</p> <p>Eine Liste der gesicherten Funktionen finden Sie unter „Datenbankserveroption -sf“ [<i>SQL Anywhere Server - Datenbankadministration</i>]</p> <p>Änderungen, die Sie vornehmen, um den Zugriff auf Funktionen zu ermöglichen bzw. zu verhindern, werden für den Datenbankserver sofort wirksam. Die Verbindung, die die <code>sa_server_option</code>-Systemprozedur ausführt, ist möglicherweise betroffen, je nachdem, welchen Schlüssel für gesicherte Funktionen die Verbindung verwendet und ob dieser der Verbindung den Zugriff auf die angegebenen Funktionen ermöglicht.</p> <p>Wenn Sie beispielsweise zwei Funktionen sichern möchten, verwenden Sie die folgende Syntax:</p> <pre>CALL sa_server_option('SecureFeatures', 'CONSOLE_LOG,WEBCLIENT_LOG');</pre> <p>Nach dem Ausführen dieser Anweisung wird die Liste der gesicherten Funktionen gemäß den vorgenommenen Änderungen festgelegt.</p> <p>Siehe „Datenbankserveroption -sf“ [<i>SQL Anywhere Server - Datenbankadministration</i>] und „Schlüssel für gesicherte Funktionen erstellen“ [<i>SQL Anywhere Server - Datenbankadministration</i>].</p>

Name der Option	Werte	Zusätzliche Informationen
StatisticsCleaner	ON, OFF	<p>Der Aufräumvorgang für Statistiken repariert Statistiken, die schlechte Schätzungen zurückgeben, mithilfe von Tabellen-Scans. Standardmäßig wird der Aufräumvorgang für Statistiken im Hintergrund ausgeführt und hat nur geringfügige Auswirkungen auf die Performance.</p> <p>Das Deaktivieren des Aufräumvorgangs für Statistiken deaktiviert nicht den Statistikwächter, aber wenn der Aufräumvorgang für Statistiken deaktiviert ist, werden Statistiken nur während der Ausführung einer Abfrage erstellt oder repariert.</p>
WebClientLogFile	<i>filename</i>	<p>Der Name der Webdienst-Clientlogdatei. Die Webdienst-Clientlogdatei wird jedes Mal gekürzt, wenn Sie die Serveroption <code>-zoc</code> oder die <code>WebClientLogFile</code>-Eigenschaft verwenden, um den Dateinamen einzustellen oder zu entfernen. Alle Backslash-Zeichen im Pfad müssen verdoppelt werden, weil dieser Wert eine Zeichenfolge ist.</p> <p>Siehe „Datenbankserveroption <code>-zoc</code>“ [SQL Anywhere Server - Datenbankadministration] und <code>WebClientLogFile</code>-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>
WebClientLogging	ON, OFF	<p>Diese Option aktiviert und deaktiviert die Protokollierung für Webdienst-Clients. Die protokollierten Informationen enthalten HTTP-Anforderungen und Antwortdaten. Geben Sie ON an, um die Protokollierung in der Webdienst-Clientlogdatei zu starten und OFF, um die Protokollierung in der Datei zu stoppen.</p> <p>Siehe „Datenbankserveroption <code>-zoc</code>“ [SQL Anywhere Server - Datenbankadministration] und <code>WebClientLogging</code>-Servereigenschaft [SQL Anywhere Server - Datenbankadministration].</p>

Privilegien

Sie müssen das `MANAGE PROFILING`-Systemprivileg haben, um die folgenden Optionen verwenden zu können, die mit Anwendungsprofilerstellung oder Anforderungsprotokollierung zusammenhängen:

- ProcedureProfiling
- ProfileFilterConn
- ProfileFilterUser
- RequestFilterConn
- RequestFilterDB
- RequestLogFile
- RequestLogging
- RequestLogMaxSize
- RequestLogNumFiles

Für alle anderen Optionen müssen Sie das `SERVER OPERATOR`-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Prozedurprofilerstellung mit Systemprozeduren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Tipp: Spaltenstatistiken aktualisieren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „sa_performance_diagnostics-Systemprozedur“ auf Seite 1279
- „Liste der Verbindungseigenschaften“ [*SQL Anywhere Server - Datenbankadministration*]
- „log_deadlocks-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -sk“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -sf“ [*SQL Anywhere Server - Datenbankadministration*]
- „sa_db_option-Systemprozedur“ auf Seite 1199

Beispiel

Die folgende Anweisung bewirkt, dass die Cacheinformationen im Meldungsfenster des Datenbankservers angezeigt werden, sobald sich die Cachegröße ändert.

```
CALL sa_server_option( 'CacheSizingStatistics', 'YES' );
```

Die folgende Anweisung verbietet neue Verbindungen zur aktuellen Datenbank:

```
CALL sa_server_option( 'ConnsDisabledForDB', 'YES' );
```

Die folgende Anweisung aktiviert die Protokollierung für alle SQL-Anweisungen, Prozeduraufrufe, Pläne sowie blockierenden bzw. entblockierenden Ereignisse und startet ein neues Anforderungslog:

```
CALL sa_server_option( 'RequestLogging', 'SQL+PROCEDURES+BLOCKS+PLAN  
+REPLACE' );
```

sa_set_http_header-Systemprozedur

Mit dieser Funktion kann ein Webdienst einen HTTP-Antwortheader festlegen.

Syntax

```
sa_set_http_header(  
  fldname  
  , val  
)
```

Argumente

- **fldname** Verwenden Sie diesen CHAR(128)-Parameter, um eine Zeichenfolge anzugeben, die den Namen eines der HTTP-Headerfelder enthält.
- **val** Verwenden Sie diesen LONG VARCHAR-Parameter, um den Wert anzugeben, auf den der benannte Parameter gesetzt werden soll. Wenn Sie einen Antwortheader auf NULL setzen, wird dieser entfernt.

Bemerkungen

Mit dem speziellen Headerfeld @HttpStatus wird der mit der Anforderung zurückgegebene Statuscode festgelegt. Der Statuscode wird auch "Antwortcode" genannt. Mit dem folgenden Befehl wird z.B. der Statuscode auf "404 Not Found" gesetzt:

```
CALL sa_set_http_header( '@HttpStatus', '404' );
```

Sie können eine benutzerdefinierte Statusmeldung erstellen, indem Sie einen dreistelligen Statuscode mit einer optionalen, durch Doppelpunkt getrennten Textnachricht angeben. Zum Beispiel gibt das folgende Skript einen Statuscode mit der Nachricht "999 User Code" aus:

```
CALL sa_set_http_header( '@HttpStatus', '999:User Code' );
```

Hinweis

Eine benutzerdefinierte Status-Textnachricht wird nicht in einen Datenbankzeichensatz konvertiert, wenn sie unter Verwendung der Protokolloption LogOptions protokolliert wurde.

Der Hauptteil der Fehlermeldung wird automatisch eingefügt. Es können nur gültige HTTP-Fehlercodes benutzt werden. Wird der Status auf einen ungültigen Code gesetzt, erscheint ein SQL-Fehler.

Die sa_set_http_header-Prozedur überschreibt stets den vorhandenen Headerwert im Headerfeld, wenn sie aufgerufen wird.

Automatisch vom Datenbankserver erstellte Antwortheader können entfernt werden. Mit dem folgenden Befehl wird beispielsweise der Antwortheader "Expires" entfernt:

```
CALL sa_set_http_header( 'Expires', NULL );
```

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „NEXT_HTTP_RESPONSE_HEADER-Funktion [Webdienst]“ auf Seite 326
- „HTTP_RESPONSE_HEADER-Funktion [Webdienst]“ auf Seite 285
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162
- „LogOptions-Protokolloption (LOPT)“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Das folgende Beispiel stellt das Content-Type-Headerfeld auf "text/html" ein.

```
CALL sa_set_http_header( 'Content-Type', 'text/html' );
```

sa_set_http_option-Systemprozedur

Mit dieser Funktion kann ein Webdienst eine HTTP-Option für die Prozesssteuerung festlegen.

Syntax

```
sa_set_http_option(  
  optname  
  , val  
)
```

Argumente

- **optname** Verwenden Sie diesen CHAR(128)-Parameter, um eine Zeichenfolge anzugeben, die den Namen einer der HTTP-Optionen enthält.

Die unterstützten Optionen sind folgende:

- **CharsetConversion** Verwenden Sie diese Option, um zu steuern, ob die Ergebnismenge automatisch von der Zeichensatzkodierung der Datenbank in die Zeichensatzkodierung des Clients konvertiert werden soll. Die einzigen zulässigen Werte sind ON und OFF. Der Standardwert ist ON.
- **AcceptCharset** Verwenden Sie diese Option, um die Voreinstellungen des Webservers für die Zeichensatzkodierung von Antworten anzugeben. Eine oder mehrere Zeichensatzkodierungen können in der Reihenfolge der Präferenz angegeben werden. Die Syntax dieser Option entspricht der Syntax, die bei der Angabe des Anforderungsheader-Felds "Accept-Charset" im RFC2616 Hypertext Transfer Protocol verwendet wird.

Ein HTTP-Client, wie z.B. ein Webbrowser, liefert möglicherweise einen Accept-Charset-Anforderungsheader, der eine nach Präferenz sortierte Liste von Zeichensatzkodierungen enthält. Optional kann jeder einzelnen Kodierung ein Qualitätswert zugeordnet werden ($q=qvalue$), der die Präferenz des Clients für die Kodierung darstellt. Standardmäßig ist der Qualitätswert 1 ($q=1$). Ein Beispiel:

```
Accept-Charset: iso-8859-5, utf-8;q=0.8
```

Ein Pluszeichen (+) im Wert der HTTP-Option AcceptCharset kann als Abkürzung für die aktuelle Zeichensatzkodierung der Datenbank verwendet werden. Das Pluszeichen zeigt außerdem an, dass die Zeichensatzkodierung der Datenbank Vorrang haben soll, wenn der Client die Kodierung ebenfalls in seiner Liste angibt, und zwar unabhängig von dem vom Client zugeordneten Qualitätswert.

Mit einem Sternchen (*) in der HTTP-Option AcceptCharset kann angezeigt werden, dass der Webdienst bei Fehlen einer Schnittmenge zwischen den Listen von Server und Client eine vom Client bevorzugte Zeichensatzkodierung verwenden soll, sofern diese auch vom Server unterstützt wird.

Beim Senden der Antwort wird die erste Zeichensatzkodierung verwendet, die sowohl vom Client als auch vom Webdienst bevorzugt wird. Die Präferenzreihenfolge des Clients hat Vorrang. Wenn keine gemeinsame Kodierungspräferenz vorhanden ist, wird die vom Webdienst bevorzugte Kodierung verwendet, es sei denn, in der Webdienst-Liste erscheint ein Sternchen (*). In diesem Fall wird die vom Client bevorzugte Kodierung verwendet.

Ohne die HTTP-Option AcceptCharset wird die Zeichensatzkodierung mit der höchsten Präferenz verwendet, die vom Client angegeben und vom Server unterstützt wird. Wenn keine der vom

Client angegebenen Kodierungen unterstützt wird (oder der Client keinen Accept-Charset-Anforderungsheader sendet), wird die Zeichensatzkodierung der Datenbank verwendet.

Wenn ein Client keinen Accept-Charset-Anforderungsheader sendet, wird eine der folgenden Aktionen ausgeführt:

- Falls die HTTP-Option AcceptCharset nicht angegeben wurde, verwendet der Webserver die Zeichensatzkodierung der Datenbank.
- Falls die HTTP-Option AcceptCharset angegeben wurde, verwendet der Webserver seine bevorzugte Zeichensatzkodierung.

Wenn ein Client einen Accept-Charset-Anforderungsheader sendet, wird eine der folgenden Aktionen ausgeführt:

- Falls die HTTP-Option AcceptCharset nicht angegeben wurde, versucht der Webserver, eine der vom Client bevorzugten Zeichensatzkodierungen zu verwenden, beginnend mit der höchsten Präferenz. Wenn der Webserver keine der vom Client bevorzugten Kodierungen unterstützt, verwendet er die Zeichensatzkodierung der Datenbank.
- Falls die HTTP-Option AcceptCharset angegeben wurde, versucht der Webserver, die Zeichensatzkodierung mit der höchsten Präferenz zu verwenden, die in beiden Listen enthalten ist, beginnend mit der Kodierung, die für den Client die höchste Präferenz aufweist. Wenn der Client beispielsweise einen Accept-Charset-Anforderungsheader mit einer Liste sendet, die in der Reihenfolge der Präferenz die Kodierungen iso-a, iso-b und iso-c enthält, und der Webserver iso-b gegenüber iso-a und schließlich iso-c bevorzugt, wird iso-a ausgewählt.

```
Web client: iso-a, iso-b, iso-c
Web server: iso-b, iso-a, iso-c
```

Wenn die Schnittmenge der beiden Tabellen leer ist, wird der vom Webserver bevorzugte Zeichensatz verwendet. Im folgenden Beispiel wird Kodierung iso-d verwendet.

```
Web client: iso-a, iso-b, iso-c
Web server: iso-d, iso-e, iso-f
```

Wenn in der HTTP-Option AcceptCharset ein Sternchen (*) enthalten wäre, würde den vom Client bevorzugten Kodierungen der Vorrang gegeben, woraus sich die Verwendung von iso-a ergibt. Im Wesentlichen sorgt die Verwendung eines Sternchens dafür, dass die Schnittmenge der beiden Listen nicht leer sein kann.

Die ideale Situation tritt ein, wenn sowohl der Client als auch der Webdienst die Zeichensatzkodierung der Datenbank verwenden, weil dann die Zeichensatzkonvertierung entfällt und sich dadurch die Antwortzeit des Webserver verbessert.

Wenn die CharsetConversion-Option auf OFF gesetzt wurde, wird die AcceptCharset-Verarbeitung nicht ausgeführt.

- **SessionID** Verwenden Sie diese Option, um eine HTTP-Sitzung zu erstellen, zu löschen oder umzubenennen. Die Datenbankverbindung bleibt erhalten, wenn ein Webdienst diese Option zum Erstellen einer HTTP-Sitzung setzt, aber Sitzungen bleiben nicht über einen Neustart des Servers hinaus erhalten. Wenn Sie sich bereits in einem Sitzungskontext befinden, benennen Sie mit

diesem Aufruf die Sitzung in die neue Sitzungs-ID um. Wenn die Sitzung mit einem Nullwert aufgerufen wurde, wird sie beim Beenden des Webdiensts gelöscht.

Die generierten Sitzungsschlüssel sind auf eine Länge von 128 Zeichen beschränkt und, falls mehrere Datenbanken geladen sind, datenbankübergreifend eindeutig.

- **SessionTimeout** Verwenden Sie diese Option, um anzugeben, wie viele Minuten die HTTP-Sitzung während einer Inaktivitätsphase erhalten bleibt. Diese Zeitüberschreitung wird jedes Mal zurückgesetzt, wenn eine HTTP-Anforderung die gegebene Sitzung verwendet. Die Sitzung wird automatisch gelöscht, wenn der SessionTimeout-Wert überschritten wird.
- **val** Verwenden Sie diesen LONG VARCHAR-Parameter, um den Wert anzugeben, auf den die benannte Option gesetzt werden soll.

Bemerkungen

Verwenden Sie diese Prozedur innerhalb von Anweisungen oder Prozeduren im Zusammenhang mit Webdiensten, um Optionen zu setzen.

Wenn sa_set_http_option aus einer über einen Webdienst aufgerufenen Prozedur aufgerufen wird und entweder die Option oder ein Optionswerts ungültig ist, wird ein Fehler zurückgegeben.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Hinweise zur Zeichensatzkonvertierung“ [[SQL Anywhere Server - Programmierung](#)]
- „HTTP-Sitzungsverwaltung auf einem HTTP-Server“ [[SQL Anywhere Server - Programmierung](#)]
- „Liste der Verbindungseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Beispiele

Das folgende Beispiel zeigt die Verwendung von sa_set_http_option zum Anzeigen der Präferenzen des Webdiensts im Hinblick auf die Zeichensatzkodierung der Datenbank. Die UTF-8-Kodierung ist als zweite Wahl angegeben. Das Sternchen (*) bedeutet, dass der Webdienst auch die vom Client bevorzugte Zeichensatzkodierung verwenden wird, vorausgesetzt, diese wird vom Webserver unterstützt.

```
CALL sa_set_http_option( 'AcceptCharset', '+,UTF-8,*');
```

Das folgende Beispiel zeigt die Verwendung von sa_set_http_option zum korrektem Identifizieren der vom Webdienst verwendeten Zeichenkodierung. In diesem Beispiel ist der Webserver mit einer 1251CYR-Datenbank verbunden und kann jeden Webbrowser mit HTML-Dokumenten versorgen, die das kyrillische Alphabet enthalten.

```
CREATE OR REPLACE PROCEDURE cyrillic_html()  
RESULT (html_doc XML)
```

```

BEGIN
  DECLARE pos INT;
  DECLARE charset VARCHAR(30);
  CALL sa_set_http_option( 'AcceptCharset', 'iso-8859-5, utf-8' );
  SET charset = CONNECTION_PROPERTY( 'CharSet' );
  -- Change any IANA labels like ISO_8859-5:1988
  -- to ISO_8859-5 for Firefox.
  SET pos = LOCATE( charset, ':' );
  IF pos > 0 THEN
    SET charset = LEFT( charset, pos - 1 );
  END IF;
  CALL sa_set_http_header( 'Content-Type', 'text/html; charset=' ||
    charset );
  SELECT '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">' ||
    XMLCONCAT(
      XMLELEMENT('HTML',
        XMLELEMENT('HEAD',
          XMLELEMENT('TITLE', 'Cyrillic characters')
        ),
        XMLELEMENT('BODY',
          XMLELEMENT('H1', 'First 5 lowercase Russian letters'),
          XMLELEMENT('P', UNISTR('\u0430\u0431\u0432\u0433\u0434'))
        )
      )
    );
END;

CREATE SERVICE cyrillic
TYPE 'RAW'
AUTHORIZATION OFF
USER DBA
AS CALL cyrillic_html();

```

Der folgende Accept-Charset-Header veranschaulicht den Prozess zur Festlegung der zu verwendenden Zeichensatzkodierung, die von einem Webbrowser wie Firefox an den Webdienst geliefert wird. Er zeigt an, dass der Browser die Kodierungen ISO-8859-1 und UTF-8 bevorzugt, aber auch andere akzeptiert.

```
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

Der Webdienst akzeptiert nicht die Zeichensatzkodierung ISO-8859-1, da die zu übertragende Webseite kyrillische Zeichen enthält. Der Webdienst bevorzugt die Kodierung ISO-8859-5 oder UTF-8, wie durch den Aufruf von `sa_set_http_option` angegeben. In diesem Beispiel wird die UTF-8-Kodierung gewählt, da sie von beiden Seiten bevorzugt wird. Die `CharSet`-Datenbankverbindungseigenschaft zeigt an, welche Kodierung vom Webdienst ausgewählt wurde. Die `sa_set_http_header`-Prozedur wird verwendet, um die Kodierung des HTML-Dokuments dem Webbrowser anzuzeigen.

```
Content-Type: text/html; charset=UTF-8
```

Wenn der Webbrowser keinen Accept-Charset-Wert angibt, verwendet der Webdienst standardmäßig seine erste Präferenz, ISO-8859-5. Die `sa_set_http_header`-Prozedur wird verwendet, um die Kodierung des HTML-Dokuments anzuzeigen.

```
Content-Type: text/html; charset=ISO_8859-5
```

Im folgenden Beispiel wird eine eindeutige HTTP-Sitzungs-ID festgelegt:

```

BEGIN
  DECLARE sessionid VARCHAR(30);
  DECLARE tm TIMESTAMP;

```

```
SET tm = NOW(*);
SET sessionid = 'MySessions_' ||
    CONVERT( VARCHAR, SECONDS(tm)*1000 + DATEPART(millisecond,tm));
SELECT sessionid;
CALL sa_set_http_option('SessionID', sessionid);
END;
```

Im folgenden Beispiel wird die Zeitüberschreitung für eine HTTP-Sitzung auf 5 Minuten festgelegt:

```
CALL sa_set_http_option('SessionTimeout', '5');
```

sa_set_soap_header-Systemprozedur

Ermöglicht das Setzen von SOAP-Headern für SOAP-Antworten. Diese Prozedur wird in gespeicherten Prozeduren verwendet, die von SOAP-Webdiensten aufgerufen werden.

Syntax

```
sa_set_soap_header(
    fldname
    , val
)
```

Argumente

- **fldname** Verwenden Sie diesen CHAR(128)-Parameter, um den Headerschlüssel anzugeben, eine eindeutige Zeichenfolge, die den angegebenen Headereintrag referenziert (und nicht mit dem localname-Eintrag von *val* identisch sein muss).
- **val** Verwenden Sie diesen LONG VARCHAR-Parameter, um den unformatierten XML-Code eines Headereintrags der obersten Ebene und seiner untergeordneten Objekte im Bereich eines SOAP-Header-Elements anzugeben.

Bemerkungen

Alle SOAP-Headereinträge, die mit dieser Prozedur gesetzt werden, werden innerhalb des SOAP-Header-Elements serialisiert, wenn die SOAP-Antwortnachricht gesendet wird. Ein *val* von NULL wird nicht serialisiert. Wenn keine Headereinträge für eine SOAP-Antwort vorhanden sind, wird innerhalb des SOAP-Envelopes kein einschließendes Header-Element erstellt.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Praktische Einführung: Verwenden von SQL Anywhere für den Zugriff auf einen SOAP/DISH-Dienst“ [[SQL Anywhere Server - Programmierung](#)]
- „Webdienstfunktionen“ auf Seite 162
- „Webdienst-Systemprozeduren“ auf Seite 1162

Beispiel

Das folgende Beispiel setzt den SOAP-Header "welcome" auf "Hello".

```
CALL sa_set_soap_header( 'welcome', '<welcome>Hello</welcome>' )
```

sa_set_tracing_level-Systemprozedur

Initialisiert die Stufe der Protokollierungsinformationen, die in Diagnoseprotokollierungstabellen gespeichert werden sollen

Syntax

```
sa_set_tracing_level(  
    level  
    [, specified_scope  
    [, specified_name  
    [, do_commit ] ] ]  
)
```

Argumente

- **level** Verwenden Sie diesen INTEGER-Parameter, um die Stufe der durchzuführenden Diagnoseprotokollierung anzugeben. Zu den möglichen Werten gehören:
 - **0** Keine Protokollierungsdaten generieren. Diese Stufe hält die Protokollierungssitzung geöffnet, sendet aber keine Protokollierungsdaten an die Diagnoseprotokollierungstabellen.
 - **1** Legt eine Grundstufe der Protokollierung fest
 - **2** Legt eine mittlere Stufe der Protokollierung fest
 - **3** Legt eine hohe Stufe der Protokollierung fest
- **specified_scope** Verwenden Sie diesen optionalen LONG VARCHAR-Parameter, um den Bereich der Protokollierung anzugeben, z.B. USER, DATABASE, CONNECTION_NAME, TRIGGER usw. Der Standardwert ist NULL.
- **specified_name** Verwenden Sie diesen optionalen LONG VARCHAR-Parameter, um den Bezeichner für das in *specified_scope* angegebene Objekt festzulegen. Der Standardwert ist NULL.
- **do_commit** Mit diesem optionalen TINYINT-Parameter können Sie angeben, ob von dieser Prozedur eingefügte Zeilen automatisch festgeschrieben werden. Der Standardwert ist 1. Geben Sie 1 an, um die Zeilen automatisch festzuschreiben (empfohlen), und 0, um sie nicht automatisch festzuschreiben.

Bemerkungen

Diese Prozedur ersetzt die Zeilen in der sa_diagnostic_tracing_level-Tabelle, wobei die Protokollierungsstufen und der Bereich auf die angegebenen Einstellungen geändert werden, wenn die Prozedur aufgerufen wird.

Eine Einstellung von 0 stoppt nicht die Protokollierungssitzung. Stattdessen bleibt die Protokollierungssitzung mit der Protokollierungsdatenbank verbunden, aber es werden keine Protokollierungsdaten gesendet. Die Protokollierungssitzung bleibt weiterhin aktiv, wenn die Stufe 0 ist.

Die Systemprozedur muss von der Datenbank, deren Profil erstellt wird, aufgerufen werden.

Privilegien

Sie müssen die DIAGNOSTICS-Systemrolle und das MANAGE PROFILING-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Benutzerdefinierte Diagnoseprotokollierungsstufen“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Bereiche der Diagnoseprotokollierung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „sa_diagnostic_tracing_level-Tabelle“ auf Seite 1156
- „Diagnoseprotokollierung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Beispiele

Im folgenden Beispiel wird die Protokollierungsebene auf 1 gesetzt. Für die gesamte Datenbank wird ein Profil mit Performance-Zählerdaten erstellt. Außerdem werden Beispiele für ausgeführte Anweisungen gezeigt:

```
CALL sa_set_tracing_level( 1 );
```

Im folgenden Beispiel wird die Protokollierungsstufe auf 3 gesetzt und der Benutzer AG84756 angegeben. Nur dem Benutzer AG84756 zugeordnete Aktivitäten werden protokolliert:

```
CALL sa_set_tracing_level( 3, 'user', 'AG84756' );
```

sa_snapshots-Systemprozedur

Gibt eine Liste von derzeit aktiven Snapshots zurück

Syntax

```
sa_snapshots( )
```

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
connection_num	INTEGER	Die Verbindungs-ID der Verbindung, auf der der Snapshot läuft
start_sequence_num	UNSIGNED BIGINT	Eine eindeutige, den Snapshot identifizierende Nummer

Spaltenname	Datentyp	Beschreibung
statement_level	BIT	TRUE, wenn der Snapshot mit der Isolationsstufe "statement-snapshot" oder mit der Isolationsstufe "readonly-statement-snapshot" erstellt wurde. Sonst FALSE.

Bemerkungen

Es können mehrere Anweisungs-Snapshots auf einer Verbindung vorhanden sein. Im Fall von verschachtelten oder überlappenden Anweisungen, die mit der Isolationsstufe statement-snapshot laufen, beginnt jede einen anderen Anweisungs-Snapshot beim ersten Lese- oder Aktualisierungsvorgang.

Gewöhnlich gibt es nur einen Transaktions-Snapshot pro Verbindung (ein Eintrag pro Verbindung in sa_snapshots mit statement_level=0). Ein mit einem Cursor verknüpfter Snapshot allerdings ändert sich nicht mehr nach dem ersten Abruf des Cursors, und ein mit WITH HOLD geöffneter Cursor bleibt bis zum Festschreiben oder Zurücksetzen geöffnet. Wenn der Cursor einen zugeordneten Snapshot hat, bleibt auch der Snapshot bestehen. Daher ist es möglich, dass mehrere Transaktions-Snapshots für dieselbe connection_num bestehen: Einer für den aktuellen Transaktions-Snapshot und einer oder mehrere für alte Transaktions-Snapshots, die aufgrund von WITH HOLD-Cursor weiter bestehen.

Privilegien

Sie müssen das MONITOR-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „sa_transactions-Systemprozedur“ auf Seite 1347
- „Snapshot-Isolation“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Beispiel

Sie können die folgende Abfrage ausführen, um die derzeit aktiven Snapshots zu ermitteln.

```
CALL sa_snapshots( );
```

sa_split_list-Systemprozedur

Nimmt eine Zeichenfolge von durch einen Begrenzer getrennten Werten und gibt eine Reihe von Zeilen zurück, wobei eine Zeile für jeden Wert steht

Syntax

```
sa_split_list(  
  str  
  [, delim  
  [, maxlen ] ]  
)
```

Argumente

- **str** Verwenden Sie diesen LONG VARCHAR-Parameter, um die Zeichenfolge anzugeben, die die aufzuteilenden und durch *delim* getrennten Werte enthält.
- **delim** Verwenden Sie diesen optionalen CHAR(10)-Parameter, um den Begrenzer anzugeben, der in *str* zum Trennen der Werte verwendet werden soll. Der Begrenzer kann eine Zeichenfolge aus beliebigen Zeichen sein (bis zu 10 Byte). Wenn kein *delim* angegeben ist, wird standardmäßig ein Komma verwendet.
- **maxlen** Verwenden Sie diesen optionalen INTEGER-Parameter, um die Maximallänge der zurückgegebenen Werte anzugeben. Beispiel: Wenn *maxlen* auf 3 gesetzt ist, werden die Werte in den Ergebnissen auf die Länge von 3 Zeichen gekürzt. Wenn Sie 0 angeben (Standardwert), können Werte von beliebiger Länge sein.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
line_num	INTEGER	Sequenzielle Nummer für die Zeile
row_value	LONG VARCHAR	Wert aus der Zeichenfolge, ggf. auf <i>maxlen</i> gekürzt

Bemerkungen

Die Prozedur `sa_split_list` akzeptiert eine Zeichenfolge mit einer begrenzten Liste von Werten und gibt eine Ergebnismenge mit einem Wert pro Zeile zurück. Dies ist das Gegenteil der Aktion, die durch die LIST-Funktion [Aggregate] ausgelöst wird. Eine leere Zeichenfolge wird für `row_value` zurückgegeben, wenn für die Zeichenfolge Folgendes gilt:

- Sie beginnt mit *delim*.
- Sie enthält zwei aufeinanderfolgende Instanzen von *delim* in der Mitte der Zeichenfolge.
- Sie endet mit *delim*.

Ein Leerzeichen in der Eingabe-Zeichenfolge ist signifikant. Wenn der Begrenzer ein Leerzeichen ist und in der Eingabe-Zeichenfolge zusätzliche Leerzeichen stehen, werden in der Ergebnismenge zusätzliche Zeilen eingefügt. Wenn der Begrenzer kein Leerzeichen ist, werden Leerzeichen in der Eingabe-Zeichenfolge aus den Werten in der Ergebnismenge nicht entfernt.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- [„LIST-Funktion \[Aggregat\]“ auf Seite 302](#)

Beispiele

Die folgende Abfrage gibt eine Liste von Produkten mit schwarzer Farbe zurück.

```
SELECT list( Name )
FROM Products
WHERE Color = 'Black';
```

list (Products.Name)
Tee Shirt,Baseball Cap,Visor,Shorts

Im folgenden Beispiel wird die Prozedur sa_split_list benutzt, um die ursprüngliche Ergebnismenge aus der zusammengestellten Liste zu entfernen.

```
SELECT *
FROM sa_split_list( 'Tee Shirt,Baseball Cap,Visor,Shorts' );
```

line_num	row_value
1	Tee Shirt
2	Baseball Cap
3	Visor
4	Shorts

Das folgende Beispiel gibt eine Zeile für jedes Wort zurück. Um eine Rückgabe von Wörtern zu vermeiden, wenn row_value eine leere Zeichenfolge ist, muss die WHERE-Klausel angegeben werden.

```
SELECT *
FROM sa_split_list( 'one|three|four|six|', '|' )
WHERE row_value <> '';
```

line_num	row_value
1	one
3	three
4	four
6	six

Im folgenden Beispiel wird eine Prozedur namens "ProductsWithColor" erstellt. Wenn sie aufgerufen wird, verwendet die ProductsWithColor-Prozedur sa_split_list, um die vom Benutzer angegebenen Farbenwerte syntaktisch zu analysieren, überprüft die Color-Spalte der Products-Tabelle und gibt Namen, Beschreibung, Größe und Farbe für jedes Produkt zurück, das einer der vom Benutzer angegebenen Farben entspricht.

Das Ergebnis des untenstehenden Prozeduraufrufs besteht aus Namen, Beschreibung, Größe und Farbe aller Produkte, die entweder weiß oder schwarz sind.

```
CREATE OR REPLACE PROCEDURE ProductsWithColor( IN color_list LONG VARCHAR )
BEGIN
```

```

SELECT Name, Description, Size, Color
FROM Products
WHERE Color IN ( SELECT row_value FROM sa_split_list( color_list ) );
END;

SELECT * from ProductsWithColor( 'white,black' );

```

sa_stack_trace-Systemprozedur

Gibt das Stack-Trace zurück, das zum aktuellen Aufruf-Speicherort führt.

Syntax

```
sa_stack_trace()
```

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
StackLevel	UNSIGNED SMALLINT	Die aktuelle Zeile hat Stack-Stufe 1.
UserName	CHAR(128)	Der Name des Eigentümers der Prozedur oder des Triggers bzw. NULL, wenn sich die aktuelle Stufe in einem Batch befindet.
ProcName	CHAR(128)	Der Name der Prozedur oder des Triggers, wo der Aufruf ausgeführt wurde, oder der Batch-Typ.
LineNumber	UNSIGNED INTEGER	Die Zeilennummer des Aufrufs in der Prozedur, dem Trigger oder dem Batch.

Bemerkungen

Jeder Datensatz in der Ergebnismenge stellt einen einzelnen Aufruf an den Stack dar. Wenn die zusammengesetzte Anweisung nicht Teil einer Prozedur, einer Funktion, eines Triggers oder eines Ereignisses ist, wird statt des Prozedurnamens der Batch-Typ (watcom_batch oder tsql_batch) zurückgegeben.

Diese Prozedur gibt dieselben Informationen zurück wie die STACK_TRACE-Funktion.

Privilegien

Keine

Nebenwirkungen

Keine.

Siehe auch

- „TRY-Anweisung“ auf Seite 1093
- „BEGIN-Anweisung“ auf Seite 557
- „ERROR_LINE-Funktion [Verschiedene]“ auf Seite 244
- „ERROR_MESSAGE-Funktion [Verschiedene]“ auf Seite 245
- „ERROR_PROCEDURE-Funktion [Funktionstyp]“ auf Seite 246
- „ERROR_SQLCODE-Funktion [Verschiedene]“ auf Seite 247
- „ERROR_SQLSTATE-Funktion [Verschiedene]“ auf Seite 248
- „ERROR_STACK_TRACE-Funktion [Verschiedene]“ auf Seite 249
- „STACK_TRACE-Funktion [Verschiedene]“ auf Seite 392
- „sa_error_stack_trace-Systemprozedur“ auf Seite 1216
- „Verschachtelte zusammengesetzte Anweisungen und Ausnahmeroutinen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Beispiel

Dieses Beispiel zeigt, wie Sie die Spalten der Ergebnismenge aus der sa_stack_trace-Systemprozedur abrufen:

```
SELECT StackLevel, UserName, ProcName, LineNumber FROM sa_stack_trace();
```

Wenn diese Anweisung außerhalb des Kontexts einer gespeicherten Prozedur ausgeführt wird, ist die Ergebnismenge leer.

Das folgende Beispiel zeigt die Implementierung einer allgemeinen Stack-Trace-Prozedur, die ihre Ergebnisse an das Clientfenster sendet. Ein Beispiel für ihre Verwendung ist enthalten.

```
CREATE OR REPLACE PROCEDURE StackDump( MSG CHAR(128) )
BEGIN
    DECLARE myStackLevel UNSIGNED SMALLINT;
    DECLARE myUserName CHAR(128);
    DECLARE myProcName CHAR(128);
    DECLARE myLineNumber UNSIGNED SMALLINT;
    DECLARE err_notfound
        EXCEPTION FOR SQLSTATE '02000';
    DECLARE myStack CURSOR FOR
        SELECT StackLevel, UserName, ProcName, LineNumber FROM
sa_stack_trace();
    MESSAGE 'Stack Trace: ' || MSG TO CLIENT;
    OPEN myStack;
    StackLoop: LOOP
        FETCH NEXT myStack
            INTO myStackLevel, myUserName, myProcName, myLineNumber;
        IF SQLSTATE = err_notfound THEN
            LEAVE StackLoop;
        END IF;
        IF myStackLevel != 1 THEN
            MESSAGE myStackLevel - 1 || ' ' || myUserName || ' ' ||
myProcName || ' ' || myLineNumber
                TO CLIENT;
        ENDIF
    END LOOP StackLoop;
    CLOSE myStack;
END;

CREATE OR REPLACE PROCEDURE Proc1()
BEGIN
```

```

        CALL Proc2();
    END;

    CREATE OR REPLACE PROCEDURE Proc2()
    BEGIN
        CALL Proc3();
    END;

    CREATE OR REPLACE PROCEDURE Proc3()
    BEGIN
        CALL StackDump('Snapshot from Proc3');
    END;

    CALL Proc1();

```

Die Ausgabe im Interactive SQL-Fenster **Meldungen** ist unten dargestellt.

```

Stack Trace: Snapshot from proc3
1 DBA proc3 3
2 DBA proc2 3
3 DBA proc1 3

```

sa_statement_text-Systemprozedur

Formatiert eine SELECT-Anweisung, sodass jedes Element auf einer eigenen Zeile erscheint. Dies ist nützlich, wenn Sie lange Anweisungen aus dem Anforderungslog anzeigen möchten, bei denen alle Zeilenschaltungen entfernt wurden.

Syntax

sa_statement_text(*txt*)

Argumente

- **txt** Verwenden Sie diesen LONG VARCHAR-Parameter, um eine SELECT-Anweisung anzugeben.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
stmt_text	LONG VARCHAR	Eine Klausel der SELECT-Anweisung.

Bemerkungen

Beim eingegebenen Text muss es sich um eine Zeichenfolge (in Apostrophen) oder einen Zeichenfolgenausdruck handeln.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „sa_get_request_times-Systemprozedur“ auf Seite 1229
- „sa_get_request_profile-Systemprozedur“ auf Seite 1228

Beispiel

Der folgende Aufruf formatiert eine SELECT-Anweisung, sodass jedes Element in einer eigenen Zeile erscheint.

```
CALL sa_statement_text( 'SELECT * FROM car WHERE name='Audi' ' );
```

	stmt_text
1	SELECT *
2	FROM car
3	WHERE name = 'Audi'

sa_table_fragmentation-Systemprozedur

Gibt Auskunft über die Fragmentierung von Datenbanktabellen

Syntax

```
sa_table_fragmentation(  
[ tbl_name  
[, owner_name ] ]  
)
```

Argumente

- **tbl_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Namen Tabelle anzugeben, die auf Fragmentierung überprüft werden soll. Der Standardwert ist NULL.
- **owner_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümer von tbl_name anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
TableName	CHAR(128)	Der Name der Tabelle
rows	UNSIGNED INTEGER	Die Anzahl der Zeilen in der Tabelle
row_segments	UNSIGNED BIGINT	Die Anzahl der Zeilensegmente in der Tabelle
segs_per_row	DOUBLE	Die Anzahl der Segmente pro Zeile

Bemerkungen

Datenbankadministratoren können diese Prozedur einsetzen, um Informationen über die Fragmentierung in den Datenbanktabellen zu erhalten. Wenn keine Argumente angegeben wurden, werden Ergebnisse für alle Tabellen in der Datenbank zurückgegeben.

Wenn Datenbanktabellen zu stark fragmentiert sind, können Sie die Prozedur REORGANIZE TABLE ausführen oder die Datenbank neu aufbauen, um die Tabellenfragmentierung zu vermindern und die Performance zu verbessern.

Privilegien

Sie müssen das MANAGE ANY STATISTICS-Systemprivileg oder das MONITOR-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Tabellenfragmentierung reduzieren“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Neuaufbau von Datenbanken“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „REORGANIZE TABLE-Anweisung“ auf Seite 998

Beispiele

```
CALL sa_table_fragmentation( 'Products', 'GROUPO' );

CALL sa_table_fragmentation( owner_name='GROUPO' );
```

sa_table_page_usage-Systemprozedur

Gibt Auskunft über die Seitenbelegung durch Datenbanktabellen

Syntax

```
sa_table_page_usage( )
```

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
TableId	UNSIGNED INTEGER	Die Tabellen-ID.
TablePages	INTEGER	Die Anzahl der von der Tabelle benutzten Tabellenseiten
PctUsedT	INTEGER	Der Prozentsatz des Tabellenseiten-Speicherplatzes
IndexPages	INTEGER	Die Anzahl der von der Tabelle benutzten Indexseiten
PctUsedI	INTEGER	Der Prozentsatz des Indexseiten-Speicherplatzes

Spaltenname	Datentyp	Beschreibung
PctOfFile	INTEGER	Der Prozentsatz der gesamten Datenbankdatei, den die Tabelle belegt
TableName	CHAR(128)	Der Tabellename.

Bemerkungen

Das Ergebnis umfasst dieselben Informationen, die vom Informations-Dienstprogramm geboten werden. Wenn die progress_messages-Datenbankoption auf "Raw" oder "Formatted" gesetzt ist, werden Meldungen zum Verarbeitungsfortschritt vom Datenbankserver an den Client gesendet, während die sa_table_page_usage-Systemprozedur läuft.

Privilegien

Sie müssen das MANAGE ANY DBSPACE-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Informations-Dienstprogramm (dbinfo)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „progress_messages-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Im folgenden Beispiel werden Informationen zur Seitenbelegung durch die Tabelle "SalesOrderItems" abgerufen.

```
SELECT * FROM sa_table_page_usage( )  
WHERE TableName = 'SalesOrderItems';
```

sa_table_stats-Systemprozedur

Gibt Auskunft darüber, wie viele Seiten von jeder Tabelle gelesen wurden

Syntax

sa_table_stats()

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
table_id	INTEGER	Die Tabellen-ID.
creator	CHAR(128)	Der Benutzername des Erstellers der Tabelle.
table_name	CHAR(128)	Der Tabellename.

Spaltenname	Datentyp	Beschreibung
"count"	UNSIGNED BIGINT	Die geschätzte Anzahl der Zeilen, von SYSTAB bezogen.
table_page_count	UNSIGNED BIGINT	Die Anzahl der von der Tabelle benutzten Hauptseiten.
table_page_cached	UNSIGNED BIGINT	Die Anzahl der derzeit im Cache gespeicherten Tabellenseiten.
table_page_reads	UNSIGNED BIGINT	Die Anzahl der Seiten-Lesevorgänge, die bei Seiten in der Haupttabelle durchgeführt wurden.
ext_page_count	UNSIGNED BIGINT	Die geschätzte Anzahl der Seiten in der Tabelle.
ext_page_cached	UNSIGNED BIGINT	Spalte für spätere Verwendung reserviert.
ext_page_reads	UNSIGNED BIGINT	Spalte für spätere Verwendung reserviert.

Bemerkungen

Jede von der Prozedur `sa_table_stats` zurückgegebene Zeile beschreibt eine Tabelle, bei der der Optimierer Seitenstatistiken führt. Die Prozedur `sa_table_stats` kann verwendet werden, um herauszufinden, welche Tabellen Cachespeicher verwenden und wie viele Festplatten-Lesezugriffe für jede Tabelle durchgeführt werden. Sie können die Prozedur `sa_table_stats` z.B. verwenden, um die Tabelle zu bestimmen, die die meisten Festplatten-Lesezugriffe generiert. Die Ergebnisse der Prozedur stellen Schätzungen dar und sollten nur für Diagnosezwecke verwendet werden.

Die Spalte `table_page_cached` zeigt an, wie viele Seiten der Tabelle derzeit im Cache gespeichert sind, und die Spalte `table_page_reads` zeigt an, wie viele Tabellenseiten von der Festplatte gelesen wurden, seit der Optimierer die Zählungen bei der Tabelle gestartet hat. Diese Ergebnisse werden nicht beständig in der Datenbank gespeichert. Sie stellen Aktivitäten auf Tabellen dar, nachdem sie zum ersten Mal in den Speicher geladen wurden.

Privilegien

Sie müssen das `MONITOR`-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- [„SYSTAB-Systemansicht“ auf Seite 1492](#)

Beispiel

Sie können die folgende Abfrage ausführen, um Informationen dazu abzurufen, wie viele Seiten aus jeder Tabelle gelesen wurden.

```
CALL sa_table_stats( );
```

sa_text_index_stats-Systemprozedur

Gibt statistische Informationen über die Textindizes in der Datenbank zurück.

Syntax

sa_text_index_stats()

Bemerkungen

Verwenden Sie die sa_text_index_stats-Systemprozedur, um statistische Informationen für jeden Textindex in der Datenbank anzuzeigen. Die nachstehende Tabelle zeigt die von sa_text_index_stats zurückgegebenen Informationen.

Spaltenname	Typ	Beschreibung
owner_id	UNSIGNED INTEGER	Die ID des Eigentümers der Tabelle
table_id	UNSIGNED INTEGER	Die ID der Tabelle
index_id	UNSIGNED INTEGER	Die ID des Textindexes
text_config_id	UNSIGNED BIGINT	Die ID des Textkonfigurationsobjekts, das vom Index referenziert wird
owner_name	CHAR(128)	Der Name des Eigentümers
table_name	CHAR(128)	Der Name der Tabelle
index_name	CHAR(128)	Der Name des Textindexes
text_config_name	CHAR(128)	Der Name des Textkonfigurationsobjekts
doc_count	UNSIGNED BIGINT	Die Gesamtanzahl der indizierten Spaltenwerte im Textindex
doc_length	UNSIGNED BIGINT	Gesamtlänge der Daten im Textindex
pending_length	UNSIGNED BIGINT	Die Gesamtlänge der ausstehenden Änderungen
deleted_length	UNSIGNED BIGINT	Die Gesamtlänge der ausstehenden Löschungen
last_refresh	TIMESTAMP	Datum und Uhrzeit der letzten Aktualisierung in Ortszeit

Die Werte für pending_length, deleted_length und last_refresh sind bei Textindizes mit IMMEDIATE REFRESH gleich 0.

Bei Textindizes mit MANUAL REFRESH können Sie doc_length, pending_length und deleted_length verwenden, um zu entscheiden, ob der Textindex aktualisiert werden soll, und welche Art der Aktualisierung durchgeführt wird (neu aufsetzen oder inkrementell aktualisieren).

Privilegien

Sie müssen eines der folgenden Systemprivilegien haben:

- MANAGE ANY STATISTICS
- CREATE ANY INDEX
- ALTER ANY INDEX
- DROP ANY INDEX
- CREATE ANY OBJECT
- ALTER ANY TABLE
- DROP ANY OBJECT

Nebenwirkungen

Keine

Siehe auch

- „Volltextsuche“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Konzepte und Referenz zu Textindizes“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „DROP TEXT INDEX-Anweisung“ auf Seite 834
- „REFRESH TEXT INDEX-Anweisung“ auf Seite 990
- „TRUNCATE TEXT INDEX-Anweisung“ auf Seite 1091
- „sa_refresh_text_indexes-Systemprozedur“ auf Seite 1295
- „sa_text_index_vocab-Systemprozedur“ auf Seite 1343
- „SYSTEXTIDX-Systemansicht“ auf Seite 1501

Beispiel

Die folgende Anweisung gibt statistische Informationen für jeden Textindex in der Datenbank zurück:

```
CALL sa_text_index_stats( );
```

sa_text_index_vocab-Systemprozedur

Listet alle Begriffe in einem Textindex und die Gesamtanzahl der indizierten Werte auf, in denen jeder Begriff erscheint.

Syntax

```
sa_text_index_vocab(  
  indexname  
  , tabname  
  [, tabowner ]  
)
```

Argumente

- **indexname** Verwenden Sie diesen CHAR(128)-Parameter, um den Namen des Textindex anzugeben.
- **tablename** Verwenden Sie diesen CHAR(128)-Parameter, um den Namen der Tabelle anzugeben, für die der Textindex erstellt wird.
- **tabowner** Verwenden Sie diesen CHAR(128)-Parameter, um den Eigentümer der Tabelle anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
term	VARCHAR(60)	Ein Begriff im Textindex.
freq	BIGINT	Die Anzahl indizierter Werte, in denen der Begriff erscheint.

Bemerkungen

Die sa_text_index_vocab-Systemprozedur gibt alle Begriffe zurück, die in einem Textindex erscheinen, sowie die Gesamtanzahl der indizierten Werte, in denen jeder Begriff erscheint (und die niedriger ist als die Gesamtanzahl der Fälle, wenn der Begriff in einigen indizierten Werte mehr als einmal enthalten ist).

Privilegien

Sie müssen das SELECT-Privileg für die indizierte Tabelle oder das SELECT ANY TABLE-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „sa_text_index_vocab_nchar-Systemprozedur“ auf Seite 1345
- „Volltextsuche“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Begriffs- und Phrasensuche“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „DROP TEXT INDEX-Anweisung“ auf Seite 834
- „REFRESH TEXT INDEX-Anweisung“ auf Seite 990
- „TRUNCATE TEXT INDEX-Anweisung“ auf Seite 1091
- „sa_refresh_text_indexes-Systemprozedur“ auf Seite 1295
- „SYSTEXTIDX-Systemansicht“ auf Seite 1501

Beispiel

Im folgenden Beispiel wird ein Textindex namens VocabTxtIdx für die Spalte Products.Description in der Beispieldatenbank erstellt.

```
CREATE TEXT INDEX VocabTxtIdx2 ON Products( Description );
```

Die nächste Anweisung führt die sa_text_index_vocab-Systemprozedur aus, um alle Begriffe zurückzugeben, die im Textindex erscheinen.

```
SELECT * FROM sa_text_index_vocab( 'VocabTxtIdx2', 'Products', 'GROUPO' );
```

term	freq
Cap	2
Cloth	1
Cotton	2
Crew	1
Hooded	1
neck	2
...	...

sa_text_index_vocab_nchar-Systemprozedur

Listet alle Begriffe in einem NCHAR-Textindex und die Gesamtanzahl der indizierten Werte auf, in denen jeder Begriff erscheint.

Syntax

```
sa_text_index_vocab_nchar(
  indexname
  , tabname
  [, tabowner ]
)
```

Argumente

- **indexname** Verwenden Sie diesen CHAR(128)-Parameter, um den Namen des Textindexes anzugeben.
- **tabname** Verwenden Sie diesen CHAR(128)-Parameter, um den Namen der Tabelle anzugeben, für die der Textindex erstellt wird.
- **tabowner** Verwenden Sie diesen CHAR(128)-Parameter, um den Eigentümer der Tabelle anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
term	NVARCHAR(60)	Ein Begriff im Textindex.
freq	BIGINT	Die Anzahl indizierter Werte, in denen der Begriff erscheint.

Bemerkungen

Die `sa_text_index_vocab_nchar`-Systemprozedur gibt alle Begriffe zurück, die in einem Textindex erscheinen, sowie die Gesamtanzahl der indizierten Werte, in denen jeder Begriff erscheint (und die niedriger ist als die Gesamtanzahl der Fälle, wenn der Begriff in einigen indizierten Werte mehr als einmal enthalten ist).

Privilegien

Sie müssen das `SELECT`-Privileg für die indizierte Tabelle oder das `SELECT ANY TABLE`-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- [„sa_text_index_vocab-Systemprozedur“ auf Seite 1343](#)
- [„Volltextsuche“ \[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#)
- [„Begriffs- und Phrasensuche“ \[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#)
- [„DROP TEXT INDEX-Anweisung“ auf Seite 834](#)
- [„REFRESH TEXT INDEX-Anweisung“ auf Seite 990](#)
- [„TRUNCATE TEXT INDEX-Anweisung“ auf Seite 1091](#)
- [„sa_refresh_text_indexes-Systemprozedur“ auf Seite 1295](#)
- [„SYSTEXTIDX-Systemansicht“ auf Seite 1501](#)

Beispiel

Im folgenden Beispiel wird ein Textindex namens `VocabNTxtIdx` für die `NProducts.Description`-Spalte erstellt. Die Tabelle `"NProducts"` ist eine Version der Tabelle `"Products"` mit `NCHAR`-Spalten statt `CHAR`-Spalten.

```
CREATE TEXT INDEX VocabNTxtIdx2 ON NProducts( Description );
```

Die nächste Anweisung führt die `sa_text_index_vocab_nchar`-Systemprozedur aus, um alle im Textindex vorkommenden Begriffe zurückzugeben.

```
SELECT * FROM sa_text_index_vocab_nchar( 'VocabNTxtIdx2', 'NProducts' );
```

term	freq
Cap	2
Cloth	1
Cotton	2
Crew	1
Hooded	1
neck	2

term	freq
...	...

sa_transactions-Systemprozedur

Gibt eine Liste von derzeit aktiven Transaktionen zurück

Syntax

sa_transactions()

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
connection_num	INTEGER	Die Verbindungs-ID der Verbindung, auf der die Transaktion läuft
transaction_id	INTEGER	Die ID, die die Transaktion eindeutig identifiziert, solange der Datenbankserver sie protokolliert. IDs werden wieder verwendet, wenn alte Transaktionsinformationen verworfen werden.
start_time	TIMESTAMP	Der Zeitstempel für den Start der Transaktion
start_sequence_num	UNSIGNED BIGINT	Die Startsequenznummer der Transaktion
end_sequence_num	UNSIGNED BIGINT	Die Endsequenznummer der Transaktion, wenn sie festgeschrieben oder zurückgesetzt wurde, sonst NULL.
committed	BIT	Der Status der Transaktion: TRUE, wenn die Transaktion mit einem COMMIT endete, FALSE, wenn sie mit einem ROLLBACK endete, und NULL, wenn die Transaktion noch aktiv ist.
version_entries	UNSIGNED INTEGER	Die Zählung der Anzahl von Zeilenversionen, die die Transaktion gespeichert hat

Bemerkungen

Diese Prozedur liefert Informationen über die Transaktionen, die derzeit auf der Datenbank laufen.

Privilegien

Sie müssen das MONITOR-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „sa_snapshots-Systemprozedur“ auf Seite 1331
- „Snapshot-Isolation“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Beispiel

Sie können die folgende Abfrage ausführen, um die derzeit aktiven Transaktionen zu ermitteln.

```
CALL sa_transactions( );
```

sa_unload_cost_model-Systemprozedur

Entlädt das aktuelle Kostenmodell in die angegebene Datei

Syntax

```
sa_unload_cost_model( file_name )
```

Argumente

- **file_name** Verwenden Sie diesen CHAR(256)-Parameter, um den Namen der Datei anzugeben, in die die Daten entladen werden sollen. Da der Datenbankserver die Systemprozedur ausführt, gibt *file_name* eine Datei auf dem Datenbankserver-Computer an und ein relativer *file_name* gibt eine Datei relativ zum Startverzeichnis des Datenbankservers an.

Bemerkungen

Der Optimierer verwendet Kostenmodelle, um optimale Zugriffspläne für Abfragen zu ermitteln. Der Datenbankserver hält ein Kostenmodell für jede Datenbank vor. Das Kostenmodell für eine Datenbank kann jederzeit neu kalibriert werden, indem die CALIBRATE SERVER-Klausel der ALTER DATABASE-Anweisung verwendet wird. Sie könnten zum Beispiel das Kostenmodell neu kalibrieren, wenn Sie die Datenbank auf eine Nicht-Standard-Hardware verschieben.

Mit der sa_unload_cost_model-Systemprozedur können Sie ein Kostenmodell in einer ASCII-Datei (*file_name*) speichern. Danach können Sie sich bei einer anderen Datenbank anmelden und die sa_load_cost_model-Systemprozedur verwenden, um das Kostenmodell von der ersten in die zweite Datenbank zu laden. Das vermeidet eine Neu-Kalibrierung der zweiten Datenbank.

Hinweis

Die sa_unload_cost_model-Systemprozedur bezieht keine CALIBRATE PARALLEL READ-Informationen in die Datei ein.

Die Verwendung der sa_unload_cost_model-Systemprozedur macht wiederholte, zeitraubende Neu-Kalibrierungen unnötig, wenn es eine große Anzahl ähnlicher Hardware-Installationen gibt.

Sie müssen Schreibberechtigung dort haben, wo die Datei erstellt wird.

Privilegien

Sie müssen das SELECT ANY TABLE-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „ALTER DATABASE-Anweisung“ auf Seite 451
- „sa_load_cost_model-Systemprozedur“ auf Seite 1249
- „Fortgeschrittene Aufgaben: Abfrageoptimierung“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Beispiel

Im folgenden Beispiel wird das Kostenmodell in eine Datei namens *costmodel8* entladen:

```
CALL sa_unload_cost_model( 'costmodel8' );
```

sa_user_defined_counter_add-Systemprozedur

Passt den Wert eines benutzerdefinierten Zählers um einen angegebenen Wert an.

Syntax

```
sa_user_defined_counter_add(  
  counter_name  
  [, delta  
  [, apply_to_con  
  [, apply_to_db  
  [, apply_to_server ] ] ]  
)
```

Argumente

- **counter_name** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Namen des benutzerdefinierten Zählers anzugeben, dessen Wert Sie ändern wollen. Beispiele für Namen benutzerdefinierter Zähler sind UserDefinedCounterRate01 und UserDefinedCounterRaw01.
- **delta** Verwenden Sie diesen BIGINT-Parameter, um den Betrag anzugeben, um den der benutzerdefinierte Zähler erhöht bzw. verringert werden soll. Standardwert ist "1".
- **apply_to_con** Verwenden Sie diesen INTEGER-Parameter, um anzugeben, ob der Zählerwert für die aktuelle Verbindung angepasst werden soll. 0 bedeutet, dass der Wert nicht angepasst werden soll, und 1 bedeutet, dass der Wert angepasst werden soll. Standardwert ist "1".
- **apply_to_db** Verwenden Sie diesen INTEGER-Parameter, um anzugeben, ob der Zählerwert für die Datenbank angepasst werden soll. 0 bedeutet, dass der Wert nicht angepasst werden soll, und 1 bedeutet, dass der Wert angepasst werden soll. Standardwert ist "1".
- **apply_to_server** Verwenden Sie diesen INT-Parameter, um anzugeben, ob der Zählerwert für den Datenbankserver angepasst werden soll. 0 bedeutet, dass der Wert nicht angepasst werden soll, und 1 bedeutet, dass der Wert angepasst werden soll. Standardwert ist "1".

Rückgabe

Diese Funktion gibt einen INTEGER-Statuscode zurück.

Bemerkungen

Diese Funktion gibt den Wert 1 zurück, wenn *delta* definiert ist, 0, wenn *delta* nicht definiert ist, und einen Fehlercode, wenn ein Fehler auftritt. Beispiele für Fehler:

- Ein ungültiger Zählername
- Ein ungültiger Wert für den `apply_to_server`-, `apply_to_db`- oder `apply_to_con`-Parameter.

Gleichzeitige Zugriffe auf Zähler werden in kleinsten Einheiten angewendet, sodass ein Zählerwert aus mehreren gleichzeitigen Anforderungen inkrementiert werden kann.

Benutzerdefinierte Zähler werden als 32-Bit-UNSIGNED INTEGER-Werte implementiert.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „`sa_user_defined_counter_set`-Systemprozedur“ auf Seite 1350
- „Liste der Verbindungseigenschaften“ [*SQL Anywhere Server - Datenbankadministration*]
- „Liste der Datenbankeigenschaften“ [*SQL Anywhere Server - Datenbankadministration*]
- „Liste der Datenbankservereigenschaften“ [*SQL Anywhere Server - Datenbankadministration*]
- „Benutzerdefinierte Statistiken“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Benutzerdefinierte Eigenschaften“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Die folgende Anweisung erhöht den Wert von `UserDefinedCounterRate01` für die aktuelle Verbindung, die aktuelle Datenbank und den aktuellen Datenbankserver um 2:

```
SELECT sa_user_defined_counter_add( 'UserDefinedCounterRate01', 2, 1, 1, 1 );
```

sa_user_defined_counter_set-Systemprozedur

Setzt einen benutzerdefinierten Zähler auf einen angegebenen Wert.

Syntax

```
sa_user_defined_counter_set(  
  counter_name  
  , value  
  [, apply_to_con  
  [, apply_to_db  
  [, apply_to_server ] ] ]
```

Argumente

- **counter_name** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Namen des benutzerdefinierten Zählers anzugeben, dessen Wert Sie ändern wollen. Beispiele für Namen benutzerdefinierter Zähler sind `UserDefinedCounterRate01` und `UserDefinedCounterRaw01`.

- **value** Verwenden Sie diesen BIGINT-Parameter, um den Wert anzugeben, auf den der benutzerdefinierte Zähler gesetzt wird.
- **apply_to_con** Verwenden Sie diesen INT-Parameter, um anzugeben, ob der Zählerwert für die aktuelle Verbindung angepasst werden soll. 0 bedeutet, dass der Wert nicht angepasst werden soll, und 1 bedeutet, dass der Wert angepasst werden soll. Standardwert ist "1".
- **apply_to_db** Verwenden Sie diesen INT-Parameter, um anzugeben, ob der Zählerwert für die Datenbank angepasst werden soll. 0 bedeutet, dass der Wert nicht angepasst werden soll, und 1 bedeutet, dass der Wert angepasst werden soll. Der Standardwert ist 0.
- **apply_to_server** Verwenden Sie diesen INT-Parameter, um anzugeben, ob der Zählerwert für den Datenbankserver angepasst werden soll. 0 bedeutet, dass der Wert nicht angepasst werden soll, und 1 bedeutet, dass der Wert angepasst werden soll. Standardwert ist "0".

Rückgabe

Diese Funktion gibt einen INTEGER-Statuscode zurück.

Bemerkungen

Diese Funktion gibt den Wert 1 zurück, wenn *value* definiert ist, 0, wenn *value* nicht definiert ist, und einen Fehlercode, wenn ein Fehler auftritt. Beispiele für Fehler:

- Ein ungültiger Zählername
- Ein ungültiger Wert für den `apply_to_server`-, `apply_to_db`- oder `apply_to_con`-Parameter.

Gleichzeitige Zugriffe auf Zähler werden in kleinsten Einheiten angewendet, sodass ein Zählerwert aus mehreren gleichzeitigen Anforderungen zurückgesetzt werden kann.

Benutzerdefinierte Zähler werden als 32-Bit-UNSIGNED INTEGER-Werte implementiert.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „[sa_user_defined_counter_add](#)-Systemprozedur“ auf Seite 1349
- „Liste der Verbindungseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Liste der Datenbankeigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Liste der Datenbankserveigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Benutzerdefinierte Eigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Benutzerdefinierte Statistiken“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Beispiel

Die folgende Anweisung setzt den Wert von `UserDefinedCounterRate01` für die aktuelle Verbindung, die aktuelle Datenbank und den aktuellen Datenbankserver auf 0:

```
SELECT sa_user_defined_counter_set( 'UserDefinedCounterRate01', 0, 1, 1, 1 );
```

sa_validate-Systemprozedur

Führt eine vollständige oder teilweise Prüfsummenvalidierung einer Datenbank aus.

Syntax

```
sa_validate(  
  [ tbl_name  
  [, owner_name ] ]  
)
```

Argumente

- **tbl_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Namen einer zu validierenden Tabelle oder materialisierten Ansicht anzugeben. Der Standardwert ist NULL.
- **owner_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um einen Eigentümer anzugeben. Wenn der Name angegeben ist, werden alle Tabellen und materialisierten Ansichten validiert, die diesem Eigentümer gehörten. Der Standardwert ist NULL.

Privilegien

Sie müssen das VALIDATE ANY OBJECT-Systemprivileg haben.

Nebenwirkungen

Keine

Bemerkungen

Angegebenes Argument	Validierungstyp
Keins	Alle Tabellen, materialisierten Ansichten und Indizes in der Datenbank werden validiert. Die Datenbank selbst wird ebenfalls validiert, einschließlich Prüfsummenvalidierung.
<i>tbl_name</i>	Die angegebene Tabelle oder materialisierte Ansicht und alle dazugehörigen Indizes, die dem aktuellen Benutzer gehören, werden validiert.
<i>owner_name</i>	Alle Tabellen, materialisierten Ansichten und Indizes, die dem angegebenen Benutzer gehören, werden validiert.
<i>tbl_name</i> und <i>owner_name</i>	Die angegebene Tabelle oder materialisierte Ansicht und alle dazugehörigen Indizes, die dem angegebenen Benutzer gehören, werden validiert.

Die Prozedur gibt eine einzelne Spalte zurück, die die Bezeichnung "Messages" trägt. Fehler, die während der Validierung zurückgegeben werden, erscheinen in der Spalte. Wenn die Validierung ohne Fehler erfolgreich verläuft, enthält die Spalte `Keine Fehler gefunden`.

Vorsicht

Die Validierung einer Tabelle oder einer ganzen Datenbank darf nur durchgeführt werden, wenn keine Verbindungen Änderungen in der Datenbank durchführen, weil sonst möglicherweise Fehler über eine Datenbankbeschädigung gemeldet werden, obwohl eine solche nicht vorliegt.

Beispiel

Die folgende Anweisung führt eine Validierung von Tabellen und materialisierten Ansichten aus, deren Eigentümer der Benutzer DBA ist.

```
CALL sa_validate( owner_name = 'DBA' );
```

sa_verify_password-Systemprozedur

Validiert das Kennwort des aktuellen Benutzers

Syntax

```
sa_verify_password( curr_pswd )
```

Argumente

- **curr_pswd** Verwenden Sie diesen CHAR(128)-Parameter, um das Kennwort des aktuellen Datenbankbenutzers anzugeben.

Rückgabe

Die folgende Anweisung gibt einen INTEGER-Wert zurück.

Bemerkungen

Diese Prozedur wird von sp_password verwendet. Wenn das Kennwort übereinstimmt, wird 0 zurückgegeben und es tritt kein Fehler auf. Wenn das Kennwort nicht übereinstimmt, wird ein Fehler diagnostiziert. Die Verbindung wird nicht beendet, wenn das Kennwort nicht übereinstimmt.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- [„Systemprozeduren von Adaptive Server Enterprise“ auf Seite 1166](#)

Beispiel

Im folgenden Beispiel wird versucht, das Kennwort der aktuellen Verbindung zu validieren, wobei der aktuelle Benutzer DBA oder User1 ist. Ein Fehler tritt auf, wenn das aktuelle Kennwort nicht übereinstimmt.

```
IF USER_NAME() = 'DBA' THEN
    SELECT sa_verify_password( 'sql' );
ELSEIF USER_NAME() = 'User1' THEN
    SELECT sa_verify_password( 'user' );
END IF;
```

sp_alter_secure_feature_key-Systemprozedur

Ändert einen zuvor definierten Schlüssel für gesicherte Funktionen durch Ändern des Autorisierungsschlüssels bzw. der Funktionsliste.

Syntax

```
sp_alter_secure_feature_key(
    name
    , auth_key
    , features
)
```

Argumente

- **name** Der VARCHAR(128)-Name des Schlüssels für gesicherte Funktionen, den Sie ändern möchten. Ein Schlüssel mit dem angegebenen Namen muss bereits vorhanden sein.
- **auth_key** Der CHAR(128)-Autorisierungsschlüssel (mit Berücksichtigung von Groß- und Kleinschreibung) des Schlüssels für gesicherte Funktionen. Der Autorisierungsschlüssel muss entweder eine nicht leere Zeichenfolge von mindestens sechs Zeichen sein oder NULL, wobei Letzteres bedeutet, dass der vorhandene Autorisierungsschlüssel nicht geändert werden soll.
- **features** Die kommagetrennte LONG VARCHAR-Liste von gesicherten Funktionen, die der Schlüssel aktivieren kann. Wenn die Funktionsliste NULL ist, wird die vorhandene Funktionsliste nicht geändert.

Bemerkungen

Mit dieser Prozedur können Sie den Autorisierungsschlüssel oder die Funktionsliste eines vorhandenen Schlüssels für gesicherte Funktionen ändern.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben und die MANAGE_KEYS-Funktion muss für die Verbindung aktiviert sein.

Nebenwirkungen

Keine

Siehe auch

- „Schlüssel für gesicherte Funktionen erstellen“ [*SQL Anywhere Server - Datenbankadministration*]
- „sp_create_secure_feature_key-Systemprozedur“ auf Seite 1360
- „sp_drop_secure_feature_key-Systemprozedur“ auf Seite 1366
- „sp_list_secure_feature_keys-Systemprozedur“ auf Seite 1375
- „sp_use_secure_feature_key-Systemprozedur“ auf Seite 1410
- Liste der gesicherten Funktionen: „Datenbankserveroption -sf“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Damit das folgende Beispiel funktioniert, muss der Server mit der Option `-sk securefkey` gestartet werden.

In diesem Beispiel wird die Gruppe gesicherter Funktionen namens SYSTEM aktiviert, die `MANAGE_KEYS` enthält. Sie erstellt einen neuen Schlüssel für gesicherte Funktionen namens MYSET mit dem Autorisierungsschlüssel `securemyset`, der Groß- und Kleinschreibung berücksichtigt, ändert die Zusammenstellung der Gruppe von gesicherten Funktionen und führt anschließend `sp_list_secure_feature_keys` aus, um eine Liste der derzeit definierten Schlüssel für gesicherte Funktionen abzurufen:

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );
CALL sp_create_secure_feature_key( 'myset', 'securemyset', 'local' );
CALL sp_alter_secure_feature_key( 'myset', 'securemyset', 'local,remote' );
CALL sp_list_secure_feature_keys( );
```

sp_auth_sys_role_info-Systemprozedur

Gibt die Zuordnung von Datenbankberechtigungen aus früheren Versionen von SQL Anywhere zu den entsprechenden Kompatibilitätsrollen zurück.

Syntax

```
sp_auth_sys_role_info( )
```

Ergebnismenge

Column	Type	Description
auth	VARCHAR(20)	Der Name der Berechtigung.
role_name	CHAR(128)	Der Name der entsprechenden Kompatibilitätsrolle.
role_id	UNSIGNED INTEGER	Die ID-Nummer für die Kompatibilitätsrolle.

Bemerkungen

Keine

Privilegien

Keine

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Siehe auch

- „Kompatibilitätsrollen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Berechtigungen auf Datenbankebene werden zu Kompatibilitätsrollen“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Die folgende Anweisung gibt die Liste der Berechtigungen (auth) für SQL Anywhere-Datenbanken vor Version 16.0 zurück, zugeordnet zu den entsprechenden Kompatibilitätsrollen (role_name) ab Version 16.0.

```
CALL sp_auth_sys_role_info();
```

auth	role_name	role_id
DBA	SYS_AUTH_DBA_ROLE	2147485648
RESOURCE	SYS_AUTH_RESOURCE_ROLE	2147485649
BACKUP	SYS_AUTH_BACKUP_ROLE	2147485650
VALIDATE	SYS_AUTH_VALIDATE_ROLE	2147485651
READFILE	SYS_AUTH_READFILE_ROLE	2147485652
PROFILE	SYS_AUTH_PROFILE_ROLE	2147485653
READCLIENTFILE	SYS_AUTH_READCLIENTFILE_ROLE	2147485654
WRITECLIENTFILE	SYS_AUTH_WRITECLIENTFILE_ROLE	2147485655
WRITEFILE	SYS_AUTH_WRITEFILE_ROLE	2147485656
REMOTE DBA	SYS_RUN_REPLICATION_ROLE	2147485664

sp_copy_directory-Systemprozedur

Kopiert ein Verzeichnis an einen angegebenen Speicherort.

Syntax

```
sp_copy_directory(  
  source_path  
  , destination_path  
)
```

Parameter

- **source_path** Verwenden Sie diesen LONG NVARCHAR-Parameter, um den Pfad des zu kopierenden Verzeichnisses anzugeben. Der Pfad kann in absoluter oder relativer Form angegeben werden. Der Pfad wird auf dem Computer relativ zum aktuellen Arbeitsverzeichnis des Datenbankservers aufgelöst. Wenn die Sandboxing-Funktion aktiviert ist, beziehen sich absolute und relative Pfade auf das Verzeichnis, in dem sich die Hauptdatenbankdatei befindet.
- **destination_path** Verwenden Sie diesen LONG NVARCHAR-Parameter, um den Pfad anzugeben, in den das Verzeichnis kopiert werden soll. Der Pfad kann in absoluter oder relativer Form angegeben werden. Der Pfad wird auf dem Computer relativ zum aktuellen Arbeitsverzeichnis des Datenbankservers aufgelöst. Wenn die Sandboxing-Funktion aktiviert ist, beziehen sich absolute und relative Pfade auf das Verzeichnis, in dem sich die Hauptdatenbankdatei befindet. Das Verzeichnis wird erstellt, wenn es noch nicht vorhanden ist

Rückgabe

Diese Funktion gibt bei Erfolg 0 und bei Fehler 1 zurück.

Bemerkungen

Diese Funktion kopiert das Verzeichnis und die darin enthaltenen Dateien aus dem Quellverzeichnis in das angegebene Verzeichnis. Das Verzeichnis und die darin enthaltenen Dateien bleiben im Quellverzeichnis. Verwenden Sie die `sp_delete_directory`-Systemprozedur, um das Quellverzeichnis zu löschen.

Privilegien

Sie müssen die Systemprivilegien `READ FILE` und `WRITE FILE` haben.

Siehe auch

- [„sp_copy_file-Systemprozedur“ auf Seite 1358](#)
- [„sp_create_directory-Systemprozedur“ auf Seite 1359](#)
- [„sp_delete_directory-Systemprozedur“ auf Seite 1361](#)
- [„sp_delete_file-Systemprozedur“ auf Seite 1362](#)
- [„sp_list_directory-Systemprozedur“ auf Seite 1373](#)
- [„sp_move_directory-Systemprozedur“ auf Seite 1377](#)
- [„sp_move_file-Systemprozedur“ auf Seite 1378](#)
- [„Systemprozeduren für Verzeichnisse und Dateien“ auf Seite 1164](#)

Beispiel

Die folgende Anweisung erstellt eine Kopie der Unterverzeichnisse und Dateien im Verzeichnis *SQLAnywhere* in *SQLAnywhere.bkp*.

```
SELECT sp_copy_directory('c:\\sqlany\\samples\\SQLAnywhere', 'c:\\sqlany\\  
samples\\SQLAnywhere.bkp');
```

Das gesamte Verzeichnis einschließlich seiner Unterverzeichnisse und Dateien wird dupliziert.

sp_copy_file-Systemprozedur

Kopiert eine Datei an einen angegebenen Speicherort.

Syntax

```
sp_copy_file(  
    source_path  
    , destination_path  
)
```

Parameter

- **source_path** Verwenden Sie diesen LONG NVARCHAR-Parameter, um den Dateipfad, einschließlich des Dateinamens, der zu verschiebenden Datei anzugeben. Der Pfad kann in absoluter oder relativer Form angegeben werden. Der Pfad wird auf dem Computer relativ zum aktuellen Arbeitsverzeichnis des Datenbankservers aufgelöst. Wenn die Sandboxing-Funktion aktiviert ist, beziehen sich absolute und relative Pfade auf das Verzeichnis, in dem sich die Hauptdatenbankdatei befindet.
- **destination_path** Verwenden Sie diesen LONG NVARCHAR-Parameter, um das neue Verzeichnis der Datei anzugeben. Der Pfad kann in absoluter oder relativer Form angegeben werden. Der Pfad wird auf dem Computer relativ zum aktuellen Arbeitsverzeichnis des Datenbankservers aufgelöst. Wenn die Sandboxing-Funktion aktiviert ist, beziehen sich absolute und relative Pfade auf das Verzeichnis, in dem sich die Hauptdatenbankdatei befindet.

Rückgabe

Diese Funktion gibt bei Erfolg 0 und bei Fehler 1 zurück.

Bemerkungen

Diese Funktion kopiert eine Datei aus einem Verzeichnis in ein anderes. Die Datei bleibt im Quellverzeichnis. Mit der sp_delete_file-Systemprozedur können Sie die Datei im Quellverzeichnis löschen.

Privilegien

Sie müssen die Systemprivilegien READ FILE und WRITE FILE haben.

Siehe auch

- „sp_copy_directory-Systemprozedur“ auf Seite 1356
- „sp_create_directory-Systemprozedur“ auf Seite 1359
- „sp_delete_directory-Systemprozedur“ auf Seite 1361
- „sp_delete_file-Systemprozedur“ auf Seite 1362
- „sp_list_directory-Systemprozedur“ auf Seite 1373
- „sp_move_directory-Systemprozedur“ auf Seite 1377
- „sp_move_file-Systemprozedur“ auf Seite 1378
- „Systemprozeduren für Verzeichnisse und Dateien“ auf Seite 1164

Beispiel

Die folgende Anweisung kopiert die Datei *license.txt* in das Verzeichnis *c:\temp* und gibt ihr einen neuen Namen.

```
SELECT sp_copy_file('c:\\sqlany\\license.txt', 'c:\\temp\\license.bkp');
```

Eine weitere Kopie der *license.txt* Datei ist im Verzeichnis *temp* unter einem anderen Namen vorhanden.

sp_create_directory-Systemprozedur

Erstellt ein Verzeichnis auf dem Computer.

Syntax

```
sp_create_directory( root_path )
```

Parameter

- **root_path** Verwenden Sie diesen LONG NVARCHAR-Parameter, um den zu erstellenden Verzeichnispfad anzugeben. Der Pfad kann in absoluter oder relativer Form angegeben werden. Ein relativer Pfad wird relativ zum aktuellen Arbeitsverzeichnis des Datenbankservers aufgelöst. Wenn die Sandboxing-Funktion aktiviert ist, beziehen sich absolute und relative Pfade auf das Verzeichnis, in dem sich die Hauptdatenbankdatei befindet.

Rückgabe

Diese Funktion gibt bei Erfolg 0 und bei Fehler 1 zurück.

Bemerkungen

Diese Funktion erstellt ein neues Verzeichnis.

Privilegien

Sie müssen das WRITE FILE-Systemprivileg haben.

Siehe auch

- „Schlüssel für gesicherte Funktionen erstellen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sp_copy_directory-Systemprozedur“ auf Seite 1356
- „sp_copy_file-Systemprozedur“ auf Seite 1358
- „sp_delete_directory-Systemprozedur“ auf Seite 1361
- „sp_delete_file-Systemprozedur“ auf Seite 1362
- „sp_list_directory-Systemprozedur“ auf Seite 1373
- „sp_move_directory-Systemprozedur“ auf Seite 1377
- „sp_move_file-Systemprozedur“ auf Seite 1378
- „Systemprozeduren für Verzeichnisse und Dateien“ auf Seite 1164

Beispiel

Die folgende Anweisung erstellt ein Verzeichnis namens *SQLAnywhere.bkp* im Verzeichnis *c:\sqlany\samples*.

```
SELECT sp_create_directory( 'c:\\sqlany\\samples\\SQLAnywhere.bkp' );
```

sp_create_secure_feature_key-Systemprozedur

Erstellt einen neuen Schlüssel für gesicherte Funktionen.

Syntax

```
sp_create_secure_feature_key(  
    name  
    , auth_key  
    , features  
)
```

Argumente

- **name** Der VARCHAR(128)-Name des neuen Schlüssels für gesicherte Funktionen. Dieses Argument darf nicht NULL oder eine leere Zeichenfolge sein.
- **auth_key** Der CHAR(128)-Autorisierungsschlüssel (mit Berücksichtigung von Groß- und Kleinschreibung) des Schlüssels für gesicherte Funktionen. Der Autorisierungsschlüssel muss eine nicht leere Zeichenfolge von mindestens sechs Zeichen sein.
- **features** Die kommagetrennte LONG VARCHAR-Liste von gesicherten Funktionen, die der neue Schlüssel aktivieren kann.

Das Angeben von - vor einer Funktion bedeutet, dass die Funktion beim Festlegen des Schlüssels für gesicherte Funktionen nicht erneut aktiviert wird.

Bemerkungen

Diese Prozedur erstellt einen neuen Schlüssel für gesicherte Funktionen, der jedem Benutzer gegeben werden kann. Die SYSTEM-Schlüssel für gesicherte Funktionen wird mithilfe der Datenbankserveroption -sk erstellt.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben und die MANAGE_KEYS-Funktion muss für die Verbindung aktiviert sein.

Nebenwirkungen

Keine

Siehe auch

- Liste der gesicherten Funktionen: „Datenbankserveroption -sf“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption -sk“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Schlüssel für gesicherte Funktionen erstellen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sp_alter_secure_feature_key-Systemprozedur“ auf Seite 1354
- „sp_drop_secure_feature_key-Systemprozedur“ auf Seite 1366
- „sp_list_secure_feature_keys-Systemprozedur“ auf Seite 1375
- „sp_use_secure_feature_key-Systemprozedur“ auf Seite 1410

Beispiel

Damit das folgende Beispiel funktioniert, muss der Server mit der Option `-sk securefkey` gestartet werden.

In diesem Beispiel wird die Gruppe gesicherter Funktionen namens SYSTEM aktiviert, die MANAGE_KEYS enthält. Sie erstellt einen neuen Schlüssel für gesicherte Funktionen namens MYSET mit dem Autorisierungsschlüssel securemyset, der Groß- und Kleinschreibung berücksichtigt, und führt anschließend `sp_list_secure_feature_keys` aus, um eine Liste der derzeit definierten Schlüssel für gesicherte Funktionen abzurufen:

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );  
CALL sp_create_secure_feature_key( 'myset', 'securemyset', 'local' );  
CALL sp_list_secure_feature_keys( );
```

sp_delete_directory-Systemprozedur

Löscht das angegebene Verzeichnis.

Syntax

```
sp_delete_directory( root_path )
```

Parameter

- **root_path** Verwenden Sie diesen LONG NVARCHAR-Parameter, um den Verzeichnispfad des zu löschenden Verzeichnisses anzugeben. Der Pfad kann in absoluter oder relativer Form angegeben werden. Ein relativer Pfad wird relativ zum aktuellen Arbeitsverzeichnis des Datenbankservers aufgelöst. Wenn die Sandboxing-Funktion aktiviert ist, beziehen sich absolute und relative Pfade auf das Verzeichnis, in dem sich die Hauptdatenbankdatei befindet.

Rückgabe

Diese Funktion gibt bei Erfolg 0 und bei Fehler 1 zurück.

Bemerkungen

Diese Funktion löscht ein Verzeichnis an einem angegebenen Speicherort.

Privilegien

Sie müssen die Systemprivilegien READ FILE und WRITE FILE haben.

Siehe auch

- „Schlüssel für gesicherte Funktionen erstellen“ [*SQL Anywhere Server - Datenbankadministration*]
- „sp_copy_directory-Systemprozedur“ auf Seite 1356
- „sp_copy_file-Systemprozedur“ auf Seite 1358
- „sp_create_directory-Systemprozedur“ auf Seite 1359
- „sp_delete_file-Systemprozedur“ auf Seite 1362
- „sp_list_directory-Systemprozedur“ auf Seite 1373
- „sp_move_directory-Systemprozedur“ auf Seite 1377
- „sp_move_file-Systemprozedur“ auf Seite 1378
- „Systemprozeduren für Verzeichnisse und Dateien“ auf Seite 1164

Beispiel

Die folgende Anweisung löscht das Verzeichnis *SQLAnywhere.bkp*.

```
SELECT sp_delete_directory('c:\\sqlany\\samples\\SQLAnywhere.bkp');
```

sp_delete_file-Systemprozedur

Löscht die angegebene Datei vom Computer.

Syntax

sp_delete_file(*file_path*)

Parameter

- **file_path** Verwenden Sie diesen LONG NVARCHAR-Parameter, um den Dateipfad, einschließlich des Dateinamens, der zu löschenden Datei anzugeben. Der Pfad kann in absoluter oder relativer Form angegeben werden. Ein relativer Pfad wird relativ zum aktuellen Arbeitsverzeichnis des Datenbankservers aufgelöst. Wenn die Sandboxing-Funktion aktiviert ist, beziehen sich absolute und relative Pfade auf das Verzeichnis, in dem sich die Hauptdatenbankdatei befindet.

Rückgabe

Diese Funktion gibt bei Erfolg 0 und bei Fehler 1 zurück.

Bemerkungen

Diese Funktion löscht die angegebene Datei vom Computer.

Sie müssen mit einem Datenbankserver auf dem Computer verbunden sein, auf dem die Datei gespeichert ist, um diese Prozedur ausführen zu können.

Privilegien

Sie müssen die Systemprivilegien READ FILE und WRITE FILE haben.

Siehe auch

- „Schlüssel für gesicherte Funktionen erstellen“ [*SQL Anywhere Server - Datenbankadministration*]
- „sp_copy_directory-Systemprozedur“ auf Seite 1356
- „sp_copy_file-Systemprozedur“ auf Seite 1358
- „sp_create_directory-Systemprozedur“ auf Seite 1359
- „sp_delete_directory-Systemprozedur“ auf Seite 1361
- „sp_list_directory-Systemprozedur“ auf Seite 1373
- „sp_move_directory-Systemprozedur“ auf Seite 1377
- „sp_move_file-Systemprozedur“ auf Seite 1378
- „Systemprozeduren für Verzeichnisse und Dateien“ auf Seite 1164

Beispiel

Die folgende Anweisung löscht die Datei *license.bkp* aus dem Verzeichnis *temp*.

```
SELECT sp_delete_file('c:\\temp\\license.bkp');
```

sp_displayroles-Systemprozedur

Gibt alle Rollen zurück, die dem angegebenen Element (Systemprivileg, Systemrolle, benutzerdefinierte Rolle oder Benutzername) erteilt wurden, oder zeigt die gesamte Hierarchiestruktur der Rollen an.

Syntax

```
sp_displayroles(  
  user_role_name  
  , display_mode  
  , grant_type  
)
```

Argumente

- **user_role_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Namen eines Systemprivilegs, einer Systemrolle, einer benutzerdefinierten Rolle oder eines Benutzers anzugeben. Wenn der Name nicht angegeben wird oder NULL ist, wird standardmäßig der aktuelle Benutzer verwendet.
- **display_mode** Verwenden Sie diesen VARCHAR(30)-Parameter, um anzugeben, ob über- oder untergeordnete Hierarchieebenen relativ zu *user_role_name* zurückgegeben werden sollen. Wenn *display_mode* nicht angegeben wird oder NULL ist, werden nur ausdrücklich erteilte Rollen und Privilegien zurückgegeben (keine geerbten Rollen oder Privilegien). Zu den möglichen Werten für *display_mode* gehören folgende:
 - **expand_up** Zeigt die *user_role_name* erteilten Systemrollen in der übergeordneten Hierarchiestruktur für *user_role_name*.
 - **expand_down** Zeigt die *user_role_name* erteilten Systemrollen und Privilegien, einschließlich der Rollenhierarchiestruktur für die untergeordneten Ebenen von *user_role_name*.

- **grant_type** Verwenden Sie diesen VARCHAR(30)-Parameter, um den zurückgegebenen Erteilungstyp zu steuern. Wenn er nicht angegeben wurde, wird standardmäßig ALL verwendet. Zu den möglichen Werten für *grant_type* gehören folgende:
 - **NO_ADMIN** Zeigt alle Rollen und Systemprivilegien, die *user_role_name* mit der Klausel WITH NO ADMIN OPTION oder WITH ADMIN OPTION erteilt wurden.
 - **ADMIN** Gibt alle Rollen und Systemprivilegien zurück, die *user_role_name* mit der Klausel WITH ADMIN OPTION oder WITH ADMIN ONLY OPTION erteilt wurden.
 - **ALL** Zeigt alle *user_role_name* erteilten Rollen und Systemprivilegien an.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
role_name	CHAR(128)	Die <i>user_role_name</i> erteilte Rolle bzw. das erteilte Systemprivileg.
parent_role_name	CHAR(128)	Die Rollennamen für die übergeordneten Elemente von <i>user_role_name</i> .
grant_type	CHAR(10)	Informationen dazu, ob <i>user_role_name</i> Administrationsrechte hat. Mögliche Werte: NO ADMIN, ADMIN oder ADMIN ONLY.
role_level	SMALLINT	Beim expand_down-Modus ist die Stufe gleich 1 für direkt erteilte Rollen, 2 für die nächste Ebene darunter usw. Beim expand_up-Modus ist die Stufe gleich 0 für die Rollen, denen <i>user_role_name</i> erteilt wurde, -1 für die nächste Hierarchieebene darüber usw.

Bemerkungen

Bei Systemprivilegien zeigt das Ergebnis den Namen des Systemprivilegs statt des dazugehörigen Rollennamens. Beim expand_down-Modus ist parent_role_name für Stufe 1 (direkt erteilte Rollen) NULL. Beim Standardmodus ist die role_level-Spalte gleich 1 und parent_role_name ist NULL, da beim Standardmodus nur die direkt erteilten Rollen angezeigt werden.

Wenn diese Prozedur für einen Benutzer mit expand_up-Modus ausgeführt wird, werden keine Ergebnisse zurückgegeben, da ein Benutzer sich in jeder Rollenhierarchie auf der obersten Ebene befindet. Wenn diese Prozedur für ein unveränderliches Systemprivileg mit expand_down-Modus ausgeführt wird, werden ebenfalls keine Ergebnisse zurückgegeben, da ein unveränderliches Systemprivileg sich in jeder Rollenhierarchie auf der untersten Ebene befindet. Im Standardmodus werden nur die direkt erteilten Rollen und Systemprivilegien angezeigt.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Privilegien

Keine Privilegien sind erforderlich, um diese Prozedur für Sie selbst auszuführen. Um jedoch die Systemprivilegien oder Rollen für eine andere Benutzer-ID oder eine Rolle zurückgeben zu können, müssen Sie das **MANAGE ROLES**-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Rollen“ [*SQL Anywhere Server - Datenbankadministration*]
- „Rollen und Privilegien für einen Benutzer oder eine Rolle anzeigen (SQL)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Rollen und Privilegien für einen Benutzer oder eine Rolle anzeigen (Sybase Central)“ [*SQL Anywhere Server - Datenbankadministration*]
- „sp_sys_priv_role_info-Systemprozedur“ auf Seite 1399
- „sp_has_role-Systemprozedur“ auf Seite 1371
- „sp_proc_priv-Systemprozedur“ auf Seite 1384
- „sp_objectpermission-Systemprozedur“ auf Seite 1379

Beispiel

Die folgende Anweisung gibt alle Rollen zurück, die dem Benutzer erteilt wurden, der den Befehl ausgibt.

```
CALL sp_displayroles();
```

In diesem Beispiel wird die Liste der Systemprivilegien zurückgegeben, die der **SYS_SPATIAL_ADMIN_ROLE**-Systemrolle erteilt wurden:

```
CALL sp_displayroles( 'SYS_SPATIAL_ADMIN_ROLE' );
```

role_name	parent_role_name	grant_type	role_level
PUBLIC	(NULL)	NO ADMIN	1
MANAGE ANY SPATIAL OBJECT	(NULL)	NO ADMIN	1

In diesem Beispiel wird die Liste der Systemprivilegien zurückgegeben, die der **SYS_SPATIAL_ADMIN_ROLE**-Systemrolle erteilt wurden, einschließlich aller Rollen, die sich in der Rollenhierarchie darüber befinden:

```
CALL sp_displayroles( 'SYS_SPATIAL_ADMIN_ROLE', 'expand_up' );
```

role_name	parent_role_name	grant_type	role_level
SYS_AUTH_DBA_ROLE	dbo	ADMIN	-3
SYS_AUTH_DBA_ROLE	SYS_RUN_REPLICATION_ROLE	ADMIN	-3

role_name	parent_role_name	grant_type	role_level
SYS_AUTH_SSO_ROLE	SYS_AUTH_DBA_ROLE	ADMIN	-2
MANAGE ROLES	SYS_AUTH_SSO_ROLE	ADMIN	-1
MANAGE ROLES	SYS_REPLICATION_ADMIN_ROLE	NO ADMIN	-1
SYS_SPATIAL_ADMIN_ROLE	MANAGE ROLES	ADMIN ONLY	0

Die folgende Anweisung gibt alle Systemprivilegien zurück, die dem Benutzer User1 erteilt wurden:

```
CALL sp_displayroles( 'User1' );
```

In diesem Beispiel wird die Liste der zu Ansichten gehörenden Systemprivilegien zurückgegeben:

```
SELECT sys_priv_name FROM sp_sys_priv_role_info()  
WHERE sys_priv_name LIKE '%VIEW%'
```

sp_drop_secure_feature_key-Systemprozedur

Löscht einen Schlüssel für gesicherte Funktionen.

Syntax

```
sp_drop_secure_feature_key( name )
```

Argumente

- **name** Der VARCHAR(128)-Name des zu löschenden Schlüssels für gesicherte Funktionen.

Bemerkungen

Wenn der benannte Schlüssel nicht existiert, wird ein Fehler zurückgegeben. Wenn der benannte Schlüssel vorhanden ist, wird er gelöscht, sofern er nicht der letzte Schlüssel für gesicherte Funktionen ist, die für die Verwaltung von gesicherten Funktionen und Schlüsseln für gesicherte Funktionen verwendet werden darf. Der SYSTEM-Schlüssel für gesicherte Funktionen kann nicht gelöscht werden, es sei denn, für einen anderen Schlüssel sind die gesicherten Funktionen MANAGE_FEATURES und MANAGE_KEYS aktiviert.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben und die MANAGE_KEYS-Funktion muss für die Verbindung aktiviert sein.

Nebenwirkungen

Keine

Siehe auch

- „Schlüssel für gesicherte Funktionen erstellen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sp_alter_secure_feature_key-Systemprozedur“ auf Seite 1354
- „sp_create_secure_feature_key-Systemprozedur“ auf Seite 1360
- „sp_list_secure_feature_keys-Systemprozedur“ auf Seite 1375
- „sp_use_secure_feature_key-Systemprozedur“ auf Seite 1410
- MANAGE_FEATURES und MANAGE_KEYS: „Datenbankserveroption -sf“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Damit das folgende Beispiel funktioniert, muss der Server mit der Option `-sk securefkey` gestartet werden.

In diesem Beispiel wird die Gruppe gesicherter Funktionen namens SYSTEM aktiviert, die MANAGE_KEYS enthält. Sie löscht anschließend den Schlüssel für gesicherte Funktionen namens MYSET:

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );  
CALL sp_drop_secure_feature_key( 'myset' );
```

sp_forward_to_remote_server-Systemprozedur

Syntax

```
sp_forward_to_remote_server(  
    @server_name  
    , @sql  
)
```

Argumente

- **@server_name** Verwenden Sie diesen VARCHAR(128)-Parameter, um den Namen des Fremdservers anzugeben, auf dem die SQL-Anweisung ausgeführt wird.
- **@sql** Verwenden Sie diesen LONG VARCHAR-Parameter, um die SQL-Anweisung anzugeben, die auf dem Fremdserver ausgeführt werden soll.

Bemerkungen

Diese Prozedur ermöglicht es einer Anwendung, eine SQL-Anweisung auf einem Fremdserver auszuführen und die von dieser Anweisung generierten Ergebnismengen abzurufen. Die SQL-Anweisung wird wortgetreu an den Fremdsender gesendet und daher muss SQL Anywhere nicht in der Lage sein, die Anweisung syntaktisch zu analysieren.

Um diese Systemprozedur verwenden zu können, müssen Sie den Fremdsender mit der CREATE SERVER-Anweisung definieren.

Anders als bei der FORWARD TO-Anweisung kann sp_forward_to_remote_server innerhalb von Prozeduren verwendet werden. Jedoch kann diese gespeicherte Prozedur nicht in der FROM-Klausel einer SELECT-Anweisung verwendet werden, da das Schema der entfernten Ergebnismengen beliebig ist. Sie

können entfernte Ergebnismengen abrufen, indem Sie einen Cursor für eine gespeicherte Prozedur deklarieren, der in der `sp_forward_to_remote_server`-Prozedur aufgerufen wird.

Hinweis

Wenn die SQL-Anweisung mehrere Ergebnismengen zurückgibt, liefert die gespeicherte Prozedur `sp_forward_to_remote_server` jede entfernte Ergebnismenge einzeln.

Privilegien

Keine

Nebenwirkungen

Es gibt keine lokalen Nebenwirkungen bei der Ausführung dieser gespeicherten Prozedur, aber da die SQL-Anweisung, die auf dem Fremdserver ausgeführt wird, beliebig ist, können auf dem Fremdserver Nebenwirkungen auftreten.

Siehe auch

- „[FORWARD TO-Anweisung](#)“ auf Seite 861
- „[CREATE SERVER-Anweisung](#)“ auf Seite 701

Beispiel

Im folgenden Beispiel wird die gespeicherte Prozedur `sp_forward_to_remote_server` verwendet, um die Anzahl der Nicht-Systemtabellen in einer entfernten SQL Anywhere 16-Datenbank namens `RemoteSA` zu ermitteln.

```
CALL sp_forward_to_remote_server( 'RemoteSA',  
    'SELECT COUNT(*) FROM sys.systable WHERE CREATOR NOT IN (0,3,6)' );
```

Im folgenden Beispiel wird die gespeicherte Prozedur `sp_forward_to_remote_server` verwendet, um die Spalten aus einer Excel-Tabellenkalkulation namens `newSalesData` zu lesen.

```
call sp_forward_to_remote_server( 'RemoteExcel', 'SELECT * FROM  
newSalesData' );
```

sp_get_last_synchronize_result-Systemprozedur

Gibt Informationen über die letzte durch eine `SYNCHRONIZE`-Anweisung eingeleitete Synchronisation zurück.

Syntax

```
sp_get_last_synchronize_result(  
[ @conn_id  
, @complete_only ] ]  
)
```

Argumente

- **@conn_id** Verwenden Sie diesen optionalen INTEGER-Parameter, um die Verbindungs-ID-Nummer für eine Verbindung anzugeben, auf der die `SYNCHRONIZE`-Anweisung ausgeführt wurde.

Der Standardwert ist NULL. Wenn der Parameter nicht angegeben wird oder NULL ist, wird die Verbindungs-ID der aktuellen Verbindung verwendet.

- **@complete_only** Setzen Sie diesen optionalen BIT-Parameter auf 1, wenn die gespeicherte Prozedur Informationen zu abgeschlossenen Synchronisationen zurückgeben soll. Setzen Sie den Parameter auf 0, um Informationen über zurzeit aktive Synchronisationen zurückzugeben. Der Standardwert ist 1.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
row_id	BIGINT	Der Primärschlüssel der Tabelle, mit dessen Hilfe die Reihenfolge bestimmt wird, in der Zeilen in die Tabelle eingefügt wurden.
conn_id	UNSIGNED INTEGER	Die Verbindungs-ID-Nummer der Verbindung, für die die SYNCHRONIZE-Anweisung ausgeführt wird, die dieses Ereignis generiert hat.
result_time	TIMESTAMP	Der Zeitpunkt, zu dem das Ereignis der Tabelle synchronize_results hinzugefügt wurde.
result_type	CHAR(128)	Der Typ des Ereignisses.
parm_id	INTEGER	Jedem Ereignis können null oder mehr Parameter zugeordnet sein. Die parm_id-Spalte sortiert die Parameter, die den einzelnen Ereignissen zugeordnet sind.
parm_result	LONG VARCHAR	Der Nachrichtentext im Zusammenhang mit dem Ereignisparameter.

Bemerkungen

Um Details früherer oder aktueller Synchronisationen anzuzeigen, können Sie die gespeicherte Prozedur `sp_get_last_synchronize_result` als Alternative zum direkten Abfragen der globalen gemeinsamen temporären Tabellen "synchronize_results" und "synchronize_parameters" verwenden. Die gespeicherte Prozedur liefert nur die Ergebnisse der letzten Synchronisation für die angegebene Verbindungs-ID-Nummer. Wenn Sie keine Parameter angeben, wird die letzte abgeschlossene Synchronisation für die aktuelle Verbindung zurückgegeben.

Sie können diese gespeicherte Prozedur auch dazu verwenden, den Fortschritt einer Synchronisation auf einer anderen Verbindung als Ihrer aktuellen Verbindung zu überwachen. So überwachen Sie den Fortschritt einer Synchronisation auf einer anderen Verbindung:

1. Führen Sie eine `SELECT CONNECTION_PROPERTY`-Anweisung aus, um die Verbindungs-ID der aktuellen Verbindung zu bestimmen.
2. Führen Sie eine `SYNCHRONIZE`-Anweisung mit der Verbindungs-ID aus, die von der `SELECT CONNECTION_PROPERTY`-Anweisung zurückgegeben wurde.

3. Führen Sie auf einer anderen Verbindung eine SELECT CONNECTION_PROPERTY-Anweisung aus und setzen Sie den Parameter *complete_only* auf 0. Informationen über die letzte Synchronisation für die angegebene Verbindung werden zurückgegeben, auch wenn die Synchronisation unvollständig ist.

Im Folgenden finden Sie eine Liste von Ereignissen und ihren zugeordneten parm_id-Werten aus der synchronize_parameters-Tabelle:

Event	parm_id-Wert	Beschreibung
DBSC_EVENT-TYPE_ERROR_MSG	0	Der Text der Fehlermeldung.
	1	Die der Meldung zugeordnete ID.
DBSC_EVENT-TYPE_WARNING_MSG	0	Der Text der Warnmeldung.
	1	Die der Meldung zugeordnete ID.
DBSC_EVENT-TYPE_INFO_MSG	0	Der Text der Informationsmeldung.
	1	Die der Meldung zugeordnete ID.
DBSC_EVENTTYPE_PROGRESS_INDEX	0	Der neue Fortschrittsindexwert.
DBSC_EVENTTYPE_PROGRESS_TEXT	0	Der neue Fortschrittstext.
DBSC_EVENT-TYPE_TITLE	0	Der neue Fenstertitel.
DBSC_EVENT-TYPE_SYNC_DONE	0	Der Exit-Code aus der Synchronisation. 0 bedeutet Erfolg.
DBSC_EVENT-TYPE_ML_CONNECT	0	Das verwendete Kommunikationsprotokoll.
	1	Die verwendeten Netzwerkprotokolloptionen.
DBSC_EVENT-TYPE_DOWNLOAD_COMMITTED	0	Die Anzahl der festgeschriebenen Einfüge- bzw. Aktualisierungsvorgänge.
	1	Die Anzahl der festgeschriebenen Löschvorgänge.

Event	parm_id-Wert	Beschreibung
DBSC_EVENT-TYPE_UPLOAD_SENT	0	Die Anzahl der hochgeladenen Einfügevorgänge.
	1	Die Anzahl der hochgeladenen Aktualisierungsvorgänge.
	2	Die Anzahl der hochgeladenen Löschvorgänge.

Privilegien

Sie benötigen das SELECT-Privileg für die gemeinsam genutzten globalen temporären Tabellen `synchronize_results` und `synchronize_parameters`.

Nebenwirkungen

Keine

Siehe auch

- „[SYNCHRONIZE-Anweisung \[MobiLink\]](#)“ auf Seite 1081
- [DBSC_Event-Struktur \[Dbmlsync .NET\] \[MobiLink - Clientadministration\]](#)

Beispiel

Im folgenden Beispiel werden Informationen zu der letzten Synchronisation zurückgegeben, die für die aktuelle Verbindung abgeschlossen wurde.

```
CALL sp_get_last_synchronize_result();
```

Im folgenden Beispiel werden beim Aufrufen der `sp_get_last_synchronize_result`-Systemprozedur benannte Parameter verwendet und Informationen zur letzten abgeschlossenen Synchronisation zurückgegeben, die über Verbindungs-ID 25 initiiert wurde.

```
CALL sp_get_last_synchronize_result(
    @conn_id=25,
    @complete_only=1);
```

sp_has_role-Systemprozedur

Gibt die Information zurück, ob dem Aufrufer der Prozedur das angegebene Systemprivileg bzw. die angegebene benutzerdefinierte Rolle erteilt wurde.

Syntax

```
sp_has_role(
    rolename
    [, grant_type
    [, throw_error]]
)
```

Parameter

- **rolename** Verwenden Sie diesen CHAR(128)-Parameter, um den Namen eines Systemprivilegs oder einer benutzerdefinierten Rolle anzugeben, nach dem gesucht werden soll.
- **grant_type** Verwenden Sie diesen optionalen CHAR(20)-Parameter, um den Erteilungstyp anzugeben, nach dem gesucht werden soll. Die möglichen Werte sind ADMIN und NO ADMIN (Standardwert).

Bei der Einstellung ADMIN überprüft die sp_has_role-Prozedur, ob der aufrufende Benutzer Administrationsrechte für das Privileg bzw. die Rolle hat. Bei der Einstellung NO ADMIN überprüft sp_has_role, ob der aufrufende Benutzer die Ausübungsrechte für das Privileg bzw. die Rolle hat.

- **throw_error** Verwenden Sie diesen optionalen INTEGER-Parameter, um anzugeben, ob ein Wert zurückgegeben werden soll, der das Ergebnis der Privilegüberprüfung darstellt. Beim Wert 1 wird eine Meldung zurückgegeben, wenn das angegebene Systemprivileg bzw. die angegebene benutzerdefinierte Rolle nicht dem Aufrufer erteilt wurde. Beim Wert 0 (Standardwert) wird in keinem Fall eine Meldung zurückgegeben.

Rückgabe

Zurückgegebener Wert	Beschreibung
1	Das Systemprivileg bzw. die benutzerdefinierte Rolle ist dem aufrufenden Benutzer erteilt und die <i>grant_type</i> -Überprüfung wird ebenfalls bestanden.
0 oder "Berechtigung verweigert: Sie haben nicht die Berechtigung, diesen Befehl bzw. diese Prozedur auszuführen."	Das Systemprivileg bzw. die benutzerdefinierte Rolle ist nicht dem aufrufenden Benutzer erteilt bzw. die <i>grant_type</i> -Überprüfung ist ebenfalls fehlgeschlagen. Die Fehlermeldung ersetzt den Wert 0, wenn <i>throw_error</i> auf 1 gesetzt ist.
-1	Das Systemprivileg bzw. die benutzerdefinierte Rolle ist nicht vorhanden. Es wird keine Fehlermeldung angezeigt, auch wenn <i>throw_error</i> auf 1 gesetzt ist.

Bemerkungen

Das *throw_error*-Argument ist nützlich, um Fehlermeldungen vom Typ "Berechtigung verweigert" zurückzugeben, wenn ein Benutzer die Berechtigungsüberprüfung in einer gespeicherten Prozedur nicht besteht.

Privilegien

Keine

Siehe auch

- „Rollen“ [*SQL Anywhere Server - Datenbankadministration*]
- „sp_sys_priv_role_info-Systemprozedur“ auf Seite 1399
- „sp_proc_priv-Systemprozedur“ auf Seite 1384
- „sp_displayroles-Systemprozedur“ auf Seite 1363
- „sp_objectpermission-Systemprozedur“ auf Seite 1379

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

- Die folgende Anweisung überprüft, ob Ihnen das CREATE ANY PROCEDURE-Systemprivileg ohne Administrationsrechte erteilt wurde.

```
SELECT sp_has_role( 'CREATE ANY PROCEDURE' );
```

- Die folgende Anweisung überprüft, ob Ihnen das CREATE ANY PROCEDURE-Systemprivileg mit Administrationsrechten erteilt wurde, und gibt nur dann einen Fehler zurück, wenn es Ihnen nicht erteilt wurde.

```
SELECT sp_has_role( 'CREATE ANY PROCEDURE', 'ADMIN', 1 );
```

- Die folgende Anweisung überprüft, ob Ihnen Rolle_A ohne Administrationsrechte erteilt wurde.

```
SELECT sp_has_role( 'Role_A' );
```

- Die folgende Anweisung überprüft, ob Ihnen Rolle_A mit Administrationsrechten erteilt wurde, und gibt nur dann einen Fehler zurück, wenn sie Ihnen nicht erteilt wurde.

```
SELECT sp_has_role( 'Role_A', 'ADMIN', 1 );
```

sp_list_directory-Systemprozedur

Gibt Informationen zum Inhalt eines Verzeichnisses zurück.

Syntax

```
sp_list_directory(  
  root_path  
  [, max_depth ]  
)
```

Parameter

- **root_path** Verwenden Sie diesen LONG NVARCHAR-Parameter, um den absoluten oder relativen Verzeichnispfad anzugeben. Der relative Pfad ist relativ zum Arbeitsverzeichnis des Servers. Wenn die Sandboxing-Funktion aktiviert ist, beziehen sich absolute und relative Pfade auf das Verzeichnis, in dem sich die Hauptdatenbankdatei befindet.
- **max_depth** Verwenden Sie diesen optionalen INTEGER-Parameter, um die maximale Anzahl der zu durchsuchenden Verzeichnisse anzugeben. Wenn max_depth NULL, 0 oder ein negativer Wert ist, werden alle Unterverzeichnisse von root_path durchsucht. Der Standardwert ist NULL.

Ergebnismenge

Diese Prozedur gibt Informationen zu den Dateien und Unterverzeichnissen in einem angegebenen Verzeichnis zurück. Diese Prozedur gibt eine Tabelle mit den folgenden Spalten zurück: `file_path`, `file_type` und `file_size`.

Spaltenname	Spaltenbeschreibung
<code>file_path</code>	Der Pfad zu einer Datei oder einem Unterverzeichnis innerhalb des angegebenen Verzeichnisses. Wenn <code>directory_path</code> als relativer Pfad angegeben wird, ist auch der zurückgegebene <code>file_path</code> -Wert relativ. Andernfalls ist der <code>file_path</code> -Wert absolut. Wenn die Sandboxing-Funktion aktiviert ist, beziehen sich absolute und relative Pfade auf das Verzeichnis, in dem sich die Hauptdatenbankdatei befindet.
<code>file_type</code>	Gibt F an, wenn der <code>file_path</code> -Wert eine Datei ist, bzw. D, wenn der <code>file_path</code> -Wert ein Verzeichnis ist.
<code>file_size</code>	Gibt die Größe (in Byte) der Datei an bzw. NULL, wenn der <code>file_path</code> -Wert ein Verzeichnis ist.

Bemerkungen

Der `max_depth`-Parameter gibt die maximale Tiefe an, bis zu der das Verzeichnis durchsucht wird. Wenn `max_depth` NULL, 0 oder ein negativer Wert ist, bedeutet dies, dass es keine Einschränkung gibt.

Privilegien

Sie müssen das READ FILE-Systemprivileg haben.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Siehe auch

- „Schlüssel für gesicherte Funktionen erstellen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „`sp_copy_directory`-Systemprozedur“ auf Seite 1356
- „`sp_copy_file`-Systemprozedur“ auf Seite 1358
- „`sp_create_directory`-Systemprozedur“ auf Seite 1359
- „`sp_delete_directory`-Systemprozedur“ auf Seite 1361
- „`sp_delete_file`-Systemprozedur“ auf Seite 1362
- „`sp_move_directory`-Systemprozedur“ auf Seite 1377
- „`sp_move_file`-Systemprozedur“ auf Seite 1378
- „Systemprozeduren für Verzeichnisse und Dateien“ auf Seite 1164

Beispiele

Die folgende Anweisung listet die Dateien und Unterverzeichnisse des Verzeichnisses `c:\sqlany\samples\SQLAnywhere` auf.

```
CALL sp_list_directory('c:\sqlany\samples\SQLAnywhere');
```

Die folgende Anweisung gibt die Gesamtzahl der Dateien und Ordner in den Verzeichnissen *c:\sqlany\samples\SQLAnywhere* und *c:\sqlany\samples\UltraLite* zurück.

```
SELECT COUNT(*) FROM sp_list_directory( 'c:\\sqlany\\samples\\SQLAnywhere' )
UNION ALL
SELECT COUNT(*) FROM sp_list_directory( 'c:\\sqlany\\samples\\UltraLite' );
```

Wenn die Verzeichnisse *c:\sqlany\samples\SQLAnywhere* und *c:\sqlany\samples\UltraLite* dieselben Dateinamen, Dateitypen und Dateigrößen enthalten, kombiniert die folgende Anweisung die Ergebnisse aus den beiden Verzeichnissen und gibt die Gesamtzahl der eindeutigen Ergebnisse zurück.

```
SELECT COUNT(*) FROM
(
    SELECT REPLACE( file_path, 'c:\\sqlany\\samples\\SQLAnywhere', '' ) AS
file_path, file_type, file_size
    FROM sp_list_directory( 'c:\\sqlany\\samples\\SQLAnywhere' )
    UNION
    SELECT REPLACE( file_path, 'c:\\sqlany\\samples\\UltraLite', '' ) AS
file_path, file_type, file_size
    FROM sp_list_directory( 'c:\\sqlany\\samples\\UltraLite' )
) d;
```

sp_list_secure_feature_keys-Systemprozedur

Gibt eine Liste der definierten Schlüssel für gesicherte Funktionen zurück.

Syntax

```
sp_list_secure_feature_keys( )
```

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
name	VARCHAR(128)	Der Name des Schlüssels für gesicherte Funktionen.
features	LONG VARCHAR	Die gesicherten Funktionen, die durch den Schlüssel für gesicherte Funktionen aktiviert werden.

Bemerkungen

Diese Prozedur gibt die Namen vorhandener Schlüssel für gesicherte Funktionen zurück sowie die gesicherten Funktionen, die durch die einzelnen Schlüssel aktiviert werden können.

Wenn für den Benutzer die gesicherten Funktionen `MANAGE_FEATURES` und `MANAGE_KEYS` aktiviert sind, gibt die Prozedur eine Liste aller Schlüssel für gesicherte Funktionen zurück.

Wenn für den Benutzer nur die gesicherte Funktion `MANAGE_KEYS` aktiviert ist, gibt die Prozedur Schlüssel zurück, die dieselben Funktionen oder eine Teilmenge davon umfassen, die für den aktuellen Benutzer aktiviert sind.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben und die MANAGE_KEYS-Funktion muss für die Verbindung aktiviert sein.

Nebenwirkungen

Keine

Siehe auch

- „Schlüssel für gesicherte Funktionen erstellen“ [*SQL Anywhere Server - Datenbankadministration*]
- „sp_alter_secure_feature_key-Systemprozedur“ auf Seite 1354
- „sp_create_secure_feature_key-Systemprozedur“ auf Seite 1360
- „sp_drop_secure_feature_key-Systemprozedur“ auf Seite 1366
- „sp_use_secure_feature_key-Systemprozedur“ auf Seite 1410

Beispiel

Damit das folgende Beispiel funktioniert, muss der Server mit der Option `-sk securefkey` gestartet werden.

In diesem Beispiel wird die Gruppe gesicherter Funktionen namens SYSTEM aktiviert, die die gesicherten Funktionen MANAGE_FEATURES und MANAGE_KEYS enthält. Sie führt anschließend `sp_list_secure_feature_keys` aus, um eine Liste der derzeit definierten Schlüssel für gesicherte Funktionen abzurufen:

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );  
CALL sp_list_secure_feature_keys( );
```

sp_login_environment-Systemprozedur

Setzt Verbindungsoptionen, wenn sich Benutzer anmelden

Syntax

```
sp_login_environment( )
```

Bemerkungen

`sp_login_environment` ist die Standardprozedur, die von der Datenbankoption `login_procedure` aufgerufen wird.

Es wird empfohlen, diese Prozedur nicht zu bearbeiten. Um die Login-Umgebung zu ändern, setzen Sie die Option `login_procedure` so, dass sie auf eine andere Prozedur zeigt.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „login_procedure-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

sp_move_directory-Systemprozedur

Diese Funktion verschiebt das Verzeichnis, auf das source_path zeigt, in das Ziel, auf das destination_path zeigt.

Syntax

```
sp_move_directory(  
    source_path  
    , destination_path  
)
```

Parameter

- **source_path** Verwenden Sie diesen LONG NVARCHAR-Parameter, um den Verzeichnispfad des zu verschiebenden Verzeichnisses anzugeben. Der Pfad kann in absoluter oder relativer Form angegeben werden. Ein relativer Pfad wird relativ zum aktuellen Arbeitsverzeichnis des Datenbankservers aufgelöst. Wenn die Sandboxing-Funktion aktiviert ist, beziehen sich absolute und relative Pfade auf das Verzeichnis, in dem sich die Hauptdatenbankdatei befindet.
- **destination_path** Verwenden Sie diesen LONG NVARCHAR-Parameter, um den Pfad anzugeben, in den das Verzeichnis verschoben werden soll. Das Verzeichnis wird erstellt, wenn es noch nicht vorhanden ist. Der Pfad kann in absoluter oder relativer Form angegeben werden. Ein relativer Pfad wird relativ zum aktuellen Arbeitsverzeichnis des Datenbankservers aufgelöst. Wenn die Sandboxing-Funktion aktiviert ist, beziehen sich absolute und relative Pfade auf das Verzeichnis, in dem sich die Hauptdatenbankdatei befindet.

Rückgabe

Diese Funktion gibt bei Erfolg 0 und bei Fehler 1 zurück.

Bemerkungen

Diese Funktion verschiebt alle Dateien im Quellverzeichnis in das angegebene Verzeichnis und löscht anschließend das Quellverzeichnis.

Privilegien

Sie müssen die Systemprivilegien READ FILE und WRITE FILE haben.

Siehe auch

- „Schlüssel für gesicherte Funktionen erstellen“ [*SQL Anywhere Server - Datenbankadministration*]
- „sp_copy_directory-Systemprozedur“ auf Seite 1356
- „sp_copy_file-Systemprozedur“ auf Seite 1358
- „sp_create_directory-Systemprozedur“ auf Seite 1359
- „sp_delete_directory-Systemprozedur“ auf Seite 1361
- „sp_delete_file-Systemprozedur“ auf Seite 1362
- „sp_list_directory-Systemprozedur“ auf Seite 1373
- „sp_move_file-Systemprozedur“ auf Seite 1378
- „Systemprozeduren für Verzeichnisse und Dateien“ auf Seite 1164

Beispiele

Die folgende Anweisung verschiebt das Verzeichnis *SQLAnywhere* einschließlich der dazugehörigen Dateien und Unterverzeichnisse nach *c:\temp\SQLAnywhere*.

```
SELECT sp_move_directory('c:\\sqlany\\samples\\SQLAnywhere', 'c:\\temp\\SQLAnywhere');
```

Das ursprüngliche Verzeichnis wird entfernt.

Die folgende Anweisung verschiebt das Verzeichnis *SQLAnywhere* wieder an seinen ursprünglichen Speicherort.

```
SELECT sp_move_directory('c:\\temp\\SQLAnywhere', 'c:\\sqlany\\samples\\SQLAnywhere');
```

sp_move_file-Systemprozedur

Verschiebt die angegebene Datei in ein neues Verzeichnis auf dem Computer.

Syntax

```
sp_move_file(  
    source_path  
    , destination_path  
)
```

Parameter

- **source_path** Verwenden Sie diesen LONG NVARCHAR-Parameter, um den Dateipfad, einschließlich des Dateinamens, der zu verschiebenden Datei anzugeben. Der Pfad kann in absoluter oder relativer Form angegeben werden. Ein relativer Pfad wird relativ zum aktuellen Arbeitsverzeichnis des Datenbankservers aufgelöst. Wenn die Sandboxing-Funktion aktiviert ist, beziehen sich absolute und relative Pfade auf das Verzeichnis, in dem sich die Hauptdatenbankdatei befindet.
- **destination_path** Verwenden Sie diesen LONG NVARCHAR-Parameter, um den Zielpfad, einschließlich des Dateinamens, auf dem Computer anzugeben. Der Pfad kann in absoluter oder relativer Form angegeben werden. Ein relativer Pfad wird relativ zum aktuellen Arbeitsverzeichnis des Datenbankservers aufgelöst. Wenn die Sandboxing-Funktion aktiviert ist, beziehen sich absolute und relative Pfade auf das Verzeichnis, in dem sich die Hauptdatenbankdatei befindet.

Rückgabe

Diese Funktion gibt bei Erfolg 0 und bei Fehler 1 zurück.

Bemerkungen

Diese Funktion verschiebt eine Datei aus einem Verzeichnis in ein anderes.

Privilegien

Sie müssen die Systemprivilegien READ FILE und WRITE FILE haben.

Siehe auch

- „Schlüssel für gesicherte Funktionen erstellen“ [*SQL Anywhere Server - Datenbankadministration*]
- „sp_copy_directory-Systemprozedur“ auf Seite 1356
- „sp_copy_file-Systemprozedur“ auf Seite 1358
- „sp_create_directory-Systemprozedur“ auf Seite 1359
- „sp_delete_directory-Systemprozedur“ auf Seite 1361
- „sp_delete_file-Systemprozedur“ auf Seite 1362
- „sp_list_directory-Systemprozedur“ auf Seite 1373
- „sp_move_directory-Systemprozedur“ auf Seite 1377
- „Systemprozeduren für Verzeichnisse und Dateien“ auf Seite 1164

Beispiele

Die folgende Anweisung verschiebt die Datei *license.txt* in das Verzeichnis *c:\temp* und gibt ihr einen neuen Namen.

```
SELECT sp_move_file('c:\\sqlany\\license.txt', 'c:\\temp\\license.bkp');
```

Die Datei ist an ihrem ursprünglichen Speicherort nicht mehr vorhanden.

Die folgende Anweisung verschiebt die Datei wieder an ihren ursprünglichen Speicherort und gibt ihr wieder ihren ursprünglichen Namen.

```
SELECT sp_move_file('c:\\temp\\license.bkp', 'c:\\sqlany\\license.txt');
```

sp_objectpermission-Systemprozedur

Generiert einen Bericht über Objektprivilegien, die einer angegebenen Rolle oder Benutzer-ID erteilt wurden, oder einen Bericht über Objektprivilegien, die für ein angegebenes Objekt oder einen angegebenen DBSpace erteilt wurden.

Syntax

```
sp_objectpermission(  
  object_name  
  , object_owner  
  , object_type  
)
```

Argumente

- **object_name** Der CHAR(128)-Name für ein Objekt, einen DBSpace, einen Benutzer oder eine Rolle. Wenn dieses Argument nicht angegeben wird, werden die Privilegien des aktuellen Benutzers angezeigt. Der Standardwert ist NULL.
- **object_owner** Der CHAR(128)-Name des Objekteigentümers für den angegebenen Objektnamen. Die Objektprivilegien für das angegebene Objekt mit dem angegebenen Objekteigentümer werden angezeigt. Der Standardwert ist NULL.
- **object_type** Der CHAR(20)-Typ des Datenbankobjekts. Wenn kein Wert angegeben wird, werden Privilegien für alle Objekttypen zurückgegeben. Der Standardwert ist NULL. Folgende Werte sind gültig:
 - DBSPACE
 - FUNCTION
 - MATERIALIZED VIEW
 - PROCEDURE
 - SEQUENCE
 - TABLE (wobei Objektprivilegien auf Spaltenebene ebenfalls angezeigt werden)
 - USER
 - VIEW

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
grantor	CHAR(128)	Gibt die Benutzer-ID des Berechtigungsgebers zurück.
grantee	CHAR(128)	Gibt die Benutzer-ID des Berechtigungsempfängers zurück.
object_name	CHAR(128)	Gibt den Namen des Objekts zurück.
owner	CHAR(128)	Gibt den Namen des Objekteigentümers zurück.
object_type	CHAR(20)	Gibt den Typ des Objekts zurück.
column_name	CHAR(128)	Gibt den Namen der Spalte zurück.
permission	CHAR(20)	Gibt den Namen des Privilegs zurück.
grantable	CHAR(1)	Gibt einen Wert zurück, der angibt, ob das Privileg erteilbar ist.

Bemerkungen

Alle Argumente sind optional und können die folgenden Berichte generieren:

- Wenn die Eingabe ein Objekt ist (Tabelle, Ansicht, Prozedur, Funktion, Sequenz usw.), zeigt die Prozedur eine Liste aller Benutzer und Rollen an, die unterschiedliche Objektprivilegien für das Objekt haben.

- Wenn die Eingabe eine Rolle oder ein Benutzer ist, zeigt die Prozedur eine Liste aller Objektprivilegien an, die der Rolle bzw. dem Benutzer erteilt wurden.
- Wenn die Eingabe ein DBSpace-Name ist, zeigt die Prozedur eine Liste aller Benutzer und Rollen an, die das CREATE-Privileg für den angegebenen DBSpace haben.

Wenn Sie `sp_objectpermission` ausführen, um Objektprivilegien für einen Benutzer oder eine Rolle anzuzeigen, werden die durch Rollenerteilungen geerbten Objektprivilegien ebenfalls angezeigt. Standardmäßig ist `object_type` NULL und die Objektprivilegien werden für alle vorhandenen Objekttypen angezeigt, die mit dem angegebenen Objektnamen übereinstimmen.

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Privilegien

- Keine Privilegien sind erforderlich, um diese Prozedur für Sie selbst auszuführen oder für Objekte, deren Eigentümer Sie sind. Um diese Prozedur jedoch für eine andere Benutzer-ID oder für ein im Eigentum einer anderen Benutzer-ID stehendes Objekt aufrufen zu können, müssen Sie das **MANAGE ANY OBJECT PRIVILEGE**-Systemprivileg haben.
- Um diese Prozedur für einen DBSpace ausführen zu können, müssen Sie das **MANAGE ANY DBSPACE**-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Rollen“ [*SQL Anywhere Server - Datenbankadministration*]
- „Rollen und Privilegien für einen Benutzer oder eine Rolle anzeigen (SQL)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Rollen und Privilegien für einen Benutzer oder eine Rolle anzeigen (Sybase Central)“ [*SQL Anywhere Server - Datenbankadministration*]
- „`sp_sys_priv_role_info`-Systemprozedur“ auf Seite 1399
- „`sp_has_role`-Systemprozedur“ auf Seite 1371
- „`sp_proc_priv`-Systemprozedur“ auf Seite 1384
- „`sp_displayroles`-Systemprozedur“ auf Seite 1363

Beispiel

Die folgende Anweisung gibt die Privilegien auf Objektebene zurück, die der **DIAGNOSTICS**-Systemrolle erteilt wurden. In diesem Beispiel wurden die Ergebnisse gekürzt.

```
CALL sp_objectpermission( 'diagnostics' );
```

grantor	grantee	object_name	owner	object_type	column_name	permission	grantable
SYS	diag-nostics	sa_tmp_diag-nostic_opt-rewrite	dbo	TABLE	(NULL)	SELECT	N
SYS	diag-nostics	sa_tmp_diag-nostic_opt-rewrite	dbo	TABLE	(NULL)	INSERT	N
SYS	diag-nostics	sa_tmp_diag-nostic_opt-rewrite	dbo	TABLE	(NULL)	UPDATE	N
SYS	diag-nostics	sa_tmp_diag-nostic_opt-rewrite	dbo	TABLE	(NULL)	DELETE	N
SYS	diag-nostics	sa_tmp_diag-nostic_optorder	dbo	TABLE	(NULL)	SELECT	N
SYS	diag-nostics	sa_tmp_diag-nostic_optorder	dbo	TABLE	(NULL)	INSERT	N
SYS	diag-nostics	sa_tmp_diag-nostic_optorder	dbo	TABLE	(NULL)	UPDATE	N
SYS	diag-nostics	sa_tmp_diag-nostic_optorder	dbo	TABLE	(NULL)	DELETE	N
SYS	diag-nostics	sa_tmp_diag-nostic_optquantifier	dbo	TABLE	(NULL)	SELECT	N
SYS	diag-nostics	sa_tmp_diag-nostic_optquantifier	dbo	TABLE	(NULL)	UPDATE	N
...

Die Ergebnisse zeigen, dass es viele Tabellen gibt, für die die DIAGNOSTICS-Systemrolle Berechtigungen auf Objektebene hat. Dieses Ergebnis ist sinnvoll, weil die Tabellen zum Speichern von Diagnosedaten für Diagnoseprotokollierung und Anwendungsprofilerstellung verwendet werden.

Die folgende Anweisung gibt die Privilegien auf Objektebene zurück, die dem Benutzer ml_server erteilt wurden.

```
CALL sp_objectpermission( 'ml_server' );
```

Die folgende Anweisung gibt die Privilegien auf Objektebene für den System-DBSpace zurück.

```
CALL sp_objectpermission( object_name='system', object_type='DBSPACE' );
```

sp_parse_json-Systemprozedur

Gibt eine Darstellung von JSON-Daten mit SQL-Datentypen zurück.

Syntax

```
sp_parse_json(  
  var  
  , "json"  
  [, maxlen ]  
)
```

Argumente

- **var** Verwenden Sie diesen LONG VARCHAR-Parameter, um den Namen der zu erstellenden lokalen Variablen anzugeben. Der Typ der Variablen wird zum Ausführungszeitpunkt bestimmt. Wenn die Variable noch nicht vorhanden ist, wird sie erstellt.
- **json** Verwenden Sie diesen LONG NVARCHAR-Parameter, um eine Zeichenfolgendarstellung der JSON-Datenstruktur anzugeben.
- **maxlen** Verwenden Sie diesen INTEGER-Parameter, um die Grenze der Rekursion anzugeben. Knoten unterhalb der festgelegten Tiefe werden nicht verarbeitet. Stattdessen werden sie als JSON-Fragment zurückgegeben.

Bemerkungen

Diese Prozedur verarbeitet ein JSON-Objekt und gibt die verarbeiteten Daten als SQL-Datentyp zurück. Der Typ der Rückgabeveriablen wird bestimmt, wenn die Prozedur ausgeführt wird. In den meisten Fällen weisen die Rückgabeveriablen entweder ROW- oder ARRAY SQL-Datentypen auf.

Die sp_parse_json-Systemprozedur gibt ein VARCHAR-Fragment für seine zugrunde liegenden ARRAY- oder OBJECT-Daten zurück, wenn nachfolgende Instanzen eine andere Anzahl von Knoten zurückgeben.

JSON-Objektbezeichner müssen den im Datenbankserver definierten Bezeichnerregeln entsprechen. Außerdem erzwingt der Datenbankserver für JSON-Datentypen dieselben Grenzen wie für die zugrunde liegenden ROW- und ARRAY-Datentypen.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Zusammengesetzte Datentypen“ auf Seite 136
- „Bezeichner“ auf Seite 4
- „ROW-Konstruktor [zusammengesetzt]“ auf Seite 371
- „ARRAY-Konstruktor [zusammengesetzt]“ auf Seite 168

Beispiel

Im folgenden Beispiel wird eine Tabelle mit einigen Daten so eingerichtet, dass sie eine JSON-Zeichenfolge ([{"name": "Frank", "age": 51}, {"name": "Bill", "age": 22}, {"name": "Jackie", "age": 37}]) generiert, die syntaktisch analysiert werden kann. Anschließend wird die JSON-Zeichenfolge syntaktisch analysiert und ein SQL-Array-Datentyp wird zurückgegeben.

```
BEGIN
  DECLARE json_data LONG VARCHAR;
  CREATE LOCAL TEMPORARY TABLE test (
    name AS VARCHAR(64),
    age AS INT);

  INSERT INTO test (name, age) VALUES ('Frank', 51);
  INSERT INTO test (name, age) VALUES ('Bill', 22);
  INSERT INTO test (name, age) VALUES ('Jackie', 37);

  SELECT * INTO json_data FROM test FOR JSON RAW;

  CALL sp_parse_json ( 'sql_array', json_data );

  SELECT sql_array [[row_num]].name AS name, sql_array [[row_num]].age
  AS age
  FROM sa_rowgenerator ( 1, 3 );
END;
```

sp_proc_priv-Systemprozedur

Gibt die Liste der Systemprivilegien zurück, die zum Ausführen einer Prozedur erforderlich sind.

Syntax

```
sp_proc_priv(
  [ 'proc_name' ]
)
```

Argumente

- **proc_name** Dieser CHAR(128)-Parameter gibt den Namen der Prozedur an, für die Privilegien zurückgegeben werden sollen. Wenn *proc_name* nicht angegeben wird, werden für alle Prozeduren die erforderlichen Privilegien zurückgegeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
proc_name	CHAR(128)	Der Name der Prozedur.
privilege	LONG VARCHAR	Die Liste der Privilegien, die zum Ausführen der Prozedur erforderlich sind.

Bemerkungen

Wenn in der Ergebnismenge für eine gespeicherte Prozedur eine kommasetrennte Liste von Privilegien angezeigt wird, bedeutet dies, dass jedes einzelne dieser Privilegien ausreicht. Wenn für eine gespeicherte Prozedur mehrere Zeilen angezeigt werden, ist ein Privileg aus jeder Zeile erforderlich, um die gespeicherte Prozedur auszuführen.

Wenn `sp_proc_priv` ohne `proc_name` aufgerufen wird, werden Privileginformationen für alle Prozeduren zurückgegeben, die Privilegien erfordern. Systemprozeduren, die keine Privilegien erfordern, werden nicht in die Ergebnisse einbezogen.

Hinweis

Diese Prozedur listet nur Privilegien für eine gespeicherte Prozedur auf, bei denen die Berechtigungsüberprüfung für die Prozedur nie fehlschlägt. Möglicherweise gibt es weitere Privilegien, die die Berechtigungsüberprüfung zum Ausführen der Prozedur unter bestimmten Bedingungen bestehen, aber diese Privilegien werden nicht aufgelistet.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Rollen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „`sp_sys_priv_role_info`-Systemprozedur“ auf Seite 1399
- „`sp_has_role`-Systemprozedur“ auf Seite 1371
- „`sp_displayroles`-Systemprozedur“ auf Seite 1363
- „`sp_objectpermission`-Systemprozedur“ auf Seite 1379

Beispiel

Im folgenden Beispiel werden die Privilegien zurückgegeben, die erforderlich sind, um die `sa_table_fragmentation`-Systemprozedur auszuführen. Da nur eine Zeile zurückgegeben wird, reicht jedes der Privilegien in der kommasetrennten Liste aus, um `sa_table_fragmentation` auszuführen.

```
CALL sp_proc_priv( 'sa_table_fragmentation' );
```

proc_name	privilege
sa_table_fragmentation	MANAGE ANY STATISTICS, MONITOR

Im folgenden Beispiel werden die Privilegien für die sa_install_feature-Systemprozedur zurückgegeben. Da mehrere Zeilen zurückgegeben werden, ist ein Privileg aus jeder Zeile erforderlich, um sa_install_feature auszuführen.

```
CALL sp_proc_priv( 'sa_install_feature' );
```

proc_name	privilege
sa_install_feature	SELECT ANY TABLE
sa_install_feature	MANAGE ANY SPATIAL OBJECT
sa_install_feature	MANAGE ANY OBJECT PRIVILEGE
sa_install_feature	CREATE ANY PROCEDURE, CREATE ANY OBJECT

sp_remote_columns-Systemprozedur

Erzeugt eine Liste der Spalten in einer entfernten Tabelle und eine Beschreibung ihrer Datentypen.

Der Server muss mit der CREATE SERVER-Anweisung definiert werden, um diese Systemprozedur verwenden zu können.

Syntax

```
sp_remote_columns(  
  @server_name  
, @table_name  
[, @table_owner  
[, @table_qualifier ] ]  
)
```

Argumente

- **@server_name** Verwenden Sie diesen CHAR(128)-Parameter, um eine Zeichenfolge anzugeben, die den Servernamen enthält, wie er durch die CREATE SERVER-Anweisung festgelegt wurde.
- **@table_name** Verwenden Sie diesen CHAR(128)-Parameter, um den Namen der entfernten Tabelle anzugeben.
- **@table_owner** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümer von *table_name* anzugeben. Der Standardwert ist '% '.
- **@table_qualifier** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Namen der Datenbank anzugeben, in der sich *table_name* befindet. Der Standardwert ist '% '.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
database	CHAR(128)	Der Name der Datenbank
owner	CHAR(128)	Der Name des Datenbankeigentümers
table_name	CHAR(128)	Der Tabellenname.
column_name	CHAR(128)	Der Name einer Spalte
domain_id	SMALLINT	Eine Ganzzahl, die den Datentyp der Spalte anzeigt
width	INTEGER	Die Bedeutung dieser Spalte hängt vom Datentyp ab. Bei Zeichentypen stellt "width" die Anzahl der Zeichen dar.
scale	SMALLINT	Die Bedeutung dieser Spalte hängt vom Datentyp ab. Bei numerischen Datentypen ist "scale" die Anzahl der Ziffern hinter dem Dezimalzeichen.
nullable	SMALLINT	Wenn NULL-Spaltenwerte zulässig sind, ist der Wert 1. Andernfalls ist der Wert 0.
base_type_str	CHAR(4096)	Die zugehörige Typzeichenfolge, die den physischen Spaltentyp darstellt.

Bemerkungen

Wenn Sie eine CREATE EXISTING TABLE-Anweisung eingeben und eine Spaltenliste angeben, sollten Sie eine Liste der Spalten erstellen, die in einer entfernten Tabelle zur Verfügung stehen. sp_remote_columns erzeugt eine Liste der Spalten in einer entfernten Tabelle und eine Beschreibung ihrer Datentypen. Wenn Sie eine Datenbank angeben, müssen Sie entweder einen Eigentümer oder den Wert NULL angeben.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Ferndatenzugriff“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Serverklassen für den Ferndatenzugriff“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „CREATE SERVER-Anweisung“ auf Seite 701

Standards und Kompatibilität

- **Sybase** Wird von Open Client/Open Server unterstützt

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
pk_database	CHAR(128)	Legt die Datenbank fest, die die Primärschlüsseltabelle enthält
pk_owner	CHAR(128)	Der Eigentümer der Primärschlüsseltabelle
pk_table	CHAR(128)	Die Primärschlüsseltabelle
pk_column	CHAR(128)	Der Name der Primärschlüsselspalte
fk_database	CHAR(128)	Die Datenbank, die die Fremdschlüsseltabelle enthält
fk_owner	CHAR(128)	Der Eigentümer der Fremdschlüsseltabelle
fk_table	CHAR(128)	Die Fremdschlüsseltabelle
fk_column	CHAR(128)	Der Name der Fremdspalte
key_seq	SMALLINT	Die Schlüsselsequenznummer
fk_name	CHAR(128)	Der Fremdschlüsselname
pk_name	CHAR(128)	Der Primärschlüsselname

Bemerkungen

Diese Prozedur liefert Informationen über die entfernten Tabellen, die einen Fremdschlüssel auf einer bestimmten Primärschlüsseltabelle haben. Die Ergebnismenge der `sp_remote_exported_keys`-Systemprozedur enthält Datenbank, Eigentümer, Tabelle, Spalte und Name sowohl für den Primär- als auch für den Fremdschlüssel sowie die Fremdschlüsselsequenz für die Fremdschlüsselspalten. Die Ergebnismenge kann wegen der zugrunde liegenden ODBC- und JDBC-Aufrufe variieren, es wird jedoch immer eine Information über Tabelle und Spalte eines Fremdschlüssels zurückgegeben.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- [„CREATE SERVER-Anweisung“ auf Seite 701](#)
- [„Fremdschlüssel“ \[*SQL Anywhere Server - SQL-Benutzerhandbuch*\]](#)

Beispiel

In diesem Beispiel werden Informationen zu den Fremdschlüsselbeziehungen in der Tabelle "ULEmployee" auf dem Fremdserver RemoteSA zurückgegeben:

- **@server_name** Verwenden Sie diesen CHAR(128)-Parameter, um den Server anzugeben, auf dem sich die Fremdschlüsseltabelle befindet. Für diesen Parameter muss ein Wert angegeben werden.
- **@table_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um die Tabelle anzugeben, die den Fremdschlüssel enthält. Für diesen Parameter muss ein Wert angegeben werden.
- **@table_owner** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümer der Fremdschlüsseltabelle anzugeben. Der Standardwert ist '%'
- **@table_qualifier** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um die Datenbank anzugeben, die die Fremdschlüsseltabelle enthält. Der Standardwert ist '%'

Spaltenname	Datentyp	Beschreibung
pk_database	CHAR(128)	Legt die Datenbank fest, die die Primärschlüsseltabelle enthält
pk_owner	CHAR(128)	Der Eigentümer der Primärschlüsseltabelle
pk_table	CHAR(128)	Die Primärschlüsseltabelle
pk_column	CHAR(128)	Der Name der Primärschlüsselspalte
fk_database	CHAR(128)	Die Datenbank, die die Fremdschlüsseltabelle enthält
fk_owner	CHAR(128)	Der Eigentümer der Fremdschlüsseltabelle
fk_table	CHAR(128)	Die Fremdschlüsseltabelle

Spaltenname	Datentyp	Beschreibung
fk_column	CHAR(128)	Der Name der Fremdspalte
key_seq	SMALLINT	Die Schlüsselsequenznummer
fk_name	CHAR(128)	Der Fremdschlüsselname
pk_name	CHAR(128)	Der Primärschlüsselname

Bemerkungen

Fremdschlüssel referenzieren eine Zeile in einer separaten Tabelle, die den entsprechenden Primärschlüssel enthält. Mit dieser Prozedur können Sie eine Liste der entfernten Tabellen mit Primärschlüsseln erhalten, die einer bestimmten Fremdschlüsseltabelle entsprechen. Die Ergebnismenge der gespeicherten Prozedur `sp_remote_imported_keys` beinhaltet Datenbank, Eigentümer, Tabelle, Spalte und Name sowohl für den Primär- als auch für den Fremdschlüssel sowie die Fremdschlüsselsequenz für die Fremdspalten. Die Ergebnismenge kann wegen der zugrunde liegenden ODBC- und JDBC-Aufrufe variieren, es wird jedoch immer eine Information über Tabelle und Spalte eines Primärschlüssels zurückgegeben.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „CREATE SERVER-Anweisung“ auf Seite 701
- „Fremdschlüssel“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Beispiel

Im folgenden Beispiel werden die Tabellen mit Primärschlüsseln zurückgegeben, die einem Fremdschlüssel für die Tabelle "ULOrder" auf dem Fremdserver RemoteSA entsprechen:

```
CALL sp_remote_imported_keys( 'RemoteSA', 'ULOrder', 'DBA' );
```

sp_remote_pcols-Systemprozedur

Erzeugt eine Liste der Spalten in einer entfernten Tabelle und eine Beschreibung ihrer Datentypen.

Der Server muss mit der CREATE SERVER-Anweisung definiert werden, um diese Systemprozedur verwenden zu können.

Syntax

```
sp_remote_pcols(
  @server_name
```

```
, @sp_name
[, @sp_owner
[, @sp_qualifier ] ]
)
```

Argumente

- **@server_name** Verwenden Sie diesen CHAR(128)-Parameter, um eine Zeichenfolge anzugeben, die den Servernamen enthält, wie er durch die CREATE SERVER-Anweisung festgelegt wurde.
- **@sp_name** Verwenden Sie diesen CHAR(128)-Parameter, um den Namen der entfernten Tabelle anzugeben.
- **@sp_owner** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümer von *sp_name* anzugeben. Der Standardwert ist '% '.
- **@sp_qualifier** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Namen der Datenbank anzugeben, in der sich *sp_name* befindet. Der Standardwert ist '% '.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
database	CHAR(128)	Der Name der Datenbank
owner	CHAR(128)	Der Name des Datenbankeigentümers
proc_name	CHAR(128)	Der Name der gespeicherten Prozedur.
parm_name	CHAR(128)	Der Name des Parameters oder der Ergebnismengenspalte.
parm_mode	CHAR(10)	Der Modus des Parameters oder der Ergebnismengenspalte (IN, OUT, INOUT, RESULT).
domain_id	SMALLINT	Ein INTEGER-Wert, der den Datentyp des Parameters oder der Ergebnismengenspalte anzeigt
width	INTEGER	Die Bedeutung dieser Spalte hängt vom Datentyp ab. Bei Zeichentypen stellt "width" die Anzahl der Zeichen dar.
scale	SMALLINT	Die Bedeutung dieser Spalte hängt vom Datentyp ab. Bei numerischen Datentypen ist "scale" die Anzahl der Ziffern hinter dem Dezimalzeichen.
nullable	SMALLINT	Wenn NULL-Parameterwerte zulässig sind, ist der Wert 1. Andernfalls ist der Wert 0.

Bemerkungen

Wenn Sie eine CREATE PROCEDURE...AT-Anweisung eingeben und dabei eine Parameterliste angeben möchten oder Informationen zu eventuell zurückgegebenen Ergebnismengen benötigen, kann es hilfreich sein, eine Liste der Parameter und Ergebnismengenspalten abzurufen, die für eine entfernte

gespeicherte Prozedur verfügbar sind. `sp_remote_pcols` erzeugt eine Liste der Parameter und Ergebnismengenspalten einer entfernten gespeicherten Prozedur sowie eine Beschreibung der dazugehörigen Datentypen. Wenn Sie eine Datenbank angeben, müssen Sie entweder einen Eigentümer oder den Wert NULL angeben.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Ferndatenzugriff“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Serverklassen für den Ferndatenzugriff“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „CREATE SERVER-Anweisung“ auf Seite 701

Standards und Kompatibilität

- **Sybase** Wird von Open Client/Open Server unterstützt

Beispiel

Im folgenden Beispiel werden Informationen zu den Parametern und Ergebnismengenspalten zurückgegeben, die für die gespeicherte Prozedur `ULOrderDownload` im entfernten SQL Anywhere-Datenbankserver `RemoteSA` vorhanden sind. Der Eigentümer der gespeicherten Prozedur ist `DBA`.

```
CALL sp_remote_pcols('RemoteSA', 'ULOrderDownload', 'DBA');
```

Im folgenden Beispiel werden mithilfe des Fremdservers `RemoteASE` Informationen zu den Parametern und Ergebnismengenspalten zurückgegeben, die für die gespeicherte Prozedur `col_name` in der Adaptive Server Enterprise-Datenbank "Production" vorhanden sind. Der Eigentümer der entfernten Prozedur wird nicht angegeben.

```
CALL sp_remote_pcols( 'RemoteASE', 'col_name', null, 'Production' );
```

sp_remote_primary_keys-Systemprozedur

Liefert Primärschlüsselangaben zu entfernten Tabellen über entfernten Datenzugriff

Syntax

```
sp_remote_primary_keys(  
    @server_name  
    , @table_name  
    [, @table_owner  
    [, @table_qualifier ] ]  
)
```

Argumente

- **@server_name** Verwenden Sie diesen CHAR(128)-Parameter, um den Namen des Fremdservers anzugeben.

- **@table_name** Verwenden Sie diesen CHAR(128)-Parameter, um den Namen der entfernten Tabelle anzugeben.
- **@table_owner** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümer der entfernten Tabelle anzugeben. Der Standardwert ist '% '.
- **@table_qualifier** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Namen der entfernten Datenbank anzugeben. Der Standardwert ist '% '.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
database	CHAR(128)	Der Name der entfernten Datenbank
owner	CHAR(128)	Eigentümer der Tabelle
table_name	CHAR(128)	Der Name der Tabelle.
column_name	CHAR(128)	Der Name der Primärschlüsselspalte
key_seq	SMALLINT	Die Primärschlüssel-Sequenznummer
pk_name	CHAR(128)	Der Primärschlüsselname

Bemerkungen

Diese Systemprozedur liefert Primärschlüsselangaben zu entfernten Tabellen über entfernten Datenzugriff.

Aufgrund von Unterschieden in den zugrunde liegenden ODBC-Aufrufen weichen die zurückgegebenen Informationen leicht vom Katalog-/Datenbankwert ab, abhängig von der für den Server angegebenen Ferndatenzugriffsklasse.

Privilegien

Keine

Standards und Kompatibilität

- **Sybase** Wird von Open Client/Open Server unterstützt

Nebenwirkungen

Keine

Siehe auch

- „Ferndatenzugriff“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Serverklassen für den Ferndatenzugriff“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „CREATE SERVER-Anweisung“ auf Seite 701

Beispiele

Im folgenden Beispiel werden Informationen zu den Primärschlüsseln zurückgegeben, die in Tabellen in einem SQL Anywhere-Fremdserver namens RemoteSA enthalten sind, deren Eigentümer DBA ist.

```
CALL sp_remote_primary_keys( 'RemoteSA', null, 'DBA' );
```

Führen Sie die folgende Anweisung aus, um eine Liste der Primärschlüssel abzurufen, die in allen Tabellen der Produktionsdatenbank in einem Adaptive Server Enterprise-Server namens RemoteASE enthalten sind, deren Eigentümer Fred ist:

```
CALL sp_remote_primary_keys( 'RemoteASE', null, 'Fred', 'production' );
```

sp_remote_procedures-Systemprozedur

Gibt eine Liste der Prozeduren auf einem Fremdserver zurück.

Syntax

```
sp_remote_procedures(
    @server_name
    [, @sp_name
    [, @sp_owner
    [, @sp_qualifier ] ] ]
)
```

Argumente

- **@server_name** Verwenden Sie diesen CHAR(128)-Parameter, um den Namen des Fremdservers anzugeben.
- **@sp_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Namen der entfernten gespeicherten Prozedur anzugeben. Der Standardwert ist '%'.
 Wenn Sie diesen Parameter angeben, werden nur die Prozeduren mit dem angegebenen Namen zurückgegeben.
- **@sp_owner** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümer der entfernten gespeicherten Prozedur anzugeben. Der Standardwert ist '%'.
 Wenn Sie diesen Parameter angeben, werden nur die Prozeduren mit dem angegebenen Eigentümer zurückgegeben.
- **@sp_qualifier** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um die Datenbank anzugeben, in der sich *sp_name* befindet. Der Standardwert ist '%'.
 Wenn Sie diesen Parameter angeben, werden nur die Prozeduren in der angegebenen Datenbank zurückgegeben.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
database	CHAR(128)	Der Name der entfernten Datenbank
owner	CHAR(128)	Der Name des Eigentümers der gespeicherten Prozedur.
proc_name	CHAR(128)	Der Name der gespeicherten Prozedur.

Bemerkungen

Der Server muss mit der CREATE SERVER-Anweisung definiert werden, um diese Systemprozedur verwenden zu können.

Beim Konfigurieren Ihres Datenbankservers kann hilfreich sein, eine Liste der entfernten gespeicherten Prozeduren abzurufen, die auf einem bestimmten Server verfügbar sind. Diese Prozedur gibt eine Liste der gespeicherten Prozeduren auf einem Server zurück.

Die Prozedur akzeptiert vier Parameter. Wenn eine gespeicherte Prozedur, ein Eigentümer oder ein Datenbankname angegeben wird, beschränkt sich die Liste der gespeicherten Prozeduren auf diejenigen, die mit den Argumenten übereinstimmen.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „Ferndatenzugriff“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Serverklassen für den Ferndatenzugriff“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „CREATE PROCEDURE-Anweisung“ auf Seite 681
- „CREATE SERVER-Anweisung“ auf Seite 701

Standards und Kompatibilität

- **Sybase** Wird von Open Client/Open Server unterstützt

Beispiele

Im folgenden Beispiel werden Informationen zu den gespeicherten Prozeduren in einem SQL Anywhere-Fremdserver namens RemoteSA zurückgegeben, deren Eigentümer DBA ist.

```
CALL sp_remote_procedures( 'RemoteSA', null, 'DBA' );
```

Im folgenden Beispiel werden Informationen zu den gespeicherten Prozeduren in der Produktionsdatenbank in einem Adaptive Server Enterprise-Fremdserver namens RemoteASE zurückgegeben, deren Eigentümer Fred ist:

```
CALL sp_remote_procedures( RemoteASE', null, 'Fred', 'production' );
```

sp_remote_tables-Systemprozedur

Gibt eine Liste der Tabellen auf einem Server zurück

Syntax

```
sp_remote_tables(  
    @server_name  
    [, @table_name  
    [, @table_owner  
    [, @table_qualifier  
    [, @with_table_type ] ] ] ]  
)
```

Argumente

- **@server_name** Verwenden Sie diesen CHAR(128)-Parameter, um den Namen des Fremdservers anzugeben.
- **@table_name** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Namen der entfernten Tabelle anzugeben. Der Standardwert ist '%'.
- **@table_owner** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um den Eigentümer der entfernten Tabelle anzugeben. Der Standardwert ist '%'.
- **@table_qualifier** Verwenden Sie diesen optionalen CHAR(128)-Parameter, um die Datenbank anzugeben, in der sich *table_name* befindet. Der Standardwert ist '%'.
- **@with_table_type** Mit diesem optionalen BIT-Parameter können Sie angeben, welche Typen von entfernten Tabellen einbezogen werden sollen. Der Standardwert ist 0. Geben Sie 1 an, wenn Sie möchten, dass die Ergebnismenge eine Spalte mit Tabellentypen enthält, oder 0, wenn dies nicht der Fall ist.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
database	CHAR(128)	Der Name der entfernten Datenbank
owner	CHAR(128)	Der Name des Tabelleneigentümers.
table_name	CHAR(128)	Der Name der Tabelle.
table_type	CHAR(128)	Gibt den Tabellentyp an. Der Wert hängt vom Typ des entfernten Servers ab. Beispiel: TABLE, VIEW, SYS und GBL TEMP sind zulässige Werte.

Bemerkungen

Der Server muss mit der CREATE SERVER-Anweisung definiert werden, um diese Systemprozedur verwenden zu können.

Wenn Sie Ihren Datenbankserver konfigurieren, kann es sinnvoll sein, eine Liste der entfernten Tabellen auszugeben, die auf einem bestimmten Server verfügbar sind. Diese Prozedur gibt eine Liste der Tabellen auf einem Server zurück.

Die Prozedur akzeptiert fünf Parameter. Wenn ein Tabellenbesitzer oder ein Datenbankname angegeben wird, beschränkt sich die Liste der Tabellen auf jene, die den Argumenten entsprechen.

Privilegien

Keine

Nebenwirkungen

Keine

Standards und Kompatibilität

- **Sybase** Wird von Open Client/Open Server unterstützt

Siehe auch

- „Ferndatenzugriff“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Serverklassen für den Ferndatenzugriff“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „CREATE SERVER-Anweisung“ auf Seite 701

Beispiele

Im folgenden Beispiel werden Informationen zu den Tabellen in einem SQL Anywhere-Fremdserver namens RemoteSA zurückgegeben, deren Eigentümer DBA ist.

```
CALL sp_remote_tables( 'RemoteSA', null, 'DBA' );
```

Führen Sie die folgende Anweisung aus, um eine Liste aller Tabellen der Produktionsdatenbank in einem Adaptive Server Enterprise-Server namens RemoteASE abzurufen, deren Eigentümer Fred ist:

```
CALL sp_remote_tables( 'RemoteASE', null, 'Fred', 'production' );
```

Führen Sie die folgende Anweisung aus, um eine Liste aller Microsoft Excel-Arbeitsmappen abzurufen, die aus einer durch den Server RemoteExcel referenzierten ODBC-Datenquelle verfügbar sind:

```
CALL sp_remote_tables( 'RemoteExcel' );
```

sp_servercaps-Systemprozedur

Zeigt Informationen über die Fähigkeiten eines entfernten Servers an.

Syntax

```
sp_servercaps( @server_name )
```

Argumente

- **@server_name** Verwenden Sie diesen CHAR(128)-Parameter, um einen Server anzugeben, der mit der CREATE SERVER-Anweisung festgelegt wird. @server_name ist derselbe Servername, der in der CREATE SERVER-Anweisung verwendet wurde.

Ergebnisse

Column	Type	Description
capid	INTEGER	Der Funktionsbezeichner.
capname	CHAR(128)	Der Name der Fähigkeit.
capvalue	CHAR(128)	Die Einstellung der Funktion, normalerweise T (TRUE) oder F (FALSE).

Bemerkungen

Der Server muss mit der CREATE SERVER-Anweisung definiert werden, um diese Systemprozedur verwenden zu können.

Diese Prozedur zeigt Informationen über die Fähigkeiten eines entfernten Servers an. Die Informationen über die Fähigkeiten werden verwendet, um zu ermitteln, wie viel von einer SQL-Anweisung an einen Fremdserver weitergeleitet werden kann. Die Systemtabelle ISYSCAPABILITY, die eine Liste der Serverfähigkeiten enthält, wird erst gefüllt, wenn eine Verbindung zum ersten Fremdserver hergestellt wird.

Standards und Kompatibilität

- **Sybase** Wird von Open Client/Open Server unterstützt

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „SYSCAPABILITY-Systemansicht“ auf Seite 1439
- „SYSCAPABILITYNAME-Systemansicht“ auf Seite 1439
- „Ferndatenzugriff“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Serverklassen für den Ferndatenzugriff“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „CREATE SERVER-Anweisung“ auf Seite 701

Beispiel

Führen Sie die folgende Anweisung aus, um Informationen zum Fremdserver RemoteSA abzurufen:

```
CALL sp_servercaps( 'RemoteSA' );
```

sp_sys_priv_role_info-Systemprozedur

Gibt die 1:1-Zuordnung zwischen Systemprivilegien und Systemrollen zurück.

Syntax

```
sp_sys_priv_role_info( )
```

Ergebnismenge

Column	Type	Description
sys_priv_name	CHAR(128)	Der Name des Systemprivilegs.
sys_priv_role_name	CHAR(128)	Der Name der entsprechenden Systemrolle.

Column	Type	Description
sys_priv_id	UNSIGNED INTEGER	Die ID-Nummer für die Systemrolle.

Bemerkungen

sp_sys_priv_role_info meldet die vollständige 1:1-Zuordnung zwischen Systemprivilegien und Systemrollen sowie Rollen-IDs. Diese Prozedur gibt für jedes Systemprivileg eine Zeile zurück.

Diese Prozedur ist hauptsächlich für die interne Verwendung vorgesehen und wird von Dienstprogrammen ausgeführt, die eine erteilte Systemrolle dem entsprechenden Systemprivileg zuordnen müssen und umgekehrt.

Privilegien

Keine

Siehe auch

- „Rollen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sp_has_role-Systemprozedur“ auf Seite 1371
- „sp_proc_priv-Systemprozedur“ auf Seite 1384
- „sp_displayroles-Systemprozedur“ auf Seite 1363
- „sp_objectpermission-Systemprozedur“ auf Seite 1379

Standards und Kompatibilität

- **SQL/2008** Erweiterung des Herstellers.

Beispiel

Die folgende Anweisung gibt die Zuordnung von Systemprivilegien zur entsprechenden Systemrolle zurück.

```
CALL sp_sys_priv_role_info();
```

sys_priv_name	sys_priv_role_name	sys_priv_id
VALIDATE ANY OBJECT	SYS_VALIDATE_ANY_OBJECT_ROLE	2147483819
REORGANIZE ANY OBJECT	SYS_REORGANIZE_ANY_OBJECT_ROLE	2147483820
BACKUP DATABASE	SYS_BACKUP_DATABASE_ROLE	2147483821
MANAGE ANY EVENT	SYS_MANAGE_ANY_EVENT_ROLE	2147483822
ALTER DATABASE	SYS_ALTER_DATABASE_ROLE	2147483823
SERVER OPERATOR	SYS_SERVER_OPERATOR_ROLE	2147483824
UPGRADE ROLE	SYS_UPGRADE_ROLE_ROLE	2147483825

sys_priv_name	sys_priv_role_name	sys_priv_id
MANAGE ANY LDAP SERVER	SYS_MANAGE_ANY_LDAP_SERVER_ROLE	2147483827
MANAGE CERTIFICATES	SYS_MANAGE_CERTIFICATES_ROLE	2147483828
CREATE ANY SEQUENCE	SYS_CREATE_ANY_SEQUENCE_ROLE	2147483829
...

sp_trace_event_fields-Systemprozedur

Gibt Informationen zu den Feldern des angegebenen Trace-Ereignisses zurück.

Syntax

```
sp_trace_event_fields( [ event_name ] )
```

Argumente

- event_name** Verwenden Sie diesen optionalen CHAR(256)-Parameter, um den Trace-Ereignisnamen anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
event_name	CHAR(256)	Gibt den Namen des Trace-Ereignisses zurück. Namen von System-Trace-Ereignissen haben das Präfix SYS_ .
field_name	CHAR(256)	Gibt den Feldnamen zurück.
field_id	INTEGER	Gibt die Ordnungszahl des Felds innerhalb des Trace-Ereignisses zurück.
field_domain	INTEGER	Gibt die Domänen-ID des Felds zurück. Fremdschlüssel für SYSDOMAIN.domain_id.
field_description	LONG VARCHAR	Gibt eine Beschreibung des Feldinhalts zurück.

Bemerkungen

Wenn *event_name* NULL ist, gibt diese Prozedur die Felder für alle Trace-Ereignisse zurück.

Privilegien

Sie müssen das MANAGE ANY TRACE SESSION-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Ereignisprotokollierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE TEMPORARY TRACE EVENT-Anweisung“ auf Seite 753
- „CREATE TEMPORARY TRACE EVENT SESSION-Anweisung“ auf Seite 755
- „ALTER TRACE EVENT SESSION-Anweisung“ auf Seite 538
- „DROP TRACE EVENT-Anweisung“ auf Seite 835
- „DROP TRACE EVENT SESSION-Anweisung“ auf Seite 836
- „NOTIFY TRACE EVENT-Anweisung“ auf Seite 963
- „sp_trace_events-Systemprozedur“ auf Seite 1407
- „sp_trace_event_sessions-Systemprozedur“ auf Seite 1406
- „sp_trace_event_session_events-Systemprozedur“ auf Seite 1402
- „sp_trace_event_session_targets-Systemprozedur“ auf Seite 1405
- „sp_trace_event_session_target_options-Systemprozedur“ auf Seite 1403
- „Dienstprogramm für die Verwaltung von ETD-Dateien (dbmanagetd)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Die folgende Anweisung gibt Informationen zu den Feldern für alle Trace-Ereignissitzungen in der Datenbank zurück:

```
SELECT * FROM sp_trace_event_fields( );
```

sp_trace_event_session_events-Systemprozedur

Listet die Trace-Ereignisse auf, die Teil einer bestimmten Trace-Ereignissitzung sind.

Syntax

```
sp_trace_event_session_events( [ session_name ] )
```

Parameter

- **session_name** Verwenden Sie diesen optionalen CHAR(256)-Parameter, um den Namen der Trace-Ereignissitzung anzugeben. Der Standardwert ist NULL. Wenn der Sitzungsname nicht angegeben wird oder NULL ist, werden Informationen für alle Trace-Ereignissitzungen zurückgegeben.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
session_name	CHAR(256)	Gibt den Namen der Trace-Ereignissitzung zurück.
event_name	CHAR(256)	Gibt den Namen des Trace-Ereignisses zurück.

Bemerkungen

Verwenden Sie diese Systemprozedur, um zu ermitteln, welche Trace-Ereignisse (sowohl systemdefinierte als auch benutzerdefinierte) für eine Trace-Ereignissitzung protokolliert werden.

Privilegien

Sie müssen das `MANAGE ANY TRACE SESSION`-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Ereignisprotokollierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE TEMPORARY TRACE EVENT-Anweisung“ auf Seite 753
- „CREATE TEMPORARY TRACE EVENT SESSION-Anweisung“ auf Seite 755
- „ALTER TRACE EVENT SESSION-Anweisung“ auf Seite 538
- „DROP TRACE EVENT-Anweisung“ auf Seite 835
- „DROP TRACE EVENT SESSION-Anweisung“ auf Seite 836
- „NOTIFY TRACE EVENT-Anweisung“ auf Seite 963
- „sp_trace_events-Systemprozedur“ auf Seite 1407
- „sp_trace_event_fields-Systemprozedur“ auf Seite 1401
- „sp_trace_event_sessions-Systemprozedur“ auf Seite 1406
- „sp_trace_event_session_targets-Systemprozedur“ auf Seite 1405
- „sp_trace_event_session_target_options-Systemprozedur“ auf Seite 1403
- „Dienstprogramm für die Verwaltung von ETD-Dateien (dbmanagetd)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Diese Anweisung gibt Informationen zu den Ereignissen zurück, die Teil aller Ereignisprotokollierungssitzungen für die aktuelle Datenbank sind:

```
SELECT * FROM sp_trace_event_session_events( );
```

sp_trace_event_session_target_options-Systemprozedur

Listet die Zieloptionen für eine Trace-Ereignissitzung auf.

Syntax

```
sp_trace_event_session_target_options( [ session_name ] )
```

Argumente

- **session_name** Verwenden Sie diesen optionalen CHAR(256)-Parameter, um den Namen der Trace-Ereignissitzung anzugeben. Der Standardwert ist NULL. Wenn der Sitzungsname nicht angegeben wird oder NULL ist, werden Informationen für alle Trace-Ereignissitzungen zurückgegeben.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
session_name	CHAR(256)	Gibt den Sitzungsnamen zurück.
target_type	CHAR(256)	Gibt den Zieltyp zurück (z.B. FILE).
option_name	CHAR(256)	Gibt den Optionsnamen zurück.
option_value	LONG VARCHAR	Gibt den Optionswert zurück.

Bemerkungen

Diese Prozedur gibt Informationen zu den Optionen für eine oder alle Trace-Ereignissitzungen in der aktuellen Datenbank zurück.

Privilegien

Sie müssen das `MANAGE ANY TRACE SESSION`-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Ereignisprotokollierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE TEMPORARY TRACE EVENT-Anweisung“ auf Seite 753
- „CREATE TEMPORARY TRACE EVENT SESSION-Anweisung“ auf Seite 755
- „ALTER TRACE EVENT SESSION-Anweisung“ auf Seite 538
- „DROP TRACE EVENT-Anweisung“ auf Seite 835
- „DROP TRACE EVENT SESSION-Anweisung“ auf Seite 836
- „NOTIFY TRACE EVENT-Anweisung“ auf Seite 963
- „sp_trace_events-Systemprozedur“ auf Seite 1407
- „sp_trace_event_fields-Systemprozedur“ auf Seite 1401
- „sp_trace_event_sessions-Systemprozedur“ auf Seite 1406
- „sp_trace_event_session_events-Systemprozedur“ auf Seite 1402
- „sp_trace_event_session_targets-Systemprozedur“ auf Seite 1405
- „Dienstprogramm für die Verwaltung von ETD-Dateien (dbmanageetd)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Die folgende Anweisung gibt Informationen zu den Zieloptionen für alle Trace-Ereignissitzungen in der Datenbank zurück:

```
SELECT * FROM sp_trace_event_session_target_options( );
```

sp_trace_event_session_targets-Systemprozedur

Listet die Ziele einer Trace-Sitzung auf.

Syntax

```
sp_trace_event_session_targets( [ session_name ] )
```

Argumente

- **session_name** Verwenden Sie diesen optionalen CHAR(256)-Parameter, um den Namen der Trace-Ereignissitzung anzugeben. Der Standardwert ist NULL. Wenn der Sitzungsname nicht angegeben wird oder NULL ist, werden Informationen zu allen Trace-Ereignissitzungen in der Datenbank zurückgegeben.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
session_name	CHAR(256)	Gibt den Sitzungsnamen zurück.
target_type	CHAR(256)	Gibt den Zieltyp an (z.B. eine Datei).

Bemerkungen

Ein Ziel ist der Speicherort, an dem die Informationen der Trace-Ereignis-Sitzung protokolliert werden (z.B. eine Datei oder ein Datenbankserver-Meldungslog).

Privilegien

Sie müssen das `MANAGE ANY TRACE SESSION`-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Ereignisprotokollierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE TEMPORARY TRACE EVENT-Anweisung“ auf Seite 753
- „CREATE TEMPORARY TRACE EVENT SESSION-Anweisung“ auf Seite 755
- „ALTER TRACE EVENT SESSION-Anweisung“ auf Seite 538
- „DROP TRACE EVENT-Anweisung“ auf Seite 835
- „DROP TRACE EVENT SESSION-Anweisung“ auf Seite 836
- „NOTIFY TRACE EVENT-Anweisung“ auf Seite 963
- „sp_trace_events-Systemprozedur“ auf Seite 1407
- „sp_trace_event_fields-Systemprozedur“ auf Seite 1401
- „sp_trace_event_sessions-Systemprozedur“ auf Seite 1406
- „sp_trace_event_session_events-Systemprozedur“ auf Seite 1402
- „sp_trace_event_session_target_options-Systemprozedur“ auf Seite 1403
- „Dienstprogramm für die Verwaltung von ETD-Dateien (dbmanageetd)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Die folgende Anweisung gibt Informationen zu allen Trace-Ereignissitzungen in der Datenbank zurück:

```
SELECT * FROM sp_trace_event_session_targets( );
```

sp_trace_event_sessions-Systemprozedur

Gibt eine Liste der Trace-Ereignissitzungen zurück, die für die Datenbank definiert sind.

Syntax

```
sp_trace_event_sessions( [ session_name ] )
```

Parameter

- **session_name** Verwenden Sie diesen CHAR(256)-Parameter, um die Trace-Ereignissitzung anzugeben, zu der Sie Informationen abrufen möchten. Wenn kein Sitzungsname angegeben wird, werden Informationen zu allen Trace-Ereignissitzungen in der Datenbank zurückgegeben.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
session_name	CHAR(256)	Gibt den Namen der Sitzung zurück.
description	LONG VAR-CHAR	Gibt eine Beschreibung der Sitzung zurück.
started	BIT	Gibt 1 zurück, wenn die Sitzung gestartet ist, und andernfalls 0.
is_temporary	BIT	Gibt 1 zurück, wenn es sich um eine temporäre Trace-Sitzung handelt, und andernfalls 0.

Bemerkungen

Sie können Trace-Ereignissitzungen mithilfe der STATE-Klausel für die ALTER TRACE EVENT SESSION-Anweisung manuell starten und stoppen.

Privilegien

Sie müssen das MANAGE ANY TRACE SESSION-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Ereignisprotokollierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE TEMPORARY TRACE EVENT-Anweisung“ auf Seite 753
- „CREATE TEMPORARY TRACE EVENT SESSION-Anweisung“ auf Seite 755
- „ALTER TRACE EVENT SESSION-Anweisung“ auf Seite 538
- „DROP TRACE EVENT-Anweisung“ auf Seite 835
- „DROP TRACE EVENT SESSION-Anweisung“ auf Seite 836
- „NOTIFY TRACE EVENT-Anweisung“ auf Seite 963
- „sp_trace_events-Systemprozedur“ auf Seite 1407
- „sp_trace_event_fields-Systemprozedur“ auf Seite 1401
- „sp_trace_event_session_events-Systemprozedur“ auf Seite 1402
- „sp_trace_event_session_targets-Systemprozedur“ auf Seite 1405
- „sp_trace_event_session_target_options-Systemprozedur“ auf Seite 1403
- „Dienstprogramm für die Verwaltung von ETD-Dateien (dbmanageetd)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Die folgende Anweisung gibt eine Liste der Trace-Ereignissitzungen für die aktuelle Datenbank zurück:

```
SELECT * FROM sp_trace_event_sessions( );
```

sp_trace_events-Systemprozedur

Gibt Informationen zu den Trace-Ereignissen in der Datenbank zurück.

Syntax

```
sp_trace_events( [ event_name ] )
```

Argumente

- **event_name** Verwenden Sie diesen optionalen CHAR(256)-Parameter, um den Trace-Ereignisnamen anzugeben. Der Standardwert ist NULL.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
event_name	CHAR(256)	Gibt den Namen des Trace-Ereignisses zurück. Namen von System-Trace-Ereignissen haben das Präfix SYS_ .
description	LONG VARCHAR	Gibt eine Beschreibung dessen zurück, was das Trace-Ereignis erfasst.
severity	TINYINT	Gibt den Schweregrad der Trace-Ereignisses zurück.
is_system	BIT	Gibt bei System-Trace-Ereignissen 1 zurück und andernfalls 0.

Spaltenname	Datentyp	Beschreibung
is_temporary	BIT	Gibt bei temporären Trace-Ereignissen 1 zurück und andernfalls 0.

Bemerkungen

Diese Prozedur gibt Informationen zu allen benutzerdefinierten und systemdefinierten Trace-Ereignissen in der Datenbank zurück. Sie gibt die Details für das angegebene Trace-Ereignis zurück bzw. für alle Trace-Ereignisse, wenn *event_name* nicht angegeben wird oder NULL ist.

Ebene	Schweregrad des Wertebereichs
ALWAYS	0
CRITICAL	1-50
ERROR	51-100
WARNING	101-150
INFORMATION	151-200
DEBUG	201-255

Privilegien

Sie müssen das **MANAGE ANY TRACE SESSION**-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „Ereignisprotokollierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE TEMPORARY TRACE EVENT-Anweisung“ auf Seite 753
- „CREATE TEMPORARY TRACE EVENT SESSION-Anweisung“ auf Seite 755
- „ALTER TRACE EVENT SESSION-Anweisung“ auf Seite 538
- „DROP TRACE EVENT-Anweisung“ auf Seite 835
- „DROP TRACE EVENT SESSION-Anweisung“ auf Seite 836
- „NOTIFY TRACE EVENT-Anweisung“ auf Seite 963
- „sp_trace_event_fields-Systemprozedur“ auf Seite 1401
- „sp_trace_event_sessions-Systemprozedur“ auf Seite 1406
- „sp_trace_event_session_events-Systemprozedur“ auf Seite 1402
- „sp_trace_event_session_targets-Systemprozedur“ auf Seite 1405
- „sp_trace_event_session_target_options-Systemprozedur“ auf Seite 1403
- „Dienstprogramm für die Verwaltung von ETD-Dateien (dbmanageetd)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Die folgende Anweisung gibt eine Liste der benutzerdefinierten Trace-Ereignisse in der Datenbank zurück:

```
SELECT * FROM sp_trace_events( )  
WHERE is_system = 0;
```

sp_tsql_environment-Systemprozedur

Setzt Verbindungsoptionen, wenn sich Benutzer von jConnect- oder Open Client-Anwendungen verbinden

Syntax

sp_tsql_environment()

Bemerkungen

Die sp_login_environment-Prozedur ist die Standardprozedur, die von der Datenbankoption login_procedure angegeben wird. Bei jeder neuen Verbindung wird die von login_procedure angegebene Prozedur aufgerufen. Wenn die Verbindung das TDS-Kommunikationsprotokoll verwendet (d.h., wenn es sich um eine Open Client- oder jConnect-Verbindung handelt), ruft sp_login_environment wiederum sp_tsql_environment auf.

Diese Prozedur setzt Datenbankoptionen so, dass sie mit dem Standardverhalten von Adaptive Server Enterprise kompatibel sind.

Wenn Sie das Standardverhalten ändern möchten, erstellen Sie neue Prozeduren und ändern Sie die login_procedure-Option so, dass sie auf diese neuen Prozeduren zeigt.

Die folgende Liste enthält die Optionen, die durch die sp_tsql_environment-Prozedur festgelegt werden:

```
if db_property( 'IQStore' ) = 'Off' then  
    -- SQL Anywhere datastore  
    SET TEMPORARY OPTION close_on_endtrans='OFF';  
end if;  
SET TEMPORARY OPTION ansinull='OFF';  
SET TEMPORARY OPTION tsql_variables='ON';  
SET TEMPORARY OPTION ansi_blanks='ON';  
SET TEMPORARY OPTION chained='OFF';  
SET TEMPORARY OPTION quoted_identifier='OFF';  
SET TEMPORARY OPTION allow_nulls_by_default='OFF';  
SET TEMPORARY OPTION on_tsql_error='CONTINUE';  
SET TEMPORARY OPTION isolation_level='1';  
SET TEMPORARY OPTION date_format='YYYY-MM-DD';  
SET TEMPORARY OPTION timestamp_format='YYYY-MM-DD HH:NN:SS.SSS';  
SET TEMPORARY OPTION time_format='HH:NN:SS.SSS';  
SET TEMPORARY OPTION date_order='MDY';  
SET TEMPORARY OPTION escape_character='OFF';
```

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- „sp_login_environment-Systemprozedur“ auf Seite 1376
- „login_procedure-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Im folgenden Beispiel wird die sp_tsql_environment-Prozedur aufgerufen:

```
CALL sp_tsql_environment();
```

sp_use_secure_feature_key-Systemprozedur

Aktivieren Sie die im angegebenen Schlüssel enthaltenen gesicherten Funktionen für die aktuelle Verbindung.

Syntax

```
sp_use_secure_feature_key(  
    name  
, auth_key  
)
```

Argumente

- **name** Der VARCHAR(128)-Name des zu aktivierenden Schlüssels für gesicherte Funktionen.
- **auth_key** Der CHAR(128)-Autorisierungsschlüssel (mit Berücksichtigung von Groß- und Kleinschreibung) des zu aktivierenden Schlüssels für gesicherte Funktionen. Der Autorisierungsschlüssel muss mindestens sechs Zeichen lang sein.

Bemerkungen

Diese Prozedur aktiviert die gesicherten Funktionen, die im angegebenen Schlüssel für gesicherte Funktionen enthalten sind, nur für die aktuelle Verbindung.

Privilegien

Keine.

Nebenwirkungen

Keine

Siehe auch

- „Schlüssel für gesicherte Funktionen erstellen“ [*SQL Anywhere Server - Datenbankadministration*]
- „sp_alter_secure_feature_key-Systemprozedur“ auf Seite 1354
- „sp_create_secure_feature_key-Systemprozedur“ auf Seite 1360
- „sp_drop_secure_feature_key-Systemprozedur“ auf Seite 1366
- „sp_list_secure_feature_keys-Systemprozedur“ auf Seite 1375

Beispiel

Damit die folgenden Beispiele funktionieren, muss der Server mit der folgenden Option gestartet werden:
`-sk securefkey.`

In diesem Beispiel wird die Gruppe gesicherter Funktionen namens SYSTEM, die MANAGE_KEYS enthält, für die aktuelle Verbindung aktiviert.

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );
```

In diesem Beispiel wird die Gruppe gesicherter Funktionen namens SYSTEM aktiviert, die MANAGE_KEYS enthält. Sie erstellt einen neuen Schlüssel für gesicherte Funktionen namens MYSET mit dem Autorisierungsschlüssel securemyset, der Groß- und Kleinschreibung berücksichtigt, und verwendet anschließend den neuen Schlüssel für gesicherte Funktionen:

```
CALL sp_use_secure_feature_key( 'system', 'securefkey' );  
CALL sp_create_secure_feature_key( 'myset', 'securemyset', 'local,remote' );  
CALL sp_use_secure_feature_key( 'myset', 'securemyset' );
```

Nach dem Wechsel zum neuen Schlüssel für gesicherte Funktionen verwendet die aktuelle Verbindung nicht mehr die gesicherte Funktion SYSTEM und hat daher keinen Zugriff auf MANAGE_KEYS.

st_geometry_dump-Systemprozedur

Zerlegt eine Geometrie in ihre Komponenten der untersten Ebene.

Syntax

```
st_geometry_dump(  
  geometry  
  [, "options"]  
)
```

Argumente

- **geometry** Der ST_Geometry-Wert der zu zerlegenden Geometrie.
- **"options"** Eine optionale VARCHAR(255)-Zeichenfolge aus durch Semikola getrennten Parametern und Werten, die Sie verwenden können, um die Ausgabe der Prozedur zu konfigurieren. Der Standardwert ist NULL.

Die folgende Tabelle enthält eine Liste der Parameter, die Sie angeben können:

Parameter	Standardwert	Zulässige Werte	Beschreibung
Format	Original	Original, Internal oder Mixed	Das Format, in dem die Geometrie zurückgegeben werden soll. Wenn Sie "Original" angeben, wird die Geometrie in ihrem ursprünglichen Format zurückgegeben. Wenn Sie "Internal" angeben, wird die Geometrie in ihrem normalisierten Format zurückgegeben. Wenn Sie "Mixed" angeben, werden alle verfügbaren gespeicherten Formate zurückgegeben, eine Zeile pro Format.
ExpandPoints	Yes	Yes, No	Standardmäßig gibt die st_geometry_dump-Systemprozedur beim Zerlegen einer Geometrie mit Punkten (wie z.B. ST_LineString oder ST_MultiPoint) die Punkte in separaten Zeilen aus. Setzen Sie ExpandPoints auf "No", wenn Sie nicht möchten, dass diese zusätzlichen Zeilen generiert werden.
MaxDepth	-1	-1, eine beliebige Zahl größer oder gleich Null	Standardmäßig fährt die st_geometry_dump-Systemprozedur mit dem Zerlegen einer Objekthierarchie fort, bis sie die Blattobjekte erreicht. Der MaxDepth-Parameter kann gesetzt werden, um die Anzahl der Ebenen in der Hierarchie zu begrenzen, auf denen die Geometrie zerlegt wird. Bei einem Wert von 0 wird nur die Stammgeometrie zurückgegeben. Bei einem Wert von 1 werden die Geometrie und die ihr direkt untergeordneten Objekte zurückgegeben usw.
SetGeom	Yes	Yes, No	Die st_geometry_dump-Systemprozedur gibt eine Spalte zurück mit der ST_Geometry, die einem Objekt in der ursprünglichen Hierarchie zugeordnet ist. Wenn diese Spalte nicht benötigt wird, kann der Parameter SetGeom auf "No" gesetzt werden, um die Ausführungszeit und die Ausgabegröße der Prozedur zu verringern.

Parameter	Standardwert	Zulässige Werte	Beschreibung
Validate	Basic	None, Basic, Full	Standardmäßig wendet die st_geometry_dump-Systemprozedur die Validierungsregeln an, die der Datenbankserver beim Laden von Geometrien verwendet, und setzt die Spalte "Valid" der Ergebnismenge auf 1, wenn das Objekt in der Zeile diesen Regeln entspricht. Der Validate-Parameter kann auf "None" gesetzt werden, um diese Prüfung zu deaktivieren, oder auf "Full", um die zusätzlichen Prüfungen durch die ST_IsValid-Methode anzuwenden. Eine vollständige Überprüfung nimmt mehr Zeit in Anspruch.

Ergebnismenge

In der folgenden Tabelle werden die von der st_geometry_dump-Prozedur zurückgegebenen Ergebnisse beschrieben:

Spalte	Datentyp	Beschreibung
id	UNSIGNED BIGINT	Eine eindeutige ID für diese Zeile in den Ergebnissen.
parent_id	UNSIGNED BIGINT	Die ID des diesem Objekt direkt übergeordneten Objekts.
depth	INTEGER	Die Tiefe vom Stammobjekt bis zu dem dieser Zeile zugeordneten Objekt.
format	VARCHAR(128)	Gibt an, ob es sich bei der Geometrie um die ursprüngliche Darstellung ("Original") oder die normalisierte Darstellung ("Internal") handelt.
valid	BIT	Gibt an, ob die Geometrie entsprechend der Validate-Option festgelegten Prüfungsstufe gültig ist (1).
geom_type	VARCHAR(128)	Der Geometrietyp, wie von ST_GeometryType zurückgegeben.
geom	ST_Geometry	Die Geometriespezifikation. Wenn der SetGeom-Parameter auf "No" gesetzt ist, wird die Geometriespezifikation nicht in der Ergebnismenge zurückgegeben.
xmin	DOUBLE	Der minimale x-Wert für die Geometrie.
xmax	DOUBLE	Der maximale x-Wert für die Geometrie.

Spalte	Datentyp	Beschreibung
ymin	DOUBLE	Der minimale y-Wert für die Geometrie.
ymax	DOUBLE	Der maximale y-Wert für die Geometrie.
zmin	DOUBLE	Der minimale z-Wert für die Geometrie.
zmax	DOUBLE	Die maximale z-Wert für die Geometrie.
mmin	DOUBLE	Der minimale m-Wert für die Geometrie.
mmax	DOUBLE	Die maximale m-Wert für die Geometrie.
details	LONG VAR-CHAR	Weitere Informationen zur Geometrie, einschließlich zusätzlicher Angaben dazu, warum das Objekt nicht gültig ist.

Bemerkungen

Die `st_geometry_dump`-Systemprozedur zerlegt eine Geometriehierarchie in eine Ergebnismenge mit einer Zeile für jedes der Objekte in der Hierarchie (einschließlich des Stammobjekts). Jede Geometrie in der Hierarchie kann validiert werden, um festzustellen, ob sie gültig ist und ggf. warum sie ungültig ist.

Einige Funktionen der `st_geometry_dump`-Systemprozedur können mithilfe von typspezifischen Methoden wie `ST_GeometryN` oder `ST_PointN` zugeordnet werden.

Die `st_geometry_dump`-Systemprozedur kann verwendet werden, um ungültige Geometrien zu korrigieren.

Privilegien

Keine

Nebenwirkungen

Keine

Siehe auch

- [ST_IsValid-Methode für den ST_Geometry-Datentyp \[SQL Anywhere Server - Unterstützung für räumliche Daten\]](#)
- „CREATE SPATIAL REFERENCE SYSTEM-Anweisung“ auf Seite 719
- „st_geometry_on_invalid-Option“ [SQL Anywhere Server - Datenbankadministration]
- STORAGE FORMAT-Klausel, CREATE SPATIAL REFERENCE SYSTEM-Anweisung auf Seite 725

Beispiel

Im folgenden Beispiel wird das Polygon, 'Polygon ((0 0, 3 0, 3 3, 0 3, 0 0))', in seine Komponentengeometrien zerlegt:

```
SELECT * FROM st_geometry_dump( 'Polygon ((0 0, 3 0, 3 3, 0 3, 0 0))',
  'SetGeom=No' );
```

id	parent_id	depth	format	valid	geom_type	geom	xmin	xmax	ymin	ymax	...
1	1	0	Internal	1	ST_Polygon	Polygon ((0 0, 3 0, 3 3, 0 3, 0 0))	0	3	0	3	...
2	1	1	Internal	1	ST_LineString	LineString (0 0, 3 0, 3 3, 0 3, 0 0)	0	3	0	3	...
3	2	2	Internal	1	ST_Point	Point (0 0)	0	0	0	0	...
4	2	2	Internal	1	ST_Point	Point (3 0)	3	3	0	0	...
5	2	2	Internal	1	ST_Point	Point (3 3)	3	3	3	3	...
6	2	2	Internal	1	ST_Point	Point (0 3)	0	0	3	3	...
7	2	2	Internal	1	ST_Point	Point (0 0)	0	0	0	0	...

Das folgende Beispiel zeigt, wie die st_geometry_dump-Systemprozedur verwendet werden kann, um die ungültigen Punkte in einer Geometrie zu finden. In diesem Beispiel enthält die Linienfolge einen Punkt mit Länge 1200. Deshalb werden sowohl der Punkt als auch die Linienfolge in den Ergebnissen als ungültig gemeldet (valid=0).

```

SET TEMPORARY OPTION st_geometry_on_invalid='Ignore';
CREATE OR REPLACE VARIABLE @geo ST_Geometry;
SET @geo = new ST_LineString( 'LineString(1200 2, 80 10)', 4326 );

SELECT * FROM st_geometry_dump( @geo, 'SetGeom=No' );

```

id	parent_id	depth	format	valid	geom_type	geom	xmin	xmax	ymin	ymax	...	details
1	1	0	Original	0	ST_LineString	(NULL)	80	1,200	2	10	...	Value 1200.000000 out of range for coordinate longitude (SRS allows -180.000000 to 180.000000).
2	1	1	Original	0	ST_LineString	(NULL)	1,200	1,200	2	2	...	Value 1200.000000 out of range for coordinate longitude (SRS allows -180.000000 to 180.000000).
3	1	1	Original	1	ST_Point	(NULL)	80	80	10	10	...	

Wenn ungültige Daten identifiziert wurden, kann die `st_geometry_dump`-Systemprozedur mit anderen als räumlichen Methoden verwendet werden, um die ungültigen Elemente zu korrigieren und eine gültige Geometrie zusammenzustellen. Das folgende Beispiel zeigt, wie ein ungültiger Punkt mit Länge 1200 in Länge 120.0 korrigiert werden kann:

```
SET TEMPORARY OPTION st_geometry_on_invalid='Ignore';
CREATE OR REPLACE VARIABLE @geo ST_Geometry;
SET @geo = new ST_LineString( 'LineString(1200 2, 80 10)', 4326 );

SELECT ST_LineString::ST_LineStringAggr(
    new ST_Point( IF xmax = 1200 then 120.0 ELSE xmax ENDIF,
                  ymax, 4326 ) ORDER BY id )
FROM   st_geometry_dump( @geo )
WHERE  geom_type='ST_Point';
```

st_geometry_load_shapefile-Systemprozedur

Erstellt eine Tabelle und lädt eine ESRI-Formdatei in diese.

Syntax

```
st_geometry_load_shapefile(
  shp_filename
, srid
, table_name
[, table_owner
```

```
[, shp_encoding ] ]  
)
```

Argumente

- **shp_filename** Verwenden Sie diesen VARCHAR(512)-Parameter, um den Speicherort der ESRI-Formdatei anzugeben. Der Dateiname muss die Erweiterung *.shp* haben und ihm müssen eine *.shx*- und eine *.dbf*-Datei zugeordnet sein, die denselben Basisnamen haben und sich in demselben Verzeichnis befinden. Der Pfad ist relativ zum Datenbankserver, nicht zur Clientanwendung.
- **srld** Verwenden Sie diesen INTEGER-Parameter, um die SRID anzugeben, die den Daten aus der Formdatei zugeordnet werden soll. Damit die Geometrien aussagekräftig werden, muss die SRID, die Sie angeben, für die Geometrien geeignet sein, auch wenn sie nicht mit der in der Formdatei verwendeten SRID identisch ist.
- **table_name** Verwenden Sie diese VARCHAR(128)-Spalte, um den Namen der Tabelle anzugeben, die zum Speichern der Formdatei-Daten erstellt werden soll.
- **table_owner** Verwenden Sie diese optionale VARCHAR(128)-Spalte, um den Eigentümer für die neue Tabelle anzugeben, die erstellt wird. Der Standardeigentümer ist der aktuelle Benutzer.
- **shp_encoding** Verwenden Sie diesen optionalen VARCHAR(50)-Parameter, um die Kodierung anzugeben, die beim Lesen der Formdatei verwendet werden soll. Die Standardkodierung ist ISO-8859-1.

Bemerkungen

Die `st_geometry_load_shapefile`-Systemprozedur erstellt zuerst die Tabelle "Tabelleneigentümer"."Tabellenname" mit den Spalteninformationen (Namen und Typen), die in der ESRI-Formdatei gefunden wurden. Anschließend lädt `st_geometry_load_shapefile` die Daten aus der Formdatei in die neue Tabelle.

Diese Prozeduren benutzen die `sa_describe_shapefile`-Systemprozedur und die `LOAD TABLE...FORMAT SHAPEFILE`-Anweisung.

Privilegien

Wenn die Option `-gl` auf `NONE` gesetzt ist, schlägt das Laden der Daten fehl.

Wenn die Option `-gl` auf `DBA` gesetzt ist, gilt Folgendes:

- Wenn Sie *table_owner* sind, müssen Sie das `CREATE TABLE`-Systemprivileg, das `CREATE ANY TABLE`-Systemprivileg oder das `CREATE ANY OBJECT`-Systemprivileg haben sowie das `ALTER ANY TABLE`-Systemprivileg, das `ALTER ANY OBJECT`-Systemprivileg oder das `LOAD ANY TABLE`-Systemprivileg.
- Wenn Sie nicht *table_owner* sind, müssen Sie das `CREATE ANY TABLE`-Systemprivileg oder das `CREATE ANY OBJECT`-Systemprivileg haben sowie das `ALTER ANY TABLE`-Systemprivileg, das `ALTER ANY OBJECT`-Systemprivileg oder das `LOAD ANY TABLE`-Systemprivileg.

Wenn die Option `-gl` auf `ALL` gesetzt ist, gilt Folgendes:

- Wenn Sie *table_owner* sind, müssen Sie das CREATE TABLE-Systemprivileg, das CREATE ANY TABLE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben.
- Wenn Sie nicht *table_owner* sind, müssen Sie das CREATE ANY TABLE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben sowie das ALTER ANY TABLE-Systemprivileg, das ALTER ANY OBJECT-Systemprivileg oder das LOAD ANY TABLE-Systemprivileg.

Nebenwirkungen

Wenn das Laden fehlschlägt, bleibt die erstellte Tabelle bestehen.

Siehe auch

- „sa_describe_shapefile-Systemprozedur“ auf Seite 1209
- „Lektion 3: Die Daten der ESRI-Formdatei einlesen“ [*SQL Anywhere Server - Unterstützung für räumliche Daten*]
- „LOAD TABLE-Anweisung“ auf Seite 931
- „Unterstützung von ESRI-Formdateien“ [*SQL Anywhere Server - Unterstützung für räumliche Daten*]

Beispiel

Die folgende Anweisung erstellt die Tabelle *esri_load* und lädt sie mit den Daten aus einer fiktiven Formdatei, *c:\esri\shapefile.shp*, wobei die Daten der SRID 1000004326 zugeordnet werden:

```
CALL st_geometry_load_shapefile( 'c:\\esri\\tgr36069trt00.shp', 1000004326,  
'esri_load' );
```

xp_cmdshell-Systemprozedur

Führt einen Betriebssystem-Befehl von einer Prozedur aus aus

Syntax

```
xp_cmdshell(  
  command  
  [, redir_output | 'no_output' ]  
)
```

Argumente

- **command** Verwenden Sie diesen VARCHAR(8000)-Parameter, um einen Systembefehl anzugeben. Der Standardwert ist NULL.
- **redir_output** Verwenden Sie diesen optionalen CHAR(254)-Parameter, um anzugeben, ob die Ausgabe in einem Befehlsfenster angezeigt werden soll. Das Standardverhalten ist, die Ausgabe in einem Befehlsfenster anzuzeigen. Wenn Sie "no_output" angeben, wird die Ausgabe nicht in einem Befehlsfenster angezeigt. Der Standardwert ist ''.

Rückgabe

Diese Funktion gibt einen INTEGER-Exit-Code zurück.

Bemerkungen

xp_cmdshell führt einen Systembefehl aus und gibt dann die Kontrolle an die aufrufende Umgebung zurück. Der von xp_cmdshell zurückgegebene Wert ist der Beendigungscode aus dem ausgeführten Shell-Prozess. Der Rückgabewert ist 2, wenn ein Fehler beim Start des untergeordneten Prozesses auftritt.

Der zweite Parameter betrifft nur Befehlszeilenanwendungen in Windows-Betriebssystemen. In UNIX wird ungeachtet der Einstellung für den zweiten Parameter kein Befehlsfenster angezeigt.

Für Windows Mobile sind alle ausgeführten Befehle ungeachtet der Einstellung des zweiten Parameters im Meldungslog des Datenbankservers ersichtlich. Die Konsolen-Shell `\\windows\cmd.exe` ist erforderlich, um die Prozedur auszuführen.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Siehe auch

- „CALL-Anweisung“ auf Seite 564
- „Systemprozeduren für Verzeichnisse und Dateien“ auf Seite 1164
- „Allgemeine Sicherheitstipps“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiele

Die folgende Anweisung listet die Dateien im aktuellen Verzeichnis in der Datei `c:\temp.txt` auf:

```
CALL xp_cmdshell( 'dir > c:\\temp.txt' );
```

Die folgende Anweisung führt den gleichen Vorgang aus, allerdings ohne Anzeige eines **Befehlsfensters**.

```
CALL xp_cmdshell( 'dir > c:\\temp.txt', 'no_output' );
```

xp_get_mail_error_code-Systemprozedur

Gibt Informationen zum letzten SMTP- oder MAPI-Fehler zurück.

Syntax

```
xp_get_mail_error_code( )
```

Rückgabe

Diese Funktion gibt einen INTEGER-Wert zurück, die den SMTP- oder MAPI-Fehlercode darstellt.

Bemerkungen

Wenn der Rückgabewert einer E-Mail-Prozedur (xp_startmail, xp_startsmtp, xp_sendmail, xp_stopmail und xp_stopsmt) gleich -1 ist, können Sie mit dieser Funktion den SMTP- oder MAPI-Fehlercode abrufen.

Wenn der Rückgabewert einer E-Mail-Prozedur gleich 5, 6 oder 7 ist, können Sie mit dieser Funktion den errno-Wert des letzten Socketfehlers abrufen.

Bei Verwendung von MAPI ist der zurückgegebene Wert der Rückgabecode der MAPI-Funktion. Bei Verwendung von SMTP ist der zurückgegebene Wert entweder ein Fehlercode (und `xp_get_mail_error_text` gibt den SMTP-Fehlertext zurück) oder ein `errno`-Wert (und `xp_get_mail_error_text` gibt eine leere Zeichenfolge zurück).

Privilegien

Sie müssen das SEND EMAIL-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- [Rückgabecodes für MAPI- und SMTP-Systemprozeduren auf Seite 1163](#)
- [„xp_get_mail_error_text-Systemprozedur“ auf Seite 1420](#)
- [„xp_startmail-Systemprozedur“ auf Seite 1431](#)
- [„xp_startsmtp-Systemprozedur“ auf Seite 1431](#)
- [„xp_sendmail-Systemprozedur“ auf Seite 1426](#)
- [„xp_stopmail-Systemprozedur“ auf Seite 1434](#)
- [„xp_stopsmtplib-Systemprozedur“ auf Seite 1434](#)

Beispiel

In diesem Beispiel wird der letzte SMTP- oder MAPI-Fehlercode zurückgegeben.

```
SELECT xp_get_mail_error_code( )
```

In diesem Beispiel wird SMTP verwendet, um das Senden einer reinen Textnachricht zu initiieren.

```
BEGIN
  DECLARE err_smtp INTEGER;
  DECLARE err_code INTEGER;
  DECLARE err_msg LONG VARCHAR;

  SELECT xp_startsmtp( 'doe@sample.com', 'corporatemail.sample.com' ) INTO
err_smtp;
  SELECT xp_get_mail_error_code( ), xp_get_mail_error_text( ) INTO
err_code, err_msg;
  SELECT err_smtp, err_code, err_msg;
END;
```

xp_get_mail_error_text-Systemprozedur

Gibt den Text der letzten Fehler- oder Statusmeldung für SMTP zurück.

Syntax

```
xp_get_mail_error_text( )
```

Rückgabewert

Diese Funktion gibt einen LONG VARCHAR-Wert zurück, der den Fehler- oder Statusmeldungstext für SMTP oder MAPI darstellt. Wenn kein Fehlertext verfügbar ist, wird eine leere Zeichenfolge oder NULL zurückgegeben.

Bemerkungen

Verwenden Sie diese Funktion, um den Fehler- oder Statusmeldungstext für eine der E-Mail-Prozeduren (xp_startmail, xp_startsmtp, xp_sendmail, xp_stopmail und xp_stopsmtp) abzurufen.

Privilegien

Sie müssen das SEND EMAIL-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- [Rückgabecodes für MAPI- und SMTP-Systemprozeduren auf Seite 1163](#)
- [„xp_get_mail_error_code-Systemprozedur“ auf Seite 1419](#)
- [„xp_startmail-Systemprozedur“ auf Seite 1431](#)
- [„xp_startsmtp-Systemprozedur“ auf Seite 1431](#)
- [„xp_sendmail-Systemprozedur“ auf Seite 1426](#)
- [„xp_stopmail-Systemprozedur“ auf Seite 1434](#)
- [„xp_stopsmtp-Systemprozedur“ auf Seite 1434](#)

Beispiel

In diesem Beispiel wird der Text der letzten SMTP- oder MAPI-Meldung zurückgegeben.

```
SELECT xp_get_mail_error_text( )
```

In diesem Beispiel wird SMTP verwendet, um das Senden einer reinen Textnachricht zu initiieren.

```
BEGIN
    DECLARE err_smtp INTEGER;
    DECLARE err_code INTEGER;
    DECLARE err_msg LONG VARCHAR;

    SELECT xp_startsmtp( 'doe@sample.com', 'corporatemail.sample.com' ) INTO
err_smtp;
    SELECT xp_get_mail_error_code( ), xp_get_mail_error_text( ) INTO
err_code, err_msg;
    SELECT err_smtp, err_code, err_msg;
END;
```

xp_getenv-Systemprozedur

Gibt den Wert einer Umgebungsvariablen zurück.

Syntax

```
xp_getenv( environment_variable )
```

Argumente

- **environment_variable** Verwenden Sie diesen VARCHAR(8000)-Parameter, um die Umgebungsvariable anzugeben. Dieser Parameter berücksichtigt die Groß- und Kleinschreibung unter Windows-Betriebssystemen nicht und berücksichtigt sie unter allen anderen Betriebssystemen, unabhängig von der Berücksichtigung der Groß- und Kleinschreibung in der Datenbank. Der Standardwert ist NULL.

Rückgabe

Diese Funktion gibt einen LONG BINARY-Wert zurück.

Bemerkungen

Wenn die angegebene Umgebungsvariable NULL oder nicht gesetzt ist, wird NULL zurückgegeben.

Privilegien

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Nebenwirkungen

Keine

Siehe auch

- „SQL Anywhere-Umgebungsvariablen“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Im folgenden Beispiel wird die xp_getenv-Systemprozedur verwendet, um den Wert der Umgebungsvariablen PATH zurückzugeben.

```
SELECT CAST(xp_getenv('PATH') AS LONG VARCHAR);
```

Im folgenden Beispiel werden die Systemprozeduren xp_getenv und sa_split_list verwendet, um den Wert des Windows-Umgebungsvariablen PATH als Liste zurückzugeben. Verwenden Sie unter Unix-Betriebssystemen ":" als Trennzeichen.

```
CALL sa_split_list(CAST(xp_getenv('PATH') AS LONG VARCHAR), ';');
```

Im folgenden Beispiel wird die Umgebungsvariable NONEXISTENT verwendet, von der angenommen wird, dass sie nicht vorhanden ist. Daher sollte die Abfrage NULL zurückgeben.

```
SELECT xp_getenv( 'NONEXISTENT' );
```

xp_msver-Systemprozedur

Ruft Informationen über Version und Namen des Datenbankservers ab

Syntax

xp_msver(the_option)

Argumente

- **the_option** Verwenden Sie diesen CHAR(254)-Parameter, um eine Zeichenfolge anzugeben. Die Zeichenfolge muss eine der folgenden Zeichenfolgen innerhalb von Zeichenfolge-Begrenzern sein.

Argument	Beschreibung
ProductName	Gibt den Name des Produkts (SQL Anywhere) zurück. ProductName ist der Standardwert des Arguments.
ProductVersion	Gibt die Versionsnummer zurück, gefolgt von der Build-Nummer. Das Format ist Folgendes: <code>16.0.0.1403</code>
CompanyName	Gibt den Namen des Unternehmens zurück.
FileDescription	Gibt den Namen des Produkts zurück, gefolgt vom Namen des Betriebssystems
LegalCopyright	Gibt eine Copyright-Zeichenfolge für die Software zurück
LegalTrademarks	Gibt die Markenzeicheninformationen für die Software zurück.

Rückgabe

Diese Funktion gibt einen CHAR(254)-Wert zurück.

Bemerkungen

Diese Funktion gibt Produkt-, Unternehmens-, Versions- und andere Informationen zurück.

Privilegien

Keine

Siehe auch

- „Systemfunktionen“ auf Seite 165

Beispiel

Die folgende Anweisung fordert die Versions- und Betriebssystem-Beschreibung an:

```
SELECT xp_msver( 'ProductVersion') Version,
       xp_msver( 'FileDescription' ) Description;
```

Es folgt eine Beispielausgabe. Der Wert für Version ist auf Ihrem System wahrscheinlich unterschiedlich.

Version	Description
16.0.0.1403	SQL Anywhere Windows XP

xp_read_file-Systemprozedur

Liest die Datei und gibt den Inhalt der Datei als LONG BINARY-Variable zurück.

Syntax

```
xp_read_file(  
    filename  
    [, lazy]  
)
```

Argumente

- **filename** Verwenden Sie diesen LONG VARCHAR-Parameter, um den Namen der Datei anzugeben, deren Inhalt zurückgegeben werden soll.
- **lazy** Wenn Sie diesen optionalen INTEGER-Parameter angeben und sein Wert nicht 0 ist, wird der Inhalt der Datei erst auf Anforderung gelesen. Lesevorgänge treten nur auf, wenn auf den LONG BINARY-Wert zugegriffen wird, und nur für den angeforderten Teil der Datei. Der Standardwert ist 0 oder "non-lazy".

Rückgabe

Diese Funktion gibt den Inhalt der angegebenen Datei als LONG BINARY-Wert zurück. Wenn die Datei nicht vorhanden ist oder nicht gelesen werden kann, wird NULL zurückgegeben.

Bemerkungen

Der Dateiname in *filename* ist relativ zum Startverzeichnis des Datenbankservers.

Die Funktion kann beim Einfügen ganzer, in Dateien gespeicherter Dokumente oder Bilder in Tabellen hilfreich sein. Falls die Datei nicht gelesen werden kann, gibt die Funktion NULL zurück.

Wenn die Datendatei in einem anderen Zeichensatz geschrieben ist, können Sie die CSCONVERT-Funktion verwenden, um sie zu konvertieren.

Sie können die CSCONVERT-Funktion auch verwenden, um die für die xp_read_file-Systemprozedur erforderliche Zeichensatzkonvertierung durchzuführen.

Privilegien

Sie müssen das READ FILE-Systemprivileg haben.

Beispiel

Die folgende Anweisung fügt ein Bild in eine Spalte namens Photo in der Tabelle "Products" ein.

```
UPDATE Products  
SET Photo=xp_read_file( 'c:\\sqlany\\scripts\\adata\\HoodedSweatshirt.jpg' )  
WHERE Products.ID=600;
```

Die folgende Anweisung liest eine Textdatei und zeigt jede Zeile mit Zeilennummer an.

```
SELECT * FROM sa_split_list( CAST(xp_read_file('\\Windows\\win.ini') AS LONG  
VARCHAR), 0x0a);
```

Siehe auch

- „CSCONVERT-Funktion [Zeichenfolge]“ auf Seite 211
- „xp_write_file-Systemprozedur“ auf Seite 1435
- „Systemprozeduren für Verzeichnisse und Dateien“ auf Seite 1164
- „CALL-Anweisung“ auf Seite 564
- Verwenden von OPENXML mit xp_read_file [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Einfügen von Dokumenten und Grafiken“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

xp_scanf-Systemprozedur

Extrahiert Teilzeichenfolgen aus einer Eingabezeichenfolge und einer Formatzeichenfolge

Syntax

```
xp_scanf(
    input_buffer
    , format
)
```

Argumente

- **input_buffer** Verwenden Sie diesen CHAR(254)-Parameter, um die Eingabezeichenfolge anzugeben.
- **format** Verwenden Sie diesen CHAR(254)-Parameter, um das Format der Eingabezeichenfolge anzugeben, mit Platzhaltern (%s) für jedes *parm*-Argument. Das *format*-Argument kann bis zu 50 Platzhalter enthalten und die Anzahl der Platzhalter muss mit der Anzahl der *parm*-Argumente übereinstimmen.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
parm	CHAR(254)	Gibt aus <i>input_buffer</i> extrahierte Teilzeichenfolgen im angegebenen <i>format</i> zurück. Es können bis zu 50 dieser Parameter verwendet werden.

Privilegien

Keine

Siehe auch

- „CALL-Anweisung“ auf Seite 564

Beispiel

Die folgenden Anweisungen extrahieren die Teilzeichenfolgen "Hello" und "World!" aus dem Eingabepuffer "Hello World!" und setzen sie in die Variablen "string1" bzw. "string2", um sie anschließend auszuwählen:

```
CREATE VARIABLE string1 CHAR( 254 );
CREATE VARIABLE string2 CHAR( 254 );
CALL xp_scanf( 'Hello World!', '%s %s', string1, string2 );
SELECT string1, string2;
```

xp_sendmail-Systemprozedur

Sendet eine E-Mail-Nachricht

Syntax

```
xp_sendmail(
  recipient = mail-address
  [, subject = subject ]
  [, cc_recipient = mail-address ]
  [, bcc_recipient = mail-address ]
  [, query = sql-query ]
  [, "message" = message-body ]
  [, attachname = attach-name ]
  [, attach_result = attach-result ]
  [, echo_error = echo-error ]
  [, include_file = filename ]
  [, no_column_header = no-column-header ]
  [, no_output = no-output ]
  [, width = width ]
  [, separator = separator-char ]
  [, dbuser = user-name ]
  [, dbname = db-name ]
  [, type = type ]
  [, include_query = include-query ]
  [, content_type = content-type ]
)
```

Argumente

Einige Argumente liefern feste Werte und sind verfügbar, um Transact-SQL-Kompatibilität zu gewährleisten (s.u.).

- **recipient** Dieser LONG VARCHAR-Parameter gibt die Mail-Adresse des Empfängers an. Wenn Sie mehrere Empfänger angeben, müssen die E-Mail-Adressen durch Semikolons voneinander getrennt werden.
- **subject** Dieser LONG VARCHAR-Parameter gibt das Betreff-Feld der Nachricht an. Der Standardwert ist NULL.
- **cc_recipient** Dieser LONG VARCHAR-Parameter gibt die Mail-Adresse des Cc-Empfängers an. Wenn Sie mehrere Cc-Empfänger angeben, müssen die E-Mail-Adressen durch Semikolons voneinander getrennt werden. Der Standardwert ist NULL.
- **bcc_recipient** Dieser LONG VARCHAR-Parameter gibt die Mail-Adresse des Bcc-Empfängers an. Wenn Sie mehrere Bcc-Empfänger angeben, müssen die E-Mail-Adressen durch Semikolons voneinander getrennt werden. Der Standardwert ist NULL.

- **query** Dieser LONG VARCHAR-Parameter wird für die Kompatibilität mit Transact-SQL verfügbar gemacht. Er wird von SQL Anywhere nicht benutzt. Der Standardwert ist NULL.
- **"message"** Dieser LONG VARCHAR-Parameter gibt den Nachrichteninhalt an. Der Standardwert ist NULL. Der Parametername "message" muss in Anführungszeichen gesetzt werden, da "MESSAGE" ein reserviertes Wort ist.
- **attachname** Dieser LONG VARCHAR-Parameter wird für die Kompatibilität mit Transact-SQL verfügbar gemacht. Er wird von SQL Anywhere nicht benutzt. Der Standardwert ist NULL.
- **attach_result** Dieser INTEGER-Parameter wird aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt. Er wird von SQL Anywhere nicht benutzt. Standardwert ist "0".
- **echo_error** Dieser INTEGER-Parameter wird aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt. Er wird von SQL Anywhere nicht benutzt. Standardwert ist "1".
- **include_file** Dieser LONG VARCHAR-Parameter gibt die Anhangdatei an. Der Standardwert ist NULL.
- **no_column_header** Dieser INTEGER-Parameter wird aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt. Er wird von SQL Anywhere nicht benutzt. Standardwert ist "0".
- **no_output** Dieser INTEGER-Parameter wird aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt. Er wird von SQL Anywhere nicht benutzt. Standardwert ist "0".
- **width** Dieser INTEGER-Parameter wird aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt. Er wird von SQL Anywhere nicht benutzt. Der Standardwert ist 80.
- **separator** Dieser CHAR(1)-Parameter wird für die Kompatibilität mit Transact-SQL verfügbar gemacht. Er wird von SQL Anywhere nicht benutzt. Standardwert ist CHAR(9).
- **dbuser** Dieser LONG VARCHAR-Parameter wird für die Kompatibilität mit Transact-SQL verfügbar gemacht. Er wird von SQL Anywhere nicht benutzt. Standardwert ist "guest".
- **dbname** Dieser LONG VARCHAR-Parameter wird für die Kompatibilität mit Transact-SQL verfügbar gemacht. Er wird von SQL Anywhere nicht benutzt. Standardwert ist "master".
- **type** Dieser LONG VARCHAR-Parameter wird für die Kompatibilität mit Transact-SQL verfügbar gemacht. Er wird von SQL Anywhere nicht benutzt. Der Standardwert ist NULL.
- **include_query** Dieser INTEGER-Parameter wird aus Gründen der Kompatibilität mit Transact-SQL bereitgestellt. Er wird von SQL Anywhere nicht benutzt. Standardwert ist "0".
- **content_type** Dieser LONG VARCHAR-Parameter gibt den Inhaltstyp des Nachrichtenparameters "message" an (z.B. text/html, ASIS usw.). Der Standardwert ist NULL. Der Wert von content_type wird nicht validiert. Das Setzen eines ungültigen Inhaltstyps bewirkt, dass eine ungültige oder unverständliche E-Mail gesendet wird.

Wenn Sie Header manuell festlegen möchten, setzen Sie den content_type-Parameter auf ASIS. Wenn Sie dies tun, nimmt die xp_sendmail-Prozedur an, dass die Daten, die an den message-Parameter übergeben wurden, eine korrekt formatierte E-Mail mit Headern ist und fügt daher keine weiteren

Header hinzu. Wenn Sie ASIS angeben, müssen Sie alle Header im Nachrichtenparameter manuell einstellen, auch Header, die normalerweise durch die Übergabe von Daten an die anderen Parameter ausgefüllt werden.

Rückgabe

Diese Funktion gibt einen INTEGER-Statuscode zurück.

Siehe [Rückgabecodes für MAPI- und SMTP-Systemprozeduren auf Seite 1163](#).

Bemerkungen

xp_sendmail ist eine Systemprozedur, die eine E-Mail-Nachricht an angegebene Empfänger sendet, sobald eine Sitzung mit xp_startmail oder xp_startsmtp gestartet wurde. Die Prozedur akzeptiert Nachrichten beliebiger Länge. Die Argumentwerte für xp_sendmail sind Zeichenfolgen. Die Länge der einzelnen Argumente ist durch den Umfang des auf Ihrem System verfügbaren Speichers begrenzt.

Das content_type-Argument ist für Benutzer bestimmt, die mit den Erfordernissen von MIME-E-Mail vertraut sind. xp_sendmail akzeptiert ASIS als content_type. Wenn content_type auf ASIS gesetzt ist, nimmt xp_sendmail an, dass der Hauptteil der Nachricht ("message") eine korrekt formatierte E-Mail mit Headern ist, und fügt keine zusätzlichen Header hinzu. Geben Sie ASIS an, um mehrteilige Nachrichten zu senden, die mehr als einen Inhaltstyp enthalten. Weitere Hinweise zu MIME finden Sie unter RFCs 2045-2049 (<http://www.ietf.org/>).

Alle durch den include_file-Parameter angegebenen Anhänge werden als MIME-Typ "application/octet-stream" mit base64-Kodierung gesendet und müssen auf dem Datenbankserver vorhanden sein.

Ab SQL Anywhere 10.0.0 sind E-Mails, die mit einem SMTP-E-Mailsystem versendet werden, kodiert, wenn die Betreffzeile Zeichen außerhalb des 7-Bit-ASCII-Bereichs enthält. Außerdem wird eine an ein SMS-fähiges Gerät gesendete E-Mail möglicherweise nicht richtig dekodiert, wenn die Betreffzeile Zeichen außerhalb des 7-Bit-ASCII-Bereichs enthält.

"xp_startmail" muss ausgeführt worden sein, um eine E-Mail-Sitzung unter MAPI starten zu können, und "xp_startsmtp" für eine E-Mail-Sitzung unter SMTP.

Wenn Sie Mail mit MAPI versenden, wird der content_type-Parameter nicht unterstützt.

Privilegien

Sie müssen das SEND EMAIL-Systemprivileg haben.

Siehe auch

- „MAPI- und SMTP-Systemprozeduren“ auf Seite 1162
- „xp_startmail-Systemprozedur“ auf Seite 1431
- „xp_startsmtp-Systemprozedur“ auf Seite 1431
- „xp_stopmail-Systemprozedur“ auf Seite 1434
- „xp_stopsmtp-Systemprozedur“ auf Seite 1434
- „CALL-Anweisung“ auf Seite 564
- „Reservierte Wörter“ auf Seite 1

Beispiel

In diesem Beispiel wird SMTP verwendet, um eine reine Textnachricht zu senden.

```
CALL xp_startsmtp( 'doe@sample.com', 'corporatemail.sample.com' );
CALL xp_sendmail( recipient='jane.smith@sample.com',
                  subject='This is my subject line',
                  "message"='This text is the body of my email.\n' );
CALL xp_stopsmt( );
```

In diesem Beispiel wird SMTP verwendet, um eine Nachricht mit HTML-Formatierung und einem Anhang zu senden.

```
CALL xp_startsmtp( 'doe@sample.com', 'corporatemail.sample.com' );
CALL xp_sendmail( recipient='jane.smith@sample.com',
                  subject='HTML mail example with attachment',
                  "message"='Plain text.<BR><BR><B>Bold text.</B><BR><BR>' ||
                           ' <a href="www.sap.com">SAP Home Page</a>',
                  content_type = 'text/html',
                  include_file = '\\temp\\sendmail2.sql' );
CALL xp_stopsmt( );
```

In diesem Beispiel wird SMTP verwendet, um eine Nachricht mit HTML-Inline-Formatierung und einem Anhang zu senden.

```
CALL xp_startsmtp( 'doe@sample.com', 'corporatemail.sample.com' );
CALL xp_sendmail( recipient='jane.smith@sample.com',
                  subject='Inline HTML mail example with attachment',
                  "message"='Content-Type: text/html;\nContent-Disposition:
inline; \n\n' ||
                           'Plain text.<BR><BR><B>Bold text.</B><BR><BR>' ||
                           ' <a href="www.sap.com">SAP Home Page</a>',
                  content_type = 'ASIS',
                  include_file = '\\temp\\sendmail3.sql' );
CALL xp_stopsmt( );
```

In diesem Beispiel wird SMTP verwendet, um eine Nachricht mit HTML-Inline-Formatierung zu senden, die eine Signatur und zwei Anhänge, darunter eine ZIP-Datei, enthält.

```
BEGIN
DECLARE content LONG VARCHAR;

SET content =
'Content-Type: multipart/mixed; boundary="xxxxx";\n' ||
'This part of the email should not be shown. If this ' ||
'is shown then the email client is not MIME compatible\n\n' ||
'--xxxxx\n' ||
'Content-Type: text/html;\n' ||
'Content-Disposition: inline;\n\n' ||
'Plain text.<BR><BR><B>Bold text.</B><BR><BR>' ||
' <a href="www.sap.com">SAP Home Page</a>\n\n' ||
xp_read_file( '\\temp\\johndoe.sig.html' ) ||
'--xxxxx\n' ||
'Content-Type: application/zip; name="sendmail4.zip"\n' ||
'Content-Transfer-Encoding: base64\n' ||
'Content-Disposition: attachment; filename="sendmail4.zip"\n\n' ||
base64_encode( xp_read_file( '\\temp\\sendmail4.zip' ) ) ||
'\n\n' ||
'--xxxxx--\n';

CALL xp_startsmtp( 'doe@sample.com', 'corporatemail.sample.com' );
CALL xp_sendmail( recipient='jane.smith@sample.com',
```

```
        subject='Inline HTML mail example with signature and 2
attachments',
        "message"=content,
        content_type = 'ASIS',
        include_file = '\\temp\\sendmail4.sql' );
CALL xp_stopsmtplib( );
END
```

xp_sprintf-Systemprozedur

Erstellt eine Ergebniszeichenfolge aus einer Reihe von Eingabezeichenfolgen

Syntax

```
xp_sprintf(
format
, parm [, parm2 ... ]
)
```

Argumente

- **format** Mit diesem CHAR(254)-Parameter können Sie angeben, wie die Ergebniszeichenfolge formatiert werden soll, wobei Platzhalter (%s) für jedes *parm*-Argument verwendet werden. Das *format*-Argument kann bis zu 50 Platzhalter enthalten und die Anzahl der Platzhalter muss mit der Anzahl der *parm*-Argumente übereinstimmen.
- **parm** Die Eingabezeichenfolgen, die in der Ergebniszeichenfolge verwendet werden. Sie können bis zu 50 dieser CHAR(254)-Argumente angeben.

Ergebnismenge

Spaltenname	Datentyp	Beschreibung
output_buffer	CHAR(254)	Gibt eine Ergebniszeichenfolge zurück, die aus den Argumente für <i>format</i> und <i>parm</i> erstellt wird.

Privilegien

Keine

Siehe auch

- „CALL-Anweisung“ auf Seite 564

Beispiel

Die folgenden Anweisungen setzen die Zeichenfolge "Hello World!" in die Ergebnisvariable.

```
CREATE VARIABLE result CHAR( 254 );
CALL xp_sprintf( result, '%s %s', 'Hello', 'World!' );
```

xp_startmail-Systemprozedur

Startet eine E-Mail-Sitzung unter MAPI.

Syntax

```
xp_startmail(  
  [ mail_user = mail-login-name  
  [, mail_password = mail-password ] ]  
)
```

Argumente

- **mail_user** Verwenden Sie diesen LONG VARCHAR-Parameter, um den MAPI-Loginnamen anzugeben. Der Standardwert ist NULL.
- **mail_password** Verwenden Sie diesen LONG VARCHAR-Parameter, um das MAPI-Kennwort anzugeben. Der Standardwert ist NULL.

Rückgabe

Diese Funktion gibt einen INTEGER-Statuscode zurück.

Siehe [Rückgabecodes für MAPI- und SMTP-Systemprozeduren auf Seite 1163](#).

Bemerkungen

xp_startmail ist eine Systemprozedur, die eine E-Mail-Sitzung startet.

Wenn Sie Microsoft Exchange verwenden, ist das Argument *mail-login-name* ein Exchange-Profilname, und Sie sollten kein Kennwort im Prozeduraufruf mit einschließen.

Nicht unterstützt unter Unix.

Privilegien

Sie müssen das SEND EMAIL-Systemprivileg haben.

Siehe auch

- „MAPI- und SMTP-Systemprozeduren“ auf Seite 1162
- „xp_stopmail-Systemprozedur“ auf Seite 1434
- „xp_sendmail-Systemprozedur“ auf Seite 1426
- „xp_startsmtp-Systemprozedur“ auf Seite 1431
- „xp_stopsmtp-Systemprozedur“ auf Seite 1434
- „CALL-Anweisung“ auf Seite 564
- „Allgemeine Sicherheitstipps“ [*SQL Anywhere Server - Datenbankadministration*]

xp_startsmtp-Systemprozedur

Startet eine E-Mail-Sitzung unter SMTP

Syntax

```
xp_startsmtp(  
  smtp_sender = email-address  
  , smtp_server = smtp-server  
  [, smtp_port = port-number ]  
  [, timeout = timeout ]  
  [, smtp_sender_name = username ]  
  [, smtp_auth_username = auth-username ]  
  [, smtp_auth_password = auth-password ]  
  [, trusted_certificates = public-certificate ]  
  [, certificate_company = organization ]  
  [, certificate_unit = organization-unit ]  
  [, certificate_name = common-name ]  
)
```

Argumente

- **smtp_sender** Dieser LONG VARCHAR-Parameter gibt die E-Mail-Adresse des Absenders an.
- **smtp_server** Dieser LONG VARCHAR-Parameter gibt an, welcher SMTP-Server verwendet werden soll und ist der Servername oder die IP-Adresse.
- **smtp_port** Dieser optionale INTEGER-Parameter gibt die Portnummer für die Verbindung zum SMTP-Server an. Standardwert ist "25".
- **timeout** Dieser optionale INTEGER-Parameter gibt in Sekunden an, wie lange auf eine Antwort vom Datenbankserver gewartet werden soll, bevor der aktuelle Aufruf von xp_sendmail abgebrochen werden soll. Der Standardwert beträgt 60 Sekunden.
- **smtp_sender_name** Dieser optionale LONG VARCHAR-Parameter gibt einen Alias für die E-Mail-Adresse des Absenders an. Beispiel: "JSmith" anstelle von *email-address*. Der Standardwert ist NULL.
- **smtp_auth_username** Dieser optionale LONG VARCHAR-Parameter gibt den Benutzernamen zur Übergabe an SMTP-Server an, die eine Authentifizierung erfordern. Der Standardwert ist NULL.
- **smtp_auth_password** Dieser optionale LONG VARCHAR-Parameter gibt das Kennwort zur Übergabe an SMTP-Server an, die eine Authentifizierung erfordern. Der Standardwert ist NULL.
- **trusted_certificates** Dieser optionale LONG VARCHAR-Parameter gibt ein Schlüssel/Wert-Paar an oder Pfad und Dateiname einer Datei, die ein oder mehrere vertrauenswürdige Zertifikate enthält. Der Standardwert ist NULL. Wenn dieser Parameter NULL ist, wird eine Standard-SMTP-Verbindung hergestellt.

Schlüssel	Value
file=	Pfad und Dateiname einer Datei, die ein oder mehrere vertrauenswürdige Zertifikate enthält.
cert_name=	Der Name eines in der Datenbank gespeicherten Zertifikats.

Das vertrauenswürdige Zertifikat kann ein selbstsigniertes Zertifikat des Servers, ein öffentliches Unternehmensstammzertifikat oder ein Zertifikat einer kommerziellen Zertifizierungsstelle sein. Sie müssen Ihre Zertifikate mit RSA erstellen.

- **certificate_company** Dieser optionale LONG VARCHAR-Parameter gibt an, dass der Client Serverzertifikate nur akzeptiert, wenn das Feld "Organisation" auf dem Zertifikat zu diesem Wert passt. Dieser Parameter wird ignoriert, wenn der trusted_certificates-Wert NULL ist. Der Standardwert ist NULL.
- **certificate_unit** Dieser optionale LONG VARCHAR-Parameter gibt an, dass der Client Serverzertifikate nur akzeptiert, wenn das Feld "Organisationseinheit" (Organisation Unit) auf dem Zertifikat zu diesem Wert passt. Dieser Parameter wird ignoriert, wenn der trusted_certificates-Wert NULL ist. Der Standardwert ist NULL.
- **certificate_name** Dieser optionale LONG VARCHAR-Parameter gibt an, dass der Client Serverzertifikate nur akzeptiert, wenn das Namensfeld (Common Name) auf dem Zertifikat zu diesem Wert passt. Dieser Parameter wird ignoriert, wenn der trusted_certificates-Wert NULL ist. Der Standardwert ist NULL.

Rückgabe

Diese Funktion gibt einen INTEGER-Statuscode zurück.

Siehe [Rückgabecodes für MAPI- und SMTP-Systemprozeduren auf Seite 1163](#).

Bemerkungen

xp_startsmtp ist eine Systemprozedur, die eine E-Mail-Sitzung für eine angegebene E-Mail-Adresse startet, indem sie sich beim SMTP-Server anmeldet. Nach Zeitüberschreitung wird diese Verbindung getrennt. Darum wird empfohlen, dass Sie xp_start_smtp aufrufen, kurz bevor Sie xp_sendmail ausführen wollen.

Wenn Sie smtp_auth_username und smtp_auth_password angeben und der Server die SMTP-Authentifizierungsfähigkeit nicht unterstützt, wird Fehlercode 104 zurückgegeben.

xp_startsmtp kann durch Virusprogramme beeinträchtigt werden und gibt dann den Fehlercode 100 zurück. Bei McAfee VirusScan Version 8.0.0 und später beeinträchtigen die Einstellungen für den Schutz vor Massenversendungen von E-Mailwürmern xp_sendmail. Wenn Ihre Virenschutz-Software es Ihnen ermöglicht, Prozesse anzugeben, die den Massenmailingschutz umgehen, geben Sie *dbeng16.exe* und *db_srv16.exe* an. Zum Beispiel können Sie mit McAfee VirusScan Massenmailing verhindern, indem Sie diese zwei Prozesse im Bereich **Eigenschaften** zur Liste der **ausgeschlossenen Prozesse** hinzufügen.

Privilegien

Sie müssen das SEND EMAIL-Systemprivileg haben.

Siehe auch

- „MAPI- und SMTP-Systemprozeduren“ auf Seite 1162
- „xp_startmail-Systemprozedur“ auf Seite 1431
- „xp_stopmail-Systemprozedur“ auf Seite 1434
- „xp_sendmail-Systemprozedur“ auf Seite 1426
- „xp_stopsmtplib-Systemprozedur“ auf Seite 1434
- „CALL-Anweisung“ auf Seite 564
- „CREATE CERTIFICATE-Anweisung“ auf Seite 582
- „Allgemeine Sicherheitstipps“ [*SQL Anywhere Server - Datenbankadministration*]

xp_stopmail-Systemprozedur

Schließt eine MAPI-E-Mail-Sitzung

Syntax

xp_stopmail()

Rückgabe

Diese Funktion gibt einen INTEGER-Statuscode zurück.

Siehe [Rückgabecodes für MAPI- und SMTP-Systemprozeduren auf Seite 1163](#).

Bemerkungen

xp_stopmail ist eine Systemprozedur, die eine E-Mail-Sitzung stoppt.

Nicht unterstützt unter Unix.

Privilegien

Sie müssen das SEND EMAIL-Systemprivileg haben.

Siehe auch

- „MAPI- und SMTP-Systemprozeduren“ auf Seite 1162
- „xp_startmail-Systemprozedur“ auf Seite 1431
- „xp_sendmail-Systemprozedur“ auf Seite 1426
- „xp_startsmtp-Systemprozedur“ auf Seite 1431
- „xp_stopsmtplib-Systemprozedur“ auf Seite 1434
- „CALL-Anweisung“ auf Seite 564
- „Allgemeine Sicherheitstipps“ [*SQL Anywhere Server - Datenbankadministration*]

xp_stopsmtplib-Systemprozedur

Schließt eine SMTP-E-Mail-Sitzung

Syntax

xp_stopsmtplib()

Rückgabe

Diese Funktion gibt einen INTEGER-Statuscode zurück.

Siehe [Rückgabecodes für MAPI- und SMTP-Systemprozeduren auf Seite 1163](#).

Bemerkungen

xp_stopsmtplib ist eine Systemprozedur, die eine E-Mail-Sitzung stoppt.

Privilegien

Sie müssen das SEND EMAIL-Systemprivileg haben.

Siehe auch

- „MAPI- und SMTP-Systemprozeduren“ auf Seite 1162
- „xp_startmail-Systemprozedur“ auf Seite 1431
- „xp_stopmail-Systemprozedur“ auf Seite 1434
- „xp_sendmail-Systemprozedur“ auf Seite 1426
- „xp_startsmtp-Systemprozedur“ auf Seite 1431
- „CALL-Anweisung“ auf Seite 564
- „Allgemeine Sicherheitstipps“ [[SQL Anywhere Server - Datenbankadministration](#)]

xp_write_file-Systemprozedur

Schreibt Daten aus einer SQL-Anweisung in eine Datei

Syntax

```
xp_write_file(  
    filename  
    , file_contents  
)
```

Argumente

- **filename** Verwenden Sie diesen LONG VARCHAR-Parameter, um den Dateinamen anzugeben.
- **file_contents** Verwenden Sie diesen LONG BINARY-Parameter, um den Inhalt anzugeben, der in die Datei geschrieben werden soll.

Rückgabe

Diese Funktion gibt einen INTEGER-Statuscode zurück.

Bemerkungen

Diese Funktion schreibt den *file_contents* in die Datei *filename*. Sie gibt "0" zurück, falls erfolgreich, und nicht Null bei Fehlschlag.

Dem *filename*-Wert kann ein absoluter oder ein relativer Pfad vorangestellt werden. Wenn *filename* ein relativer Pfad vorangestellt wird, ist der Dateiname relativ zum aktuellen Arbeitsverzeichnis des Datenbankservers. Falls die Datei bereits besteht, wird ihr Inhalt überschrieben.

Diese Funktion kann zum Entladen umfangreicher Binärdaten in Dateien hilfreich sein.

Sie können die CSCONVERT-Funktion auch verwenden, um die für die xp_write_file-Systemprozedur erforderliche Zeichensatzkonvertierung durchzuführen.

Privilegien

Sie müssen das WRITE FILE-Systemprivileg haben.

Siehe auch

- „CSCONVERT-Funktion [Zeichenfolge]“ auf Seite 211
- „xp_read_file-Systemprozedur“ auf Seite 1424
- „Einfügen von Dokumenten und Grafiken“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Systemprozeduren für Verzeichnisse und Dateien“ auf Seite 1164

Beispiele

Dieses Beispiel verwendet xp_write_file, um eine Datei *accountnum.txt* zu erstellen, die die Daten 123456 enthält:

```
CALL xp_write_file( 'accountnum.txt', '123456' );
```

Dieses Beispiel fragt die Contacts-Tabelle der Beispieldatenbank ab und erstellt dann eine Textdatei für jeden Kontakt, der in New Jersey wohnt. Die Textdateien werden benannt, wobei eine Verkettung des Vornamens (GivenName), des Nachnamens (Surname) und der Zeichenfolge *.txt* verwendet wird (z.B. *Reeves_Scott.txt*), und enthalten auf separaten Zeilen die Straßenadresse (Street), Stadt (City) und den Bundesstaat (State).

```
SELECT xp_write_file(
  Surname || '_' || GivenName || '.txt',
  Street || '\n' || City || '\n' || State )
FROM Contacts WHERE State = 'NJ';
```

Dieses Beispiel verwendet xp_write_file, um eine Bilddatei (JPG) für jedes Produkt in der Products-Tabelle zu erstellen. Jeder Wert in der ID-Spalte wird zu einem Dateinamen für eine Datei, die den entsprechenden Wert der Photo-Spalte enthält:

```
SELECT xp_write_file( ID || '.jpg', Photo ) FROM Products;
```

Im obenstehenden Beispiel ist ID eine Zeile mit einer UNIQUE-Integritätsregel. Das ist wichtig, damit eine Datei nicht mit dem Inhalt von nachfolgenden Zeilen überschrieben wird. Auch müssen Sie die Dateierweiterung angeben, die den in der Spalte gespeicherten Daten entspricht. In diesem Fall werden in Products.Photo Bilddaten (JPEG-Dateien) gespeichert.

Ansichten

Systemansichten

Der Katalog enthält Systemtabellen, die durch Schlüssel und Indizes miteinander verknüpft sind. In SQL Anywhere sind die Systemtabellen verborgen. Es gibt allerdings eine Systemansicht für jede Tabelle. Eine Systemansicht kann auch Spalten aus mehr als einer Systemtabelle enthalten, um einem gebräuchlichen Join zu entsprechen.

Um die Kompatibilität mit zukünftigen Versionen des SQL Anywhere-Katalogs zu gewährleisten, müssen Anwendungen Systemansichten und nicht die zugrunde liegenden Systemtabellen verwenden, die sich ändern können.

Detaillierte Systeminformationen für Ansichten und Definitionen anzeigen

Datenbankadministratoren können über Sybase Central auf Informationen zu Systemansichten zugreifen, einschließlich ihrer Definitionen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Systemansichten werden aktualisiert, wenn ein Checkpoint auftritt.

Aufgabe

1. Stellen Sie eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie auf die Datenbank und klicken Sie auf **Eigentümerfilter konfigurieren**.
3. Klicken Sie auf **SYS** und dann auf **OK**.
4. Doppelklicken Sie im linken Fensterausschnitt auf **Ansichten**.
5. Klicken Sie im linken Fensterausschnitt auf eine Ansicht mit dem Eigentümer SYS (angezeigt durch die Phrase SYS in Klammern hinter dem Namen der Ansicht) und klicken Sie im rechten Fensterausschnitt auf die Registerkarte **SQL**.
6. Klicken Sie auf die Registerkarte **Daten**.

Ergebnisse

Die Ansichtsdefinition erscheint auf der Registerkarte **Daten**.

SYSARTICLE-Systemansicht

Jede Zeile in der SYSARTICLE-Systemansicht beschreibt einen Artikel in einer Publikation. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSARTICLE.

Spaltenname	Datentyp	Beschreibung
publication_id	UNSIGNED INT	Die Publikation, zu der der Artikel gehört
table_id	UNSIGNED INT	Jeder Artikel besteht aus Spalten und Zeilen einer einzelnen Tabelle. Diese Spalte enthält die Tabellen-ID für diese Tabelle.
where_expr	LONG VARCHAR	Diese Spalte enthält die Suchbedingung für Artikel, die eine Teilmenge von Zeilen enthalten, die in einer WHERE-Klausel festgelegt sind.
subscribe_by_expr	LONG VARCHAR	Diese Spalte enthält den Ausdruck für Artikel, die eine Teilmenge von Zeilen enthalten, die in einem SUBSCRIBE BY-Ausdruck festgelegt sind.
query	CHAR(1)	Gibt Daten zum Artikeltyp an den Datenbankserver weiter
alias	VARCHAR(256)	Der Alias für den Artikel
schema_change_active	BIT	1, wenn Tabelle und Publikation Teil einer Änderung des Synchronisationsschemas sind.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (publication_id, table_id)
```

```
FOREIGN KEY (publication_id) REFERENCES SYS.ISYSPUBLICATION (publication_id)
```

```
FOREIGN KEY (table_id) REFERENCES SYS.ISYSTAB (table_id)
```

SYSARTICLECOL-Systemansicht

Jede Zeile in der SYSARTICLECOL-Systemansicht bezeichnet eine Spalte in einem Artikel. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSARTICLECOL.

Spaltenname	Datentyp	Beschreibung
publication_id	UNSIGNED INT	Ein eindeutiger Bezeichner für die Publikation, zu der die Spalte gehört
table_id	UNSIGNED INT	Die Tabelle, zu der die Spalte gehört

Spaltenname	Datentyp	Beschreibung
column_id	UNSIGNED INT	Der Spaltenbezeichner aus der Systemansicht SYSTABCOL

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (publication_id, table_id, column_id)
```

```
FOREIGN KEY (publication_id, table_id) REFERENCES SYS.ISYSARTICLE  
(publication_id, table_id)
```

```
FOREIGN KEY (table_id, column_id) REFERENCES SYS.ISYSTABCOL (table_id,  
column_id)
```

SYSCAPABILITY-Systemansicht

Jede Zeile der SYSCAPABILITY-Systemansicht gibt den Status einer Fähigkeit auf einem entfernten Datenbankserver an. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSCAPABILITY.

Spaltenname	Datentyp	Beschreibung
capid	INTEGER	Die ID der Fähigkeit, wie in der SYSCAPABILITYNAME-Systemansicht aufgelistet
srvid	UNSIGNED INT	Der Server, auf den sich die Fähigkeit bezieht, wie in der SYS-SERVER-Systemansicht aufgelistet
capvalue	CHAR(128)	Der Wert der Fähigkeit.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (capid, srvid)
```

```
FOREIGN KEY (srvid) REFERENCES SYS.ISYSSERVER (srvid)
```

Siehe auch

- „SYSCAPABILITYNAME-Systemansicht“ auf Seite 1439

SYSCAPABILITYNAME-Systemansicht

Jede Zeile in der SYSCAPABILITYNAME-Systemansicht liefert einen Namen für jede Fähigkeits-ID in der SYSCAPABILITY-Systemansicht.

Spaltenname	Datentyp	Beschreibung
capid	INTEGER	Eine Zahl, die die Fähigkeit eindeutig bezeichnet.
capname	VARCHAR(32000)	Der Name der Fähigkeit.

Bemerkungen

Die Systemansicht SYSCAPABILITYNAME wird mit einer Kombination aus sa_rowgenerator und den folgenden Servereigenschaften definiert:

- RemoteCapability
- MaxRemoteCapability

Siehe auch

- „Liste der Datenbankservereigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „SYSCAPABILITY-Systemansicht“ auf Seite 1439

SYSCERTIFICATE-Systemansicht

In jeder Zeile der SYSCERTIFICATE-Systemansicht wird ein Zertifikat als Text im PEM-Format gespeichert. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSCERTIFICATE.

Spaltenname	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Die ID des Zertifikats.
cert_name	CHAR(128)	Der Zertifikatsname.
contents	LONG BINARY	Der Inhalt des Zertifikats in komprimierter Form.
update_time	TIMESTAMP	Der Zeitpunkt in Ortszeit (Datum und Uhrzeit) der letzten Erstellung oder Ersetzung.
update_time_utc	TIMESTAMP WITH TIME ZONE	Der Zeitpunkt in UTC (Datum und Uhrzeit) der letzten Erstellung oder Ersetzung.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (object_id)

UNIQUE INDEX (cert_name)
```

SYSCHECK-Systemansicht

Jede Zeile in der SYSCHECK-Systemansicht enthält die Definition für eine benannte Prüf-Integritätsregel in einer Tabelle. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSCHECK.

Spaltenname	Datentyp	Beschreibung
check_id	UNSIGNED INT	Eine Zahl, die die Integritätsregel in der Datenbank eindeutig kennzeichnet
check_defn	LONG VARCHAR	Der CHECK-Ausdruck

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (check_id)
```

```
FOREIGN KEY (check_id) REFERENCES SYS.ISYSCONSTRAINT (constraint_id)
```

SYSCOLPERM-Systemansicht

Die GRANT-Anweisung kann verwendet werden, um UPDATE-, SELECT- oder REFERENCES-Privilegien für einzelne Spalten in einer Tabelle zu erteilen. Jede Spalte mit UPDATE-, SELECT- oder REFERENCES-Privilegien wird in einer Zeile der SYSCOLPERM-Systemansicht erfasst. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSCOLPERM.

Spaltenname	Datentyp	Beschreibung
table_id	UNSIGNED INT	Die Tabellennummer der Tabelle, die die Spalte enthält
grantee	UNSIGNED INT	Die Benutzernummer der Benutzer-ID, der das Privileg für die Spalte erteilt wurde. Wenn die PUBLIC-Rolle der Berechtigungsempfänger ist, wird das Privileg allen Benutzer-IDs erteilt.
grantor	UNSIGNED INT	Die Benutzernummer der Benutzer-ID, die das Privileg erteilt.
column_id	UNSIGNED INT	Diese Spaltennummer identifiziert zusammen mit table_id die Spalte, für die das Privileg erteilt wurde.
privilege_type	SMALLINT	Die Nummer dieser Spalte zeigt die Art des Spaltenprivilegs an (16=REFERENCES, 1=SELECT oder 8=UPDATE).
is_grantable	CHAR(1)	Zeigt an, ob das Privileg für die Spalte mit WITH GRANT OPTION erteilt wurde.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (table_id, grantee, grantor, column_id, privilege_type)
```

```
FOREIGN KEY (table_id, column_id) REFERENCES SYS.ISYSTABCOL (table_id, column_id)
```

```
FOREIGN KEY (grantor) REFERENCES SYS.ISYSUSER (user_id)
```

```
FOREIGN KEY (grantee) REFERENCES SYS.ISYSUSER (user_id)
```

SYSCOLSTAT-Systemansicht

Die SYSCOLSTAT-Systemansicht enthält Spaltenstatistiken (einschließlich Histogramme), die vom Optimierer verwendet werden. Der Inhalt dieser Ansicht wird am besten mit der gespeicherten Prozedur sa_get_histogram oder dem Histogrammdienstprogramm abgerufen. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSCOLSTAT.

Spaltenname	Datentyp	Beschreibung
table_id	UNSIGNED INT	Eine Zahl, die die Tabelle oder materialisierte Ansicht eindeutig identifiziert, zu der diese Spalte gehört
column_id	UNSIGNED INT	Eine Zahl, die zusammen mit table_id die Spalte eindeutig kennzeichnet
format_id	SMALLINT	Wird nur vom System verwendet
update_time	TIMESTAMP	Der Zeitpunkt der letzten Aktualisierung der Spaltenstatistiken in Ortszeit.
density	FLOAT	Eine Schätzung der mittleren Selektivität eines einzelnen Wertes für die Spalte, ohne die großen, in der Zeile gespeicherten Einzelwertsselektivitäten einzubeziehen.
max_steps	SMALLINT	Wird nur vom System verwendet
actual_steps	SMALLINT	Wird nur vom System verwendet
step_values	LONG BINARY	Wird nur vom System verwendet
frequencies	LONG BINARY	Wird nur vom System verwendet
update_time_utc	TIMESTAMP WITH TIME ZONE	Der Zeitpunkt der letzten Aktualisierung der Spaltenstatistiken in UTC.

Hinweis

Bei Datenbanken, die ab SQL Anywhere 16 erstellt werden, ist die zugrunde liegende Systemtabelle für diese Ansicht immer verschlüsselt, um die Daten vor nicht autorisiertem Zugriff zu schützen.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (table_id, column_id)
```

```
FOREIGN KEY (table_id, column_id) REFERENCES SYS.ISYSTABCOL (table_id, column_id)
```

Siehe auch

- „sa_get_histogram-Systemprozedur“ auf Seite 1225
- „Histogramm-Dienstprogramm (dbhist)“ [*SQL Anywhere Server - Datenbankadministration*]

SYSCONSTRAINT-Systemansicht

Jede Zeile in der SYSVIEW-Systemansicht beschreibt eine benannte Integritätsregel in der Datenbank. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSCONSTRAINT.

Spaltenname	Datentyp	Beschreibung
constraint_id	UNSIGNED INT	Die eindeutige ID für die Integritätsregel
constraint_type	CHAR(1)	Der Integritätsregeltyp: <ul style="list-style-type: none"> • C Prüf-Integritätsregel auf Spalten • T Tabellen-Integritätsregel • P Primärschlüssel • F Fremdschlüssel • U Eindeutigkeits-Integritätsregel
ref_object_id	UNSIGNED BIGINT	Die Objekt-ID der Spalte, der Tabelle oder des Indexes, auf den sich die Integritätsregel bezieht
table_object_id	UNSIGNED BIGINT	Die Objekt-ID der Tabelle, auf die sich die Integritätsregel bezieht.
constraint_name	CHAR(128)	Der Name der Integritätsregel

Integritätsregeln für die zugrunde liegende Systemtabelle

```

PRIMARY KEY (constraint_id)

FOREIGN KEY (ref_object_id) REFERENCES SYS.ISYSOBJECT (object_id)

FOREIGN KEY (table_object_id) REFERENCES SYS.ISYSOBJECT (object_id)

UNIQUE CONSTRAINT (table_object_id, constraint_name)

```

SYSDATABASE-Systemansicht

Jede Zeile in der SYSDATABASE-Systemansicht beschreibt einen DBSpace. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSDATABASE.

Spaltenname	Datentyp	Beschreibung
dbfile_id	SMALLINT	Wird nur intern verwendet.

Spaltenname	Datentyp	Beschreibung
dbspace_id	SMALLINT	Jeder DBSpace-Datei in einer Datenbank ist eine eindeutige Nummer zugewiesen. Der SYSTEM-DBSpace enthält alle Systemobjekte und hat eine dbspace_id "0".
dbfile_name	CHAR(128)	Der Dateiname für den DBSpace.
file_name	LONG VARCHAR	Ein eindeutiger Name für den DBSpace. Er wird im Befehl CREATE TABLE verwendet.
lob_map	LONG VARBIT	Wird nur intern verwendet.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (dbfile_id)

FOREIGN KEY (dbspace_id) REFERENCES SYS.ISYSDBSPACE (dbspace_id)

UNIQUE index (file_name)
```

SYSDBSpace-Systemansicht

Jede Zeile in der SYSDBSpace-Systemansicht beschreibt eine DBSpace-Datei. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSDBSPACE.

Spaltenname	Datentyp	Beschreibung
dbspace_id	SMALLINT	Eindeutige Zahl, die den DBSpace identifiziert. Der SYSTEM-DBSpace enthält alle Systemobjekte und hat eine dbspace_id "0".
object_id	UNSIGNED BIGINT	Die Objekt-ID der Tabelle
dbspace_name	CHAR(128)	Ein eindeutiger Name für den DBSpace. Er wird im Befehl CREATE TABLE verwendet.
store_type	TINYINT	Wird nur intern verwendet.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (dbspace_id)

FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE FULL
```

SYSDATABASEPERM-Systemansicht

Jede Zeile in der SYSDATABASEPERM-Systemansicht beschreibt ein Privileg für eine Database-Datei. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSDATABASEPERM.

Spaltenname	Datentyp	Beschreibung
database_id	SMALLINT	Eindeutige Zahl, die den Database identifiziert. Der SYSTEM-Database enthält alle Systemobjekte und hat eine database_id "0".
grantee	UNSIGNED INT	Die Benutzer-ID des Benutzers, der das Privileg erhält.
privilege_type	SMALLINT	Das Privileg, das dem Berechtigungsempfänger erteilt wird. CREATE gibt beispielsweise dem Berechtigungsempfänger das Privileg, Objekte im Database zu erstellen.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
FOREIGN KEY (database_id) REFERENCES SYS.ISYSDATABASE (database_id)
```

```
FOREIGN KEY (grantee) REFERENCES SYS.ISYSUSER (user_id)
```

Siehe auch

- „GRANT-Anweisung“ auf Seite 881
- „Benutzersicherheit (Rollen und Privilegien)“ [[SQL Anywhere Server - Datenbankadministration](#)]

SYSDEPENDENCY-Systemansicht

Jede Zeile in der Systemansicht SYSDEPENDENCY beschreibt eine Abhängigkeit zwischen zwei Datenbankobjekten. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSDEPENDENCY.

Zwischen zwei Datenbankobjekten besteht eine Abhängigkeit, wenn ein Objekt das andere in seiner Definition referenziert. Wenn die Abfragespezifikation für eine Ansicht beispielsweise eine Tabelle referenziert, ist die Ansicht von der Tabelle abhängig. Der Datenbankserver protokolliert Abhängigkeiten von Ansichten auf Tabellen, materialisierten Ansichten und Spalten.

Spaltenname	Datentyp	Beschreibung
ref_object_id	UNSIGNED BIGINT	Die Objekt-ID des referenzierten Objekts
dep_object_id	UNSIGNED BIGINT	Die ID des referenzierenden Objekts

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (ref_object_id, dep_object_id)
```

```
FOREIGN KEY (ref_object_id) REFERENCES SYS.ISYSOBJECT (object_id)
```

```
FOREIGN KEY (dep_object_id) REFERENCES SYS.ISYSOBJECT (object_id)
```

Siehe auch

- „sa_dependent_views-Systemprozedur“ auf Seite 1202
- „Ansichtenabhängigkeiten“ [SQL Anywhere Server - SQL-Benutzerhandbuch]

SYSDOMAIN-Systemansicht

Die SYSDOMAIN-Systemansicht protokolliert Informationen über integrierte Datentypen (auch Domänen genannt). Der Inhalt dieser Ansicht ändert sich während des Normalbetriebs nicht. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSDOMAIN.

Spaltenname	Datentyp	Beschreibung
domain_id	SMALLINT	Die eindeutige Zahl, die jedem Datentyp zugeordnet ist. Diese Zahlen können nicht geändert werden.
domain_name	CHAR(128)	Der Name des normalerweise im Befehl CREATE TABLE gefundenen Datentyps, wie CHAR oder INTEGER.
type_id	SMALLINT	Der ODBC-Datentyp. Dieser Wert entspricht dem Wert für data_type in der Transact-SQL-Kompatibilitätstabelle "dbo.SYSTYPES".
"precision"	SMALLINT	Die Anzahl der signifikanten Stellen, die mit diesem Datentyp gespeichert werden können. Der Spaltenwert ist NULL bei nichtnumerischen Datentypen.

Integritätsregeln für die zugrunde liegende Systemtabelle

PRIMARY KEY (domain_id)

SYSEVENT-Systemansicht

Jede Zeile in der SYSEVENT-Systemansicht beschreibt ein Ereignis, das mit CREATE EVENT erstellt wurde. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSEVENT.

Spaltenname	Datentyp	Beschreibung
event_id	UNSIGNED INT	Die jedem Ereignis zugeordnete eindeutige Nummer
object_id	UNSIGNED BIGINT	Die interne ID für das Ereignis, die es in der Datenbank eindeutig kennzeichnet
creator	UNSIGNED INT	Die Benutzernummer des Besitzers des Ereignisses. Der Name des Benutzers ist in der SYSUSER-Systemansicht zu finden.
event_name	VARCHAR(128)	Der Name des Ereignisses.
enabled	CHAR(1)	Zeigt an, ob das Ereignis ausgelöst werden darf.

Spaltenname	Datentyp	Beschreibung
location	CHAR(1)	<p>Der Ort, an dem das Ereignis auslösen soll:</p> <ul style="list-style-type: none"> • Y = AT ALL-Klausel und FOR PRIMARY-Klausel angegeben • E = AT CONSOLIDATED-Klausel und FOR PRIMARY-Klausel angegeben • T = AT REMOTE-Klausel und FOR PRIMARY-Klausel angegeben • P = (AT-Klausel nicht angegeben) FOR PRIMARY-Klausel angegeben • B = AT ALL-Klausel und FOR ALL-Klausel angegeben • D = AT CONSOLIDATED-Klausel und FOR ALL-Klausel angegeben • S = AT REMOTE-Klausel und FOR ALL-Klausel angegeben • M = (AT-Klausel nicht angegeben) FOR ALL-Klausel angegeben • C = AT CONSOLIDATED (FOR-Klausel nicht angegeben) • R = AT REMOTE (FOR-Klausel nicht angegeben) • A = AT ALL-Klausel (FOR-Klausel nicht angegeben)
event_type_id	UNSIGNED INT	Bei Systemereignissen der Ereignistyp, wie in der SYSEVENT-TYPE-Systemansicht aufgeführt.
action	LONG VAR-CHAR	Die Event-Handler-Definition. Ein ausgeblendeter Wert zeigt ein verstecktes Ereignis an.
external_action	LONG VAR-CHAR	Wird nur vom System verwendet
condition	LONG VAR-CHAR	Die Bedingung, mit der das Auslösen des Event-Handlers gesteuert wird
remarks	LONG VAR-CHAR	Bemerkungen zum Ereignis. Diese Spalte kommt aus ISYSREMARK.
source	LONG VAR-CHAR	Die ursprüngliche Quelle für das Ereignis. Diese Spalte kommt aus ISYSSOURCE.

Integritätsregeln für die zugrunde liegende Systemtabelle

PRIMARY KEY (event_id)

FOREIGN KEY (creator) REFERENCES SYS.ISYSUSER (user_id)

FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE FULL

UNIQUE INDEX (event_name)

Siehe auch

- [„SYSEVENTTYPE-Systemansicht“ auf Seite 1448](#)

SYSEVENTTYPE-Systemansicht

Die SYSEVENTTYPE-Systemansicht definiert Systemereignistypen, die von CREATE EVENT referenziert werden können.

Spaltenname	Datentyp	Beschreibung
event_type_id	INT	Die jedem Ereignistyp zugeordnete eindeutige Zahl
name	VARCHAR(32000)	Der Name des Systemereignistyps
description	VARCHAR(32000)	Eine Beschreibung des Systemereignistyps

Bemerkungen

Die Systemansicht SYSEVENTTYPE wird mit einer Kombination aus sa_rowgenerator und den folgenden Servereigenschaften definiert:

- EventTypeName
- EventTypeDesc
- MaxEventType

Siehe auch

- [„Liste der Datenbankservereigenschaften“ \[SQL Anywhere Server - Datenbankadministration\]](#)
- [„SYSEVENT-Systemansicht“ auf Seite 1446](#)

SYSEXTERNENV-Systemansicht

Viele externe Laufzeitumgebungen werden unterstützt: Embedded SQL und ODBC-Anwendungen, die in C/C++ geschrieben wurden, sowie in Java, Perl, PHP oder in Sprachen wie C# und Visual Basic geschriebene Anwendungen, die auf der Common Language Runtime (CLR) von Microsoft basieren.

Jede Zeile in der SYSEXTERNENV-Systemansicht beschreibt die Informationen, die benötigt werden, um die jeweiligen externen Umgebungen zu erkennen und zu starten. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSEXTERNENV.

Spaltenname	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Ein eindeutiger Bezeichner für die externe Umgebung.

Spaltenname	Datentyp	Beschreibung
name	CHAR(128)	Diese Spalte enthält den Namen der externen Umgebung oder Sprache. Namen können sein: java , perl , php , clr , c_esql32 , c_esql64 , c_odbc32 oder c_odbc64 .
scope	CHAR(1)	<p>Diese Spalte ist C für CONNECTION bzw. D für DATABASE. Die Bereichsspalte zeigt an, ob die externe Umgebung einmal pro Verbindung oder einmal pro Datenbank gestartet wird.</p> <p>Externe Umgebungen, die einmal pro Verbindung gestartet werden (wie PERL, PHP, C_ESQL32, C_ESQL64, C_ODBC32 und C_ODBC64), läuft eine Instanz der externen Umgebung für jede Verbindung, die die externe Umgebung verwendet. Einmal pro Verbindung gestartete externe Umgebungen werden beendet, wenn die Verbindung beendet wird.</p> <p>Bei einmal pro Datenbank gestarteten externen Umgebungen (wie JAVA und CLR) läuft eine Instanz für jede Datenbank, die die externe Umgebung verwendet. Die einmal pro Datenbank gestartete externe Umgebung wird beendet, wenn die Datenbank gestoppt wird.</p>
support_result_sets	CHAR(1)	Diese Spalte identifiziert die externen Umgebungen, die Ergebnismengen zurückgeben können. Alle externe Umgebungen können Ergebnismengen zurückgeben, ausgenommen Perl und PHP.
location	LONG VAR-CHAR	Diese Spalte gibt den Speicherort auf dem Datenbankservercomputer an, an dem sich die Programm- oder Binärdatei für die externe Umgebung befindet. Sie enthält den Namen des Programms oder der Binärdatei. Der Pfad kann voll qualifiziert oder relativ sein. Wenn der Pfad relativ ist, muss sich das Programm oder die Binärdatei an einem Speicherort befinden, an dem sie der Datenbankserver finden kann.
options	LONG VAR-CHAR	Diese Spalte enthält alle Optionen, die in der Eingabeaufforderung erforderlich sind, um die der externen Umgebung zugeordnete Programmdatei zu starten. Diese Spalte sollte nicht geändert werden.

Spaltenname	Datentyp	Beschreibung
user_id	UNSIGNED INT	Wenn eine externe Umgebung erstmals gestartet wird, muss sie eine Verbindung mit der Datenbank herstellen, um die für die Nutzung der externen Umgebung erforderlichen Einstellungen einzurichten. Standardmäßig wird diese Verbindung mit der DBA-Benutzer-ID hergestellt. Wenn der Datenbankadministrator jedoch die externe Umgebung unter einer anderen Benutzer-ID mit MANAGE ANY EXTERNAL OBJECT -Systemprivileg laufen lassen möchte, enthält die user_id-Spalte stattdessen diese andere Benutzer-ID. Normalerweise ist diese Spalte NULL und der Datenbankserver verwendet standardmäßig die DBA-Benutzer-ID.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (object_id)
```

```
FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE FULL
```

```
FOREIGN KEY (user_id) REFERENCES SYS.ISYSUSER (user_id)
```

```
UNIQUE INDEX (name)
```

Siehe auch

- „Unterstützung für externe Umgebungen in SQL Anywhere“ [[SQL Anywhere Server - Programmierung](#)]

SYSEXTERNENVOBJECT-Systemansicht

Viele externe Laufzeitumgebungen werden unterstützt: Embedded SQL und ODBC-Anwendungen, die in C/C++ geschrieben wurden, sowie in Java, Perl, PHP oder in Sprachen wie C# und Visual Basic geschriebene Anwendungen, die auf der Common Language Runtime (CLR) von Microsoft basieren.

Jede Zeile in der Systemansicht SYSEXTERNENVOBJECT gibt ein installiertes externes Objekt an. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSEXTERNENVOBJECT.

Spaltenname	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Ein eindeutiger Bezeichner für das externe Objekt.
extenv_id	UNSIGNED BIGINT	Der eindeutige Bezeichner für die externe Umgebung (SYSEXTERNENV.object_id).
owner	UNSIGNED INT	Diese Spalte identifiziert den Ersteller/Eigentümer des externen Objekts.

Spaltenname	Datentyp	Beschreibung
name	LONG VAR-CHAR	Diese Spalte enthält den Namen des externen Objekts, wie in der INSTALL EXTERNAL OBJECT-Anweisung angegeben.
contents	LONG BINARY	Der Inhalt des externen Objekts.
update_time	TIMESTAMP	Diese Spalte enthält den Zeitpunkt in Ortszeit, zu dem das Objekt zum letzten Mal geändert (oder installiert) wurde.
update_time_utc	TIMESTAMP WITH TIME ZONE	Diese Spalte enthält den Zeitpunkt in UTC, zu dem das Objekt zum letzten Mal geändert (oder installiert) wurde.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (object_id)

FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE
FULL

FOREIGN KEY (extenv_id) REFERENCES SYS.ISYSEXTERNENV (object_id)

FOREIGN KEY (owner) REFERENCES SYS.ISYSUSER (user_id)

UNIQUE INDEX (name)
```

Siehe auch

- „Unterstützung für externe Umgebungen in SQL Anywhere“ [[SQL Anywhere Server - Programmierung](#)]

SYSEXTERNLOGIN-Systemansicht

Jede Zeile in der SYSEXTERNLOGIN-Systemansicht beschreibt ein externes Login für Ferndatenzugriff. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSEXTERNLOGIN.

Hinweis

Frühere Versionen dieses Katalogs enthielten eine SYSEXTERNLOGINS-Systemtabelle. Diese Tabelle wurde zu ISYSEXTERNLOGIN (ohne "S") umbenannt und ist die Basistabelle für diese Ansicht.

Spaltenname	Datentyp	Beschreibung
user_id	UNSIGNED INT	Die Benutzer-ID in der lokalen Datenbank
srvid	UNSIGNED INT	Der entfernte Server, wie in der SYSSERVERS-Systemansicht aufgelistet
remote_login	VARCHAR(128)	Der Login-Name für den Benutzer für den entfernten Server

Spaltenname	Datentyp	Beschreibung
remote_password	VARBINARY(128)	Das Kennwort des entfernten Servers für den Benutzer

Hinweis

Bei Datenbanken, die ab SQL Anywhere 16 erstellt werden, ist die zugrunde liegende Systemtabelle für diese Ansicht immer verschlüsselt, um die Daten vor nicht autorisiertem Zugriff zu schützen.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (user_id, srvid)

FOREIGN KEY (user_id) REFERENCES SYS.ISYSUSER (user_id)

FOREIGN KEY (srvid) REFERENCES SYS.ISYSSERVER (srvid)
```

SYSFKEY-Systemansicht

Jede Zeile in der SYSFKEY-Systemansicht beschreibt eine Fremdschlüssel-Integritätsregel im System. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSFKEY.

Spaltenname	Datentyp	Beschreibung
foreign_table_id	UNSIGNED INT	Die Tabellennummer der Fremdtabelle
foreign_index_id	UNSIGNED INT	Die Indexnummer für den Fremdschlüssel
primary_table_id	UNSIGNED INT	Die Tabellennummer der Primärtabelle
primary_index_id	UNSIGNED INT	Die Indexnummer des Primärschlüssels

Spaltenname	Datentyp	Beschreibung
match_type	TINYINT	<p>Die übereinstimmende Typ für die Integritätsregel. Übereinstimmende Typen sind:</p> <ul style="list-style-type: none"> • 0 Standardübereinstimmung verwenden • 1 SIMPLE • 2 FULL • 129 SIMPLE UNIQUE • 130 FULL UNIQUE <p>Weitere Hinweise zu Übereinstimmungstypen finden Sie unter der MATCH-Klausel der „CREATE TABLE-Anweisung“ auf Seite 737.</p>
check_on_commit	CHAR(1)	Zeigt an, ob INSERT und UPDATE-Anweisungen bis zum COMMIT warten sollen, um zu prüfen, ob Fremdschlüssel weiterhin gültig sind.
nulls	CHAR(1)	Zeigt an, ob die Spalten im Fremdschlüssel NULL enthalten dürfen. Diese Einstellung ist unabhängig von der Einstellung für NULL in den Spalten, die im Fremdschlüssel enthalten sind.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (foreign_table_id, foreign_index_id)
```

```
FOREIGN KEY (foreign_table_id, foreign_index_id) REFERENCES SYS.ISYSIDX  
(table_id, index_id)
```

```
FOREIGN KEY (primary_table_id, primary_index_id) REFERENCES SYS.ISYSIDX  
(table_id, index_id)
```

SYSHISTORY-Systemansicht

Jede Zeile in der SYSHISTORY-Systemansicht protokolliert einen Systemvorgang in der Datenbank, wie z.B. einen Datenbankstart, eine Datenbankkalibrierung usw. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSHISTORY.

Spaltenname	Datentyp	Beschreibung
operation	CHAR(128)	<p>Der Typ des Vorgangs, der in der Datenbankdatei durchgeführt wird. Es gilt einer der folgenden Werte:</p> <ul style="list-style-type: none"> • INIT Angaben über den Zeitpunkt der Datenbankerstellung. • UPGRADE Angaben über den Zeitpunkt des Datenbank-Upgrades. • START Angaben über den Zeitpunkt des Datenbankstarts mit Hilfe einer bestimmten Version des Datenbankservers auf einem bestimmten Betriebssystem. • LAST_START Angaben über den Zeitpunkt, an dem der Datenbankserver zum letzten Mal gestartet wurde. Ein LAST_START-Vorgang wird in einen START-Vorgang umgewandelt, wenn Sie die Datenbank mit einer anderen Version des Datenbankservers bzw. auf einem anderen Betriebssystem als den in der LAST_START-Zeile gespeicherten Werte starten. • DTT Informationen über den <i>vorletzten</i> auf dem DBSpace durchgeführten Disk Transfer Time-Kalibrationsvorgang (DTT). Das sind Informationen über die vorletzte Ausführung einer ALTER DATABASE CALIBRATE- oder einer ALTER DATABASE RESTORE DEFAULT CALIBRATION-Anweisung. • LAST_DTT Informationen über den <i>zuletzt</i> auf dem DBSpace durchgeführten DTT-Kalibrationsvorgang. Das sind Informationen über die letzte Ausführung einer ALTER DATABASE CALIBRATE- oder einer ALTER DATABASE RESTORE DEFAULT CALIBRATION-Anweisung. • LAST_BACKUP Informationen über die letzte Sicherung, einschließlich Datum und Uhrzeit der Sicherung, den Sicherungstyp, die gesicherten Dateien und die Version des Datenbankservers, der die Sicherung durchgeführt hat.
object_id	UNSIGNED INT	Für jeden Vorgang außer DTT und LAST_DTT ist der Wert in der Spalte 0. Für DTT- und LAST_DTT-Vorgänge ist dies die dbspace_id des DBSpace gemäß der Definition in der SYSDBSPACE-Systemansicht.

Spaltenname	Datentyp	Beschreibung
sub_operation	CHAR(128)	Bei allen Vorgängen außer DTT und LAST_DTT ist der Wert in dieser Spalte ein Satz von leeren Apostrophen ("). Bei DTT- und LAST_DTT-Vorgängen enthält diese Spalte den Typ des auf dem DBSpace durchgeführten untergeordneten Vorgangs. Werte sind: <ul style="list-style-type: none"> • DTT_SET Die DBSpace-Kalibrierung wurde gesetzt. • DTT_UNSET Die DBSpace-Kalibrierung wurde auf die Standardeinstellung zurückgesetzt.
version	CHAR(128)	Die Versions- und die Build-Nummer des Datenbankservers, der den Vorgang ausführt.
platform	CHAR(128)	Das Betriebssystem, auf dem der Vorgang ausgeführt wurde.
first_time	TIME-STAMP	Der Zeitpunkt in Ortszeit (Datum und Uhrzeit), zu dem die Datenbank erstmals unter einem bestimmten Betriebssystem mit einer bestimmten Version der Software gestartet wurde.
last_time	TIME-STAMP	Der Zeitpunkt in Ortszeit (Datum und Uhrzeit), zu dem die Datenbank zuletzt unter einem bestimmten Betriebssystem mit einer bestimmten Version der Software gestartet wurde.
details	LONG VARCHAR	Diese Spalte enthält Angaben über Eingabeaufforderungsoptionen, mit denen der Datenbankserver gestartet wird, oder über die Fähigkeitsbits der Datenbank. Diese Informationen sind für die Verwendung durch den technischen Support bestimmt.
first_time_utc	TIME-STAMP WITH TIME ZONE	Der Zeitpunkt in UTC (Datum und Uhrzeit), zu dem die Datenbank erstmals unter einem bestimmten Betriebssystem mit einer bestimmten Version der Software gestartet wurde.
last_time_utc	TIME-STAMP WITH TIME ZONE	Der Zeitpunkt in UTC (Datum und Uhrzeit), zu dem die Datenbank zuletzt unter einem bestimmten Betriebssystem mit einer bestimmten Version der Software gestartet wurde.

Integritätsregeln für die zugrunde liegende Systemtabelle

`PRIMARY KEY (operation, object_id, version, platform)`

Siehe auch

- „SYSDBSPACE-Systemansicht“ auf Seite 1444

SYSIDX-Systemansicht

Jede Zeile in der SYSIDX-Systemtabelle definiert einen logischen Index in der Datenbank. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSIDX.

Spaltenname	Datentyp	Beschreibung
table_id	UNSIGNED INT	Identifiziert eindeutig die Tabelle, auf die der Index angewendet wird
index_id	UNSIGNED INT	Eine eindeutige Nummer, die den Index innerhalb seiner Tabelle kennzeichnet
object_id	UNSIGNED BIGINT	Die interne ID für den Index, die ihn in der Datenbank eindeutig kennzeichnet
phys_index_id	UNSIGNED INT	Identifiziert den zugrunde liegenden physischen Index, der zum Implementieren des logischen Indexes verwendet wird. Dieser Wert ist bei Indizes in temporären oder entfernten Tabellen NULL. Ansonsten entspricht der Wert der object_id eines physischen Indexes in der SYSPHYSIDX-Systemansicht.
dbspace_id	SMALLINT	Die ID der Datei, in der der Index enthalten ist. Dieser Wert entspricht einem Eintrag in der SYSDBSPACE-Systemansicht.
index_category	TINYINT	Der Indextyp. Werte sind: <ul style="list-style-type: none"> • 1 Primärschlüssel • 2 Fremdschlüssel • 3 Sekundärer Index (umfasst Eindeutigkeits-Integritätsregeln) • 4 Textindizes
unique	TINYINT	Gibt an, ob der Index ein eindeutiger Index (1) ist, eine Eindeutigkeits-Integritätsregel (2), reserviert (3), ein nicht eindeutiger Index (4) oder ein eindeutiger Index mit WITH NULLS NOT DISTINCT-Klausel. Ein eindeutiger Index verhindert, dass zwei Zeilen in der indizierten Tabelle dieselben Werte in der Index-Spalte haben.
index_name	CHAR(128)	Der Name des Indexes.
not_enforced	CHAR(1)	Wird nur vom System verwendet

Spaltenname	Datentyp	Beschreibung
file_id	SMALLINT	DEPRECATED (nicht mehr empfohlen). Diese Spalte ist in SYS-VIEW vorhanden, nicht aber in der zugrunde liegenden Systemtabelle ISYSIDX. Der Inhalt dieser Spalte ist derselbe wie in dbspace_id und wird aus Kompatibilitätsgründen bereitgestellt. Verwenden Sie stattdessen dbspace_id.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (table_id, index_id)
```

```
FOREIGN KEY (table_id) REFERENCES SYS.ISYSTAB (table_id)
```

```
FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE FULL
```

```
FOREIGN KEY (table_id, phys_index_id) REFERENCES SYS.ISYSPHYSIDX (table_id, phys_index_id)
```

```
UNIQUE INDEX (index_name, table_id, index_category)
```

Siehe auch

- „SYSIDXCOL-Systemansicht“ auf Seite 1457
- „SYSPHYSIDX-Systemansicht“ auf Seite 1468
- „SYSDBSPACE-Systemansicht“ auf Seite 1444

SYSIDXCOL-Systemansicht

Jede Zeile in der SYSIDXCOL-Systemansicht beschreibt eine Spalte eines Indexes in der SYSIDX-Systemansicht. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSIDXCOL.

Spaltenname	Datentyp	Beschreibung
table_id	UNSIGNED INT	Identifiziert die Tabelle, auf die der Index angewendet wird
index_id	UNSIGNED INT	Kennzeichnet den Index, auf den sich die Spalte bezieht. Gemeinsam kennzeichnen table_id und index_id einen Index, der in der SYSIDX-Systemansicht beschrieben wird.
sequence	SMALLINT	Jeder Spalte in einem Index ist eine eindeutige Nummer, beginnend mit "0", zugewiesen. Die Reihenfolge dieser Nummern bestimmt die relative Wichtigkeit der Spalten im Index. Die wichtigste Spalte hat die Sequenznummer "0".
column_id	UNSIGNED INT	Kennzeichnet, welche Spalte der Tabelle indiziert ist. Gemeinsam kennzeichnen table_id und column_id eine Spalte, die in der SYSCOLUMN-Systemansicht beschrieben wird.

Spaltenname	Datentyp	Beschreibung
order	CHAR(1)	Zeigt an, ob die Spalte im Index in aufsteigender (A) oder absteigender (D) Reihenfolge gehalten wird. Dieser Wert ist für Textindizes NULL.
primary_column_id	UNSIGNED INT	Die ID der Primärschlüsselspalte, die dieser Fremdschlüsselspalte entspricht. Der Wert ist NULL bei Nicht-Fremdschlüsselspalten.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (table_id, index_id, column_id)
```

```
FOREIGN KEY (table_id, index_id) REFERENCES SYS.ISYSIDX (table_id, index_id)
```

```
FOREIGN KEY (table_id, column_id) REFERENCES SYS.ISYSTABCOL (table_id, column_id)
```

Siehe auch

- „SYSIDX-Systemansicht“ auf Seite 1456

SYSJAR-Systemansicht

Jede Zeile in der SYSJAR-Systemansicht definiert eine in der Datenbank gespeicherte JAR-Datei. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSJAR.

Spaltenname	Datentyp	Beschreibung
jar_id	INTEGER	Eine eindeutige Nummer, die die JAR-Datei kennzeichnet
object_id	UNSIGNED BIGINT	Die interne ID für die JAR-Datei, die sie in der Datenbank eindeutig kennzeichnet
creator	UNSIGNED INT	Die Benutzernummer des Erstellers der JAR-Datei. Kann durch die AS USER-Klausel der INSTALL JAVA-Anweisung festgelegt werden.
jar_name	LONG VARCHAR	Der Name der JAR-Datei
jar_file	LONG VARCHAR	Diese Spalte wird nicht mehr verwendet und enthält NULL.
update_time	TIMESTAMP	Der Zeitpunkt in Ortszeit, zu dem die JAR-Datei zuletzt aktualisiert wurde

Spaltenname	Datentyp	Beschreibung
update_time_utc	TIMESTAMP WITH TIME ZONE	Der Zeitpunkt in UTC, zu dem die JAR-Datei zuletzt aktualisiert wurde

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (jar_id)
```

```
FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE  
FULL
```

```
UNIQUE INDEX (jar_name)
```

Siehe auch

- „SYSJARCOMPONENT-Systemansicht“ auf Seite 1459
- „SYSJAVACLASS-Systemansicht“ auf Seite 1460

SYSJARCOMPONENT-Systemansicht

Jede Zeile in der SYSJARCOMPONENT-Systemansicht definiert eine JAR-Dateikomponente. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSJARCOMPONENT.

Spaltenname	Datentyp	Beschreibung
component_id	INTEGER	Der Primärschlüssel, der die Identifizierung der Komponente enthält
jar_id	INTEGER	Ein Feld, das die ID-Nummer der JAR-Datei enthält.
component_name	LONG VAR- CHAR	Der Name der Komponente
component_type	CHAR(1)	Diese Spalte wird nicht mehr verwendet und enthält NULL.
contents	LONG BINARY	Der Byte-Code der JAR-Datei

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (component_id)
```

```
FOREIGN KEY (jar_id) REFERENCES SYS.ISYSJAR (jar_id)
```

Siehe auch

- „SYSJAR-Systemansicht“ auf Seite 1458
- „SYSJAVACLASS-Systemansicht“ auf Seite 1460

SYSJAVACLASS-Systemansicht

Jede Zeile in der SYSJAVACLASS-Systemansicht beschreibt eine in der Datenbank gespeicherte Java-Klasse. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSJAVACLASS.

Spaltenname	Datentyp	Beschreibung
class_id	INTEGER	Die eindeutige Nummer für die Java-Klasse. Gleichzeitig der Primärschlüssel der Tabelle
object_id	UNSIGNED BIGINT	Die interne ID für die Java-Klasse, die sie in der Datenbank eindeutig kennzeichnet
creator	UNSIGNED INT	Die Benutzernummer des Erstellers der Klasse. Kann durch die AS USER-Klausel der INSTALL JAVA-Anweisung festgelegt werden.
jar_id	INTEGER	Die ID der JAR-Datei, von der die Klasse stammt
class_name	LONG VAR-CHAR	Der Name der Java-Klasse
public	CHAR(1)	Zeigt an, ob die Klasse öffentlich (Y) oder privat (N) ist
component_id	INTEGER	Die ID der Komponente in der SYSJARCOMPONENT-Systemansicht
update_time	TIMESTAMP	Die letzte Aktualisierungszeit der Klasse in Ortszeit.
update_time_utc	TIMESTAMP WITH TIME ZONE	Die letzte Aktualisierungszeit der Klasse in UTC.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (class_id)

FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE
FULL

FOREIGN KEY (creator) REFERENCES SYS.ISYSUSER (user_id)

FOREIGN KEY (component_id) REFERENCES SYS.ISYSJARCOMPONENT (component_id)
```

Siehe auch

- „SYSJAR-Systemansicht“ auf Seite 1458
- „SYSJARCOMPONENT-Systemansicht“ auf Seite 1459

SYSLDAPSERVER-Systemansicht

Die SYSLDAPSERVER-Systemansicht enthält eine Zeile für jedes in der Datenbank konfigurierte LDAP-Serverkonfigurationsobjekt. Ein LDAP-Serverkonfigurationsobjekt enthält die Konfigurationsinformationen, die benötigt werden, um eine Verbindung mit einem LDAP-Server außerhalb von SQL Anywhere herzustellen. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSLDAPSERVER.

Spaltenname	Datentyp	Beschreibung
ldsrv_id	UNSIGNED BIGINT	Ein eindeutiger Bezeichner für den LDAP-Server. Diese ID wird von der Login-Richtlinie verwendet, um diesen LDAP-Server zu referenzieren. Diese Spalte ist ein Fremdschlüssel für ISYBJECT.
ldsrv_name	CHAR(128)	Der Name des LDAP-Servers.
ldsrv_state	CHAR(9)	Der Zustand des LDAP-Servers. Gültige Werte: <ul style="list-style-type: none"> • RESET • READY • ACTIVE • FAILED • SUSPENDED
ldsrv_start_tls	TINYINT	Gültige Werte: 1 (ON) und 0 (OFF). Beim Wert ON wird LDAP über TLS verwendet, um eine Verbindung mit dem LDAP-Server herzustellen. Dieses Protokoll ermöglicht verschlüsselte Kommunikation für Verbindungen und Suchvorgänge mit dem LDAP-Server.
ldsrv_num_retries	TINYINT	Die Anzahl an Versuchen einer Authentifizierung mit dem LDAP-Server, nach der ein Fehler oder Failover (falls angegeben) zurückgegeben wird. Gültiger Bereich: 1-60.
ldsrv_timeout	UNSIGNED INT	Der Timeoutwert für Verbindungen oder Suchvorgänge in Millisekunden. Gültiger Bereich: 1-3600000 (1 Stunde).
ldsrv_last_state_change	TIMESTAMP	Der Zeitpunkt der letzten Zustandsänderung. Der Wert wird unabhängig von der lokalen Zeitzone des Servers in Coordinated Universal Time (UTC) gespeichert.
ldsrv_search_url	CHAR(1024)	Die LDAP-URL, die die Suche nach dem Distinguished Name (DN) für einen Benutzer auf Basis der Benutzer-ID definiert.

Spaltenname	Datentyp	Beschreibung
ldsrv_auth_url	CHAR(1024)	Die LDAP-Suchzeichenfolge, mit der unter der angegebenen Benutzer-ID nach dem DN eines Benutzers gesucht wird.
ldsrv_access_dn	CHAR(1024)	Der DN, der für den Zugriff auf den LDAP-Server verwendet wird, um mithilfe von Suchvorgängen DNs für andere Benutzer-IDs abzurufen.
ldsrv_access_dn_pwd	VARBINARY(1024)	Das Kennwort für das Zugriffskonto ein. Das Kennwort wird beim Speichern auf der Festplatte symmetrisch verschlüsselt.

Siehe auch

- „trusted_certificates_file-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

SYSLOGINMAP-Systemansicht

Die SYSLOGINMAP-Systemansicht enthält eine Zeile für jeden Benutzer, der sich mit der Datenbank unter Verwendung eines integrierten Logins oder eines Kerberos-Logins verbinden kann. Deshalb ist der Zugriff auf diese Ansicht eingeschränkt. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSLOGINMAP.

Spaltenname	Datentyp	Beschreibung
login_mode	TINYINT	Der Typ des Logins: 1 für integrierte Logins, 2 für Kerberos-Logins
login_id	VARCHAR(1024)	Entweder der Benutzerprofilname des integrierten Logins oder der Kerberos-Prinzipal, der database_uid zugeordnet ist
object_id	UNSIGNED BIGINT	Ein eindeutiger Bezeichner, einer für jede Entsprechung zwischen Benutzer-ID und Datenbankbenutzer-ID
database_uid	UNSIGNED INT	Die Datenbankbenutzer-ID, der die Login-ID zugeordnet wird

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (login_mode, login_id)
```

```
FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE FULL
```

```
FOREIGN KEY (database_uid) REFERENCES SYS.ISYSUSER (user_id)
```

SYSLOGINPOLICY-Systemansicht

Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSLOGINPOLICY.

Spaltenname	Datentyp	Beschreibung
login_policy_id	UNSIGNED BIGINT	Ein eindeutiger Bezeichner für die Login-Richtlinie.
login_policy_name	CHAR(128)	Der Name der Login-Richtlinie

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (login_policy_id)
```

```
FOREIGN KEY (login_policy_id) REFERENCES SYS.ISYSOBJECT (object_id)
```

```
UNIQUE INDEX (login_policy_name)
```

Siehe auch

- „SYSLOGINPOLICYOPTION-Systemansicht“ auf Seite 1463
- „SYSUSER-Systemansicht“ auf Seite 1506

SYSLOGINPOLICYOPTION-Systemansicht

Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSLOGINPOLICYOPTION.

Spaltenname	Datentyp	Beschreibung
login_policy_id	UNSIGNED BIGINT	Ein eindeutiger Bezeichner für die Login-Richtlinie.
login_option_name	CHAR(128)	Der Name der Login-Richtlinie
login_option_value	LONG VARCHAR	Der Wert der Login-Richtlinie zum Zeitpunkt ihrer Erstellung.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (login_policy_id, login_option_name)
```

```
FOREIGN KEY (login_policy_id) REFERENCES SYS.ISYSLOGINPOLICY  
(login_policy_id)
```

Siehe auch

- „SYSLOGINPOLICY-Systemansicht“ auf Seite 1463
- „SYSUSER-Systemansicht“ auf Seite 1506

SYSMIRROROPTION-Systemansicht

Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSMIRROROPTION.

Spaltenname	Datentyp	Beschreibung
option_name	CHAR(128)	Der Name der Option
option_value	LONG VAR-CHAR	Der Wert der Option, als der Spiegel erstellt wurde. Werte in dieser Spalte werden für Benutzer verborgen, die nicht das SELECT ANY TABLE-Systemprivileg haben.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (option_name)
```

Siehe auch

- „SYSMIRRORSERVER-Systemansicht“ auf Seite 1464
- „SYSMIRRORSERVEROPTION-Systemansicht“ auf Seite 1465
- „SET MIRROR OPTION-Anweisung“ auf Seite 1034

SYSMIRRORSERVER-Systemansicht

Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSMIRRORSERVER.

Spaltenname	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Ein eindeutiger Bezeichner für den Spiegelserver.
server_name	CHAR(128)	Der Name des Servers.
server_type	CHAR(20)	Der Servertyp. Der Wert kann einer der folgenden sein: PRIMARY, MIRROR, ARBITER, PARTNER oder COPY.
parent	UNSIGNED BIGINT	Der übergeordnete Server. Wenn der Wert Null ist, handelt es sich um den Primär- oder Spiegelserver in einem Datenbankspiegelungssystem. Wenn diese Spalte einen Wert enthält, ist es die ID des Servers, der dem aktuellen Server übergeordnet ist.
alternate_parent	UNSIGNED BIGINT	Die ID des Servers, der als alternativer übergeordneter Server verwendet wird, wenn der aktuelle übergeordnete Server ausfällt.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (object_id)
```

```
FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id)
```

```
UNIQUE INDEX (server_name)
```

Siehe auch

- „[SYSMIRROROPTION-Systemansicht](#)“ auf Seite 1463
- „[SYSMIRRORSERVEROPTION-Systemansicht](#)“ auf Seite 1465
- „Ermitteln von übergeordneten Knoten für Kopieknoten“ [*SQL Anywhere Server - Datenbankadministration*]
- „[CREATE MIRROR SERVER-Anweisung](#)“ auf Seite 656

SYSMIRRORSERVEROPTION-Systemansicht

Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSMIRRORSERVEROPTION.

Spaltenname	Datentyp	Beschreibung
server_id	UNSIGNED BIGINT	Ein eindeutiger Bezeichner für den Spiegelserver.
option_name	CHAR(128)	Der Name der Option
option_value	LONG VARCHAR	Der Wert der Option, als der Spiegel erstellt wurde.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (server_id, option_name)
```

```
FOREIGN KEY (server_id) references SYS.ISYSMIRRORSERVER (object_id)
```

Siehe auch

- „[SYSMIRROROPTION-Systemansicht](#)“ auf Seite 1463
- „[SYSMIRRORSERVER-Systemansicht](#)“ auf Seite 1464

SYSMVOPTION-Systemansicht

Jede Zeile in der SYSMVOPTION-Systemansicht beschreibt die Einstellung eines einzelnen Optionswerts für eine materialisierte Ansicht zum Zeitpunkt ihrer Erstellung. Den Namen für die Option finden Sie in der Systemansicht SYSMVOPTIONNAME. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSMVOPTION.

Spaltenname	Datentyp	Beschreibung
view_object_id	UNSIGNED BIGINT	Die Objekt-ID der materialisierten Ansicht
option_id	UNSIGNED INT	Eine eindeutige Nummer, die die Option in der Datenbank kennzeichnet. Um den Optionsnamen zu erhalten, verwenden Sie die SYSMVOPTIONNAME-Systemansicht.
option_value	LONG VARCHAR	Der Wert der Option, als die materialisierte Ansicht erstellt wurde.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (view_object_id, option_id)

FOREIGN KEY (view_object_id) REFERENCES SYS.ISYSOBJECT (object_id)

FOREIGN KEY (option_id) REFERENCES SYS.ISYSMVOPTIONNAME (option_id)
```

Siehe auch

- „[SYSMVOPTIONNAME-Systemansicht](#)“ auf Seite 1466
- „[Begriffe und Einstellungen für Textindizes anzeigen \(Sybase Central\)](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „[Fortgeschrittene Aufgaben: Anzeigen von Informationen aus materialisierten Ansichten im Katalog](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

SYSMVOPTIONNAME-Systemansicht

Jede Zeile in der SYSMVOPTION-Systemansicht enthält den Namen eines Optionswertes für eine materialisierte Ansicht oder einen Textindex zum Zeitpunkt der Erstellung. Den Wert für die Option finden Sie in der Systemansicht SYSMVOPTION. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSMVOPTIONNAME.

Spaltenname	Datentyp	Beschreibung
option_id	UNSIGNED INT	Eine Nummer, die die Option in der Datenbank eindeutig kennzeichnet
option_name	CHAR(128)	Der Name der Option

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (option_id)

UNIQUE INDEX (option_name)
```

Siehe auch

- „[SYSMVOPTION-Systemansicht](#)“ auf Seite 1465
- „[Begriffe und Einstellungen für Textindizes anzeigen \(Sybase Central\)](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „[Fortgeschrittene Aufgaben: Anzeigen von Informationen aus materialisierten Ansichten im Katalog](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

SYSOBJECT-Systemansicht

Jede Zeile in der SYSOBJECT-Systemansicht beschreibt ein Datenbankobjekt. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSOBJECT.

Spaltenname	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Die interne ID für das Objekt, die es in der Datenbank eindeutig kennzeichnet
status	TINYINT	Der Status des Objekts. Werte sind: <ul style="list-style-type: none"> • 1 (gültig) Das Objekt steht dem Datenbankserver zur Verfügung. Dieser Status ist mit ENABLED synonym. D.h., wenn Sie ein Objekt mit ENABLE aktivieren, ändert sich der Status zu VALID. • 2 (ungültig) Der Versuch, das Objekt nach einem internen Vorgang zu rekompilieren, ist fehlgeschlagen, z.B. nach einer das Schema ändernden Modifizierung eines Objekts, von dem es abhängt. Der Datenbankserver versucht weiterhin, das Objekt zu rekompilieren, sobald es in einer Anweisung referenziert wird. • 4 (deaktiviert) Das Objekt wurde explizit vom Benutzer deaktiviert, beispielsweise durch die Verwendung einer ALTER TABLE...DISABLE VIEW DEPENDENCIES-Anweisung.
object_type	TINYINT	Typ des Objekts.
creation_time	TIMESTAMP	Der Zeitpunkt in Ortszeit (Datum und Uhrzeit), zu dem das Objekt erstellt wurde.
object_type_str	CHAR(128)	Typ des Objekts.
creation_time_utc	TIMESTAMP WITH TIME ZONE	Der Zeitpunkt in UTC (Datum und Uhrzeit), zu dem das Objekt erstellt wurde.

Integritätsregeln für die zugrunde liegende Systemtabelle

PRIMARY KEY (object_id)

SYSOPTION-Systemansicht

Die SYSOPTION-Systemansicht enthält eine Zeile für jede in der Datenbank gespeicherte Optionseinstellung. Jeder Benutzer kann seine eigenen Einstellungen für eine gegebene Option haben. Außerdem definieren Einstellungen für den PUBLIC-Benutzer die Standardeinstellungen für Benutzer, die keine eigenen Einstellungen haben. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSOPTION.

Spaltenname	Datentyp	Beschreibung
user_id	UNSIGNED INT	Die Benutzernummer, auf die sich die Optionseinstellung bezieht
option	CHAR(128)	Der Name der Option
setting	LONG VARCHAR	Die aktuelle Einstellung für die Option

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (user_id, "option")
```

```
FOREIGN KEY (user_id) REFERENCES SYS.ISYSUSER (user_id)
```

SYSOPTSTAT-Systemansicht

Die SYSOPTSTAT-Systemansicht speichert Informationen über die Kostenmodellkalibrierung, wie sie von der ALTER DATABASE CALIBRATE-Anwendung berechnet werden. Der Inhalt dieser Ansicht dient nur der internen Verwendung und wird am besten mit der sa_get_dtt-Systemprozedur abgerufen. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSOPTSTAT.

Spaltenname	Datentyp	Beschreibung
stat_id	UNSIGNED INT	Wird nur vom System verwendet
group_id	UNSIGNED INT	Wird nur vom System verwendet
format_id	SMALLINT	Wird nur vom System verwendet
data	LONG BINARY	Wird nur vom System verwendet

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (stat_id, group_id, format_id)
```

SYSPHYSIDX-Systemansicht

Jede Zeile in der SYSPHYSIDX-Systemansicht definiert einen physischen Index in der Datenbank. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSPHYSIDX.

Spaltenname	Datentyp	Beschreibung
table_id	UNSIGNED INT	Die Objekt-ID der Tabelle, auf die sich der Index bezieht
phys_index_id	UNSIGNED INT	Eine eindeutige Nummer des physischen Indexes innerhalb seiner Tabelle

Spaltenname	Datentyp	Beschreibung
root	INTEGER	Kennzeichnet den Speicherort der Stammseite des physischen Indexes in der Datenbankdatei
key_value_count	UNSIGNED INT	Die Anzahl der unterschiedlichen Schlüsselwerte im Index
leaf_page_count	UNSIGNED INT	Die Anzahl der Index-Blattseiten
depth	UNSIGNED SMALLINT	Die Tiefe (Anzahl der Ebenen) des physischen Indexes
max_key_distance	UNSIGNED INT	Wird nur vom System verwendet
seq_transitions	UNSIGNED INT	Wird nur vom System verwendet
rand_transitions	UNSIGNED INT	Wird nur vom System verwendet
rand_distance	UNSIGNED INT	Wird nur vom System verwendet
allocation_bitmap	LONG VARBIT	Wird nur vom System verwendet
long_value_bitmap	LONG VARBIT	Wird nur vom System verwendet

Integritätsregeln für die zugrunde liegende Systemtabelle

`PRIMARY KEY (table_id, phys_index_id)`

Siehe auch

- „SYSIDX-Systemansicht“ auf Seite 1456
- „SYSIDXCOL-Systemansicht“ auf Seite 1457

SYSPROCEDURE-Systemansicht

Jede Zeile in der SYSPROCEDURE-Systemansicht beschreibt eine Prozedur in der Datenbank. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSPROCEDURE.

Spaltenname	Datentyp	Beschreibung
proc_id	UNSIGNED INT	Jeder Prozedur wird eine eindeutige Nummer (Prozedurnummer) zugeordnet.
creator	UNSIGNED INT	Eigentümer der Prozedur
object_id	UNSIGNED BIGINT	Die interne ID für die Prozedur, die sie in der Datenbank eindeutig kennzeichnet

Spaltenname	Datentyp	Beschreibung
proc_name	CHAR(128)	Der Name der Prozedur. Ein Ersteller kann keine zwei Prozeduren mit demselben Namen haben.
proc_defn	LONG VAR-CHAR	Der Definition der Prozedur
remarks	LONG VAR-CHAR	Bemerkungen zu dieser Prozedur. Dieser Wert wird in der ISYSREMARK-Systemtabelle gespeichert.
replicate	CHAR(1)	Diese Eigenschaft ist nur für den internen Gebrauch gedacht.
srvid	UNSIGNED INT	Zeigt den entfernten Server an, wenn die Prozedur ein Stellvertreter (Proxy) für eine Prozedur auf einem entfernten Datenbankserver ist.
source	LONG VAR-CHAR	Die beibehaltene Quelle für die Prozedur. Dieser Wert wird in der ISYSSOURCE-Systemtabelle gespeichert.
avg_num_rows	FLOAT	Für die Abfrageoptimierung gesammelte Informationen, wenn die Prozedur in der FROM-Klausel erscheint.
avg_cost	FLOAT	Für die Abfrageoptimierung gesammelte Informationen, wenn die Prozedur in der FROM-Klausel erscheint.
stats	LONG BINARY	Für die Abfrageoptimierung gesammelte Informationen, wenn die Prozedur in der FROM-Klausel erscheint.

Integritätsregeln für die zugrunde liegende Systemtabelle

```

PRIMARY KEY (proc_id)

FOREIGN KEY (srvid) REFERENCES SYS.ISYSSERVER (srvid)

FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE
FULL

FOREIGN KEY (creator) REFERENCES SYS.ISYSUSER (user_id)

UNIQUE INDEX (proc_name, creator)

```

SYSPROCARM-Systemansicht

Jede Zeile in der SYSPROCARM-Systemansicht beschreibt einen Parameter für eine Prozedur in der Datenbank. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSPROCARM.

Spaltenname	Datentyp	Beschreibung
proc_id	UNSIGNED INT	Identifiziert eindeutig die Prozedur, zu der der Parameter gehört
parm_id	SMALLINT	Jede Prozedur beginnt die Nummerierung ihrer Parameter mit 1. Die Reihenfolge der Parameter entspricht der Reihenfolge, in der sie definiert wurden. Bei Funktionen hat der erste Parameter den Namen der Funktion und stellt den Rückgabewert für die Funktion dar.
parm_type	SMALLINT	Es gibt folgende Parametertypen: <ul style="list-style-type: none"> • 0 Normaler Parameter (Variable) • 1 Ergebnisvariable - wird mit einer Prozedur, die Ergebnismengen zurückgibt, verwendet • 2 SQLSTATE-Fehlerwert • 3 SQLCODE-Fehlerwert • 4 Rückgabewert der Funktion
parm_mode_in	CHAR(1)	Zeigt an, ob der Parameter der Prozedur einen Wert übergibt (IN- oder INOUT-Parameter)
parm_mode_out	CHAR(1)	Zeigt an, ob der Parameter einen Wert von der Prozedur (OUT- oder INOUT-Parameter) oder Spalten in der RESULT-Klausel zurückgibt.
domain_id	SMALLINT	Identifiziert den Datentyp des Parameters durch die Datentypennummer, die in der SYSDOMAIN-Systemansicht aufgelistet ist.
width	BIGINT	Enthält die Länge eines Zeichenfolgenparameters, die Genauigkeit numerischer Parameter oder die Anzahl der Byte zum Speichern eines anderen Datentyps.
scale	SMALLINT	Bei numerischen Datentypen die Anzahl der Ziffern hinter dem Dezimalzeichen. Bei allen anderen Datentypen ist der Wert dieser Spalte "1".
user_type	SMALLINT	Der Benutzertyp des Parameters (falls anwendbar)
parm_name	CHAR(128)	Der Name des Prozedurparameters
default	LONG VARCHAR	Der Standardwert des Parameters. Dient nur zur Information

Spaltenname	Datentyp	Beschreibung
remarks	LONG VAR-CHAR	Gibt immer NULL zurück. Wird bereitgestellt, damit frühere Versionen von ODBC-Treibern mit neueren Personal Datenbankservern funktionieren.
base_type_str	VAR-CHAR(32767)	Die zugehörige Typzeichenfolge, die den physischen Parametertyp darstellt.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (proc_id, parm_id)
```

```
FOREIGN KEY (proc_id) REFERENCES SYS.ISYSPROCEDURE (proc_id)
```

```
FOREIGN KEY (domain_id) REFERENCES SYS.ISYSDOMAIN (domain_id)
```

```
FOREIGN KEY (user_type) REFERENCES SYS.ISYSUSERTYPE (type_id)
```

SYSPROCPerm-Systemansicht

Jede Zeile der SYSPROCPerm-Systemansicht beschreibt einen Benutzer, dem das EXECUTE-Privileg für eine Prozedur erteilt wurde. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSPROCPERM.

Spaltenname	Datentyp	Beschreibung
proc_id	UNSIGNED INT	Die Prozedurnummer identifiziert eindeutig die Prozedur, für die das EXECUTE-Privileg erteilt wurde.
grantee	UNSIGNED INT	Die Benutzernummer des Berechtigungsempfängers.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (proc_id, grantee)
```

```
FOREIGN KEY (grantee) REFERENCES SYS.ISYSUSER (user_id)
```

```
FOREIGN KEY (proc_id) REFERENCES SYS.ISYSPROCEDURE (proc_id)
```

SYSPROXYTAB-Systemansicht

Jede Zeile der SYSPROXYTAB-Systemansicht beschreibt die Fernparameter einer Proxytabelle. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSPROXYTAB.

Spaltenname	Datentyp	Beschreibung
table_object_id	UNSIGNED BIGINT	Die Objekt-ID der Proxytabelle

Spaltenname	Datentyp	Beschreibung
existing_obj	CHAR(1)	Gibt an, ob die Proxy-Tabelle zuvor auf dem Fremdserver vorhanden war.
srvid	UNSIGNED INT	Die eindeutige ID für den Fernserver, der mit der Proxytabelle verknüpft ist
remote_location	LONG VAR-CHAR	Der Standort der Proxytabelle auf dem entfernten Server

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (table_object_id)
```

```
FOREIGN KEY (table_object_id) REFERENCES ISYSOBJECT (object_id) MATCH UNIQUE FULL
```

```
FOREIGN KEY (srvid) REFERENCES SYS.ISYSSERVER (srvid)
```

SYSPUBLICATION-Systemansicht

Jede Zeile in der SYSPUBLICATION-Systemansicht beschreibt eine SQL Remote- oder MobiLink-Publikation. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSPUBLICATION.

Spaltenname	Datentyp	Beschreibung
publication_id	UNSIGNED INT	Eine Nummer, die die Publikation eindeutig kennzeichnet
object_id	UNSIGNED BIGINT	Die interne ID für die Publikation, die sie in der Datenbank eindeutig kennzeichnet
creator	UNSIGNED INT	Der Eigentümer der Publikation
publication_name	CHAR(128)	Der Name der Publikation
remarks	LONG VAR-CHAR	Bemerkungen zu der Publikation. Dieser Wert wird in der ISYSREMARK-Systemtabelle gespeichert.
type	CHAR(1)	Diese Spalte wird nicht mehr empfohlen.

Spaltenname	Datentyp	Beschreibung
sync_type	UNSIGNED INT	<p>Der Typ von Synchronisation für die Publikation. Werte sind:</p> <ul style="list-style-type: none"> • 0 (Logscan) Dies ist eine reguläre Publikation, die das Transaktionslog verwendet, um alle relevanten Daten heraufzuladen, die sich seit dem letzten Upload geändert haben. • 1 (skriptgesteuerter Upload) Bei dieser Publikation wird das Transaktionslog ignoriert und der Upload wird durch den Benutzer mittels gespeicherter Prozeduren festgelegt. Informationen über gespeicherte Prozeduren werden in der ISYSSYNCSRIPT-Systemtabelle gespeichert. • 2 (nur Download) Dies ist ausschließlich eine Download-Publikation. Es werden keine Daten heraufgeladen.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (publication_id)

FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE FULL

FOREIGN KEY (creator) REFERENCES SYS.ISYSUSER (user_id)

UNIQUE INDEX (publication_name, creator)
```

Siehe auch

- „Skriptgesteuerter Upload“ [[MobiLink - Clientadministration](#)]
- „SYSSYNCSRIPT-Systemansicht“ auf Seite 1491

SYSREMARK-Systemansicht

Jede Zeile in der SYSREMARK-Systemansicht beschreibt eine Bemerkung (oder Kommentar) für ein Objekt. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISISREMARK.

Spalte	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Die interne ID für das Objekt, das eine zugeordnete Bemerkung hat
remarks	LONG VARCHAR	Die Bemerkung oder der Kommentar, der dem Objekt zugeordnet ist

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (object_id)
```

```
FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE
FULL
```

SYSREMOTEOPTION-Systemansicht

Jede Zeile in der SYSREMOTEOPTION-Systemansicht beschreibt den Wert eines SQL Remote-Nachrichtenverbindungsparameters. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSREMOTEOPTION.

Einige Spalten in dieser Tabelle enthalten potenziell sensible Daten. Die SYSREMOTEOPTION2-Ansicht bietet öffentlichen Zugriff auf die Daten in dieser Ansicht, mit Ausnahme der potenziell sensiblen Spalten.

Spalte	Datentyp	Beschreibung
option_id	UNSIGNED INT	Eine Kennung für den Nachrichtenverbindungsparameter
user_id	UNSIGNED INT	Die Benutzer-ID, für die der Parameter festgelegt ist
setting	VARCHAR(255)	Der Wert des Nachrichtenverbindungsparameters

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (option_id, user_id)

FOREIGN KEY (option_id) REFERENCES SYS.ISYSREMOTEOPTIONTYPE (option_id)

FOREIGN KEY (user_id) REFERENCES SYS.ISYSUSER (user_id)
```

SYSREMOTEOPTIONTYPE-Systemansicht

Jede Zeile in der SYSREMOTEOPTIONTYPE-Systemansicht beschreibt einen der SQL Remote-Nachrichtenverbindungsparameter. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSREMOTEOPTIONTYPE.

Spalte	Datentyp	Beschreibung
option_id	UNSIGNED INT	Eine Kennung für den Nachrichtenverbindungsparameter
type_id	SMALLINT	Eine Kennung für den Nachrichtentyp, der den Parameter verwendet
option	VARCHAR(128)	Der Name des Nachrichtenverbindungsparameters

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (option_id)

FOREIGN KEY (type_id) REFERENCES SYS.ISYSREMOTETTYPE (type_id)
```

SYSREMOTETYPE-Systemansicht

Die SYSREMOTETYPE-Systemansicht enthält Informationen zu SQL Remote. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSREMOTETYPE.

Spaltenname	Datentyp	Beschreibung
type_id	SMALLINT	Gibt an, welches der von SQL Remote unterstützten Nachrichtensysteme zum Senden von Nachrichten an den Benutzer verwendet werden soll.
object_id	UNSIGNED BIGINT	Die interne ID für den entfernten Typ, die ihn in der Datenbank eindeutig kennzeichnet
type_name	CHAR(128)	Der Name des Nachrichtensystems, das von SQL Remote unterstützt wird.
publisher_address	LONG VARCHAR	Die Adresse des externen Datenbank-Publizierers
remarks	LONG VARCHAR	Bemerkungen über den entfernten Typ. Dieser Wert wird in der ISYSREMARK-Systemtabelle gespeichert.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (type_id)
```

```
FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE FULL
```

```
UNIQUE INDEX (type_name)
```

SYSREMOTEUSER-Systemansicht

Jede Zeile in der SYSREMOTEUSER-Systemansicht beschreibt eine Benutzer-ID mit REMOTE-Systemprivileg (einen Subskribenten) zusammen mit dem Status von SQL Remote-Nachrichten, die an diesen Benutzer und von diesem Benutzer gesendet wurden. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSREMOTEUSER.

Spaltenname	Datentyp	Beschreibung
user_id	UNSIGNED INT	Die Benutzernummer des Benutzers mit REMOTE-Privileg.
consolidate	CHAR(1)	Zeigt an, ob dem Benutzer das CONSOLIDATE-Privileg (Y) oder das REMOTE-Privileg (N) erteilt wurde.

Spaltenname	Datentyp	Beschreibung
type_id	SMALLINT	Legt fest, welches der von SQL Remote unterstützten Nachrichtensysteme zum Senden von Nachrichten an den Benutzer verwendet wird
address	LONG VARCHAR	Die Adresse, an die SQL Remote-Nachrichten gesendet werden sollen. Die Adresse muss dem address_type entsprechen.
frequency	CHAR(1)	Wie oft SQL Remote-Nachrichten gesendet werden
send_time	TIME	Der nächste Zeitpunkt, zu dem Nachrichten an diesen Benutzer zu versenden sind
log_send	UNSIGNED BIGINT	Nachrichten werden nur an Subskribenten versendet, bei denen log_send größer als log_sent ist.
time_sent	TIMESTAMP	Der Zeitpunkt in Ortszeit, zu dem die letzte Nachricht an diesen Subskribenten gesendet wurde
log_sent	UNSIGNED BIGINT	Der Log-Offset für den zuletzt gesendeten Vorgang
confirm_sent	UNSIGNED BIGINT	Der Log-Offset für den zuletzt bestätigten Vorgang von diesem Subskribenten
send_count	INTEGER	Anzahl der gesendeten SQL Remote-Nachrichten
resend_count	INTEGER	Zähler, der sicherstellt, dass Nachrichten nur einmal auf die Subskribenten-Datenbank angewendet werden
time_received	TIMESTAMP	Der Zeitpunkt in Ortszeit, zu dem die letzte Nachricht von diesem Subskribenten empfangen wurde
log_received	UNSIGNED BIGINT	Der Log-Offset in der Datenbank des Subskribenten für den Vorgang, der zuletzt in der aktuellen Datenbank empfangen wurde.
confirm_received	UNSIGNED BIGINT	Der Log-Offset in der Datenbank des Subskribenten für den letzten Vorgang, für den eine Bestätigungsnachricht gesendet wurde
receive_count	INTEGER	Anzahl der empfangenen Nachrichten
rereceive_count	INTEGER	Zähler, der sicherstellt, dass Nachrichten nur einmal auf die aktuelle Datenbank angewendet werden

Spaltenname	Datentyp	Beschreibung
time_sent_utc	TIMESTAMP WITH TIME ZONE	Der Zeitpunkt in UTC, zu dem die letzte Nachricht an diesen Subskribenten gesendet wurde
time_received_utc	TIMESTAMP WITH TIME ZONE	Der Zeitpunkt in UTC, zu dem die letzte Nachricht von diesem Subskribenten empfangen wurde

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (user_id)

FOREIGN KEY (user_id) REFERENCES SYS.ISYSUSER (user_id)

FOREIGN KEY (type_id) REFERENCES SYS.ISYSREMOType (type_id)

UNIQUE INDEX (type_id, address)
```

SYSROLEGRANT-Systemansicht

Die SYSROLEGRANT-Systemansicht speichert Informationen zu Rollenmitgliedschaft und Typ der Mitgliedschaft. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSROLEGRANT.

Spaltenname	Datentyp	Beschreibung
grant_id	UNSIGNED INT	ID für die einzelnen GRANT-Anweisungen.
role_id	UNSIGNED INT	ID der erteilten Rolle, gemäß ISYSUSER.
grantee	UNSIGNED INT	ID des Benutzers, dem die Rolle erteilt wird, gemäß ISYSUSER.

Spaltenname	Datentyp	Beschreibung
grant_type	TINYINT	<p>Beschreibt den Typ der Erteilung in 3 Ziffern. Das erste Bit von rechts gibt an, ob das Privileg erteilt wurde. Die zweite Ziffer gibt an, ob Administrationsrechte erteilt wurden. Die dritte Ziffer gibt an, ob Systemprivilegien vererbbar sind.</p> <ul style="list-style-type: none"> • 001 Privileg erteilt, ohne Vererbung und Administrationsrechte. Gilt nur für veraltete nicht vererbbare Berechtigungen außer DBA und REMOTE DBA. • 101 Privileg erteilt, mit Vererbung, aber ohne Administrationsrechte. • 110 Nur Administrationsrechte wurden erteilt. • 111 Privileg erteilt, mit Vererbung und Administrationsrechten.
grant_scope	TINYINT	<p>Wird von SET USER und CHANGE PASSWORD verwendet, um den Bereich der Erteilung festzulegen. Dies kann einer oder mehrere der folgenden Werte sein:</p> <ul style="list-style-type: none"> • 1 ANY • 2 Benutzerliste • 4 Rollenliste
grantor	CHAR(128)	Der Name des Berechtigungsgebers.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (grant_id)
```

```
FOREIGN KEY (role_id) REFERENCES SYS.ISYSUSER (user_id)
```

```
FOREIGN KEY (grantee) REFERENCES SYS.ISYSUSER (user_id)
```

```
UNIQUE INDEX (role_id, grantee, grant_scope)
```

SYSROLEGRANTEXT-Systemansicht

Wenn Sie die Systemprivilegien SET USER und CHANGE PASSWORD erteilen, können Sie eine Liste von Benutzern oder Rollen angeben, denen der Berechtigungsempfänger diese Privilegien erteilen kann. Die SYSROLEGRANTEXT-Systemansicht speichert Informationen dazu, welchen Benutzern und Rollen der Berechtigungsempfänger die Systemprivilegien SET USER und CHANGE PASSWORD erteilen kann.

Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSROLEGRANTEXT.

Spaltenname	Datentyp	Beschreibung
grant_id	UNSIGNED INT	ID für die einzelnen GRANT-Anweisungen.
user_id	UNSIGNED INT	Die Benutzer-IDs, die in <i>Benutzerliste</i> oder <i>Rollenliste</i> für eine bestimmte erweiterte GRANT-Anweisung angegeben wurden.

Bemerkungen

Wenn Sie die Systemprivilegien SET USER und CHANGE PASSWORD erteilen oder entziehen, entweder mit der *Benutzerliste*-Option oder mit der Option ANY WITH ROLES *Rollenliste*, wird diese Ansicht mit den Werten aus der erweiterten Syntax aktualisiert.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (grant_id, user_id)

FKKEY( grant_id ) REFERENCES ISYSROLEGRANT( grant_id )

FKKEY( user_id) REFERENCES ISYSUSER( user_id )
```

Konsolidierte Ansicht SYSROLEGRANTS

Die SYSROLEGRANTS-Systemansicht speichert Informationen zu Rollenmitgliedschaft und Typ der Mitgliedschaft, genauso wie die SYSROLEGRANT-Systemansicht. Die SYSROLEGRANTS enthält jedoch auch die Namen von Rollen und Berechtigungsempfängern (nicht nur IDs). Die zugrunde liegenden Systemtabellen für diese Ansicht sind ISYSROLEGRANT und ISYSUSER.

Spaltenname	Datentyp	Beschreibung
grant_id	UNSIGNED INT	ID für die einzelnen GRANT-Anweisungen.
role_id	UNSIGNED INT	ID der erteilten Rolle, gemäß ISYSUSER.
role_name	CHAR(128)	Der Name der Rolle.
grantee	UNSIGNED INT	ID des Benutzers, dem die Rolle erteilt wird, gemäß ISYSUSER.
grantee_name	CHAR(128)	Der Name des Berechtigungsempfängers.

Spaltenname	Datentyp	Beschreibung
grant_type	TINYINT	<p>Beschreibt den Typ der Erteilung in 3 Bits. Das erste Bit von rechts gibt an, ob das Privileg erteilt wurde. Die zweite Ziffer gibt an, ob Administrationsrechte erteilt wurden. Die dritte Ziffer gibt an, ob Systemprivilegien vererbbar sind.</p> <ul style="list-style-type: none"> • 001 Privileg erteilt, ohne Vererbung und Administrationsrechte. Gilt nur für veraltete nicht vererbare Berechtigungen außer DBA und REMOTE DBA. • 101 Privileg erteilt, mit Vererbung, aber ohne Administrationsrechte. • 110 Nur Administrationsrechte wurden erteilt. • 111 Privileg erteilt, mit Vererbung und Administrationsrechten.
grant_scope	TINYINT	<p>Wird von SET USER und CHANGE PASSWORD verwendet, um den Bereich der Erteilung festzulegen. Dies kann einer oder mehrere der folgenden Werte sein:</p> <ul style="list-style-type: none"> • 1 ANY • 2 Benutzerliste • 4 Rollenliste
grantor	CHAR(128)	Der Name des Berechtigungsgebers.

Integritätsregeln für die zugrunde liegende Systemtabelle

```

PRIMARY KEY (grant_id)

FOREIGN KEY (role_id) REFERENCES SYS.ISYSUSER (user_id)

FOREIGN KEY (grantee) REFERENCES SYS.ISYSUSER (user_id)

INDEX (role_id, grantee, grant_scope)

```

SYSSCHEDULE-Systemansicht

Jede Zeile in der SYSSCHEDULE-Systemansicht gibt einen Zeitpunkt an, zu dem ein Ereignis ausgelöst werden soll, wie in der SCHEDULE-Klausel von CREATE EVENT festgelegt. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSSCHEDULE.

Spaltenname	Datentyp	Beschreibung
event_id	UNSIGNED INT	Die jedem Ereignis zugeordnete eindeutige Nummer
sched_name	VAR-CHAR(128)	Der Name, der dem Zeitplan für das Ereignis zugeordnet ist
recurring	TINYINT	Zeigt an, wenn ein Plan wiederholt wird
start_time	TIME	Die Planstartzeit
stop_time	TIME	Die Planstopzeit, wenn BETWEEN verwendet wurde
start_date	DATE	Das Datum der ersten geplanten Ausführung des Ereignisses
days_of_week	TINYINT	Eine Bitmaske, die die Wochentage angibt, zu denen ein Ereignis geplant ist: <ul style="list-style-type: none"> • x01 = Sonntag • x02 = Montag • x04 = Dienstag • x08 = Mittwoch • x10 = Donnerstag • x20 = Freitag • x40 = Samstag
days_of_month	UNSIGNED INT	Eine Bitmaske, die die Tage des Monats angibt, zu denen ein Ereignis geplant ist. Einige Beispiele: <ul style="list-style-type: none"> • x01 = erster Tag • x02 = zweiter Tag • x40000000 = 31. Tag • x80000000 = letzter Tag des Monats
interval_units	CHAR(10)	Die Intervalleinheit, angegeben durch EVERY: <ul style="list-style-type: none"> • HH = Stunden • NN = Minuten • SS = Sekunden
interval_amt	INTEGER	Die durch EVERY definierte Zeitspanne

Integritätsregeln für die zugrunde liegende Systemtabelle

PRIMARY KEY (event_id, sched_name)

FOREIGN KEY (event_id) REFERENCES SYS.ISYSEVENT (event_id)

SYSSEQUENCE-Systemansicht

Die SYSSEQUENCE-Systemansicht enthält eine Zeile für jede benutzerdefinierte Sequenz. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSSEQUENCE.

Spaltenname	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Die jeder Sequenz zugeordnete eindeutige Nummer.
owner	UNSIGNED INT	Der Eigentümer der Sequenz.
min_value	BIGINT	Der zulässige Mindestwert für die Sequenz.
max_value	BIGINT	Der zulässige Höchstwert für die Sequenz.
increment_by	BIGINT	Der Inkrementwert für die Sequenz.
start_with	BIGINT	Der Anfangswert für die Sequenz.
cache	UNSIGNED INT	Die Anzahl der Sequenzwerte, die für schnellere Zugriff im Speicher vorab zugewiesen werden sollen. Ein Wert von 0 bedeutet, dass Werte nicht vorab zugewiesen werden sollen.
cycle	TINYINT	Legt fest, ob die Werte weiter generiert werden sollen, nachdem der Höchst- oder Mindestwert erreicht wurde.
resume_at	BIGINT	Der durch die ALTER SEQUENCE-Anweisung angegebene RESTART WITH-Wert. Der Wert ist NULL, wenn keine ALTER RESTART WITH-Anweisung ausgeführt wurde.
sequence_name	CHAR(128)	Der Name der Sequenz.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (object_id)

FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT MATCH UNIQUE FULL

FOREIGN KEY (owner) REFERENCES SYS.ISYSUSER (user_id)
```

SYSSEQUENCEPERM-Systemansicht

Die SYSSEQUENCEPERM-Systemansicht enthält die Privilegien, die Benutzer oder Gruppen für Sequenzen haben. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSSEQUENCEPERM.

Spaltenname	Datentyp	Beschreibung
sequence_id	UNSIGNED BIGINT	Die jeder Sequenz zugeordnete eindeutige Nummer.
grantee	UNSIGNED INT	Die ID des Benutzers oder der Gruppe mit Privilegien, die Sequenz zu ändern oder zu löschen.
grantor	UNSIGNED INT	Die ID des Benutzers, der die Privilegien für die Sequenz erteilt hat.
privilege_type	SMALLINT	Der Typ der Privilegien, die dem Benutzer oder der Gruppe für die Sequenz erteilt wurden.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (sequence_id, grantee, privilege_type)

FOREIGN KEY (sequence_id) REFERENCES SYS.ISYSEQUENCE (object_id)

FOREIGN KEY (grantee) REFERENCES SYS.ISYSUSER (user_id)

FOREIGN KEY (grantor) REFERENCES SYS.ISYSUSER (user_id)
```

SYSSERVER-Systemansicht

Jede Zeile in der SYSSERVER-Systemansicht beschreibt einen entfernten Server. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSSERVER.

Hinweis

Frühere Versionen des Katalogs enthielten eine SYSSERVERS-Systemtabelle. Diese Tabelle wurde zu ISYSSERVER (ohne "S") umbenannt und ist die Basistabelle für diese Ansicht.

Spaltenname	Datentyp	Beschreibung
srvid	UNSIGNED INT	Ein Bezeichner für den entfernten Server
srvname	VARCHAR(128)	Der Name des Fremdservers
srvclass	LONG VARCHAR	Die Serverklasse, die in der Anweisung CREATE SERVER angegeben wurde
srvinfo	LONG VARCHAR	Serverinformationen
srvreadonly	CHAR(1)	Zeigt an, ob der Server schreibgeschützt ist.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (srvid)
```

SYSSOURCE-Systemansicht

Jede Zeile in der SYSSOURCE-Systemansicht enthält den Quellcode für ein Objekt, das in der SYSOBJECT-Systemansicht aufgelistet ist (falls anwendbar). Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSSOURCE.

Spaltenname	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Die interne ID für das Objekt, dessen Quellcode definiert wird
source	LONG VARCHAR	Diese Spalte enthält den ursprünglichen Quellcode für das Objekt, wenn die Datenbankoption preserve_source_format auf "ON" war, als das Objekt erstellt wurde.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (object_id)
```

```
FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE FULL
```

Siehe auch

- „preserve_source_format-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

SYSSPATIALREFERENCESYSTEM-Systemansicht

Jede Zeile der SYSSPATIALREFERENCESYSTEM-Systemansicht beschreibt ein räumliches Bezugssystem, das in der Datenbank festgelegt ist. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSSPATIALREFERENCESYSTEM.

Diese Ansicht bietet eine etwas andere Menge an Informationen als ST_SPATIAL_REFERENCE_SYSTEMS.

Spaltenname	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Wird nur vom System verwendet
owner	UNSIGNED INT	Eigentümer des räumlichen Bezugssystems
srs_name	CHAR(128)	Der Name des räumlichen Bezugssystems.
srs_id	INTEGER	Der numerische Bezeichner (SRID) für das räumliche Bezugssystem.

Spaltenname	Datentyp	Beschreibung
round_earth	CHAR(1)	Zeigt an, ob der SRS-Typ ROUND EARTH (Y) oder PLANAR (N) ist.
axis_order	CHAR(12)	Beschreibt, wie der Datenbankserver Punkte hinsichtlich Breiten- und Längengrad interpretiert (z.B. bei Verwendung der Methoden ST_Lat und ST_Long). Für nicht geografische räumliche Bezugssysteme gilt die Achsenreihenfolge x/y/z/m. Für geografische räumliche Bezugssysteme lautet die Achsenreihenfolge standardmäßig Länge/Breite/z/m. Breite/Länge/z/m wird ebenfalls unterstützt.
snap_to_grid	DOUBLE	Legt die Größe des Rasters fest, das SQL Anywhere beim Ausführen von Berechnungen verwendet.
tolerance	DOUBLE	Legt die Genauigkeit fest, die beim Vergleichen von Punkten verwendet werden soll.
semi_major_axis	DOUBLE	Der Abstand zwischen der Mitte des Ellipsoids und dem Äquator für ein räumliches Bezugssystem mit gewölbter Erddarstellung.
semi_minor_axis	DOUBLE	Der Abstand zwischen der Mitte des Ellipsoids und den Polen für ein räumliches Bezugssystem mit gewölbter Erddarstellung.
inv_flattening	DOUBLE	Die inverse Abplattung des Ellipsoids in einem räumlichen Bezugssystem mit gewölbter Erddarstellung. Inverse Abplattung (f) ist ein mathematischer Wert, der festlegt, wie weit die Pole eines Spheroids in Richtung des Äquators gedrückt werden. Dies reicht von keiner Abplattung (einem perfekten Kreis) bis hin zur vollständigen Abplattung (einer geraden Linie). Die inverse Abflachung ist der Wert von 1/f, wie folgt: $1/f = (\text{semi_major_axis}) / (\text{semi_major_axis} - \text{semi_minor_axis})$
min_x	DOUBLE	Der minimale in Koordinaten zulässige x-Wert.
max_x	DOUBLE	Der maximale in Koordinaten zulässige x-Wert.
min_y	DOUBLE	Der minimale in Koordinaten zulässige y-Wert.
max_y	DOUBLE	Der maximale in Koordinaten zulässige y-Wert.
min_z	DOUBLE	Der minimale in Koordinaten zulässige z-Wert.
max_z	DOUBLE	Der maximale in Koordinaten zulässige z-Wert.

Spaltenname	Datentyp	Beschreibung
min_m	DOUBLE	Der minimale in Koordinaten zulässige m-Wert.
max_m	DOUBLE	Der maximale in Koordinaten zulässige m-Wert.
organization	LONG VAR-CHAR	Der Name der Organisation, die das von dem räumlichen Bezugssystem verwendete Koordinatensystem erstellt hat.
organizati-on_coordsys_id	INTEGER	Die ID, die das Koordinatensystem von der erstellenden Organisation erhalten hat.
srs_type	CHAR(11)	<p>Der SRS-Typ gemäß der Definition im SQL/MM-Standard. Der Wert kann einer der folgenden sein:</p> <ul style="list-style-type: none"> • GEOGRAPHIC Dies ist für räumliche Bezugssysteme, die auf georeferenzierten Koordinatensystemen mit den Achsen Breitengrad, Längengrad (und Höhe) basieren. Diese räumlichen Bezugssysteme sind vom Typ PLANAR oder ROUND EARTH. • PROJECTED Dies ist für räumliche Bezugssysteme, die auf georeferenzierten Koordinatensystemen ohne die Achsen Breitengrad und Längengrad basieren. Diese räumlichen Bezugssysteme sind vom Typ PLANAR. • ENGINEERING Dies ist für räumliche Bezugssysteme, die auf nicht georeferenzierten Koordinatensystemen basieren. Diese räumlichen Bezugssysteme sind vom Typ PLANAR. • GEOCENTRIC Nicht unterstützt. • COMPOUND Nicht unterstützt. • VERTICAL Nicht unterstützt. <p>Wenn srs_type leer ist, gilt der Typ als nicht angegeben.</p>
linear_unit_of_measure	UNSIGNED BIGINT	Die von dem räumlichen Bezugssystem verwendete lineare Maßeinheit.
angular_unit_of_measure	UNSIGNED BIGINT	Die von dem räumlichen Bezugssystem verwendete Winkelmaßeinheit.
count_in_use	UNSIGNED BIGINT	Wird nur intern verwendet.

Spaltenname	Datentyp	Beschreibung
polygon_format	LONG VAR-CHAR	Die Ausrichtung der Ringe in einem Polygon. Mögliche Werte lauten CounterClockwise, ClockWise und EvenOdd.
storage_format	LONG VAR-CHAR	Legt fest, ob die Daten in normalisiertem Format (Internal), in nicht normalisiertem Format (Original) oder in beiden (Mixed) gespeichert werden.
definition	LONG VAR-CHAR	Die WKT-Definition des räumlichen Bezugssystems in dem durch den OGC-Standard definierten Format.
transform_definition	LONG VAR-CHAR	Transformationsdefinitionseinstellungen, die beim Transformieren von Daten aus diesem räumlichen Bezugssystem in ein anderes verwendet werden.

Integritätsregeln für die zugrunde liegende Systemtabelle

```

PRIMARY KEY (object_id)

FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id)

FOREIGN KEY (linear_unit_of_measure) REFERENCES SYS.ISYSUNITOFMEASURE
(object_id)

FOREIGN KEY (angular_unit_of_measure) REFERENCES SYS.ISYSUNITOFMEASURE
(object_id)

FOREIGN KEY (owner) REFERENCES SYS.ISYSUSER (user_id)

UNIQUE CONSTRAINT (srs_name)

UNIQUE CONSTRAINT (srs_id)

```

Siehe auch

- „Konsolidierte Ansicht ST_SPATIAL_REFERENCE_SYSTEMS“ auf Seite 1515
- „CREATE SPATIAL REFERENCE SYSTEM-Anweisung“ auf Seite 719

SYSSQLSERVERTYPE-Systemansicht

Die SYSSQLSERVERTYPE-Systemansicht enthält Informationen, die sich auf die Kompatibilität zu Adaptive Server Enterprise beziehen. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSSQLSERVERTYPE.

Spaltenname	Datentyp	Beschreibung
ss_user_type	SMALLINT	Der Adaptive Server Enterprise-Benutzertyp
ss_domain_id	SMALLINT	Die Adaptive Server Enterprise-Domänen-ID

Spaltenname	Datentyp	Beschreibung
ss_type_name	VARCHAR(30)	Der Adaptive Server Enterprise-Typenname
primary_sa_domain_id	SMALLINT	Die entsprechende primäre SQL Anywhere-Domänen-ID
primary_sa_user_type	SMALLINT	Der entsprechende primäre SQL Anywhere-Benutzertyp

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (ss_user_type)
```

SYSSUBSCRIPTION-Systemansicht

Jede Zeile in der SYSSUBSCRIPTION-Systemansicht beschreibt eine Subskription einer Benutzer-ID (die das REMOTE-Systemprivileg haben muss) für eine Publikation. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSSUBSCRIPTION.

Spaltenname	Datentyp	Beschreibung
publication_id	UNSIGNED INT	Die Kennung für die Publikation, bei der die Benutzer-ID subskribiert ist
user_id	UNSIGNED INT	Die ID des Benutzers, der die Publikation subskribiert hat
subscribe_by	CHAR(128)	Der Wert des Ausdrucks "SUBSCRIBE BY" für die Subskription, falls vorhanden
created	UNSIGNED BIGINT	Der Offset im Transaktionslog, an dem die Subskription erstellt wurde
started	UNSIGNED BIGINT	Der Offset im Transaktionslog, an dem die Subskription gestartet wurde

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (publication_id, user_id, subscribe_by)
```

```
FOREIGN KEY (publication_id) REFERENCES SYS.ISYSPUBLICATION (publication_id)
```

```
FOREIGN KEY (user_id) REFERENCES SYS.ISYSUSER (user_id)
```

SYSSYNC-Systemansicht

Die SYSSYNC-Systemansicht enthält Informationen, die sich auf die MobiLink-Synchronisation beziehen. Einige Spalten in dieser Tabelle enthalten potenziell sensible Daten. Deshalb ist der Zugriff auf diese Ansicht eingeschränkt. Die SYSSYNC2-Ansicht bietet öffentlichen Zugriff auf die Daten in dieser Ansicht, ausgenommen die potenziell sensiblen Spalten. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSSYNC.

Spaltenname	Datentyp	Beschreibung
sync_id	UNSIGNED INT	Eine die Zeile eindeutig kennzeichnende Nummer
type	CHAR(1)	Dieser Wert ist stets D.
publication_id	UNSIGNED INT	Eine publication_id, die in der SYSPUBLICATION-Systemansicht zu finden ist
progress	UNSIGNED BIGINT	Das Logoffset des letzten erfolgreichen Uploads
site_name	CHAR(128)	Ein MobiLink-Benutzername
option	LONG VARCHAR	Synchronisationsoptionen
server_connect	LONG VARCHAR	Die Adresse oder URL des MobiLink-Servers
server_conn_type	LONG VARCHAR	Das Kommunikationsprotokoll, wie TCP/IP, für die Synchronisation
last_download_time	TIMESTAMP	Zeigt an, wann zum letzten Mal ein Download-Datenstrom vom MobiLink-Server empfangen wurde
last_upload_time	TIMESTAMP	Zeigt an, wann diese Daten zum letzten Mal (erfasst am MobiLink-Server) erfolgreich per Upload übertragen wurden. Standardwert ist jan-1-1900.
created	UNSIGNED BIGINT	Der Offset im Transaktionslog, an dem die Subskription erstellt wurde
log_sent	UNSIGNED BIGINT	Der Log-Fortschritt, bis zu dem Informationen per Upload übertragen wurden. Für die Aktualisierung des Eintrags in dieser Spalte ist es nicht erforderlich, dass eine Bestätigung für den Upload eingeht.
generation_number	INTEGER	Für Downloads auf Dateibasis ist dies die letzte Generationsnummer für diese Subskription. Standardwert ist "0".
extended_state	VARCHAR(1024)	Wird nur intern verwendet.
script_version	CHAR(128)	Zeigt die Skriptversion an, die von den CREATE- und ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisungen und von der START SYNCHRONIZATION SCHEMA CHANGE-Anweisung verwendet wird.
subscription_name	CHAR(128)	Der Name der Subskription.

Spaltenname	Datentyp	Beschreibung
server_protocol	UNSIGNED BIGINT	Wird nur intern verwendet. Enthält einen Wert, der intern verwendet wird, um die Version des MobiLink-Servers zu identifizieren.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (sync_id)
```

```
FOREIGN KEY (publication_id) REFERENCES SYS.ISYSPUBLICATION (publication_id)
```

```
UNIQUE INDEX (publication_id, site_name)
```

```
UNIQUE INDEX (subscription_name)
```

SYSSYNCPROFILE-Systemansicht

Die SYSSYNCPROFILE-Systemansicht enthält Informationen, die sich auf Synchronisationsprofile für MobiLink beziehen.

Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSSYNCPROFILE.

Spaltenname	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Die Objekt-ID des Synchronisationsprofils.
profile_name	CHAR(128)	Der Name des Synchronisationsprofils.
profile_defn	LONG VARCHAR	Die Definition für das Synchronisationsprofil, das Sie erstellen möchten.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (object_id)
```

```
FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id)
```

```
UNIQUE INDEX (profile_name)
```

SYSSYNCSRIPT-Systemansicht

Jede Zeile in der SYSSYNCSRIPT-Systemansicht identifiziert eine gespeicherte Prozedur für den skriptgesteuerten MobiLink-Upload. Diese Ansicht ist fast identisch mit der SYSSYNCSRIPTS-Ansicht, nur dass die Werte in dieser Ansicht in ihrem unverarbeiteten Format sind.

Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSSYNCSRIPT.

Spaltenname	Datentyp	Beschreibung
pub_object_id	UNSIGNED BIGINT	Die Objekt-ID der Publikation, zu der das Skript gehört
table_object_id	UNSIGNED BIGINT	Die Objekt-ID der Tabelle, für die das Skript anwendbar ist
type	UNSIGNED INT	Der Typ der Upload-Prozedur
proc_object_id	UNSIGNED BIGINT	Die Objekt-ID der für die Publikation verwendeten gespeicherten Prozedur

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (pub_object_id, table_object_id, type)

FOREIGN KEY (pub_object_id) REFERENCES SYS.ISYSOBJECT (object_id)

FOREIGN KEY (table_object_id) REFERENCES SYS.ISYSOBJECT (object_id)

FOREIGN KEY (proc_object_id) REFERENCES SYS.ISYSOBJECT (object_id)
```

Siehe auch

- „Skriptgesteuerter Upload“ [[MobiLink - Clientadministration](#)]
- „Konsolidierte Ansicht SYSSYNCSCRIPTS“ auf Seite 1530
- „SYSPROCEDURE-Systemansicht“ auf Seite 1469
- „SYSPUBLICATION-Systemansicht“ auf Seite 1473

SYSTAB-Systemansicht

Jede Zeile in der SYSTAB-Systemansicht beschreibt eine Tabelle oder Ansicht in der Datenbank. Zusätzliche Informationen zu Ansichten finden Sie in der SYSVIEW-Systemansicht. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSTAB.

Spaltenname	Datentyp	Beschreibung
table_id	UNSIGNED INT	Jeder Tabelle wird eine eindeutige Nummer (Tabellennummer) zu geordnet
dbspace_id	SMALLINT	Ein Wert, der angibt, welcher DBSpace die Tabelle enthält
count	UNSIGNED BIGINT	Die Anzahl der Zeilen in der Tabelle oder materialisierten Ansicht. Dieser Wert wird bei jedem erfolgreichen Checkpoint aktualisiert. Diese Zahl wird verwendet, um den Datenbankzugriff zu optimieren. Der Wert ist bei einer nicht-materialisierten Ansicht oder entfernten Tabelle immer "0".

Spaltenname	Datentyp	Beschreibung
creator	UNSIGNED INT	Die Benutzernummer des Eigentümers der Tabelle oder der Ansicht
table_page_count	INTEGER	Die Gesamtzahl der von der Basistabelle verwendeten Hauptseiten.
ext_page_count	INTEGER	Die Gesamtzahl der von der Basistabelle verwendeten Erweiterungsseiten.
commit_action	INTEGER	Bei globalen temporären Tabellen gibt "0" an, dass die ON COMMIT PRESERVE ROWS-Klausel angegeben wurde, als die Tabelle erstellt wurde. "1" gibt an, dass die ON COMMIT DELETE ROWS-Klausel angegeben wurde, als die Tabelle erstellt wurde (das Standardverhalten für temporäre Tabellen), und "3" gibt an, dass die NOT TRANSACTIONAL-Klausel angegeben wurde, als die Tabelle erstellt wurde. Bei nicht-temporären Tabellen ist commit_action immer "0".
share_type	INTEGER	Bei globalen temporären Tabellen zeigt "4" an, dass die SHARE BY ALL-Klausel angegeben wurde, als die Tabelle erstellt wurde, und "5" zeigt an, dass die SHARE BY ALL-Klausel <i>nicht</i> angegeben wurde, als die Tabelle erstellt wurde. Bei nicht-temporären Tabellen ist share_type immer "5", weil die SHARE BY ALL-Klausel beim Erstellen von nicht-temporären Tabellen nicht angegeben werden kann.
object_id	UNSIGNED BIGINT	Die Objekt-ID der Tabelle
last_modified_at	TIMESTAMP	Der Zeitpunkt in Ortszeit, zu dem die Daten in der Tabelle zuletzt geändert wurden. Diese Spalte wird nur zur Check-point-Zeit aktualisiert.
table_name	CHAR(128)	Der Name der Tabelle oder der Ansicht. Ein Ersteller kann keine zwei Tabellen oder Ansichten mit demselben Namen haben.

Spaltenname	Datentyp	Beschreibung
table_type	TINYINT	Der Typ der Tabelle oder der Ansicht. Werte sind: <ul style="list-style-type: none"> • 1 Basistabelle • 2 Materialisierte Ansicht • 3 Globale temporäre Tabelle • 4 Lokale temporäre Tabelle • 5 Textindex-Basistabelle • 6 Globale temporäre Tabelle für einen Textindex • 21 Ansicht
replicate	CHAR(1)	Dieser Wert ist nur für den internen Gebrauch gedacht.
server_type	TINYINT	Der Speicherort der Daten für die Basistabelle. Werte sind: <ul style="list-style-type: none"> • 1 Lokaler Server • 3 Fremdserver
tab_page_list	LONG VAR-BIT	Wird nur intern verwendet. Die Gruppe von Seiten, die Informationen für die Tabelle enthalten, als Bitmap ausgedrückt.
ext_page_list	LONG VAR-BIT	Wird nur intern verwendet. Die Gruppe von Seiten, die Seiten mit Zeilenerweiterungen und großen Objekte (LOB) für die Tabelle enthalten, als Bitmap ausgedrückt.
pct_free	UNSIGNED INT	Die PCT_FREE-Spezifikation für die Tabelle (falls angegeben), sonst NULL.
clustered_index_id	UNSIGNED INT	Die ID des Clustered-Indexes für die Tabelle. Wenn es keine Clustered-Indizes gibt, ist dieses Feld NULL.
encrypted	CHAR(1)	Gibt an, ob die Tabelle oder materialisierte Ansicht verschlüsselt ist.
last_modified_tsn	UNSIGNED BIGINT	Eine Sequenznummer, die der Transaktion zugeteilt wurde, die die Tabelle verändert hat. Diese Spalte wird nur zur Check-point-Zeit aktualisiert.
current_schema	UNSIGNED INT	Die aktuelle Schemaversion der Tabelle.

Spaltenname	Datentyp	Beschreibung
file_id	SMALLINT	DEPRECATED (nicht mehr empfohlen). Diese Spalte ist in SYSVIEW vorhanden, nicht aber in der zugrunde liegenden Systemtabelle ISYSTAB. Der Inhalt dieser Spalte ist derselbe wie in dbspace_id und wird aus Kompatibilitätsgründen bereitgestellt. Verwenden Sie stattdessen die Spalte dbspace_id.
table_type_str	CHAR(13)	Lesbarer Wert von table_type. Werte sind: <ul style="list-style-type: none"> • BASE Basistabelle • MAT VIEW Materialisierte Ansicht • GBL TEMP Globale temporäre Tabelle • VIEW Ansicht
last_modified_at_utc	TIMESTAMP WITH TIME ZONE	Der Zeitpunkt in UTC, zu dem die Daten in der Tabelle zuletzt geändert wurden. Diese Spalte wird nur zur Checkpoint-Zeit aktualisiert.

Integritätsregeln für die zugrunde liegende Systemtabelle

```

FOREIGN KEY (dbspace_id) REFERENCES SYS.ISYSDBSPACE (dbspace_id)

FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id)

PRIMARY KEY (table_id)

FOREIGN KEY (creator) REFERENCES SYS.ISYSUSER (user_id)

UNIQUE INDEX (table_name, creator)

```

Siehe auch

- „SYSVIEW-Systemansicht“ auf Seite 1511

SYSTABCOL-Systemansicht

Die SYSTABCOL-Systemansicht enthält eine Zeile für jede Spalte in jeder Tabelle oder Ansicht in der Datenbank. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSTABCOL.

Spaltenname	Datentyp	Beschreibung
table_id	UNSIGNED INT	Die Tabellen-ID der Tabelle oder der Ansicht, zu der die Spalte gehört

Spaltenname	Datentyp	Beschreibung
column_id	UNSIGNED INT	Die ID der Spalte. Für jede Tabelle beginnt die Spaltennummerierung mit 1. Der column_id-Wert legt die Reihenfolge der Spalten in der Ergebnismenge fest, wenn SELECT * verwendet wird. Außerdem bestimmt er die Reihenfolge der Spalten für eine INSERT-Anweisung, wenn keine Liste mit Spaltennamen bereitgestellt wird.
domain_id	SMALLINT	Der Datentyp der Spalte, dargestellt durch eine Datentypnummer, die in der SYSDOMAIN-Systemansicht aufgelistet ist
nulls	CHAR(1)	Gibt an, ob NULL in der Spalte zulässig ist
width	BIGINT	Die Länge einer Zeichenfolgen-Spalte, die Genauigkeit numerischer Spalten oder die Anzahl der Byte zum Speichern eines anderen Datentyps.
scale	SMALLINT	Die Anzahl der Stellen nach dem Dezimalzeichen bei Spalten mit einem NUMERIC- oder DECIMAL-Datentypen. Bei Zeichenfolgenspalten zeigt der Wert "1" Zeichenlängensemantik und "0" Bytelänge-Semantik an.
object_id	UNSIGNED BIGINT	Die Objekt-ID der Tabellenspalte
max_identity	BIGINT	Der höchste Wert der Spalte, wenn es eine AUTOINCREMENT-, IDENTITY- oder GLOBAL AUTOINCREMENT-Spalte ist
column_name	CHAR(128)	Der Name der Spalte.
default	LONG VARCHAR	Der Standardwert für die Spalte. Dieser Wert (falls angegeben) wird nur dann verwendet, wenn eine INSERT-Anweisung keinen Wert für diese Spalte angibt.
user_type	SMALLINT	Der Datentyp, wenn die Spalte unter Verwendung eines benutzerdefinierten Datentyps definiert ist
column_type	CHAR(1)	Der Typ der Spalte (C=berechnete Spalte, R=andere Spalten)
compressed	TINYINT	Gibt an, ob die Spalte in einem komprimierten Format gespeichert ist
collect_stats	TINYINT	Gibt an, ob das System automatisch Statistiken auf dieser Spalte erfasst und aktualisiert

Spaltenname	Datentyp	Beschreibung
inline_max	SMALLINT	Die maximale Anzahl von Bytes eines in einer Zeile zu speichernden BLOB. NULL zeigt an, dass entweder der Standardwert angewendet wurde, oder dass die Spalte nicht vom Zeichen- bzw. Binärdatentyp ist. Ein Nicht-NULL-Wert von inline_max entspricht dem INLINE-Wert, der für die Spalte unter Verwendung der CREATE TABLE- oder ALTER TABLE-Anweisung angegeben wurde.
inline_long	SMALLINT	Die Anzahl der in einer Zeile zu speichernden Duplikat-Bytes eines BLOB, wenn die BLOB-Größe den inline_max-Wert überschreitet. NULL zeigt an, dass entweder der Standardwert angewendet wurde, oder dass die Spalte nicht vom Zeichen- bzw. Binärdatentyp ist. Ein Nicht-NULL-Wert von inline_long entspricht dem PREFIX-Wert, der für die Spalte unter Verwendung der CREATE TABLE- oder ALTER TABLE-Anweisung angegeben wurde.
lob_index	TINYINT	Gibt an, ob Indizes auf BLOB-Werten in der Spalte erstellt werden, die eine interne Schwellenwertgröße (ca. acht Datenbankseiten) überschreiten. NULL zeigt an, dass entweder der Standardwert angewendet wird, oder dass die Spalte kein BLOB-Typ ist. Der Wert "1" zeigt an, dass Indizes erstellt werden. Ein Wert "0" zeigt an, dass keine Indizes erstellt werden. Ein Nicht-NULL-Wert von lob_index zeigt an, ob INDEX oder NO INDEX für die Spalte unter Verwendung der CREATE TABLE- oder ALTER TABLE-Anweisung angegeben wurde.
base_type_str	VAR-CHAR(32767)	Die zugehörige Typzeichenfolge, die den physischen Spaltentyp darstellt.
nonmaterialized_value	LONG BINARY	Nur zur internen Verwendung.
start_schema	UNSIGNED INT	Die erste Version des Tabellenschemas, in der diese Spalte vorhanden ist.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (table_id, column_id)
```

```
FOREIGN KEY (table_id) REFERENCES SYS.ISYSTAB (table_id)
```

```
FOREIGN KEY (domain_id) REFERENCES SYS.ISYSDOMAIN (domain_id)
```

```
FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE FULL
```

```
FOREIGN KEY (user_type) REFERENCES SYS.ISYSUSERTYPE (type_id)
```

Siehe auch

- „CREATE TABLE-Anweisung“ auf Seite 737

SYSTABLEPERM-Systemansicht

Privilegien für Tabellen und Ansichten, die durch die GRANT-Anweisung erteilt wurden, werden in der SYSTABLEPERM-Systemansicht gespeichert. Jede Zeile in dieser Ansicht entspricht einer Tabelle, einer Benutzer-ID, die das Privileg erteilt (Berechtigungsgeber), und einer Benutzer-ID, der die Berechtigung erteilt wird (Berechtigungsempfänger). Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSTABLEPERM.

Spaltenname	Datentyp	Beschreibung
stable_id	UNSIGNED INT	Die Tabellennummer der Tabelle oder Ansicht, für die die Privilegien gelten.
grantee	UNSIGNED INT	Die Benutzernummer der Benutzer-ID, die das Privileg erhält.
grantor	UNSIGNED INT	Die Benutzernummer der Benutzer-ID, die das Privileg erteilt.
selectauth	CHAR(1)	Zeigt an, ob SELECT-Privilegien erteilt wurden. Mögliche Werte sind "Y", "N" oder "G". Im Bereich "Bemerkungen" weiter unten erfahren Sie, was diese Werte bedeuten.
insertauth	CHAR(1)	Zeigt an, ob INSERT-Privilegien erteilt wurden. Mögliche Werte sind "Y", "N" oder "G". Im Bereich "Bemerkungen" weiter unten erfahren Sie, was diese Werte bedeuten.
deleteauth	CHAR(1)	Zeigt an, ob DELETE-Privilegien erteilt wurden. Mögliche Werte sind "Y", "N" oder "G". Im Bereich "Bemerkungen" weiter unten erfahren Sie, was diese Werte bedeuten.
updateauth	CHAR(1)	Zeigt an, ob UPDATE-Privilegien für alle Spalten in der Tabelle erteilt wurden. Mögliche Werte sind "Y", "N" oder "G". Im Bereich "Bemerkungen" weiter unten erfahren Sie, was diese Werte bedeuten.
updatecols	CHAR(1)	Zeigt an, ob UPDATE-Privilegien nur für einige der Spalten in der Basistabelle erteilt wurden. Wenn updatecols den Wert "Y" hat, enthält die SYSCOLPERM-Systemansicht eine oder mehrere Zeilen, die UPDATE-Privilegien für die Spalten erteilen.
alterauth	CHAR(1)	Zeigt an, ob ALTER-Privilegien erteilt wurden. Mögliche Werte sind "Y", "N" oder "G". Im Bereich "Bemerkungen" weiter unten erfahren Sie, was diese Werte bedeuten.

Spaltenname	Datentyp	Beschreibung
referenceauth	CHAR(1)	Zeigt an, ob REFERENCE-Privilegien erteilt wurden. Mögliche Werte sind "Y", "N" oder "G". Im Bereich "Bemerkungen" weiter unten erfahren Sie, was diese Werte bedeuten.
loadauth	CHAR(1)	Zeigt an, ob LOAD-Privilegien erteilt wurden. Mögliche Werte sind "Y", "N" oder "G". Im Bereich "Bemerkungen" weiter unten erfahren Sie, was diese Werte bedeuten.
truncateauth	CHAR(1)	Zeigt an, ob TRUNCATE-Privilegien erteilt wurden. Mögliche Werte sind "Y", "N" oder "G". Im Bereich "Bemerkungen" weiter unten erfahren Sie, was diese Werte bedeuten.

Bemerkungen

Es gibt verschiedene Typen von Privilegien, die erteilt werden können. Jedes Privileg kann einen der folgenden drei Werte haben,

- **N** No: Dem Berechtigungsempfänger wurde dieses Privileg vom Berechtigungsgeber nicht erteilt.
- **Y** Yes: Dem Berechtigungsempfänger wurde dieses Privileg vom Berechtigungsgeber erteilt.
- **G** Dem Berechtigungsempfänger wurde dieses Privileg erteilt und er kann dasselbe Privileg einem anderen Benutzer erteilen.

Hinweis

Möglicherweise wurde dem Berechtigungsempfänger das Privileg für dieselbe Tabelle von einem anderen Berechtigungsgeber erteilt. Wenn ja, wird diese Information in einer anderen Zeile der SYSTABLEPERM-Systemansicht protokolliert.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (stable_id, grantee, grantor)

FOREIGN KEY (stable_id) REFERENCES SYS.ISYSTAB (table_id)

FOREIGN KEY (grantor) REFERENCES SYS.ISYSUSER (user_id)

FOREIGN KEY (grantee) REFERENCES SYS.ISYSUSER (user_id)
```

Siehe auch

- [„GRANT-Anweisung“ auf Seite 881](#)

SYSTEXTCONFIG-Systemansicht

Jede Zeile in der SYSTEXTCONFIG-Systemtabelle beschreibt ein Textkonfigurationsobjekt zur Verwendung mit der Volltextsuchfunktion. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSTEXTCONFIG.

Spaltenname	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Die Objekt-ID für das Textkonfigurationsobjekt.
creator	UNSIGNED INT	Der Ersteller des Textkonfigurationsobjekts.
term_breaker	TINYINT	Der Algorithmus, der zur Trennung einer Zeichenfolge in Begriffe oder Wörter verwendet wird. Die Werte sind 0 für GENERIC und 1 für NGRAM. Mit GENERIC wird jede Zeichenfolge mit einem oder mehr alphanumerischen Zeichen, die durch nicht alphanumerische Zeichen getrennt werden, als Begriff eingestuft. Der NGRAM-Algorithmus ist für angenäherte Übereinstimmungen oder für Dokumente sinnvoll, die keine Leerstellen zum Trennen von Begriffen verwenden.
stemmer	TINYINT	Wird nur intern verwendet.
min_term_length	TINYINT	Die zulässige Mindestlänge eines Begriffes in Zeichen. Begriffe, die kürzer sind als min_term_length, werden ignoriert. Die MINIMUM TERM LENGTH-Einstellung ist nur für den GENERIC-Begriffsegmentierer sinnvoll. Bei NGRAM-Textindizes wird die Einstellung ignoriert.
max_term_length	TINYINT	Bei GENERIC-Textindizes ist dies die maximal zulässige Länge für einen Begriff, ausgedrückt in Zeichen. Begriffe, die länger sind als max_term_length, werden ignoriert. Bei NGRAM-Textindizes ist dies die Länge der N-Gramme, in die Begriffe zerlegt werden.
collation	CHAR(128)	Wird nur intern verwendet.
text_config_name	CHAR(128)	Der Name des Textkonfigurationsobjekts.
prefilter	LONG VAR-CHAR	Der Funktions- und Bibliothekname für eine externe Vorfilter-Bibliothek.
postfilter	LONG VAR-CHAR	Wird nur intern verwendet.
char_stoplist	LONG VAR-CHAR	Begriffe, die bei einer Volltextsuche in CHAR-Spalten zu ignorieren sind. Diese Begriffe werden auch aus Textindizes ausgeschlossen. Diese Spalte wird verwendet, wenn das Textkonfigurationsobjekt aus default_char erstellt wird.

Spaltenname	Datentyp	Beschreibung
nchar_stoplist	LONG NVARCHAR	Begriffe, die bei einer Volltextsuche in NCHAR-Spalten zu ignorieren sind. Diese Begriffe werden auch aus Textindizes ausgeschlossen. Diese Spalte wird verwendet, wenn das Textkonfigurationsobjekt aus default_nchar erstellt wird.
external_term_breaker	LONG VAR- CHAR	Der Funktions- und Bibliothekname für eine externe Begriffsegmentierer-Bibliothek.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (object_id)

FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id ) MATCH UNIQUE
FULL

FOREIGN KEY (creator) REFERENCES SYS.ISYSUSER (user_id)

UNIQUE INDEX (creator, text_config_name)
```

Siehe auch

- „Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Volltextsuche“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

SYSTEXTIDX-Systemansicht

Jede Zeile in der Systemansicht SYSTEXTIDX beschreibt einen Textindex. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSTEXTIDX.

Spaltenname	Datentyp	Beschreibung
index_id	UNSIGNED BIGINT	Die Objekt-ID des Textindexes in SYSIDX.
sequence	UNSIGNED INT	Wird nur intern verwendet.
status	UNSIGNED INT	Wird nur intern verwendet.
text_config	UNSIGNED BIGINT	Die Objekt-ID des Textkonfigurationsobjekts in SYSTEXT- CONFIG.
next_handle	UNSIGNED INT	Wird nur intern verwendet.
last_handle	UNSIGNED INT	Wird nur intern verwendet.
deleted_length	UNSIGNED BIGINT	Die Gesamtgröße der gelöschten indizierten Werte im Text- index.

Spaltenname	Datentyp	Beschreibung
pending_length	UNSIGNED BIGINT	Die Gesamtgröße der indizierten Werte, die dem Textindex bei der nächsten Aktualisierung hinzugefügt werden.
refresh_type	TINYINT	Die Art der Aktualisierung. Einer der Werte: <ul style="list-style-type: none"> • 1 Manuell (MANUAL) • 2 AUTO • 3 Sofort (IMMEDIATE)
refresh_interval	UNSIGNED INT	Das AUTO REFRESH-Intervall in Minuten.
last_refresh	TIMESTAMP	Die Uhrzeit der letzten Aktualisierung in Ortszeit.
last_refresh_utc	TIMESTAMP WITH TIME ZONE	Die Uhrzeit der letzten Aktualisierung in UTC.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (index_id, sequence)
```

```
FOREIGN KEY (index_id) REFERENCES SYS.ISYSOBJECT (object_id)
```

```
FOREIGN KEY (text_config) REFERENCES SYS.ISYSTETEXTCONFIG (object_id)
```

Siehe auch

- „Volltextsuche“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

SYSTETEXTIDXTAB-Systemansicht

Jede Zeile in der SYSTETEXTIDXTAB-Systemansicht beschreibt eine generierte Tabelle, die Teil eines Textindexes ist. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSTETEXTIDXTAB.

Spaltenname	Datentyp	Beschreibung
index_id	UNSIGNED BIGINT	Wird nur intern verwendet.
sequence	UNSIGNED INT	Wird nur intern verwendet.
table_type	UNSIGNED INT	Wird nur intern verwendet.
table_id	UNSIGNED INT	Wird nur intern verwendet.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (index_id, sequence, table_type)
```

```
FOREIGN KEY (index_id, sequence) REFERENCES SYS.ISYTEXTIDX (index_id,
sequence)
```

```
FOREIGN KEY (table_id) REFERENCES SYS.ISYSTAB (table_id)
```

Siehe auch

- „Volltextsuche“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

SYSTRIGGER-Systemansicht

Jede Zeile in der SYSTRIGGER-Systemansicht beschreibt einen Trigger in der Datenbank. Die Ansicht enthält außerdem Trigger, die automatisch für Fremdschlüssel-Definitionen erstellt werden, die eine referenzielle Trigger-Aktion haben (wie ON DELETE CASCADE). Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSTRIGGER.

Spaltenname	Datentyp	Beschreibung
trigger_id	UNSIGNED INT	Eine eindeutige Nummer für den Trigger in der SYSTRIGGER-Ansicht.
table_id	UNSIGNED INT	Die Tabellen-ID der Tabelle, zu der der Trigger gehört
object_id	UNSIGNED BIGINT	Die Objekt-ID für den Trigger in der Datenbank.
event	CHAR(1)	Der Vorgang, der das Auslösen des Triggers bewirkt. <ul style="list-style-type: none"> • A INSERT, DELETE • B INSERT, UPDATE • C UPDATE COLUMNS • D DELETE • E DELETE, UPDATE • I INSERT • M INSERT, DELETE, UPDATE • U UPDATE

Spaltenname	Datentyp	Beschreibung
trigger_time	CHAR(1)	<p>Der Zeitpunkt der Auslösung des Triggers relativ zum Ereignis.</p> <ul style="list-style-type: none"> • A AFTER (Trigger auf Zeilenebene) • B BEFORE (Trigger auf Zeilenebene) • I INSTEAD OF (Trigger auf Zeilenebene) • K INSTEAD OF (Trigger auf Anweisungsebene) • R RESOLVE • S AFTER (Trigger auf Anweisungsebene)
trigger_order	SMALLINT	Die Reihenfolge, in der die Trigger ausgelöst werden, wenn mehrere Trigger desselben Typs (INSERT, UPDATE oder DELETE) auf dieselbe Auslösezeit eingestellt sind (gilt nur für BEFORE- oder AFTER-Trigger).
foreign_table_id	UNSIGNED INT	Die Tabellennummer der Tabelle, die eine Fremdschlüssel-Definition enthält, die eine referenzielle getriggerte Aktion hat (wie "ON DELETE CASCADE") Der foreign_table_id-Wert spiegelt den Wert von ISYSIDX.table_id wider.
foreign_key_id	UNSIGNED INT	Die Fremdschlüssel-Nummer des Fremdschlüssels für die von foreign_table_id referenzierte Tabelle Die foreign_key_id-Wert spiegelt den Wert von SYSIDX.index_id wider.
referential_action	CHAR(1)	<p>Die durch einen Fremdschlüssel festgelegte Aktion. Dieser Ein-Zeichen-Wert entspricht der Aktion, die beim Erstellen des Fremdschlüssels angegeben wurde.</p> <ul style="list-style-type: none"> • C CASCADE • D SET DEFAULT • N SET NULL • R RESTRICT
trigger_name	CHAR(128)	Der Name des Triggers. Eine Tabelle kann keine zwei Trigger mit demselben Namen haben.
trigger_defn	LONG VARCHAR	Der Befehl, der zum Erstellen des Triggers verwendet wurde

Spaltenname	Datentyp	Beschreibung
remarks	LONG VAR-CHAR	Bemerkungen zum Trigger. Dieser Wert wird in der ISYSRE-MARK-Systemtabelle gespeichert.
source	LONG VAR-CHAR	Die SQL-Quelle für den Trigger. Dieser Wert wird in der ISYS-SOURCE-Systemtabelle gespeichert.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (trigger_id)

FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE
FULL

FOREIGN KEY (table_id) REFERENCES SYS.ISYSTAB (table_id)

FOREIGN KEY fkey_index (foreign_table_id, foreign_key_id) REFERENCES
SYS.ISYSIDX (table_id, index_id)

UNIQUE INDEX (table_id, event, trigger_time, trigger_order)

UNIQUE INDEX (trigger_name, table_id)

UNIQUE INDEX (table_id, foreign_table_id, foreign_key_id, event)
```

SYSTYPEMAP-Systemansicht

Die SYSTYPEMAP-Systemansicht enthält die Kompatibilitäts-Zuordnungswerte für die SYSSQLSERVERTYPE-Systemansicht. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSTYPEMAP.

Spaltenname	Datentyp	Beschreibung
ss_user_type	SMALLINT	Enthält den Benutzertyp von Adaptive Server Enterprise
sa_domain_id	SMALLINT	Enthält die entsprechende Domänen-Kennung von SQL Anywhere
sa_user_type	SMALLINT	Enthält den entsprechenden Benutzertyp von SQL Anywhere
nullable	CHAR(1)	Gibt an, ob der Typ Nullwerte zulässt.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
FOREIGN KEY (sa_domain_id) REFERENCES SYS.ISYSDOMAIN (domain_id)
```

SYSUNITOFMEASURE-Systemansicht

Jede Zeile der SYSUNITOFMEASURE-Systemansicht beschreibt eine Maßeinheit, die in der Datenbank festgelegt ist. Die Basistabelle für die SYSUNITOFMEASURE-Systemansicht ist die ISYSUNITOFMEASURE-Systemtabelle.

Spaltenname	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Wird nur vom System verwendet
owner	UNSIGNED INT	Der Eigentümer der Maßeinheit.
unit_name	CHAR(128)	Der Name der Maßeinheit.
unit_type	CHAR(7)	ANGULAR (Winkel) oder LINEAR.
conversion_factor	DOUBLE	Der Konvertierungsfaktor für die Maßeinheit.

Integritätsregeln für die zugrunde liegende Systemtabelle

PRIMARY KEY (object_id)

FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id)

FOREIGN KEY (owner) REFERENCES SYS.ISYSUSER (user_id)

UNIQUE CONSTRAINT (unit_name)

SYSUSER-Systemansicht

Jede Zeile in der SYSUSER-Systemansicht beschreibt einen Benutzer im System. Eigenständige Rollen werden zwar ebenfalls in dieser Ansicht gespeichert, aber für diese Rollen sind nur die Spalten user_id, object_id, user_name und user_type aussagekräftig. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSUSER.

Spaltenname	Datentyp	Beschreibung
user_id	UNSIGNED INT	Ein eindeutiger Bezeichner für den Benutzer, der der Login-Richtlinie zugewiesen ist.
object_id	UNSIGNED BIGINT	Ein eindeutiger Bezeichner für den Benutzer in der Datenbank.
user_name	CHAR(128)	Der Loginname für den Benutzer
password	BINARY(128)	Das Kennwort für den Benutzer. Aus Sicherheitsgründen sind Daten in dieser Spalte nur für Benutzer mit SELECT ANY TABLE-Systemprivileg sichtbar.

Spaltenname	Datentyp	Beschreibung
login_policy_id	UNSIGNED BIGINT	Ein eindeutiger Bezeichner für die Login-Richtlinie.
expired_password_on_login	TINYINT	Ein Wert, der angibt, ob das Kennwort des Benutzers beim nächsten Login abläuft.
password_creation_time	TIMESTAMP	Der Zeitpunkt in Ortszeit, zu dem das Kennwort für den Benutzer erstellt wurde.
failed_login_attempts	UNSIGNED INT	Die Anzahl der Versuche, bei denen das Login eines Benutzers fehlschlagen darf, bevor das Konto gesperrt wird.
last_login_time	TIMESTAMP	Der Zeitpunkt in Ortszeit, zu dem sich der Benutzer zuletzt angemeldet hat.

Spaltenname	Datentyp	Beschreibung
user_type	TINYINT	<p>Ein Wert, der angibt, ob es sich bei dem Benutzer um einen normalen Benutzer, eine Rolle oder einen als Rolle erweiterten Benutzer handelt und ob der Benutzer, die Rolle oder die erweiterte Rolle geändert (veränderlich) oder entfernt werden kann. Mögliche Werte:</p> <ul style="list-style-type: none">• 1 Unveränderliche Systemrolle.• 5 Veränderliche Systemrolle.• 9 Unveränderliche und entfernbare Systemrolle.• 12 Veränderlicher und entfernbarer Benutzer.• 13 Veränderliche und entfernbare Rolle.• 14 Veränderlicher und entfernbarer als Rolle erweiterter Benutzer.
user_dn	CHAR(1024)	<p>Ein LDAP-Distinguished Name (DN) als Bezeichner für den Benutzer, der innerhalb einer Domäne und über Domänen hinweg eindeutig ist. Der DN wird für die Authentifizierung bei einem LDAP Server verwendet.</p>

Spaltenname	Datentyp	Beschreibung
user_dn_cached_at	TIMESTAMP	Der Zeitpunkt, zu dem die user_dn-Spalte zuletzt im Cache gespeichert wurde. Anhand dieses Werts wird ermittelt, ob ein alter DN bereinigt werden muss. Der Wert wird unabhängig von der lokalen Zeitzone des Datenbankservers in Coordinated Universal Time (UTC) gespeichert.
password_creation_time_utc	TIMESTAMP WITH TIME ZONE	Der Zeitpunkt in UTC, zu dem das Kennwort für den Benutzer erstellt wurde.
last_login_time_utc	TIMESTAMP WITH TIME ZONE	Der Zeitpunkt in UTC, zu dem sich der Benutzer zuletzt angemeldet hat.
dual_password	BINARY(128)	Der erste bzw. zweite Teil des zweiteiligen Kennworts für den Benutzer. Aus Sicherheitsgründen sind Daten in dieser Spalte nur für Benutzer mit SELECT ANY TABLE-Systemprivileg sichtbar.

Hinweis

Bei Datenbanken, die ab SQL Anywhere 16 erstellt werden, ist die zugrunde liegende Systemtabelle für diese Ansicht immer verschlüsselt, um die Daten vor nicht autorisiertem Zugriff zu schützen.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (user_id)
```

```
FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE FULL
```

```
FOREIGN KEY (login_policy_id) REFERENCES SYS.ISYSLOGINPOLICY (login_policy_id)
```

```
UNIQUE INDEX (user_name)
```

Siehe auch

- „sp_sys_priv_role_info-Systemprozedur“ auf Seite 1399
- „SYSLOGINPOLICY-Systemansicht“ auf Seite 1463
- „SYSLOGINPOLICYOPTION-Systemansicht“ auf Seite 1463

SYSUSERMESSAGE-Systemansicht

Jede Zeile in der SYSUSERMESSAGE-Systemansicht enthält eine benutzerdefinierte Meldung für eine Fehlerbedingung. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSUSERMESSAGE.

Hinweis

Frühere Versionen dieses Katalogs enthielten eine SYSUSERMESSAGES-Systemtabelle. Diese Tabelle wurde zu ISYSUSERMESSAGE (ohne "S") umbenannt und ist die Basistabelle für diese Ansicht.

Spaltenname	Datentyp	Beschreibung
error	INTEGER	Eine eindeutige Identifikationsnummer für die Fehlerbedingung
uid	UNSIGNED INT	Die Benutzernummer, die die Meldung festgelegt hat
description	VARCHAR(255)	Die Meldung, die der Fehlerbedingung entspricht
langid	SMALLINT	Reserviert

Integritätsregeln für die zugrunde liegende Systemtabelle

```
FOREIGN KEY (uid) REFERENCES SYS.ISYSUSER (user_id)
```

```
UNIQUE CONSTRAINT (error, langid)
```

SYSUSERTYPE-Systemansicht

Jede Zeile in der SYSUSERTYPE-Systemansicht enthält eine Beschreibung eines benutzerdefinierten Datentyps. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSUSERTYPE.

Spaltenname	Datentyp	Beschreibung
type_id	SMALLINT	Eine eindeutige Kennung für den benutzerdefinierten Datentyp
creator	UNSIGNED INT	Die Benutzernummer des Eigentümers des Datentyps
domain_id	SMALLINT	Der Datentyp, auf dem dieser benutzerdefinierte Datentyp basiert, angezeigt durch eine Datentypnummer, die in der SYS-DOMAIN-Systemansicht aufgelistet ist
nulls	CHAR(1)	Gibt an, ob der benutzerdefinierte Datentyp Nullwerte zulässt. Mögliche Werte sind Y, N oder U. Der Wert U gibt an, dass die Nullwertfähigkeit nicht angegeben ist.
width	BIGINT	Die Länge einer Zeichenfolgen-Spalte, die Genauigkeit einer numerischen Spalte oder die Anzahl von Byte zum Speichern eines anderen Datentyps

Spaltenname	Datentyp	Beschreibung
scale	SMALLINT	Die Anzahl der Stellen nach dem Dezimalzeichen für Spalten mit numerischem Datentyp und Null für alle anderen Datentypen.
type_name	CHAR(128)	Der Name für den Datentyp
default	LONG VARCHAR	Der Standardwert für den Datentyp
check	LONG VARCHAR	Die CHECK-Bedingung für den Datentyp
base_type_str	VAR-CHAR(32767)	Die zugehörige Typzeichenfolge, die den Typ des Benutzertyps darstellt.

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (type_id)

FOREIGN KEY (creator) REFERENCES SYS.ISYSUSER (user_id)

FOREIGN KEY (domain_id) REFERENCES SYS.ISYSDOMAIN (domain_id)

UNIQUE CONSTRAINT (type_name)
```

SYSVIEW-Systemansicht

Jede Zeile in der SYSVIEW-Systemansicht beschreibt eine Ansicht in der Datenbank. Zusätzliche Informationen zu Ansichten finden Sie in der SYSTAB-Systemansicht. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSVIEW.

Sie können auch die sa_materialized_view_info-Systemprozedur verwenden, um ein besser lesbares Format der Informationen zu materialisierten Ansichten zu erhalten.

Spaltenname	Datentyp	Beschreibung
view_object_id	UNSIGNED BIGINT	Die Objekt-ID der Ansicht
view_def	LONG VARCHAR	Die Definition (Abfragespezifizierung) der Ansicht
mv_build_type	TINYINT	Derzeit nicht verwendet
mv_refresh_type	TINYINT	Der für die Ansicht definierte Aktualisierungstyp. Mögliche Werte sind IMMEDIATE (1) und MANUAL (2).

Spaltenname	Datentyp	Beschreibung
mv_use_in_optimization	TINYINT	Gibt an, ob die materialisierte Ansicht während der Abfrageoptimierung verwendet werden kann (0 = in der Optimierung nicht verwendbar, 1 = in der Optimierung verwendbar)
mv_last_refreshed_at	TIME-STAMP	Zeigt den Zeitpunkt in Ortszeit (Datum und Uhrzeit) an, zu dem die materialisierte Ansicht zuletzt aktualisiert wurde.
mv_known_stale_at	TIME-STAMP	Der Zeitpunkt in Ortszeit, seit dem die materialisierte Ansicht als veraltet gilt. Dieser Wert entspricht dem Zeitpunkt, an dem festgestellt wurde, dass eine der zugrunde liegenden Basistabellen geändert wurde. Der Wert "0" zeigt an, dass die Ansicht aktualisiert wurde. Allerdings könnte sie bereits wieder veraltet sein, ohne dass sie der Datenbankserver entsprechend markiert hat, weil die Ansicht seitdem nicht verwendet wurde. Verwenden Sie die sa_materialized_view_info-Systemprozedur, um den Status der materialisierten Ansicht zu bestimmen.
mv_last_refreshed_tsn	UNSIGNED BIGINT	Die Sequenznummer, die der Transaktion zugewiesen wurde, die die materialisierte Ansicht aktualisiert hat.
mv_last_refreshed_at_utc	TIME-STAMP WITH TIME ZONE	Zeigt den Zeitpunkt in UTC (Datum und Uhrzeit) an, zu dem die materialisierte Ansicht zuletzt aktualisiert wurde.
mv_known_stale_at_utc	TIME-STAMP WITH TIME ZONE	Der Zeitpunkt in UTC, seit dem die materialisierte Ansicht als veraltet gilt. Dieser Wert entspricht dem Zeitpunkt, an dem festgestellt wurde, dass eine der zugrunde liegenden Basistabellen geändert wurde. Der Wert "0" zeigt an, dass die Ansicht aktualisiert wurde. Allerdings könnte sie bereits wieder veraltet sein, ohne dass sie der Datenbankserver entsprechend markiert hat, weil die Ansicht seitdem nicht verwendet wurde. Verwenden Sie die sa_materialized_view_info-Systemprozedur, um den Status einer materialisierten Ansicht zu ermitteln. Diese Spalte enthält 0, wenn mv_last_refreshed_at 0 ist, und NULL, wenn mv_last_refreshed_at NULL ist.

Bemerkungen

Wenn eine materialisierte Ansicht mit der SNAPSHOT-Isolation aktualisiert wird, beziehen sich mv_last_refreshed_at und mv_last_refreshed_tsn auf die früheste Transaktion, die eine Zeile während der Berechnung der Inhalte der materialisierten Ansicht geändert hat.

Integritätsregeln für die zugrunde liegende Systemtabelle

PRIMARY KEY (view_object_id)

```
FOREIGN KEY (view_object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

Siehe auch

- „Aktivieren und Deaktivieren der Verwendung einer materialisierten Ansicht durch den Optimierer“ [SQL Anywhere Server - SQL-Benutzerhandbuch]
- „Einstellen des Aktualisierungstyps auf "Manuell" oder "Sofort"“ [SQL Anywhere Server - SQL-Benutzerhandbuch]
- „sa_materialized_view_info-Systemprozedur“ auf Seite 1258
- „SYSTAB-Systemansicht“ auf Seite 1492
- „CREATE MATERIALIZED VIEW-Anweisung“ auf Seite 652
- „REFRESH MATERIALIZED VIEW-Anweisung“ auf Seite 987
- „CREATE VIEW-Anweisung“ auf Seite 773

SYSWEBSERVICE-Systemansicht

Jede Zeile in der SYSWEBSERVICE-Systemansicht enthält eine Beschreibung eines Webdienstes. Die zugrunde liegende Systemtabelle für diese Ansicht ist ISYSWEBSERVICE.

Spaltenname	Datentyp	Beschreibung
service_id	UNSIGNED INT	Eine eindeutige Kennung für den Webdienst
object_id	UNSIGNED BIGINT	Die ID des Webdienstes
service_name	CHAR(128)	Der dem Webdienst zugeordnete Name
service_type	VARCHAR(40)	Der Typ des Dienstes, z.B. RAW, HTTP, XML, SOAP oder DISH
auth_required	CHAR(1)	Gibt an, ob alle Anforderungen einen gültigen Benutzernamen und ein Kennwort enthalten müssen.
secure_required	CHAR(1)	Gibt an, ob unsichere Verbindungen wie HTTP akzeptiert werden sollen, oder nur sichere Verbindungen wie HTTPS.
url_path	CHAR(1)	Steuert die Interpretation von URL
user_id	UNSIGNED INT	Wenn die Authentifizierung aktiviert ist, wird hiermit der Benutzer bzw. die Benutzergruppe gekennzeichnet, die berechtigt ist, diesen Dienst zu benutzen. Wenn die Authentifizierung deaktiviert ist, wird das Konto angegeben, das verwendet wird, wenn Anforderungen verarbeitet werden.
parameter	LONG VARCHAR	Ein Präfix, das die SOAP-Dienste kennzeichnet, die in einen DISH-Dienst einbezogen werden sollen

Spaltenname	Datentyp	Beschreibung
statement	LONG VARCHAR	Eine SQL-Anweisung, die immer als Antwort auf eine Anforderung ausgeführt wird. Wenn NULL, werden stattdessen beliebige Anweisungen aus den jeweiligen Anforderungen ausgeführt. Ignoriert bei Diensten vom Typ DISH.
remarks	LONG VARCHAR	Bemerkungen zum Webdienst. Dieser Wert wird in der ISYS-REMARK-Systemtabelle gespeichert.
enabled	CHAR(1)	Gibt an, ob der Webdienst zurzeit aktiviert oder deaktiviert ist (siehe CREATE SERVICE).

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (service_id)
```

```
FOREIGN KEY (object_id) REFERENCES SYS.ISYSOBJECT (object_id) MATCH UNIQUE FULL
```

```
UNIQUE CONSTRAINT (service_name)
```

Konsolidierte Ansichten

Konsolidierte Ansichten enthalten Daten in einem von Benutzern häufig verwendetem Format. Beispielsweise machen konsolidierte Ansichten allgemein benötigte Joins verfügbar. Konsolidierte Ansichten unterscheiden sich von Systemansichten, da sie nicht nur eine direkte Ansicht der unformatierten Daten in einer oder mehreren darunterliegenden Systemtabellen liefern. So sind zum Beispiel viele Spalten in den Systemansichten unverständliche ID-Werte, während sie in konsolidierten Ansichten lesbare Namen sind.

Konsolidierte Ansicht ST_GEOMETRY_COLUMNS

Jede Zeile der ST_GEOMETRY_COLUMNS-Systemansicht beschreibt eine Spalte mit räumlichen Daten, die in der Datenbank festgelegt ist.

Spaltenname	Datentyp	Beschreibung
table_catalog	VARCHAR(128)	Wird nur intern verwendet.
table_schema	CHAR(128)	Der Name des Schemas, zu dem die Tabelle mit der Spalte gehört. Dies entspricht dem Tabelleneigentümer.
table_name	CHAR(128)	Der Name der Tabelle, die die Spalte enthält.
column_name	CHAR(128)	Der Name der räumlichen Spalte.

Spaltenname	Datentyp	Beschreibung
srs_name	CHAR(128)	Der Name des räumlichen Bezugssystems, das der Spalte zugeordnet ist. Wenn der Spalte kein räumliches Bezugssystem zugeordnet wurde, ist srs_name NULL.
srs_id	INTEGER	Die SRID des räumlichen Bezugssystems, das der Spalte zugeordnet ist.
table_id	UNSIGNED INT	Die Tabellen-ID der Tabelle, die die Spalte enthält.
column_id	UNSIGNED INT	Die numerische ID der Spalte.
geometry_type_name	VARCHAR(32767)	Der räumliche Datentyp der in der Spalte enthaltenen Geometrien (z.B. ST_Point, ST_Geometry usw.).

Siehe auch

- „Räumliche Bezugssysteme (SRS) und räumliche Referenz-IDs (SRID)“ [[SQL Anywhere Server - Unterstützung für räumliche Daten](#)]
- „Unterstützte räumliche Datentypen und ihre Hierarchie“ [[SQL Anywhere Server - Unterstützung für räumliche Daten](#)]

Konsolidierte Ansicht ST_SPATIAL_REFERENCE_SYSTEMS

Jede Zeile der ST_SPATIAL_REFERENCE_SYSTEMS-Systemansicht beschreibt ein räumliches Bezugssystem, das in der Datenbank festgelegt ist. Diese Ansicht bietet eine etwas andere Menge an Informationen als SYSSPATIALREFERENCINGSYSTEM.

Spaltenname	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Wird nur vom System verwendet
owner	UNSIGNED INT	Eigentümer des räumlichen Bezugssystems
srs_name	CHAR(128)	Der Name des räumlichen Bezugssystems.
srs_id	INTEGER	Der numerische Bezeichner (SRID) für das räumliche Bezugssystem.

Spaltenname	Datentyp	Beschreibung
srs_type	CHAR(11)	<p>Der SRS-Typ gemäß der Definition im SQL/MM-Standard. Der Wert kann einer der folgenden sein:</p> <ul style="list-style-type: none"> • GEOGRAPHIC Dies ist für räumliche Bezugssysteme, die auf georeferenzierten Koordinatensystemen mit den Achsen Breitengrad, Längengrad (und Höhe) basieren. Diese räumlichen Bezugssysteme sind vom Typ PLANAR oder ROUND EARTH. • PROJECTED Dies ist für räumliche Bezugssysteme, die auf georeferenzierten Koordinatensystemen ohne die Achsen Breitengrad und Längengrad basieren. Diese räumlichen Bezugssysteme sind vom Typ PLANAR. • ENGINEERING Dies ist für räumliche Bezugssysteme, die auf nicht georeferenzierten Koordinatensystemen basieren. Diese räumlichen Bezugssysteme sind vom Typ PLANAR. • GEOCENTRIC Nicht unterstützt. • COMPOUND Nicht unterstützt. • VERTICAL Nicht unterstützt. <p>Wenn srs_type leer ist, gilt der Typ als nicht angegeben.</p>
round_earth	CHAR(1)	Zeigt an, ob der SRS-Typ ROUND EARTH (Y) oder PLANAR (N) ist.
axis_order	CHAR(12)	Beschreibt, wie der Datenbankserver Punkte hinsichtlich Breiten- und Längengrad interpretiert (z.B. bei Verwendung der Methoden ST_Lat und ST_Long). Für nicht geografische räumliche Bezugssysteme gilt die Achsenreihenfolge x/y/z/m. Für geografische räumliche Bezugssysteme lautet die Achsenreihenfolge standardmäßig Länge/Breite/z/m. Breite/Länge/z/m wird ebenfalls unterstützt.
snap_to_grid	DOUBLE	Legt die Größe des Rasters fest, das SQL Anywhere beim Ausführen von Berechnungen verwendet.
tolerance	DOUBLE	Legt die Genauigkeit fest, die beim Vergleichen von Punkten verwendet werden soll.
semi_major_axis	DOUBLE	Der Abstand zwischen der Mitte des Ellipsoids und dem Äquator für ein räumliches Bezugssystem mit gewölbter Erdarstellung.

Spaltenname	Datentyp	Beschreibung
semi_minor_axis	DOUBLE	Der Abstand zwischen der Mitte des Ellipsoids und den Polen für ein räumliches Bezugssystem mit gewölbter Erddarstellung.
inv_flattening	DOUBLE	Die inverse Abplattung des Ellipsoids in einem räumlichen Bezugssystem mit gewölbter Erddarstellung. Dies ist ein Verhältnis, dass sich aus der folgenden Gleichung ergibt: $1/f = (\text{semi-major-axis}) / (\text{semi-major-axis} - \text{semi-minor-axis})$
min_x	DOUBLE	Der minimale in Koordinaten zulässige x-Wert.
max_x	DOUBLE	Der maximale in Koordinaten zulässige x-Wert.
min_y	DOUBLE	Der minimale in Koordinaten zulässige y-Wert.
max_y	DOUBLE	Der maximale in Koordinaten zulässige y-Wert.
min_z	DOUBLE	Der minimale in Koordinaten zulässige z-Wert.
max_z	DOUBLE	Der maximale in Koordinaten zulässige z-Wert.
min_m	DOUBLE	Der minimale in Koordinaten zulässige m-Wert.
max_m	DOUBLE	Der maximale in Koordinaten zulässige m-Wert.
min_lat	DOUBLE	Der minimale in Koordinaten zulässige Breitengradwert.
max_lat	DOUBLE	Der maximale in Koordinaten zulässige Breitengradwert.
min_long	DOUBLE	Der minimale in Koordinaten zulässige Längengradwert.
max_long	DOUBLE	Der maximale in Koordinaten zulässige Längengradwert.
organization	LONG VARCHAR	Der Name der Organisation, die das von dem räumlichen Bezugssystem verwendete Koordinatensystem erstellt hat.
organization_coord-sys_id	INTEGER	Die ID, die das Koordinatensystem von der erstellenden Organisation erhalten hat.
linear_unit_of_measure	CHAR(128)	Die vom räumlichen Bezugssystem verwendete lineare Maßeinheit.
angular_unit_of_measure	CHAR(128)	Die vom räumlichen Bezugssystem verwendete Winkelmaßeinheit.

Spaltenname	Datentyp	Beschreibung
polygon_format	LONG VARCHAR	Die Ausrichtung der Ringe in einem Polygon. Mögliche Werte lauten CounterClockwise, ClockWise und EvenOdd.
storage_format	LONG VARCHAR	Legt fest, ob die Daten in normalisiertem Format (Internal), in nicht normalisiertem Format (Original) oder in beiden (Mixed) gespeichert werden.
definition	LONG VARCHAR	Zusätzliche Definitionseinstellungen.
transform_definition	LONG VARCHAR	Transformationsdefinitionseinstellungen, die beim Transformieren von Daten aus diesem räumlichen Bezugssystem in ein anderes verwendet werden.
description	LONG VARCHAR	Beschreibung des räumlichen Bezugssystems.

Siehe auch

- „SYSSPATIALREFERENCESYSTEM-Systemansicht“ auf Seite 1485
- „CREATE SPATIAL REFERENCE SYSTEM-Anweisung“ auf Seite 719

Konsolidierte Ansicht ST_UNITS_OF_MEASURE

Jede Zeile der ST_UNITS_OF_MEASURE-Systemansicht beschreibt eine Maßeinheit, die in der Datenbank festgelegt ist. Diese Ansicht bietet mehr Informationen als SYSUNITOFMEASURE.

Spaltenname	Datentyp	Beschreibung
object_id	UNSIGNED BIGINT	Wird nur vom System verwendet
owner	UNSIGNED INT	Der Eigentümer der Maßeinheit.
unit_name	CHAR(128)	Der Name der Maßeinheit.
unit_type	CHAR(7)	ANGULAR (Winkel) oder LINEAR.
conversion_factor	DOUBLE	Der Konvertierungsfaktor für die Maßeinheit.
description	LONG VARCHAR	Eine Beschreibung für die Maßeinheit.

Konsolidierte Ansicht SYSARTICLECOLS

Jede Zeile in der SYSARTICLECOLS-Ansicht kennzeichnet eine Spalte in einem Artikel.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSARTICLECOLS"
as select p.publication_name,t.table_name,c.column_name
from SYS.ISYSARTICLECOL as ac
join SYS.ISYSPUBLICATION as p on p.publication_id = ac.publication_id
join SYS.ISYSTAB as t on t.table_id = ac.table_id
join SYS.ISYSTABCOL as c on c.table_id = ac.table_id
and c.column_id = ac.column_id
```

Siehe auch

- „SYSARTICLECOL-Systemansicht“ auf Seite 1438
- „SYSPUBLICATION-Systemansicht“ auf Seite 1473
- „SYSTAB-Systemansicht“ auf Seite 1492
- „SYSTABCOL-Systemansicht“ auf Seite 1495

Konsolidierte Ansicht SYSARTICLES

Jede Zeile in der SYSARTICLES-Ansicht beschreibt einen Artikel in einer Publikation.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSARTICLES"
as select u1.user_name as publication_owner,p.publication_name,
u2.user_name as table_owner,t.table_name,
a.where_expr,a.subscribe_by_expr,a.alias
from SYS.ISYSARTICLE as a
join SYS.ISYSPUBLICATION as p on(a.publication_id = p.publication_id)
join SYS.ISYSTAB as t on(a.table_id = t.table_id)
join SYS.ISYSUSER as u1 on(p.creator = u1.user_id)
join SYS.ISYSUSER as u2 on(t.creator = u2.user_id)
```

Siehe auch

- „SYSARTICLE-Systemansicht“ auf Seite 1438
- „SYSPUBLICATION-Systemansicht“ auf Seite 1473
- „SYSTAB-Systemansicht“ auf Seite 1492
- „SYSUSER-Systemansicht“ auf Seite 1506

Konsolidierte Ansicht SYSCAPABILITIES

Jede Zeile der SYSCAPABILITIES-Ansicht gibt den Status einer Fähigkeit für einen entfernten Datenbankserver an. Diese Ansicht bezieht ihre Daten von der ISYSCAPABILITY- und der ISYSCAPABILITYNAME-Systemtabelle.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSCAPABILITIES"  
as select  
  ISYSCAPABILITY.capid, ISYSCAPABILITY.srvid, property('RemoteCapability', ISYSCAP  
  ABILITY.capid) as capname, ISYSCAPABILITY.capvalue  
  from SYS.ISYSCAPABILITY
```

Siehe auch

- [„SYSCAPABILITY-Systemansicht“ auf Seite 1439](#)
- [„SYSCAPABILITYNAME-Systemansicht“ auf Seite 1439](#)

Konsolidierte Ansicht SYSCATALOG

Jede Zeile in der SYSCATALOG-Ansicht beschreibt eine Systemtabelle.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSCATALOG"( creator,  
  tname, dbspacename, tabletype, ncols, primary_key, "check",  
  remarks )  
as select u.user_name, tab.table_name, dbs.dbpace_name,  
  if tab.table_type_str = 'BASE' then 'TABLE' else tab.table_type_str  
endif,  
  (select count() from SYS.ISYSTABCOL  
    where ISYSTABCOL.table_id = tab.table_id),  
  if ix.index_id is null then 'N' else 'Y' endif,  
  null,  
  rmk.remarks  
from SYS.SYSTAB as tab  
  join SYS.ISYSDBSPACE as dbs on (tab.dbpace_id = dbs.dbpace_id)  
  join SYS.ISYSUSER as u on u.user_id = tab.creator  
  left outer join SYS.ISYSIDX as ix on (tab.table_id = ix.table_id and  
  ix.index_id = 0)  
  left outer join SYS.ISYSREMARK as rmk on (tab.object_id = rmk.object_id)
```

Siehe auch

- [„SYSTAB-Systemansicht“ auf Seite 1492](#)
- [„SYSTABCOL-Systemansicht“ auf Seite 1495](#)
- [„SYSDBSPACE-Systemansicht“ auf Seite 1444](#)
- [„SYSUSER-Systemansicht“ auf Seite 1506](#)
- [„SYSIDX-Systemansicht“ auf Seite 1456](#)
- [„SYSREMARK-Systemansicht“ auf Seite 1474](#)

Konsolidierte Ansicht SYSCOLAUTH

Jede Zeile in der SYSCOLAUTH-Ansicht beschreibt eine Reihe von Privilegien (UPDATE, SELECT- oder REFERENCES), die für eine Spalte erteilt wurden. Die SYSCOLAUTH-Ansicht bietet eine benutzerfreundliche Darstellung der Daten in der [„SYSCOLPERM-Systemansicht“ auf Seite 1441](#).

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSCOLAUTH"( grantor,grantee,creator,tname,colname,
    privilege_type,is_grantable )
as select u1.user_name,u2.user_name,u3.user_name,tab.table_name,
    col.column_name,cp.privilege_type,cp.is_grantable
    from SYS.ISYSCOLPERM as cp
        join SYS.ISYSUSER as u1 on u1.user_id = cp.grantor
        join SYS.ISYSUSER as u2 on u2.user_id = cp.grantee
        join SYS.ISYSTAB as tab on tab.table_id = cp.table_id
        join SYS.ISYSUSER as u3 on u3.user_id = tab.creator
        join SYS.ISYSTABCOL as col on col.table_id = cp.table_id
        and col.column_id = cp.column_id
```

Siehe auch

- „SYSCOLPERM-Systemansicht“ auf Seite 1441
- „SYSTABCOL-Systemansicht“ auf Seite 1495
- „SYSUSER-Systemansicht“ auf Seite 1506
- „SYSTAB-Systemansicht“ auf Seite 1492

Konsolidierte Ansicht SYSCOLSTATS

Die SYSCOLSTATS-Ansicht enthält die Spaltenstatistiken gespeichert, die als Histogramme gespeichert und vom Optimierer verwendet werden.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSCOLSTATS" AS SELECT u.user_name, t.table_name,
    c.column_name, s.format_id,
        dateadd(mi, PROPERTY('TimeZoneAdjustment'), s.update_time) as
    update_time, s.density, s.max_steps, s.actual_steps,
        s.step_values, s.frequencies , TODATETIMEOFFSET( s.update_time, 0 ) as
    update_time_utc
    FROM SYS.ISYSCOLSTAT s
        JOIN SYS.ISYSTABCOL c on (s.table_id = c.table_id and s.column_id =
    c.column_id)
        JOIN SYS.ISYSTAB t on (t.table_id = c.table_id)
        JOIN SYS.ISYSUSER u on (u.user_id = t.creator)
```

Siehe auch

- „SYSCOLSTAT-Systemansicht“ auf Seite 1441
- „SYSTABCOL-Systemansicht“ auf Seite 1495
- „SYSTAB-Systemansicht“ auf Seite 1492
- „SYSUSER-Systemansicht“ auf Seite 1506

Konsolidierte Ansicht SYSCOLUMNS

Jede Zeile in der SYSCOLUMNS-Ansicht beschreibt eine Spalte von jeder Tabelle und Ansicht im Katalog.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSCOLUMNS"( creator,cname,tname,coltype,nulls,length,
    syslength,in_primary_key,colno,default_value,
    column_kind,remarks )
as select u.user_name,col.column_name,tab.table_name,dm.domain_name,
    col.nulls,col.width,col.scale,if ixcol.sequence is null then 'N' else
'Y' endif,col.column_id,
    col."default",col.column_type,rmk.remarks
from SYS.SYSTABCOL as col
    left outer join SYS.ISYSIDXCOL as ixcol on(col.table_id =
ixcol.table_id and col.column_id = ixcol.column_id and ixcol.index_id = 0)
    join SYS.ISYSTAB as tab on(tab.table_id = col.table_id)
    join SYS.ISYSDOMAIN as dm on(dm.domain_id = col.domain_id)
    join SYS.ISYSUSER as u on u.user_id = tab.creator
    left outer join SYS.ISYSREMARK as rmk on(col.object_id = rmk.object_id)
```

Siehe auch

- „SYSTABCOL-Systemansicht“ auf Seite 1495
- „SYSIDXCOL-Systemansicht“ auf Seite 1457
- „SYSTAB-Systemansicht“ auf Seite 1492
- „SYSDOMAIN-Systemansicht“ auf Seite 1446
- „SYSUSER-Systemansicht“ auf Seite 1506
- „SYSREMARK-Systemansicht“ auf Seite 1474

Konsolidierte Ansicht SYSFORIGNKEYS

Jede Zeile in der SYSFORIGNKEYS-Ansicht beschreibt einen Fremdschlüssel für jede Tabelle im Katalog.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSFORIGNKEYS"( foreign_creator,
    foreign_tname,
    primary_creator,primary_tname,role,columns )
as select fk_up.user_name,fk_tab.table_name,pk_up.user_name,
    pk_tab.table_name,ix.index_name,
    (select list(string(fk_col.column_name,' IS ',
    pk_col.column_name)
    order by fkc.table_id,fkc.index_id,fkc."sequence")
    from SYS.ISYSIDXCOL as fkc
        join SYS.ISYSTABCOL as fk_col on(
            fkc.table_id = fk_col.table_id
            and fkc.column_id = fk_col.column_id)
        ,SYS.ISYSTABCOL as pk_col
    where fkc.table_id = fk.foreign_table_id
```

```

        and fkc.index_id = fk.foreign_index_id
        and pk_col.table_id = fk.primary_table_id
        and pk_col.column_id = fkc.primary_column_id)
from SYS.ISYSFKEY as fk
join SYS.ISYSTAB as fk_tab on fk_tab.table_id = fk.foreign_table_id
join SYS.ISYSUSER as fk_up on fk_up.user_id = fk_tab.creator
join SYS.ISYSTAB as pk_tab on pk_tab.table_id = fk.primary_table_id
join SYS.ISYSUSER as pk_up on pk_up.user_id = pk_tab.creator
join SYS.ISYSIDX as ix on ix.table_id = fk.foreign_table_id and
ix.index_id = fk.foreign_index_id

```

Siehe auch

- „SYSTAB-Systemansicht“ auf Seite 1492
- „SYSTABCOL-Systemansicht“ auf Seite 1495
- „SYSIDX-Systemansicht“ auf Seite 1456
- „SYSIDXCOL-Systemansicht“ auf Seite 1457
- „SYSFKEY-Systemansicht“ auf Seite 1452
- „SYSUSER-Systemansicht“ auf Seite 1506
- „SYSDOMAIN-Systemansicht“ auf Seite 1446
- „SYSREMARK-Systemansicht“ auf Seite 1474

Konsolidierte Ansicht SYSINDEXES

Jede Zeile in der SYSINDEXES-Ansicht beschreibt einen Index in der Datenbank. Als Alternative zu dieser Ansicht können Sie auch die SYSIDX- und die SYSIDXCOL-Systemansichten verwenden.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```

ALTER VIEW "SYS"."SYSINDEXES"( icreator,
    iname, fname, creator, tname, indextype,
    colnames, interval, level_num )
as select u.user_name, idx.index_name, dbs.dbSPACE_name, u.user_name,
    tab.table_name,
    case idx.index_category
    when 1 then 'Primary Key'
    when 2 then 'Foreign Key'
    when 3 then(
        if idx."unique" = 4 then 'Non-unique'
        else if idx."unique" = 2 then 'UNIQUE constraint'
            else if idx."unique" = 5 then 'UNIQUE NULLS NOT DISTINCT'
            else 'UNIQUE'
        endif
    endif
    endif
    when 4 then 'Text Index' end, (select list(string(c.column_name,
    if idx."order" = 'A' then ' ASC' else ' DESC' endif) order by
    idx.table_id asc, idx.index_id asc, idx.sequence asc)
from SYS.ISYSIDXCOL as idx
join SYS.ISYSTABCOL as c on(
    c.table_id = idx.table_id
    and c.column_id = idx.column_id)
where idx.index_id = idx.index_id
    and idx.table_id = idx.table_id),
0,0
from SYS.ISYSTAB as tab

```

```
join SYS.ISYSDBSPACE as dbs on(tab.dbSPACE_id = dbs.dbSPACE_id)
join SYS.ISYSIDX as idx on(idx.table_id = tab.table_id)
join SYS.ISYSUSER as u on u.user_id = tab.creator
```

Siehe auch

- „SYSIDX-Systemansicht“ auf Seite 1456
- „SYSTABCOL-Systemansicht“ auf Seite 1495
- „SYSTAB-Systemansicht“ auf Seite 1492
- „SYSDBSPACE-Systemansicht“ auf Seite 1444
- „SYSIDXCOL-Systemansicht“ auf Seite 1457
- „SYSUSER-Systemansicht“ auf Seite 1506

Konsolidierte Ansicht SYSOPTIONS

Jede Zeile in der SYSOPTIONS-Ansicht beschreibt eine Option, die mit dem SET-Befehl erstellt wurde. Jeder Benutzer kann seine eigenen Einstellungen für jede Option haben. Zusätzlich legen Einstellungen für den PUBLIC-Benutzer die Standardeinstellungen fest, die für die Benutzer verwendet werden, die keine eigenen Einstellungen haben.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSOPTIONS"( user_name,"option",setting )
as select u.user_name,opt."option",opt.setting
from SYS.ISYSOPTION as opt
join SYS.ISYSUSER as u on opt.user_id = u.user_id
```

Siehe auch

- „SYSOPTION-Systemansicht“ auf Seite 1467
- „SYSUSER-Systemansicht“ auf Seite 1506

Konsolidierte Ansicht SYSPROCAUTH

Jede Zeile in der SYSPROCAUTH-Ansicht beschreibt Privilegien, die für eine Prozedur erteilt wurden. Als Alternative können Sie die SYSPROCPERM-Systemansicht verwenden.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSPROCAUTH"( grantee,
creator,procname )
as select u1.user_name,u2.user_name,p.proc_name
from SYS.ISYSPROCEDURE as p
join SYS.ISYSPROCPERM as pp on(p.proc_id = pp.proc_id)
join SYS.ISYSUSER as u1 on u1.user_id = pp.grantee
join SYS.ISYSUSER as u2 on u2.user_id = p.creator
```

Siehe auch

- „SYSPROCEDURE-Systemansicht“ auf Seite 1469
- „SYSPROCPERM-Systemansicht“ auf Seite 1472
- „SYSUSER-Systemansicht“ auf Seite 1506

Konsolidierte Ansicht SYSPROCPARMS

Jede Zeile in der SYSPROCPARMS-Ansicht beschreibt einen Parameter für eine Prozedur in der Datenbank.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSPROCPARMS"( creator,
    procname,paramname,param_id,paramtype,parammode,paramdomain,
    length,scale,"default",user_type )
as select up.user_name,p.proc_name,pp.param_name,pp.param_id,pp.param_type,
    if pp.param_mode_in = 'Y' and pp.param_mode_out = 'N' then 'IN'
    else if pp.param_mode_in = 'N' and pp.param_mode_out = 'Y' then 'OUT'
    else 'INOUT'
    endif
    endif,dm.domain_name,pp.width,pp.scale,pp."default",ut.type_name
from SYS.SYSPROCPARM as pp
    join SYS.ISYSPROCEDURE as p on p.proc_id = pp.proc_id
    join SYS.ISYSUSER as up on up.user_id = p.creator
    join SYS.ISYSDOMAIN as dm on dm.domain_id = pp.domain_id
    left outer join SYS.ISYSUSERTYPE as ut on ut.type_id = pp.user_type
```

Siehe auch

- „SYSPROCPARM-Systemansicht“ auf Seite 1470
- „SYSPROCEDURE-Systemansicht“ auf Seite 1469
- „SYSUSER-Systemansicht“ auf Seite 1506
- „SYSDOMAIN-Systemansicht“ auf Seite 1446
- „SYSUSERTYPE-Systemansicht“ auf Seite 1510

Konsolidierte Ansicht SYSPROCS

Die SYSPROCS-Ansicht enthält den Prozedur- oder Funktionsnamen, den Namen des Erstellers und etwaige für die Prozedur oder Funktion aufgezeichnete Kommentare.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der ALTER VIEW-Anweisung unten geliefert.

```
ALTER VIEW "SYS"."SYSPROCS"( creator,
    procname,remarks )
as select u.user_name,p.proc_name,r.remarks
    from SYS.ISYSPROCEDURE as p
        join SYS.ISYSUSER as u on u.user_id = p.creator
        left outer join SYS.ISYSREMARK as r on(p.object_id = r.object_id)
```

Siehe auch

- „SYSPROCEDURE-Systemansicht“ auf Seite 1469
- „SYSUSER-Systemansicht“ auf Seite 1506
- „SYSREMARK-Systemansicht“ auf Seite 1474

Konsolidierte Ansicht SYSPUBLICATIONS

Jede Zeile in der SYSPUBLICATIONS-Ansicht beschreibt eine SQL Remote- oder MobiLink-Publikation.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSPUBLICATIONS"
as select u.user_name as creator,
       p.publication_name,
       r.remarks,
       p.type,
       case p.sync_type
       when 0 then 'logscan'
       when 1 then 'scripted upload'
       when 2 then 'download only'
       else 'invalid'
       end as sync_type
from SYS.ISYSPUBLICATION as p
     join SYS.ISYSUSER as u on u.user_id = p.creator
     left outer join SYS.ISYSREMARK as r on (p.object_id = r.object_id)
```

Siehe auch

- „SYSPUBLICATION-Systemansicht“ auf Seite 1473
- „SYSREMARK-Systemansicht“ auf Seite 1474

Konsolidierte Ansicht SYSREMOTEOPTION2

Verbindet die Spalten aus den Systemansichten SYSREMOTEOPTION und SYSREMOTEOPTIONTYPE und zeigt sie in einem besser lesbaren Format an.

Werte in der setting-Spalte werden für Benutzer verborgen, die nicht das SELECT ANY TABLE-Systemprivileg haben.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSREMOTEOPTION2"
as select ISYSREMOTEOPTION.option_id,
       ISYSREMOTEOPTION.user_id,
       SYS.HIDE_FROM_NON_DBA(ISYSREMOTEOPTION.setting) as setting
from SYS.ISYSREMOTEOPTION
```

Siehe auch

- „SYSREMOTEOPTION-Systemansicht“ auf Seite 1475

Konsolidierte Ansicht SYSREMOTEOPTIONS

Jede Zeile der SYSREMOTEOPTIONS-Ansicht beschreibt den Wert eines SQL Remote-Nachrichtenverbindungsparameters.

Werte in der setting-Spalte werden für Benutzer verborgen, die nicht das SELECT ANY TABLE-Systemprivileg haben. Die Ansicht SYSREMOTEOPTION2 bietet öffentlichen Zugriff auf nicht sensible Daten.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSREMOTEOPTIONS"
as select srt.type_name,
       sup.user_name,
       srot."option",
       SYS.HIDE_FROM_NON_DBA(sro.setting) as setting
from SYS.ISYSREMOTETYPE as srt
     ,SYS.ISYSREMOTEOPTIONTYPE as srot
     ,SYS.ISYSREMOTEOPTION as sro
     ,SYS.ISYSUSER as sup
where srt.type_id = srot.type_id
and srot.option_id = sro.option_id
and sro.user_id = sup.user_id
```

Siehe auch

- „SYSREMOTETYPE-Systemansicht“ auf Seite 1476
- „SYSREMOTEOPTIONTYPE-Systemansicht“ auf Seite 1475
- „SYSREMOTEOPTION-Systemansicht“ auf Seite 1475
- „SYSUSER-Systemansicht“ auf Seite 1506

Konsolidierte Ansicht SYSREMOTETYPES

Jede Zeile in der SYSREMOTETYPES-Ansicht beschreibt einen der SQL Remote-Nachrichtentypen sowie die Adresse des Publikationseigentümers.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSREMOTETYPES"
as select rt.type_id,rt.type_name,rt.publisher_address,rm.remarks
       from SYS.ISYSREMOTETYPE as rt
       left outer join SYS.ISYSREMARK as rm on(rt.object_id = rm.object_id)
```

Siehe auch

- „SYSREMOTETYPE-Systemansicht“ auf Seite 1476
- „SYSREMARK-Systemansicht“ auf Seite 1474

Konsolidierte Ansicht SYSREMOTEUERS

Jede Zeile der SYSREMOTEUERS-Ansicht beschreibt eine Benutzer-ID mit REMOTE-Systemprivileg (einen Subskribenten) zusammen mit dem Status von SQL Remote-Nachrichten, die an diesen Benutzer und von diesem Benutzer gesendet wurden.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSREMOTEUERS" AS SELECT u.user_name, r.consolidate,
t.type_name, r.address, r.frequency, r.send_time,
(if r.frequency = 'A' then NULL
else if r.frequency = 'P' then
if r.time_sent IS NULL then CURRENT_TIMESTAMP
else (select min( minutes( dateadd(mi, PROPERTY('TimeZoneAdjustment'),
a.time_sent),
60*hour(a.send_time) + minute( seconds( a.send_time, 59 ) ) ) )
FROM SYS.ISYSREMOTEUER a WHERE a.frequency = 'P' AND a.send_time =
r.send_time ) endif
else if CURRENT DATE + r.send_time > coalesce( dateadd(mi,
PROPERTY('TimeZoneAdjustment'), r.time_sent), CURRENT_TIMESTAMP)
then CURRENT DATE + r.send_time
else CURRENT DATE + r.send_time + 1
endif endif endif) as next_send, r.log_send ,
dateadd(mi, PROPERTY('TimeZoneAdjustment'), r.time_sent)
as time_sent , r.log_sent, r.confirm_sent, r.send_count, r.resend_count,
dateadd(mi, PROPERTY('TimeZoneAdjustment'), r.time_received) as
time_received ,
r.log_received, r.confirm_received, r.receive_count, r.rereceive_count ,
TODATETIMEOFFSET( r.time_sent, 0 ) as time_sent_utc ,
TODATETIMEOFFSET( r.time_received, 0 ) as time_received_utc
FROM SYS.ISYSREMOTEUER r JOIN SYS.ISYSUSER u ON ( u.user_id = r.user_id )
JOIN SYS.ISYSREMOTETYPE t ON ( t.type_id = r.type_id )
```

Siehe auch

- „SYSREMOTEUER-Systemansicht“ auf Seite 1476
- „SYSUSER-Systemansicht“ auf Seite 1506
- „SYSREMOTETYPE-Systemansicht“ auf Seite 1476

Konsolidierte Ansicht SYSSUBSCRIPTIONS

Jede Zeile beschreibt eine Subskription einer Benutzer-ID (die das REMOTE-Systemprivileg haben muss) für eine Publikation.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSSUBSCRIPTIONS"
as select p.publication_name,u.user_name,s.subscribe_by,s.created,
s.started
from SYS.ISYSSUBSCRIPTION as s
join SYS.ISYSPUBLICATION as p on(p.publication_id = s.publication_id)
join SYS.ISYSUSER as u on u.user_id = s.user_id
```

Siehe auch

- „SYSSUBSCRIPTION-Systemansicht“ auf Seite 1489
- „SYSPUBLICATION-Systemansicht“ auf Seite 1473
- „SYSUSER-Systemansicht“ auf Seite 1506

Konsolidierte Ansicht SYSSYNC2

Die SYSSYNC2-Ansicht bietet öffentlichen Zugriff auf die in der SYSSYNC-Systemansicht enthaltenen Daten (Informationen zur MobiLink-Synchronisation), ohne potenziell sensible Daten anzuzeigen.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

Werte in den Spalten server_connect und option werden für Benutzer verborgen, die nicht das SELECT ANY TABLE-Systemprivileg haben.

```
ALTER VIEW "SYS"."SYSSYNC2"
as select ISYSSYNC.sync_id,
ISYSSYNC.type,
ISYSSYNC.publication_id,
ISYSSYNC.progress,
ISYSSYNC.site_name,
SYS.HIDE_FROM_NON_DBA(ISYSSYNC."option") as "option",
SYS.HIDE_FROM_NON_DBA(ISYSSYNC.server_connect) as server_connect,
ISYSSYNC.server_conn_type,
ISYSSYNC.last_download_time,
ISYSSYNC.last_upload_time,
ISYSSYNC.created,
ISYSSYNC.log_sent,
ISYSSYNC.generation_number,
ISYSSYNC.extended_state,
ISYSSYNC.script_version,
ISYSSYNC.subscription_name
from SYS.ISYSSYNC
```

Siehe auch

- „SYSSYNC-Systemansicht“ auf Seite 1489

Konsolidierte Ansicht SYSSYNCPUBLICATIONDEFAULTS

Die SYSSYNCPUBLICATIONDEFAULTS-Ansicht liefert die Standard-Synchronisationseinstellungen in Verbindung mit der MobiLink-Synchronisation zugrunde liegenden Publikationen.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

Werte in den Spalten `server_connect` und `option` werden für Benutzer verborgen, die nicht das `SELECT ANY TABLE`-Systemprivileg haben.

```
ALTER VIEW "SYS"."SYSSYNCPUBLICATIONDEFAULTS"
as select s.sync_id,
p.publication_name,
SYS.HIDE_FROM_NON_DBA(s."option") as "option",
SYS.HIDE_FROM_NON_DBA(s.server_connect) as server_connect,
s.server_conn_type
from SYS.ISYSSYNC as s join SYS.ISYSPUBLICATION as p on(p.publication_id
= s.publication_id) where
s.site_name is null
```

Siehe auch

- „SYSSYNC-Systemansicht“ auf Seite 1489
- „SYSPUBLICATION-Systemansicht“ auf Seite 1473

Konsolidierte Ansicht SYSSYNCS

Die SYSSYNCS-Ansicht enthält Informationen, die sich auf die MobiLink-Synchronisation beziehen.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

Werte in den Spalten `server_connect` und `option` werden für Benutzer verborgen, die nicht das `SELECT ANY TABLE`-Systemprivileg haben.

```
ALTER VIEW "SYS"."SYSSYNCS"
as select p.publication_name,s.progress,s.site_name,
SYS.HIDE_FROM_NON_DBA(s."option") as "option",
SYS.HIDE_FROM_NON_DBA(s.server_connect) as server_connect,
s.server_conn_type,s.last_download_time,
s.last_upload_time,s.created,s.log_sent,s.generation_number,
s.extended_state
from SYS.ISYSSYNC as s
left outer join SYS.ISYSPUBLICATION as p
on p.publication_id = s.publication_id
```

Siehe auch

- „SYSSYNC-Systemansicht“ auf Seite 1489
- „SYSPUBLICATION-Systemansicht“ auf Seite 1473

Konsolidierte Ansicht SYSSYNCSSCRIPTS

Jede Zeile in der SYSSYNCSSCRIPTS-Ansicht kennzeichnet eine gespeicherte Prozedur für skriptgesteuerte Uploads von MobiLink. Diese Ansicht ist mit der SYSSYNCSSCRIPT-Systemansicht fast

identisch, nur dass hier die Werte in einem lesbaren Format dargestellt sind, und nicht als unformatierte Daten.

```
ALTER VIEW "SYS"."SYSSYNCSRIPTS"
as select p.publication_name,
        t.table_name,
        case s.type
        when 0 then 'upload insert'
        when 1 then 'upload delete'
        when 2 then 'upload update'
        else 'unknown'
        end as type,
        c.proc_name
from SYS.ISYSSYNCSRIPT as s
join SYS.ISYSPUBLICATION as p on p.object_id = s.pub_object_id
join SYS.ISYSTAB as t on t.object_id = s.table_object_id
join SYS.ISYSPROCEDURE as c on c.object_id = s.proc_object_id
```

Siehe auch

- „SYSSYNCSRIPT-Systemansicht“ auf Seite 1491
- „SYSPUBLICATION-Systemansicht“ auf Seite 1473
- „SYSTAB-Systemansicht“ auf Seite 1492
- „SYSPROCEDURE-Systemansicht“ auf Seite 1469
- „Skriptgesteuerter Upload“ [*MobiLink - Clientadministration*]

Konsolidierte Ansicht SYSSYNCSUBSCRIPTIONS

Die SYSSYNCSUBSCRIPTIONS-Ansicht enthält die Synchronisationseinstellungen, die mit MobiLink-Synchronisationssubskriptionen zusammenhängen.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

Werte in den Spalten server_connect und option werden für Benutzer verborgen, die nicht das SELECT ANY TABLE-Systemprivileg haben.

```
ALTER VIEW "SYS"."SYSSYNCSUBSCRIPTIONS"
as select s.sync_id,
        p.publication_name,
        s.progress,
        s.site_name,
        SYS.HIDE_FROM_NON_DBA(s."option") as "option",
        SYS.HIDE_FROM_NON_DBA(s.server_connect) as server_connect,
        s.server_conn_type,
        s.last_download_time,
        s.last_upload_time,
        s.created,
        s.log_sent,
        s.generation_number,
        s.extended_state
from SYS.ISYSSYNC as s join SYS.ISYSPUBLICATION as p on (p.publication_id
= s.publication_id)
where s.publication_id is not null and
s.site_name is not null and exists
```

```
(select 1 from SYS.SYSSYNCUSERS as u
 where s.site_name = u.site_name)
```

Siehe auch

- „SYSSYNC-Systemansicht“ auf Seite 1489
- „SYSPUBLICATION-Systemansicht“ auf Seite 1473
- „Konsolidierte Ansicht SYSSYNCUSERS“ auf Seite 1532

Konsolidierte Ansicht SYSSYNCUSERS

Eine Ansicht der Synchronisationseinstellungen der MobiLink-Synchronisationsbenutzer.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

Werte in den Spalten server_connect und option werden für Benutzer verborgen, die nicht das SELECT ANY TABLE-Systemprivileg haben.

```
ALTER VIEW "SYS"."SYSSYNCUSERS"
as select ISYSSYNC.sync_id,
        ISYSSYNC.site_name,
        SYS.HIDE_FROM_NON_DBA(ISYSSYNC."option") as "option",
        SYS.HIDE_FROM_NON_DBA(ISYSSYNC.server_connect) as server_connect,
        ISYSSYNC.server_conn_type
from SYS.ISYSSYNC where
        ISYSSYNC.publication_id is null
```

Siehe auch

- „SYSSYNC-Systemansicht“ auf Seite 1489

Konsolidierte Ansicht SYSTABAUTH

Die SYSTABAUTH-Ansicht enthält Informationen aus der SYSTABLEPERM-Systemansicht, aber in einem lesbaren Format.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSTABAUTH"( grantor,
 grantee, screator, stname, tcreator, tname,
 selectauth, insertauth, deleteauth,
 updateauth, updatecols, alterauth, referenceauth,
 loadauth, truncateauth )
as select u1.user_name, u2.user_name, u3.user_name, tab1.table_name,
        u4.user_name, tab2.table_name, tp.selectauth, tp.insertauth,
        tp.deleteauth, tp.updateauth, tp.updatecols, tp.alterauth,
        tp.referenceauth, tp.loadauth, tp.truncateauth
from SYS.ISYSTABLEPERM as tp
        join SYS.ISYSUSER as u1 on u1.user_id = tp.grantor
        join SYS.ISYSUSER as u2 on u2.user_id = tp.grantee
```

Siehe auch

- ## Konsolidierte Ansicht SYSTRIGGERS

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

Siehe auch

- Copyright © 2013, SAP AG oder ein SAP-Konzernunternehmen. - SAP Sybase SQL Anywhere 15.3

Konsolidierte Ansicht SYSUSEROPTIONS

Die SYSUSEROPTIONS-Ansicht enthält die aktiven Optionseinstellungen für jeden Benutzer. Wenn ein Benutzer keine Einstellung für eine Option hat, zeigt diese Ansicht die öffentliche Einstellung für die Option an.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSUSEROPTIONS"( user_name,
    "option",setting )
as select u.user_name,
    o."option",
    isnull((select s.setting
        from SYS.ISYSOPTION as s
        where s.user_id = u.user_id
        and s."option" = o."option"),
    o.setting)
from SYS.SYSOPTIONS as o,SYS.ISYSUSER as u
where o.user_name = 'PUBLIC'
```

Siehe auch

- [„Konsolidierte Ansicht SYSOPTIONS“ auf Seite 1524](#)
- [„SYSUSER-Systemansicht“ auf Seite 1506](#)

Konsolidierte Ansicht SYSVIEWS

Jede Zeile in der SYSVIEWS-Ansicht beschreibt eine Ansicht, einschließlich ihrer Ansichtsdefinition.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSVIEWS"( vcreator,
    viewname,viewtext )
as select u.user_name,t.table_name,v.view_def
from SYS.ISYSTAB as t
    join SYS.ISYSVIEW as v on(t.object_id = v.view_object_id)
    join SYS.ISYSUSER as u on(u.user_id = t.creator)
```

Siehe auch

- [„SYSTAB-Systemansicht“ auf Seite 1492](#)
- [„SYSVIEW-Systemansicht“ auf Seite 1511](#)
- [„SYSUSER-Systemansicht“ auf Seite 1506](#)

Kompatibilitätsansichten

Kompatibilitätsansichten sind Ansichten, die für die Kompatibilität mit als SQL Anywhere 10 und älteren Versionen verfügbar gemacht werden. Wo immer möglich sollten Sie System- oder konsolidierte

Ansichten verwenden, weil die Unterstützung für einige Kompatibilitätsansichten in zukünftigen Versionen möglicherweise vermindert wird.

SYSCOLLATION-Kompatibilitätsansicht (nicht mehr empfohlen)

Die SYSCOLLATION-Kompatibilitätsansicht enthält Kollationssequenzinformationen für die Datenbank. Sie ist mittels integrierter Funktionen verfügbar und ist nicht im Katalog enthalten. Die Ansicht hat folgende Definition:

```
ALTER VIEW "SYS"."SYSCOLLATION"
  as select 1 as collation_id,
           DB_PROPERTY('Collation') as collation_label,
           DB_EXTENDED_PROPERTY('Collation','Description') as collation_name,
           cast(DB_EXTENDED_PROPERTY('Collation','LegacyData') as binary(1280)) as
  collation_order
```

Siehe auch

- „Liste der Datenbankeigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „DB_PROPERTY-Funktion [System]“ auf Seite 232
- „DB_EXTENDED_PROPERTY-Funktion [System]“ auf Seite 226

SYSCOLLATIONMAPPINGS-Kompatibilitätsansicht (nicht mehr empfohlen)

Die SYSCOLLATIONMAPPINGS-Kompatibilitätsansicht enthält nur eine Zeile mit der Kollationszuordnung der Datenbank. Sie ist mittels integrierter Funktionen verfügbar und ist nicht im Katalog enthalten. Die Ansicht hat folgende Definition:

```
ALTER VIEW "SYS"."SYSCOLLATIONMAPPINGS"
  as select DB_PROPERTY('Collation') as collation_label,
           DB_EXTENDED_PROPERTY('Collation','Description') as collation_name,
           DB_PROPERTY('Charset') as cs_label,
           DB_EXTENDED_PROPERTY('Collation','ASESensitiveSortOrder') as
  so_case_label,
           DB_EXTENDED_PROPERTY('Collation','ASEInsensitiveSortOrder') as
  so_caseless_label,
           DB_EXTENDED_PROPERTY('Charset','java') as jdk_label
```

Siehe auch

- „Liste der Datenbankeigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „DB_PROPERTY-Funktion [System]“ auf Seite 232
- „DB_EXTENDED_PROPERTY-Funktion [System]“ auf Seite 226

SYSCOLUMN-Kompatibilitätsansicht (nicht mehr empfohlen)

Die SYSCOLUMN-Ansicht wird für die Kompatibilität mit älteren Versionen der Software bereitgestellt, die eine SYSCOLUMN-Systemtabelle enthielten. Die frühere SYSCOLUMN-Tabelle wurde allerdings durch die ISYSTABCOL-Systemtabelle und ihre entsprechende SYSTABCOL system view ersetzt, die Sie stattdessen verwenden sollten.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSCOLUMN"
as select b.table_id,
       b.column_id,
       if c.sequence is null then 'N' else 'Y' endif as pkey,
       b.domain_id,
       b.nulls,
       b.width,
       b.scale,
       b.object_id,
       b.max_identity,
       b.column_name,
       r.remarks,
       b."default",
       b.user_type,
       b.column_type
from SYS.SYSTABCOL as b
     left outer join SYS.ISYSREMARK as r on(b.object_id = r.object_id)
     left outer join SYS.ISYSIDXCOL as c on(b.table_id = c.table_id and
       b.column_id = c.column_id and c.index_id = 0)
```

Siehe auch

- „SYSTABCOL-Systemansicht“ auf Seite 1495
- „SYSREMARK-Systemansicht“ auf Seite 1474
- „SYSIDXCOL-Systemansicht“ auf Seite 1457

SYSFILE-Kompatibilitätsansicht (nicht mehr empfohlen)

Jede Zeile in der SYSFILE-Systemansicht beschreibt einen DBSpace für eine Datenbank. Jede Datenbank besteht aus einem oder mehreren DBSpaces und jeder DBSpace entspricht einer Betriebssystemdatei.

DBSpaces werden für die Hauptdatenbankdatei, die temporäre Datei, die Transaktionslogdatei und die Transaktionsspiegeldatei automatisch erstellt. Informationen über Transaktionslog- und Transaktionslogspiegel-DBSpaces erscheinen in der SYSFILE-Systemansicht nicht.

```
ALTER VIEW "SYS"."SYSFILE"
as select b.dbfile_id as file_id,
       if b.dbpace_id = 0 and b.dbfile_id = 0 then
         db_property('File')
       else
         if b.dbpace_id = 15 and b.dbfile_id = 15 then
           db_property('TempFileName')
         else
```

```

        b.file_name
    endif
endif as file_name,
a.dbspace_name,
a.store_type,
b.lob_map,
b.dbspace_id
from SYS.ISYSDBSPACE as a
join SYS.ISYSDBFILE as b on(a.dbspace_id = b.dbspace_id)

```

Siehe auch

- „Vordefinierte DBSpaces“ [[SQL Anywhere Server - Datenbankadministration](#)]

SYSFKCOL-Kompatibilitätsansicht (nicht mehr empfohlen)

Jede Zeile von SYSFKCOL beschreibt die Zuordnung zwischen einer Fremdspalte in der Fremdtabelle einer Beziehung und der Primärspalte in der Primärtabelle. Diese Ansicht wird nicht mehr empfohlen. Verwenden Sie stattdessen die SYSIDX- und die SYSIDXCOL-Systemansichten.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```

ALTER VIEW "SYS"."SYSFKCOL"
as select a.table_id as foreign_table_id,
a.index_id as foreign_key_id,
a.column_id as foreign_column_id,
a.primary_column_id
from SYS.ISYSIDXCOL as a
,SYS.ISYSIDX as b
where a.table_id = b.table_id
and a.index_id = b.index_id
and b.index_category = 2

```

Siehe auch

- „SYSIDX-Systemansicht“ auf Seite 1456
- „SYSIDXCOL-Systemansicht“ auf Seite 1457

SYSFOREIGNKEY-Kompatibilitätsansicht (nicht mehr empfohlen)

Die SYSFOREIGNKEY-Ansicht wird für die Kompatibilität mit älteren Versionen von SQL Anywhere, die eine SYSFOREIGNKEY-Systemtabelle enthielten, zur Verfügung gestellt. Die frühere SYSFOREIGNKEY-Tabelle wurde allerdings durch die ISYSFKEY-Systemtabelle und ihre entsprechende SYSFKEY system view ersetzt, die Sie stattdessen verwenden sollten.

Ein Fremdschlüssel ist eine Beziehung zwischen zwei Tabellen, der Fremdtabelle und der Primärtabelle. Jeder Fremdschlüssel wird durch eine Zeile in SYSFOREIGNKEY und eine oder mehrere Zeilen in SYSFKCOL definiert. SYSFOREIGNKEY enthält allgemeine Informationen über den Fremdschlüssel, während SYSFKCOL die Spalten im Fremdschlüssel identifiziert und jede Spalte im Fremdschlüssel einer Spalte im Primärschlüssel der Primärtabelle zuordnet.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSFOREIGNKEY"
as select b.foreign_table_id,
  b.foreign_index_id as foreign_key_id,
  a.object_id,
  b.primary_table_id,
  p.root,
  b.check_on_commit,
  b.nulls,
  a.index_name as role,
  r.remarks,
  b.primary_index_id,
  a.not_enforced as fk_not_enforced,
  10 as hash_limit
from(SYS.ISYSIDX as a left outer join SYS.ISYSPHYSIDX as p on(a.table_id
= p.table_id and a.phys_index_id = p.phys_index_id))
  left outer join SYS.ISYSREMARK as r on(a.object_id = r.object_id)
  ,SYS.ISYSFKEY as b
where a.table_id = b.foreign_table_id
and a.index_id = b.foreign_index_id
```

Siehe auch

- [„SYSIDX-Systemansicht“ auf Seite 1456](#)
- [„SYSPHYSIDX-Systemansicht“ auf Seite 1468](#)
- [„SYSREMARK-Systemansicht“ auf Seite 1474](#)
- [„SYSFKEY-Systemansicht“ auf Seite 1452](#)

SYSGROUP-Kompatibilitätsansicht

In der SYSGROUP-Systemansicht gibt es eine Zeile für jedes Mitglied jeder Gruppe. Diese Ansicht beschreibt die Viele-zu-Viele-Beziehung zwischen Gruppen und Mitgliedern. Eine Gruppe kann viele Mitglieder haben und ein Benutzer kann Mitglied vieler Gruppen sein.

Spaltenname	Datentyp	Beschreibung
group_id	UNSIGNED INT	Die Benutzernummer der Gruppe
group_member	UNSIGNED INT	Die Benutzernummer eines Mitglieds

Integritätsregeln für die zugrunde liegende Systemtabelle

```
PRIMARY KEY (group_id, group_member)

FOREIGN KEY group_id (group_id) REFERENCES SYS.ISYSUSER (user_id)

FOREIGN KEY group_member (group_member) REFERENCES SYS.ISYSUSER (user_id)
```

SYSGROUPS-Kompatibilitätsansicht

In der SYSGROUPS-Ansicht gibt es eine Zeile für jedes Mitglied jeder Gruppe. Diese Ansicht beschreibt die Viele-zu-Viele-Beziehung zwischen Gruppen und Mitgliedern. Eine Gruppe kann viele Mitglieder haben und ein Benutzer kann Mitglied vieler Gruppen sein.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSGROUPS"( group_name,
    member_name )
as select g.user_name,u.user_name
    from SYS.ISYSROLEGRANT,SYS.ISYSUSER as g,SYS.ISYSUSER as u
    where ISYSROLEGRANT.role_id = g.user_id and ISYSROLEGRANT.grantee =
u.user_id and(
    u.user_name in( 'SYS_SPATIAL_ADMIN_ROLE' )
    or u.user_id <= 2147483648) and(
    g.user_type = (0x02|0x04|0x08)
    or g.user_name in( 'SYS','PUBLIC','dbo','diagnostics',
    'rs_systabgroup','SA_DEBUG','SYS_SPATIAL_ADMIN_ROLE' ) )
```

Siehe auch

- „SYSUSER-Systemansicht“ auf Seite 1506
- „SYSGROUP-Kompatibilitätsansicht“ auf Seite 1538

SYSINDEX-Kompatibilitätsansicht (nicht mehr empfohlen)

Die SYSINDEX-Ansicht wird für die Kompatibilität mit älteren Versionen von SQL Anywhere, die eine SYSINDEX-Systemtabelle enthielten, zur Verfügung gestellt. Die frühere SYSINDEX-Systemtabelle wurde allerdings durch die ISYSIDX-Systemtabelle und ihre entsprechende SYSIDX system view ersetzt, die Sie statt dessen verwenden sollten.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSINDEX"
as select b.table_id,
    b.index_id,
    b.object_id,
    p.root,
    b.dbspace_id,
    case b."unique"
    when 1 then 'Y'
    when 2 then 'U'
    when 3 then 'M'
    when 4 then 'N'
    when 5 then 'Y'
    else 'I'
    end as "unique",
    t.creator,
    b.index_name,
    r.remarks,
    10 as hash_limit,
```

```
b.dbspace_id as file_id
from(SYS.ISYSIDX as b left outer join SYS.ISYSPHYIDX as p on(b.table_id
= p.table_id and b.phys_index_id = p.phys_index_id))
left outer join SYS.ISYSREMARK as r on(b.object_id = r.object_id)
,SYS.ISYSTAB as t
where t.table_id = b.table_id
and b.index_category = 3
```

Siehe auch

- „SYSIDX-Systemansicht“ auf Seite 1456
- „SYSPHYIDX-Systemansicht“ auf Seite 1468
- „SYSTABLE-Kompatibilitätsansicht (nicht mehr empfohlen)“ auf Seite 1541
- „SYSREMARK-Systemansicht“ auf Seite 1474

SYSINFO-Kompatibilitätsansicht (nicht mehr empfohlen)

Die SYSINFO-Ansicht zeigt die Merkmale der Datenbank an, die beim Erstellen der Datenbank festgelegt wurden. Sie enthält immer nur eine Zeile. Diese Ansicht ist mittels integrierter Funktionen verfügbar und ist nicht im Katalog enthalten. Die SYSINFO-Ansicht hat die folgende Definition:

```
ALTER VIEW "SYS"."SYSINFO" ( page_size,
encryption,
blank_padding,
case_sensitivity,
default_collation,
database_version )
as select db_property('PageSize'),
if db_property('Encryption') <> 'None' then 'Y' else 'N' endif,
if db_property('BlankPadding') = 'On' then 'Y' else 'N' endif,
if db_property('CaseSensitive') = 'On' then 'Y' else 'N' endif,
db_property('Collation'),
NULL
```

Siehe auch

- „Liste der Datenbankeigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „DB_PROPERTY-Funktion [System]“ auf Seite 232
- „DB_EXTENDED_PROPERTY-Funktion [System]“ auf Seite 226

SYSIXCOL-Kompatibilitätsansicht (nicht mehr empfohlen)

Die SYSIXCOL-Ansicht wird für die Kompatibilität mit älteren Versionen von SQL Anywhere, die eine SYSIXCOL-Systemtabelle enthielten, verfügbar gemacht. Die SYSIXCOL-Systemtabelle wurde allerdings durch die ISYIDXCOL-Systemtabelle und ihre entsprechende SYSIDXCOL-Systemansicht ersetzt. Sie sollten besser die SYSIDXCOL system view verwenden.

Jede Zeile in der SYSIXCOL-Ansicht beschreibt eine Spalte in einem Index. Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSIXCOL"
as select a.table_id,
```

```

a.index_id,
a.sequence,
a.column_id,
a."order"
from SYS.ISYSIDXCOL as a
,SYS.ISYSIDX as b
where a.table_id = b.table_id
and a.index_id = b.index_id
and b.index_category = 3

```

Siehe auch

- „SYSIDX-Systemansicht“ auf Seite 1456
- „SYSIDXCOL-Systemansicht“ auf Seite 1457

SYSTABLE-Kompatibilitätsansicht (nicht mehr empfohlen)

Die SYSTABLE-Ansicht wird für die Kompatibilität mit älteren Versionen von SQL Anywhere, die eine SYSTABLE-Systemtabelle enthielten, verfügbar gemacht. Die frühere SYSTABLE-Systemtabelle wurde allerdings durch die ISYSTAB-Systemtabelle und ihre entsprechende SYSTAB system view ersetzt, die Sie statt dessen verwenden sollten.

Jede Zeile der SYSTABLE-Ansicht beschreibt eine Tabelle in der Datenbank.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```

ALTER VIEW "SYS"."SYSTABLE"
as select b.table_id,
       b.file_id,
       b.count,
       0 as first_page,
       b.commit_action as last_page,
       COALESCE(ph.root,0) as primary_root,
       b.creator,
       0 as first_ext_page,
       0 as last_ext_page,
       b.table_page_count,
       b.ext_page_count,
       b.object_id,
       b.table_name,
       b.table_type_str as table_type,
       v.view_def,
       r.remarks,
       b.replicate,
       p.existing_obj,
       p.remote_location,
       'T' as remote_objtype,
       p.srvid,
       case b.server_type
       when 1 then 'SA'
       when 2 then 'IQ'
       when 3 then 'OMNI'
       else 'INVALID'
       end as server_type,
       10 as primary_hash_limit,
       0 as page_map_start,

```

```
s.source,  
b."encrypted"  
from SYS.SYSTAB as b  
  left outer join SYS.ISYSREMARK as r on(b.object_id = r.object_id)  
  left outer join SYS.ISYSSOURCE as s on(b.object_id = s.object_id)  
  left outer join SYS.ISYSVIEW as v on(b.object_id = v.view_object_id)  
  left outer join SYS.ISYSPROXYTAB as p on(b.object_id =  
p.table_object_id)  
  left outer join(SYS.ISYSIDX as i left outer join SYS.ISYSPHYSIDX as ph  
on(i.table_id = ph.table_id  
  and i.phys_index_id = ph.phys_index_id)) on(b.table_id = i.table_id  
and i.index_category = 1  
  and i.index_id = 0)
```

Siehe auch

- „SYSTAB-Systemansicht“ auf Seite 1492
- „SYSREMARK-Systemansicht“ auf Seite 1474
- „SYSSOURCE-Systemansicht“ auf Seite 1485
- „SYSVIEW-Systemansicht“ auf Seite 1511
- „SYSPROXYTAB-Systemansicht“ auf Seite 1472
- „SYSIDX-Systemansicht“ auf Seite 1456
- „SYSPHYSIDX-Systemansicht“ auf Seite 1468

SYSUSERAUTH-Kompatibilitätsansicht (nicht mehr empfohlen)

Jede Zeile der SYSUSERAUTH-Ansicht beschreibt einen Benutzer, ohne dessen user_id anzuzeigen. Stattdessen wird jeder Benutzer durch seinen Benutzernamen identifiziert. Weil in dieser Ansicht Kennwörter angezeigt werden, benötigen Sie das SELECT ANY TABLE-Systemprivileg, um die Daten anzeigen zu können.

Die SYSUSERAUTH-Ansicht wird aus Gründen der Kompatibilität mit älteren Versionen der Software bereitgestellt. Verwenden Sie stattdessen die konsolidierte Ansicht SYSROLEGRANTS.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

Hinweis

Obwohl der Titel dieser Ansicht das Wort auth (für "authorities", Berechtigungen) enthält, basiert das Sicherheitsmodell auf Rollen und Privilegien. Die Daten in der Ansicht werden daher mithilfe von Rolleninformationen aus den in der Ansichtsdefinition genannten Tabellen und Ansichten zusammengestellt.

```
ALTER VIEW "SYS"."SYSUSERAUTH" ( name,  
  password,resourceauth,dbaauth,scheduleauth,user_group )  
as select  
SYSUSERPERM.user_name,SYSUSERPERM.password,SYSUSERPERM.resourceauth,SYSUSERPERM.  
dbaauth,SYSUSERPERM.scheduleauth,SYSUSERPERM.user_group  
from SYS.SYSUSERPERM
```

Siehe auch

- „Konsolidierte Ansicht SYSROLEGRANTS“ auf Seite 1480
- „SYSROLEGRANT-Systemansicht“ auf Seite 1478
- „SYSROLEGRANTEXT-Systemansicht“ auf Seite 1479

SYSUSERAUTHORITY-Kompatibilitätsansicht (nicht mehr empfohlen)

Die SYSUSERAUTHORITY-Ansicht wird aus Gründen der Kompatibilität mit älteren Versionen der Software bereitgestellt. Verwenden Sie stattdessen die konsolidierte Ansicht SYSROLEGRANTS.

Jede Zeile in der SYSUSERAUTHORITY-Systemansicht beschreibt eine Berechtigung, die einer Benutzer-ID erteilt wurde.

Hinweis

Obwohl der Titel dieser Ansicht das Wort authority (Berechtigung) enthält, basiert das Sicherheitsmodell auf Rollen und Privilegien. Die Daten in der Ansicht werden daher mithilfe von Rolleninformationen aus den in der Ansichtsdefinition genannten Tabellen und Ansichten zusammengestellt.

```
ALTER VIEW "SYS"."SYSUSERAUTHORITY" as
  select ISYSROLEGRANT.grantee as user_id,
         sp_auth_sys_role_info.auth
  from SYS.ISYSROLEGRANT
       natural join dbo.sp_auth_sys_role_info()
 where ISYSROLEGRANT.grant_type <> (0x02|0x04) and
       not ISYSROLEGRANT.grantee = any(select sp_auth_sys_role_info.role_id
  from dbo.sp_auth_sys_role_info()) union
  select ISYSUSER.user_id,
         cast('GROUP' as varchar(20)) as auth
  from SYS.ISYSUSER
       where ISYSUSER.user_name
 in( 'SYS', 'PUBLIC', 'diagnostics', 'SYS_SPATIAL_ADMIN_ROLE', 'rs_systabgroup', 'S
A_DEBUG', 'dbo' ) union
  select ISYSUSER.user_id,
         cast('GROUP' as varchar(20)) as auth
  from SYS.ISYSUSER
       where ISYSUSER.user_type = (0x02|0x04|0x08) union
  select cast(opt.setting as unsigned integer) as user_id,
         cast('PUBLISH' as varchar(20)) as auth
  from SYS.ISYSOPTION as opt
       where opt."option" like '%db_publisher%' and opt.setting not like '%-1%'
```

Siehe auch

- „Konsolidierte Ansicht SYSROLEGRANTS“ auf Seite 1480
- „SYSROLEGRANT-Systemansicht“ auf Seite 1478
- „SYSROLEGRANTEXT-Systemansicht“ auf Seite 1479

SYSUSERLIST-Kompatibilitätsansicht (nicht mehr empfohlen)

Die SYSUSERAUTH-Ansicht steht für die Kompatibilität mit älteren Versionen der Software zur Verfügung.

Jede Zeile in der SYSUSERLIST-Ansicht beschreibt einen Benutzer, ohne dessen user_id und Kennwort anzuzeigen. Jeder Benutzer wird durch seinen Benutzernamen identifiziert.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSUSERLIST" ( name ,
    resourceauth, dbaauth, scheduleauth, user_group )
as select
SYSUSERPERM.user_name, SYSUSERPERM.resourceauth, SYSUSERPERM.dbaauth, SYSUSERPERM
M.scheduleauth, SYSUSERPERM.user_group
from SYS.SYSUSERPERM
```

Siehe auch

- „SYSUSERPERM-Kompatibilitätsansicht (nicht mehr empfohlen)“ auf Seite 1544

SYSUSERPERM-Kompatibilitätsansicht (nicht mehr empfohlen)

Jede Zeile der SYSUSERPERM-Ansicht beschreibt eine Benutzer-ID. Sie müssen das SELECT ANY TABLE-Systemprivileg haben, um Daten in dieser Ansicht anzeigen zu können.

Diese Ansicht wird nicht mehr empfohlen, weil sie nur die in früheren Versionen verfügbaren Berechtigungen anzeigt. Sie sollten Ihre Anwendung so ändern, dass sie stattdessen die konsolidierte Ansicht SYSROLEGRANTS verwendet.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```
ALTER VIEW "SYS"."SYSUSERPERM"
as select b.user_id,
    b.object_id,
    b.user_name,
    b.password,
    if AA.resourceauth is not null and AA.resourceauth > 0 then
    'Y' else 'N' endif as resourceauth,
    if AA.dbaauth is not null and AA.dbaauth > 0 then
    'Y' else 'N' endif as dbaauth,
    'N' as scheduleauth,
    if exists(select * from SYS.ISYSOPTION as opt
        where opt."option" like '%db_publisher%' and opt.setting not like '%-1'
        and b.user_id = cast(opt.setting as integer)) then
    'Y' else 'N' endif as publishauth,
    if AA.remotedbaauth is not null and AA.remotedbaauth > 0 then
```

```

        'Y' else 'N' endif as remotdbaauth,
        if b.user_type = (0x02|0x04|0x08) or b.user_name
in( 'SYS','PUBLIC','diagnostics','SYS_SPATIAL_ADMIN_ROLE','rs_systabgroup','S
A_DEBUG','dbo' ) then
        'Y' else 'N' endif as user_group,
        r.remarks
from SYS.ISYSUSER as b
        left outer join SYS.ISYSREMARK as r on(b.object_id = r.object_id)
        left outer join(select sum(if sp_auth_sys_role_info.auth = 'RESOURCE'
then 1 else 0 endif) as resourceauth,
        sum(if sp_auth_sys_role_info.auth = 'DBA' then 1 else 0 endif) as
dbaauth,
        sum(if sp_auth_sys_role_info.auth = 'REMOTE DBA' then 1 else 0
endif) as remotdbaauth,
        ISYSROLEGRANT.grantee
from SYS.ISYSROLEGRANT natural join dbo.sp_auth_sys_role_info()
where ISYSROLEGRANT.grant_type <> (0x02|0x04)
and sp_auth_sys_role_info.auth in( 'DBA','RESOURCE','REMOTE DBA' )
group by ISYSROLEGRANT.grantee) as AA
on(AA.grantee = b.user_id)

```

Siehe auch

- „Konsolidierte Ansicht SYSROLEGRANTS“ auf Seite 1480
- „SYSROLEGRANT-Systemansicht“ auf Seite 1478
- „SYSROLEGRANTEXT-Systemansicht“ auf Seite 1479
- „SYSUSER-Systemansicht“ auf Seite 1506

SYSUSERPERMS-Kompatibilitätsansicht (nicht mehr empfohlen)

Diese Ansicht wird nicht mehr empfohlen, weil sie nur die in früheren Versionen verfügbaren Datenbankberechtigungen und Berechtigungen anzeigt. Sie sollten Ihre Anwendung so ändern, dass sie stattdessen die konsolidierte Ansicht SYSROLEGRANTS verwendet.

Jede Zeile der SYSUSERPERMS-Ansicht beschreibt eine Benutzer-ID. Kennwortinformationen werden jedoch nicht einbezogen. Allen Benutzern ist es erlaubt, diese Ansicht zu verwenden.

Die Tabellen und Spalten, aus denen diese Ansicht besteht, werden von der SQL-Anweisung unten geliefert. Um weitere Informationen über eine bestimmte Tabelle oder Spalte zu erhalten, verwenden Sie die unter der Ansichtsdefinition verfügbaren Links.

```

ALTER VIEW "SYS"."SYSUSERPERMS"
as select
SYSUSERPERM.user_id,SYSUSERPERM.user_name,SYSUSERPERM.resourceauth,SYSUSERPER
M.dbaauth,

SYSUSERPERM.scheduleauth,SYSUSERPERM.user_group,SYSUSERPERM.publishauth,SYSUS
ERPERM.remotedbaauth,SYSUSERPERM.remarks
from SYS.SYSUSERPERM

```

Siehe auch

- „Konsolidierte Ansicht SYSROLEGRANTS“ auf Seite 1480
- „SYSROLEGRANT-Systemansicht“ auf Seite 1478
- „SYSROLEGRANTEXT-Systemansicht“ auf Seite 1479
- „SYSUSER-Systemansicht“ auf Seite 1506

Ansichten für Transact-SQL-Kompatibilität

Die Systemkataloge von Adaptive Server Enterprise und von SQL Anywhere sind unterschiedlich. Die Systemtabellen und -ansichten in Adaptive Server Enterprise gehören dem Benutzer dbo und sind zum Teil in der Master-Datenbank, zum Teil in der sybsecurity-Datenbank und zum Teil in jeder einzelnen Datenbank gespeichert. Die SQL Anywhere-Systemtabellen und -ansichten gehören dem besonderen Benutzer SYS und sind in jeder Datenbank separat vorhanden.

Um die Vorbereitung von kompatiblen Anwendungen zu unterstützen, stellt SQL Anywhere die folgende Gruppe von Ansichten zur Verfügung, die dem besonderen Benutzer dbo gehören, was ihren Gegenstücken in Adaptive Server Enterprise entspricht. Wenn architektonische Unterschiede den Inhalt einer bestimmten Tabelle oder einer Ansicht von Adaptive Server Enterprise in einem Kontext von SQL Anywhere bedeutungslos machen, ist die Ansicht leer und enthält nur die Spaltennamen und Datentypen.

Ansichtsname	Beschreibung
syscolumns	Eine Zeile für jede Spalte in einer Tabelle oder Ansicht und für jeden Parameter in einer Prozedur.
syscomments	Eine oder mehrere Zeilen für jede Ansicht, Regel, Prozedur, jeden Standardwert und Trigger und entsprechend der SQL-Definitionsanweisung.
sysindexes	Eine Zeile für jeden clustered oder nonclustered-Index, eine Zeile für jede Tabelle ohne Indizes und eine zusätzliche Zeile für jede Tabelle, die Text- oder Bilddaten enthält.
sysobjects	Eine Zeile für jede Tabelle, Ansicht, Prozedur, Regel, jeden Trigger-Standardwert, jedes Log oder (nur in tempdb) temporäre Objekt.
systypes	Eine Zeile für jeden systemeigenen oder benutzerdefinierten Datentyp.
sysusers	Eine Zeile für jeden in der Datenbank zugelassenen Benutzer.
syslogins	Eine Zeile für jedes gültige Benutzerkonto.

Index

Symbole

- !<
 - Vergleichsoperator,9
- !=
 - Vergleichsoperator,9
- !>
 - Vergleichsoperator,9
- %, Kommentarindikator
 - Info,92
- %, Operator
 - Modulo-Funktion,317
- &
 - Bit-Operator,20
- , Kommentarindikator
 - Info,92
- /*, Kommentarindikator
 - Info,92
- //, Kommentarindikator
 - Info,92
- 0x
 - binäre Literale,6
- 29. Februar
 - Info,122
- <
 - Vergleichsoperator,9
- <=
 - Vergleichsoperator,9
- <>
 - Vergleichsoperator,9
- =
 - Vergleichsoperator,9
- >
 - Vergleichsoperator,9
- >=
 - Vergleichsoperator,9
- @@char_convert, globale Variable
 - Info,88
- @@client_csid, globale Variable
 - Info,88
- @@client_cname, globale Variable
 - Info,88
- @@connections, globale Variable
 - Info,88
- @@cpu_busy, globale Variable
 - Info,88
- @@dbts, globale Variable
 - Info,88
- @@error, globale Variable
 - Info,88
- @@fetch_status, globale Variable
 - Info,88
- @@identity, globale Variable
 - Beschreibung,91
 - Info,88
 - Trigger,92
- @@idle, globale Variable
 - Info,88
- @@io_busy, globale Variable
 - Info,88
- @@isolation, globale Variable
 - Info,88
- @@langid, globale Variable
 - Info,88
- @@language, globale Variable
 - Info,88
- @@max_connections, globale Variable
 - Info,88
- @@maxcharlen, globale Variable
 - Info,88
- @@ncharsize, globale Variable
 - Info,88
- @@nestlevel, globale Variable
 - Info,88
- @@pack_errors, globale Variable
 - Info,88
- @@pack_received, globale Variable
 - Info,88
- @@pack_sent, globale Variable
 - Info,88
- @@procid, globale Variable
 - Info,88
- @@rowcount, globale Variable
 - Info,88
- @@servername, globale Variable
 - Info,88
- @@spid, globale Variable
 - Info,88
- @@sqlstatus, globale Variable
 - Info,88
- @@textsize, globale Variable
 - Info,88
- @@thresh_hysteresis, globale Variable

- Info,88
- @@timeticks, globale Variable
 - Info,88
- @@total_errors, globale Variable
 - Info,88
- @@total_read, globale Variable
 - Info,88
- @@total_write, globale Variable
 - Info,88
- @@tranchained, globale Variable
 - Info,88
- @@transtate, globale Variable
 - Info,88
- @@version, globale Variable
 - Info,88
- @HttpMethod
 - HTTP-HEADER-Funktion,284
- @HttpQueryString
 - HTTP-HEADER-Funktion,284
- @HttpStatus header
 - HTTP_RESPONSE_HEADER-Funktion,286
- @HttpStatus-Header
 - sa_set_http_header_info-Systemprozedur,1323
- @HttpURI
 - HTTP-HEADER-Funktion,284
- @HttpVersion
 - HTTP-HEADER-Funktion,284
- @mp:id-Meta-Eigenschaft
 - OPENXML-Operator,13
- @mp:localname-Meta-Eigenschaft
 - OPENXML-Operator,13
- @mp:namespaceuri-Meta-Eigenschaft
 - OPENXML-Operator,13
- @mp:prefix-Meta-Eigenschaft
 - OPENXML-Operator,13
- @mp:xmltext-Meta-Eigenschaft
 - OPENXML-Operator,13
- [ESQL]
 - Anweisungsindikatoren,448
- [Interactive SQL]
 - Anweisungsindikatoren,448
- [SP]
 - Anweisungsindikatoren,448
- [T-SQL]
 - Anweisungsindikatoren,448
- ^
 - Bit-Operator,20
- |

- Bit-Operator,20

- ~

- Bit-Operator,20

- NEAR-Ausdruck ersetzen,60

A

- Abfrageausdruck

- allgemeines Element der SQL-Syntax,445

- Abfrageblock

- allgemeines Element der SQL-Syntax,445

- allgemeines Element in SQL-Syntax,445

- Abgeleitete Tabellen

- Beispiel in einer FROM-Klausel,866

- FROM-Klausel, SQL-Syntax,863

- in Unterabfragen verwenden,774

- lateral,866

- Abhängige Variable

- Regressionszeile,352

- Abhängigkeiten

- sa_dependent_views-Systemprozedur

- verwenden,1202

- Ablehnen

- Privilegien erteilen,1009

- Abrufen

- Binärdaten von Spalten,876

- Informationen von Deskriptorbereichen,878

- lange Spaltennamen,796

- mehrfache Ergebnismengen,1003

- Optionswerte,880

- Zeilen von Cursor,853

- ABS-Funktion

- Syntax,166

- ABSOLUTE-Klausel

- FETCH-Anweisung,854

- ACCENT-Klausel

- CREATE DATABASE-Anweisung,585

- AccentSensitive-Eigenschaft

- DB_EXTENDED_PROPERTY-Funktion,226

- Accept-Encoding

- HTTP_HEADER-Funktion,284

- AcceptCharset-Option

- sa_set_http_option-Systemprozedur,1324

- ACCESS ACCOUNT-Klausel

- ALTER LDAP SERVER-Anweisung,469

- CREATE LDAP SERVER-Anweisung,643

- ACOS-Funktion

- Syntax,167

Adaptive Server Enterprise
 CREATE DATABASE-Anweisung,585
 gespeicherte Prozeduren in Watcom-SQL-Syntax
 konvertieren,430
 Katalogprozeduren,1167
 nach SQL Anywhere mit sa_migrate-
 Systemprozedur migrieren,1266
 Systemprozeduren,1166
 ADD OPTION-Klausel
 ALTER SYNCHRONIZATION
 SUBSCRIPTION-Anweisung [MobiLink] ,513
 ALTER SYNCHRONIZATION USER-
 Anweisung [MobiLink] ,515
 ADD PCTFREE-Klausel
 ALTER MATERIALIZED VIEW-Anweisung,477
 ADD table-constraint-Klausel
 ALTER TABLE-Anweisung ,524
 ADD | ALTER | DELETE SCHEDULE-Klausel
 ALTER EVENT-Anweisung,462
 ADD-Klausel
 ALTER DBSPACE-Anweisung ,458
 ALTER PUBLICATION-Anweisung,485
 ALTER TABLE-Anweisung ,518
 ADDRESS-Klausel
 ALTER REMOTE MESSAGE TYPE-
 Anweisung,487
 ALTER SYNCHRONIZATION
 SUBSCRIPTION-Anweisung [MobiLink] ,513
 ALTER SYNCHRONIZATION USER-
 Anweisung [MobiLink] ,515
 CREATE PUBLICATION-Anweisung [MobiLink]
 [SQL Remote],694
 CREATE SYNCHRONIZATION
 SUBSCRIPTION-Anweisung [MobiLink],734
 CREATE SYNCHRONIZATION USER,736
 GRANT CONSOLIDATE-Anweisung [SQL
 Remote],889
 GRANT REMOTE-Anweisung [SQL Remote],900
 Adressen
 SQL Remote-Publikationseigentümer,487
 AES-Verschlüsselungsalgorithmus
 CREATE DATABASE-Anweisung,583
 CREATE ENCRYPTED FILE-Anweisung,604
 DECRYPT-Funktion,234
 ENCRYPT-Funktion,241
 AES256-Verschlüsselungsalgorithmus
 CREATE DATABASE-Anweisung,583
 CREATE ENCRYPTED FILE-Anweisung,604
 DECRYPT-Funktion,234
 ENCRYPT-Funktion,241
 AES_FIPS-Verschlüsselungsalgorithmus
 CREATE DATABASE-Anweisung,583
 CREATE ENCRYPTED FILE-Anweisung,604
 DECRYPT-Funktion,234
 ENCRYPT-Funktion,241
 AFTER MESSAGE BREAK-Klausel
 WAITFOR-Anweisung,1121
 AFTER-Klausel
 CREATE TRIGGER-Anweisung,762
 AFTER-Trigger
 CREATE TRIGGER-Anweisung ,762
 Aggregatfunktionen
 alphabetische Liste,153
 Aktualisieren
 mithilfe von Joins,1112
 Publikationen und Subskription,1113
 Spalten ohne Protokollierung,1127
 Tabellen und Spalten in SQL Remote,1106
 Textindex mit REFRESH TEXT INDEX,991
 Zeilen,1109
 Aktualisierungen
 SQL Remote-Joins,1107
 Systemansichten,1437
 Systemtabellen,1129
 Aktualisierungstypen
 RefreshType-Eigenschaft,1258
 ALGORITHM-Klausel
 CREATE ENCRYPTED FILE-Anweisung,604
 CREATE ENCRYPTED TABLE DATABASE-
 Anweisung,602
 Aliase
 für Spalten,1023
 UPDATE-Anweisung,1112
 Aliasnamen
 DELETE-Anweisung,792
 ALL
 Schlüsselwort in SELECT-Anweisung,1022
 ALL PRIVILEGES-Privilegklausel
 GRANT-Anweisung,884
 ALL-Klausel
 CREATE EVENT-Anweisung,606

- DESCRIBE-Anweisung,795
- DISCONNECT-Anweisung [ESQL] [Interactive SQL]-Anweisung,802
- MEDIAN-Funktion,312
- SELECT-Anweisung,1022
- ALL-Privilegklausel
 - GRANT-Anweisung,884
- ALL-Suchbedingung
 - Syntax,45
- ALLOCATE DESCRIPTOR-Anweisung
 - Embedded SQL-Syntax,449
- ALLOW-Klausel
 - LOAD TABLE-Anweisung,934
- Alphabetische Liste der Binärdatentypen
 - Info,133
- Alphabetische Liste der Bit-Array-Datentypen
 - Info,114
- Alphabetische Liste der Datums- und Uhrzeit-Datentypen
 - Info,116
- Alphabetische Liste der numerischen Datentypen
 - Info,103
- Alphabetische Liste der Währungsdatentypen
 - Info,113
- Alphabetische Liste der zusammengesetzten Datentypen
 - Info,136
- Alphabetische Zeichen
 - Definition,4
- ALTER DATABASE UPGRADE-Anweisung
 - Syntax,451
- ALTER DATABASE-Anweisung
 - FORCE START-Klausel,455
 - SET PARTNER FAILOVER-Klausel,453
 - Syntax,451
- ALTER DATATYPE-Anweisung
 - Syntax,460
- ALTER DBSPACE-Anweisung
 - Syntax,457
- ALTER DOMAIN-Anweisung
 - Syntax,460
- ALTER EVENT-Anweisung
 - Syntax,461
- ALTER EXTERNAL ENVIRONMENT-Anweisung
 - Syntax,463
- ALTER FUNCTION-Anweisung
 - SQL-Syntax,465
- ALTER INDEX-Anweisung
 - Syntax,466
- ALTER LDAP SERVER-Anweisung
 - Syntax,468
- ALTER LOGIN POLICY-Anweisung
 - Syntax,471
- ALTER MATERIALIZED VIEW-Anweisung
 - Syntax,476
- ALTER MIRROR SERVER-Anweisung
 - Syntax,480
- ALTER OPTION-Klausel
 - ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink] ,513
 - ALTER SYNCHRONIZATION USER-Anweisung [MobiLink] ,515
- ALTER PARENT FROM-Klausel
 - ALTER MIRROR SERVER-Anweisung,481
- ALTER PROCEDURE-Anweisung
 - Syntax,483
- ALTER PUBLICATION-Anweisung
 - SQL Remote-Syntax,485
 - Syntax,485
- ALTER REMOTE MESSAGE TYPE-Anweisung
 - SQL Remote-Syntax,487
- ALTER ROLE-Anweisung
 - Syntax,488
- ALTER SEQUENCE-Anweisung
 - Syntax,490
- ALTER SERVER-Anweisung
 - Syntax,491
- ALTER SERVICE-Anweisung [HTTP-Webdienst]
 - Syntax,494
- ALTER SERVICE-Anweisung [SOAP-Webdienst]
 - Syntax,500
- ALTER SPATIAL REFERENCE SYSTEM-Anweisung
 - Syntax,506
- ALTER STATISTICS-Anweisung
 - Syntax,510
- ALTER SYNCHRONIZATION PROFILE-Anweisung
 - MobiLink-Syntax,511
- ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung
 - MobiLink-Syntax,512
- ALTER SYNCHRONIZATION USER-Anweisung
 - MobiLink-Syntax,515
- ALTER TABLE OWNER-Klausel
 - ALTER TABLE-Anweisung,528

-
- ALTER TABLE-Anweisung
 - Syntax,516
 - ALTER TEXT CONFIGURATION-Anweisung
 - Syntax,532
 - ALTER TEXT INDEX-Anweisung
 - Syntax,536
 - ALTER TRACE EVENT SESSION-Anweisung
 - Syntax,538
 - ALTER TRANSACTION LOG-Klausel
 - ALTER DATABASE-Anweisung,452
 - ALTER TRIGGER-Anweisung
 - Syntax,539
 - ALTER USER-Anweisung
 - Kennwort, maximale Länge,542
 - Kennwörter, Zeichensatzkonvertierung,542
 - Syntax,540
 - Zeichensatzkonvertierung für Kennwörter,542
 - ALTER VIEW-Anweisung
 - DISABLE-Klausel,544
 - ENABLE-Klausel,544
 - RECOMPILE-Klausel,544
 - Syntax,543
 - ALTER-Klausel
 - ALTER TABLE-Anweisung,525
 - ALTER-Privileg
 - GRANT-Anweisung,883
 - REVOKE-Anweisung,1010
 - ALTER-Privilegklausel
 - GRANT-Anweisung,884
 - Alternative Servernamen
 - mit CREATE MIRROR SERVER-Anweisung definieren,657
 - Am Raster ausrichten
 - SNAP TO GRID-Klausel, ALTER SPATIAL REFERENCE SYSTEM-Anweisung,508
 - SNAP TO GRID-Klausel, CREATE SPATIAL REFERENCE SYSTEM-Anweisung,724
 - AND
 - Bit-Operatoren,20
 - Drei-Werte-Logik,67
 - logische Operatoren, Beschreibung,10
 - Ändern
 - ALTER PUBLICATION-Anweisung,485
 - ALTER TABLE-Anweisung,516
 - Ansichten mit der ALTER VIEW-Anweisung,543
 - Datenbanken mit der ALTER DATABASE-Anweisung,451
 - Datentypen mit der ALTER DOMAIN-Anweisung,460
 - DBSpaces mit der ALTER DBSPACE-Anweisung,457
 - Domänen mit der ALTER DOMAIN-Anweisung,460
 - Ereignisse mit der ALTER EVENT-Anweisung,461
 - Fremdserverattribute mit der ALTER SERVER-Anweisung,491
 - Indizes mit der ALTER INDEX-Anweisung,466
 - LDAP-Serverkonfigurationsobjekte,468
 - materialisierte Ansichten mit der ALTER MATERIALIZED VIEW-Anweisung,476
 - Optionen der Login-Richtlinie mit der ALTER LOGIN POLICY-Anweisung,471
 - Optionen der Login-Richtlinie mit der ALTER USER-Anweisung,540
 - Prozeduren mit der ALTER PROCEDURE-Anweisung,483
 - Spalten mit der ALTER TABLE-Anweisung,516
 - SQL Remote, ALTER PUBLICATION-Anweisung,485
 - SQL Remote, entfernte Nachrichtentypen,487
 - Textindizes mit der ALTER TEXT INDEX-Anweisung,536
 - Textkonfigurationsobjekte,532
 - Trigger mit der ALTER TRIGGER-Anweisung,539
 - Änderung von Synchronisationsprofilen
 - ALTER SYNCHRONIZATION PROFILE-Anweisung [MobiLink],511
 - Anforderungen
 - Zeitfolgeinformationen abrufen,1279
 - Anforderungslog
 - mit sa_get_request_times-Profilprozedur verarbeiten,1306
 - Anforderungsprotokollierung
 - Anforderungslog mit sa_get_request_profile analysieren,1228
 - Anforderungslog mit sa_get_request_times analysieren,1229
 - von Interactive SQL aus aktivieren,1306
 - Anforderungszeit
 - sa_performance_diagnostics-Systemprozedur,1279
 - Anführungszeichen
 - Datenbankobjekte,4
 - einfache und doppelte,41

- in SQL-Bezeichnern nicht gestattet,4
 - Kompatibilität mit ASE,41
 - SQL-Bezeichner,4
- Angenäherte Datentypen
 - Info,103
- ANGULAR UNIT OF MEASURE-Klausel
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung,508
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung,722
- Anhänge
 - E-Mail-Beispiel,1429
- Annähernde Übereinstimmung
 - CONTAINS-Suchbedingung,59
- ANSI
 - Äquivalenz durch die REWRITE-Funktion,368
- ansi_nulls, Option
 - Microsoft SQL-Server, Kompatibilität,1057
- ansi_permissions-Option
 - mit Transact-SQL SET-Anweisung definieren,1056
- Ansichten
 - Abhängigkeiten ermitteln,1202
 - Aktualisierung mit INSERT-Anweisung,921
 - Änderung von materialisierten Ansichten eines anderen Eigentümers,544
 - CREATE MATERIALIZED VIEW-Anweisung,652
 - CREATE VIEW-Anweisung,773
 - DROP VIEW-Anweisung,840
 - Indizes,641
 - Info,1437
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - Kompatibilitätsansichten,1534
 - konsolidierte Ansichten,1514
 - mit der ALTER VIEW-Anweisung ändern,543
 - parametrisierte Ansichten,774
 - sa_recompile_views-Systemprozedur,1293
 - Systemansichten,1438
 - Transact-SQL-Kompatibilität,1546
- Ansichten, Abhängigkeiten
 - Datenbanken entladen/neuladen,1293
- ansinull-Option
 - mit Transact-SQL SET-Anweisung setzen,1056
- Anweisungen
 - alphabetische Liste der SQL Anywhere Server-Anweisungen,449
 - in der BEGIN-Anweisung gruppieren,557
 - in der BEGIN-Anweisung gruppieren [TSQL],560
 - in der TRY-Anweisung gruppieren,1093
 - vorbereiten,976
 - vorbereitete Anweisungen löschen,826
 - vorbereitete ausführen,846
- Anweisungslabel
 - allgemeines Element der SQL-Syntax,446
 - GOTO, Transact-SQL-Anweisung,881
- Anweisungsliste
 - allgemeines Element der SQL-Syntax,446
- Anweisungssyntax
 - alphabetische Liste der SQL Anywhere Server-Anweisungen,449
 - Konventionen in der Dokumentation,445
- Anwendungsprofilerstellung
 - Protokollierungsstufe einstellen,1330
- ANY-Suchbedingung
 - Syntax,46
- Anzahl von Zeilen
 - Systemansichten,1492
- Anzeigen
 - Ausnahmebedingungen,1000
 - Fehler,982
 - Interactive SQL Prozedurprofilinformationen,1290
 - Meldungen,959
 - Meldungen im Meldungsfenster,980
- Apostrophe
 - in SQL-Zeichenfolgen,8
- APPEND-Klausel
 - OUTPUT-Anweisung,968
 - UNLOAD-Anweisung,1099
- Arbiterserver
 - mit der CREATE MIRROR SERVER-Anweisung definieren,657
- Archive
 - Datenbanken wiederherstellen,1001
 - Datenbanksicherungen mit der BACKUP-Anweisung erstellen,548
- Archivsicherungen
 - unterstützte Betriebssysteme mit der BACKUP-Anweisung,548
- ARGN-Funktion
 - Syntax,168
- Arithmetische Operatoren
 - Modulo,11
 - SQL-Syntax,11
- Arkuskosinus-Funktion

ACOS-Funktion,167
 Arkussinus-Funktion
 ASIN-Funktion,173
 Arkustangens-Funktion
 ATAN-Funktion,173
 ATAN2-Funktion,174
 ARRAY-Datentyp
 deklarieren,136
 ARRAY-Klausel
 EXECUTE-Anweisung,846
 FETCH-Anweisung,855
 PUT-Anweisung [ESQL],981
 ARRAY-Konstruktor
 Syntax,168
 Array-Operatoren
 Syntax,19
 UNNEST-Operator,19
 ARRAY_AGG-Funktion
 Syntax,170
 ARRAY_MAX_CARDINALITY-Funktion
 Syntax,171
 Arrays (*Siehe* zusammengesetzte Datentypen)
 Artikel
 SYSARTICLE-Systemansicht,1438
 SYSARTICLECOL-Systemansicht,1438
 AS-Klausel
 ALTER MIRROR SERVER-Anweisung,480
 ALTER SERVICE-Anweisung [SOAP über HTTP],500
 ALTER SERVICE-Anweisungen [HTTP-Webdienst],494
 ALTER VIEW-Anweisung,543
 CONNECT-Anweisung [ESQL] [Interactive SQL],578
 CREATE DBSPACE-Anweisung,593
 CREATE DOMAIN-Anweisung,598
 CREATE FUNCTION-Anweisung,633
 CREATE MATERIALIZED VIEW-Anweisung,652
 CREATE MESSAGE-Anweisung,655
 CREATE MIRROR SERVER-Anweisung,657
 CREATE PROCEDURE-Anweisung [T-SQL],679
 CREATE TRIGGER-Anweisung [Transact-SQL],769
 CREATE VIEW-Anweisung,774
 DELETE-Anweisung,791
 START DATABASE-Anweisung,1063
 UPDATE-Anweisung,1109
 ASC | DESC-Klausel
 CREATE INDEX-Anweisung,639
 ASCII
 Funktion und Syntax,172
 ASE COMPATIBLE-Klausel
 CREATE DATABASE-Anweisung,585
 ASIN-Funktion
 Syntax,173
 Assertierungen
 Beispiele für reguläre Ausdrücke,36
 reguläre Ausdrücke,36
 AT-Klausel
 ALTER EVENT-Anweisung,462
 CREATE EVENT-Anweisung,610
 CREATE EXISTING TABLE-Anweisung,614
 CREATE FUNCTION-Anweisung,635
 CREATE PROCEDURE-Anweisung,685
 CREATE TABLE-Anweisung,739
 ATAN-Funktion
 Syntax,173
 ATAN2-Funktion
 Syntax,174
 ATN2-Funktion
 Syntax,174
 ATOMIC-Klausel
 BEGIN-Anweisung,558
 ATTACH TRACING-Anweisung
 Diagnoseprotokollierung,546
 Syntax,546
 ATTENDED-Klausel
 BACKUP-Anweisung,551
 Attribute
 Fremdserver mit der ALTER SERVER-Anweisung ändern,491
 Auditing
 mit der sa_disable_auditing_type-Systemprozedur deaktivieren,1211
 mit der sa_enable_auditing_type-Systemprozedur aktivieren,1214
 AUDITING-Option
 Kommentare hinzufügen,1170
 Aufrufe nativer Funktionen
 eine Schnittstelle zu nativen Funktionen mit der CREATE FUNCTION-Anweisung,617
 Funktionen,620
 Schnittstelle zu nativen Prozeduren mit CREATE PROCEDURE-Anweisung,661
 Aufrufen

- Prozeduren mit der CALL-Anweisung, 564
- Aufrufer
 - ALTER DATABASE-Anweisung, SYSTEM PROCEDURE AS-Klausel, 451
 - ALTER PROCEDURE-Anweisung, SQL SECURITY-Klausel, 681
 - CREATE DATABASE-Anweisung, 681
- Ausdrücke
 - allgemeines Element der SQL-Syntax, 445
 - arithmetische Operatoren, 11
 - Array-Operatoren, 19
 - CASE-Ausdrücke, 25
 - Datentypen, 262
 - IF-Ausdrücke, 24
 - Info, 22
 - Konstanten, 24
 - Spaltennamen, 24
 - SQL-Operator-Vorrang, 21
 - Syntax, 22
 - Transact-SQL-Kompatibilität, 41
 - Unterabfragen, 24
 - Zeichenfolgenoperatoren, 11
- Ausdruckstypen
 - Suchbedingungen, 42
- Äußere Referenzen
 - FROM-Klausel, 866
 - lateral abgeleitete Tabellen, 866
- Ausführen
 - Betriebssystembefehle, 1088
 - gespeicherte Prozeduren in Transact-SQL, 848
 - SQL-Anweisungen von Dateien, 984
 - von Prozeduren wieder aufnehmen, 1003
 - vorbereitete Anweisungen, 846
- Ausführungszeit
 - START LOGGING-Anweisung, 1068
- Ausnahmebedingungen
 - anzeigen, 1061
 - erneut anzeigen, 1000
- Ausnahmen
 - mit DECLARE-Anweisung deklarieren, 786
- Auswählen
 - Daten mit UNLOAD-Anweisung entladen, 1098
 - Mengendifferenzen bilden, 841
 - Schnittmengen bilden, 927
 - Unions bilden, 1096
 - Zeilen, 1020
- Auswählen aus DML
 - FROM-Klausel, 868
- AUTHENTICATION URL-Klausel
 - ALTER LDAP SERVER-Anweisung, 469
 - CREATE LDAP SERVER-Anweisung, 643
- authentication_string-Option
 - SET MIRROR OPTION-Anweisung, 1035
- AUTHORIZATION-Klausel
 - ALTER SERVICE-Anweisung [SOAP über HTTP], 500
 - ALTER SERVICE-Anweisungen [HTTP-Webdienst], 494
 - CREATE SERVICE-Anweisung [HTTP-Webdienst], 497, 709
 - CREATE SERVICE-Anweisung [SOAP-Webdienst], 503, 716
- AUTO COMPRESSED-Klausel
 - LOAD TABLE-Anweisung, 936
- AUTO TUNE WRITERS-Klausel
 - BACKUP-Anweisung, 551
- AUTO UPDATE-Klausel
 - ALTER STATISTICS-Anweisung, 510
- AUTO-Klausel
 - BACKUP-Anweisung, 552
- auto_add_fan_out-Option
 - SET MIRROR OPTION-Anweisung, 1035
- auto_add_server-Option
 - SET MIRROR OPTION-Anweisung, 1035
- auto_commit, Option
 - Interactive SQL-Option, 1043
- auto_failover-Option
 - SET MIRROR OPTION-Anweisung, 1035
- auto_unlock_time-Option
 - ALTER LOGIN POLICY-Anweisung, 472
 - CREATE LOGIN POLICY-Anweisung, 647
- AUTOINCREMENT
 - @ @identity, 91
 - den Wert zurücksetzen, 1299
 - GET_IDENTITY-Funktion, 267
- AUTOINCREMENT-Klausel
 - CREATE TABLE-Anweisung, 737
- AUTOINCREMENT-Standardwert
 - CREATE TABLE-Anweisung, 520, 743
- AutoMultiProgrammingLevel-Eigenschaft
 - mit sa_server_option einstellen, 1306
- AutoMultiProgrammingLevelStatistics-Eigenschaft
 - mit sa_server_option einstellen, 1306
- AUTOSTOP-Klausel
 - START DATABASE-Anweisung, 1063
- AvailForOptimization-Eigenschaft

sa_materialized_view_info-Systemprozedur,1258
AVG-Funktion
Syntax,175

B

Back Quotes
Datenbankobjekte,4
SQL-Bezeichner,4
Backslashes
in SQL-Bezeichnern nicht gestattet,4
in SQL-Zeichenfolgen,8
BACKUP-Anweisung
Syntax,548
backup.syb
Info,551
Bandlaufwerke
Datenbanksicherungen mit der BACKUP-
Anweisung,548
BASE64_DECODE-Funktion
Syntax,177
BASE64_ENCODE-Funktion
Syntax,177
Basis 10, Logarithmus
LOG10-Funktion,307
Basistabellen
CREATE TABLE-Anweisung,749
Bedingungen
allgemeines Element der SQL-Syntax,445
CONTAINS,59
Drei-Werte-Logik,67
EXISTS,66
SQL-Suchbedingungen,42
Suche,42
Beenden
Interactive SQL,850
Prozeduren,1004
Transaktionen zurücksetzen,1015
Befehle
Betriebssystem ausführen,1088
BEFORE-Klausel
CONTAINS-Suchbedingung,59
CREATE TRIGGER-Anweisung,762
BEFORE-Schlüsselwort, CONTAINS-Suchbedingung
nicht unterstützt in der Volltextsuche,63
BEFORE-Trigger
CREATE TRIGGER-Anweisung,762
BEGIN CATCH-Anweisung

Syntax,1093
BEGIN DECLARE-Anweisung
Embedded SQL-Syntax,778
BEGIN SNAPSHOT-Anweisung
Syntax,556
BEGIN TRANSACTION-Anweisung
Transact-SQL-Syntax,561
BEGIN TRY-Anweisung
Syntax,1093
BEGIN-Anweisung
Syntax,557
Transact-SQL-Syntax,560
BEGIN-Schlüsselwort
TSQL-Kompatibilität,560
Beginnen
benutzerdefinierte Transaktionen mit der BEGIN
TRANSACTION-Anweisung,561
Begrenzen
von SQL-Zeichenfolgen,4
Begrenzte Zeichenfolgen
Kompatibilität mit ASE,41
Begriffe
MAXIMUM TERM LENGTH-Klausel,533
MINIMUM TERM LENGTH-Klausel,533
TERM BREAKER-Klausel,533
Begriffsegmentierer
externe Begriffsegmentierer-Bibliothek
angeben,533
mit sa_char_terms die Aufteilung von
Zeichenfolgen in Begriffe testen,1172
mit sa_nchar_terms die Aufteilung von
Zeichenfolgen in Begriffe testen,1278
Warnung bei Verwendung von nicht-
alphanumerischen Zeichen in der
Abfragezeichenfolge,61
Behandeln
Fehler in Embedded SQL,1122
Benannte Parameter
beim Aufrufen von Prozeduren zulässig,564
Info,93
Benutzer
ALTER SYNCHRONIZATION USER-
Anweisung,515
ändern mit der ALTER USER-Anweisung,540
CREATE SYNCHRONIZATION USER-
Anweisung,735
DROP SYNCHRONIZATION USER-
Anweisung,831

- einstellen, 1059
- erstellen mit der CREATE USER-Anweisung, 770
- Kommentare mit der COMMENT-Anweisung hinzufügen, 573
- löschen, 1009
- mit der DROP USER-Anweisung löschen, 838
- Status ermitteln, 1233
- Benutzer-IDs
 - allgemeines Element der SQL-Syntax, 446
 - Ansichten, 1542
 - Einschränkungen, 888
 - entziehen, 1009
 - Systemansichten, 1492
- Benutzerdefinierte Datentypen
 - CREATE DOMAIN-Anweisung, 598
 - Info, 138
 - mit der DROP DATATYPE-Anweisung löschen, 805
 - Transact-SQL, 140
- Benutzerdefinierte Eigenschaften
 - Werte einstellen, 1350
 - Werte inkrementieren, 1349
- Benutzerdefinierte Funktionen
 - beenden, 1004
 - CREATE FUNCTION-Anweisung, 633
 - Definition, 158
 - Info, 158
 - Java, 158
 - Werte zurückgeben, 1004
- Benutzerdefinierte Optionen
 - temporäre Einstellungen nicht unterstützt, 1040
- Benutzerdefinierte Rollen
 - mit der GRANT-Anweisung erteilen, 882
 - mit der REVOKE-Anweisung entziehen, 1009
- Benutzerdefinierte Selektivitätsschätzungen
 - Info, 68
- Benutzererweiterte Rollen
 - benutzererweiterte Rollen in Benutzer zurückkonvertieren, 820
 - erstellen, CREATE ROLE-Anweisung, 696
- Benutzerschätzungen
 - Info, 68
- Berechtigungen
 - Kompatibilitätsrollen zuordnen, 1355
 - SYSCOLAUTH-Ansicht, 1520
- Bereich
 - Datentyp, 123
- Berücksichtigung von Groß- und Kleinschreibung
 - LIKE-Suchbedingung, 53
 - REGEXP-Suchbedingung, 55
 - REGEXP_SUBSTR-Funktion, 349
 - SIMILAR TO-Suchbedingung, 57
- Beschreiben
 - Cursor, 794
 - Cursor-Verhalten in benutzerdefinierten Prozeduren, 684
 - Cursor-Verhalten in externen Prozeduren, 664
- Betriebsbereite Server
 - ALTER MIRROR SERVER-Anweisung, 480
 - CREATE MIRROR SERVER-Anweisung, 656
- Betriebssysteme
 - Befehle ausführen, 1088
- BETWEEN ... AND-Klausel
 - CREATE EVENT-Anweisung, 610
- BETWEEN-Klausel
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung, 719
 - WINDOW-Klausel, 1126
- BETWEEN-Suchbedingung
 - Syntax, 48
- Bezeichner
 - in Domänen verwenden, 600
 - Info, 4
 - maximale Länge in SQL Anywhere, 4
 - Syntax, 4
- Beziehungen
 - Systemansichten, 1452
- BIGINT UNSIGNED-Datentyp
 - Syntax, 104
- BIGINT-Datentyp
 - mit Datums- und Uhrzeitangaben vergleichen, 143
 - Syntax, 104
- Bilddaten, SQL-Funktionen
 - Info, 166
- Bilder
 - aus der Datenbank lesen, 986
- Binärdateien
 - importieren, 1424
- Binärdatentypen
 - aus Spalten abrufen, 876
 - BINARY, 133
 - IMAGE, 134
 - LONG BINARY, 134
 - UNIQUEIDENTIFIER, 134
 - VARBINARY, 135
- Binäre Datentypen

- dekodieren,177
- kodieren,177
- Binäre große Objekte
 - Binärdatentypen,133
 - exportieren,1435
 - Hinweise zum Transaktionslog,457
 - mit der xp_read_file-Systemprozedur einfügen,1424
- Binäre Literale
 - Sonderzeichen,6
- Binärwerte
 - Escapezeichen,6
- Binary Large Objects
 - ASE-erzeugte BCP-Dateien importieren,938
 - GET DATA-Anweisung,876
 - SET-Anweisungsbeispiel,1056
 - von Spalten abrufen,876
- BINARY-Datentyp
 - Syntax,133
- BIND VARIABLES FOR-Klausel
 - DESCRIBE-Anweisung [ESQL],794
- Bindestriche
 - zulässige Syntax in einer CONTAINS-Klausel,63
 - zulässige Syntax in einer Volltext-Abfragezeichenfolge,63
- Bindevariablen
 - Cursor beschreibende,794
 - OPEN-Anweisung,965
- Bindungsvariable
 - EXECUTE-Anweisung,846
- BINTOHEX-Funktion
 - Syntax,178
- Bit-Array-Datentypen
 - Info,114
 - LONG VARBIT,114
 - VARBIT,115
- Bit-Arrays
 - Datentypen,114
 - Info,114
 - konvertieren,148
- BIT-Datentyp
 - Syntax,105
- Bit-Operatoren
 - Syntax,20
- BIT_AND-Funktion
 - Syntax,179
- BIT_LENGTH-Funktion
 - Syntax,180
- BIT_OR-Funktion
 - Syntax,180
- BIT_SUBSTR-Funktion
 - Syntax,181
- BIT_XOR-Funktion
 - Syntax,182
- Bits
 - konvertieren,148
- BLANK PADDING-Klausel
 - CREATE DATABASE-Anweisung,586
- BLOBs
 - ASE-erzeugte BCP-Dateien importieren,938
 - BLOB-Indizierung bei Tabellenerstellung konfigurieren,741
 - BLOB-Indizierung mit der ALTER TABLE-Anweisung konfigurieren,523
 - exportieren,1435
 - GET DATA-Anweisung,876
 - Hinweise zum Transaktionslog,457
 - INLINE-Klausel, CREATE TABLE-Anweisung,740
 - mit der SET-Anweisung einfügen,1054
 - mit der xp_read_file-Systemprozedur einfügen,1424
 - PREFIX-Klausel, CREATE TABLE-Anweisung,740
 - SET-Anweisungsbeispiel,1056
- Blobs
 - Abfrage in Blobs,867
- BLOCK-Klausel
 - FETCH-Anweisung,854
 - OPEN-Anweisung,965
- Blockabruf
 - FETCH-Anweisung,854
 - OPEN-Anweisung,965
- Blockierung
 - identifizieren,1183
- Blocks
 - Fehlerbehandlung,1250
 - identifizieren,1183
- BREAK-Anweisung
 - Transact-SQL-Syntax,563
- BY LOCK-Klausel
 - PREPARE-Anweisung [ESQL],977
- BY TIMESTAMP-Klausel
 - PREPARE-Anweisung [ESQL],977
- BY VALUES-Klausel
 - PREPARE-Anweisung [ESQL],977

- BYE-Anweisung
 - Interactive SQL-Syntax, 850
- BYTE ORDER MARK
 - Daten aus einer UTF-16- oder UTF-8-Datendatei laden, 944
 - Lese- oder Schreiboption in der CSCONVERT-Funktion, 211
- Byte Order Mark (BOM)
 - Daten aus einer UTF-8- oder UTF-16-Datendatei laden, 944
- BYTE ORDER MARK-Klausel
 - INPUT-Anweisung, 911
 - LOAD TABLE-Anweisung, 935
 - OPENXML-Operator, 15
 - OUTPUT-Anweisung, 969
 - UNLOAD-Anweisung, 1099
- BYTE_LENGTH-Funktion
 - Syntax, 183
- BYTE_SUBSTR-Funktion
 - Syntax, 184
- C**
- Cache
 - leeren, 1220
- CACHE-Klausel
 - ALTER SEQUENCE-Anweisung, 491
 - CREATE SEQUENCE-Anweisung, 700
- CacheSizingStatistics-Eigenschaft
 - mit sa_server_option einstellen, 1306
- CALIBRATE DBSPACE TEMPORARY-Klausel
 - ALTER DATABASE-Anweisung, 452
- CALIBRATE DBSPACE-Klausel
 - ALTER DATABASE-Anweisung, 452
- CALIBRATE GROUP READ-Klausel
 - ALTER DATABASE-Anweisung, 452
- CALIBRATE PARALLEL READ-Klausel
 - ALTER DATABASE-Anweisung, 452
- CALIBRATE SERVER-Klausel
 - ALTER DATABASE-Anweisung, 452
- CALL-Anweisung
 - in Transact-SQL, 848
 - Syntax, 564
- CAPABILITY-Klausel
 - ALTER SERVER-Anweisung, 492
- CARDINALITY-Funktion
 - Syntax, 185
- CASCADE-Klausel
 - CREATE TABLE-Anweisung, 737
- CASE-Anweisung
 - Syntax, 566
- CASE-Ausdruck
 - NULLIF-Funktion, 331
 - Syntax, 25
- CASE-Klausel
 - CREATE DATABASE-Anweisung, 586
- CaseSensitivity-Eigenschaft
 - DB_EXTENDED_PROPERTY-Funktion, 226
- CAST-Funktion
 - Datentypkonvertierung, 146
 - Syntax, 186
- CATALOG ONLY-Klausel
 - RESTORE DATABASE-Anweisung, 1001
- CatalogCollation-Eigenschaft
 - DB_EXTENDED_PROPERTY-Funktion, 226
- CATCH-Anweisung
 - Syntax, 1093
- CEILING-Funktion
 - Syntax, 187
- CERTIFICATE-Klausel
 - CREATE FUNCTION-Anweisung [Webclients], 628
 - CREATE PROCEDURE-Anweisung [Webclients], 674
- CHANGE PASSWORD-Systemprivileg
 - mit der GRANT-Anweisung erteilen, 883
- change_password_dual_control-Option
 - ALTER LOGIN POLICY-Anweisung, 472
 - CREATE LOGIN POLICY-Anweisung, 647
- CHAR-Datentyp
 - Bytelänge-Semantik, 95
 - DESCRIBE bei einer CHAR-Spalte, 95
 - mit Datums- und Uhrzeitangaben vergleichen, 143
 - Syntax, 95
 - vergleichen mit NCHAR-Datentyp, 141
 - Zeichenlängensemantik, 95
- CHAR-Funktion
 - Syntax, 188
- CHAR_LENGTH-Funktion
 - Syntax, 189
- CHARINDEX-Funktion
 - Syntax, 189
- CharSet-Eigenschaft
 - DB_EXTENDED_PROPERTY-Funktion, 226
- CharsetConversion-Option
 - sa_set_http_option-Systemprozedur, 1324

CHECK CONSTRAINTS-Klausel
 LOAD TABLE-Anweisung,935

CHECK EVERY-Klausel
 WAITFOR-Anweisung,1121

CHECK ON COMMIT-Klausel
 CREATE TABLE-Anweisung,748

CHECK-Bedingungen
 CREATE TABLE-Anweisung,745

CHECK-Klausel
 ALTER TABLE-Anweisung,522
 CREATE DOMAIN-Anweisung,598
 CREATE MATERIALIZED VIEW-Anweisung,653
 CREATE TABLE-Anweisung,748
 Suchbedingungen,42
 VALIDATE LDAP SERVER-Anweisung,1117

CHECKPOINT-Anweisung
 Syntax,570

Checkpoint-Logs
 CHECKPOINT-Anweisung,570

Checkpoints
 mit der CHECKPOINT-Anweisung setzen,570

Checkpoints erstellen
 für Datenbanken, mit der CHECKPOINT-Anweisung,570

CHECKSUM-Klausel
 ALTER DATABASE-Anweisung,453
 CREATE DATABASE-Anweisung,586

Chiffrierschlüssel
 Schlüssel für eine verschlüsselte Datenbank ändern,606

child_creation-Option
 SET MIRROR OPTION-Anweisung,1035

CLASS-Klausel
 ALTER SERVER-Anweisung,492
 CREATE SERVER-Anweisung,702
 REMOVE JAVA-Anweisung,997

CLEAR-Anweisung
 Interactive SQL-Syntax,571

Clientdateien
 READ_CLIENT_FILE-Funktion,348
 WRITE_CLIENT_FILE-Funktion,432

CLIENTPORT-Klausel
 CREATE FUNCTION-Anweisung [Webclients],628
 CREATE PROCEDURE-Anweisung [Webclients],674

Clientseitiges Caching von Anweisungen
 RequestLogFile-Eigenschaft,1306

Clockwise-Format
 CREATE SPATIAL REFERENCE SYSTEM-Anweisung,725

CLOSE-Anweisung
 Embedded SQL-Syntax,571
 Syntax,571

close_on_endtrans-Option
 mit Transact-SQL SET-Anweisung setzen,1056

Clustered-Indizes
 mit der ALTER INDEX-Anweisung erstellen,466

CLUSTERED-Klausel
 ALTER INDEX-Anweisung,466
 CREATE INDEX-Anweisung,639

COALESCE-Funktion
 Syntax,191

Codepages
 INPUT-Anweisung,912
 OUTPUT-Anweisung,969

COL_LENGTH-Funktion
 Syntax,165

COL_NAME-Funktion
 Syntax,165

Collation-Eigenschaft
 DB_EXTENDED_PROPERTY-Funktion,226

COLLATION-Klausel
 CREATE DATABASE-Anweisung,587
 Kollationsanpassung,587

CollectStatistics-Eigenschaft
 mit sa_server_option einstellen,1306

COLUMN DELIMITED BY-Klausel
 UNLOAD-Anweisung,1099

COLUMN WIDTHS-Klausel
 INPUT-Anweisung,911
 OUTPUT-Anweisung,969

COLUMN-Klausel
 DESCRIBE-Anweisung [ESQL],794
 GET DATA-Anweisung,877

COMMENT-Anweisung
 Syntax,573

COMMENTS INTRODUCED BY-Klausel
 LOAD TABLE-Anweisung,936

Commit
 für Zwei-Phasen-Commit vorbereiten,979

COMMIT-Anweisung
 für zwei Phasen-Commit vorbereiten,979
 referenzielle Integrität,1173
 Syntax,575

- COMPARE-Funktion
 - Kollationsanpassung,192
 - Syntax,192
- COMPRESS-Funktion
 - Syntax,193
- COMPRESSED-Klausel
 - ALTER TABLE-Anweisung,522
 - CREATE TABLE-Anweisung,740
 - LOAD TABLE-Anweisung,936
 - UNLOAD-Anweisung,1099
- COMPUTE-Klausel
 - ALTER TABLE-Anweisung,523
 - CREATE TABLE-Anweisung,748
- COMPUTES-Klausel
 - LOAD TABLE-Anweisung,936
- CONFIGURATION-Klausel
 - CREATE TEXT INDEX-Anweisung,759
- CONFIGURE-Anweisung
 - Interactive SQL-Syntax,577
- CONFLICT-Funktion
 - Syntax,194
- CONNECT
 - mit der REVOKE-Anweisung entziehen,1010
- CONNECT TO-Klausel
 - GRANT CONNECT-Anweisung,888
- CONNECT TRACING-Anweisung
 - sa_diagnostic_tracing_level-Tabelle füllen,1156
- CONNECT-Anweisung
 - Embedded SQL-Syntax,578
 - Interactive SQL-Syntax,578
- CONNECT-Privileg
 - GRANT CONNECT-Anweisung,888
- Connection
 - HTTP_RESPONSE_HEADER-Funktion,286
- CONNECTION CLOSE-Klausel
 - ALTER SERVER-Anweisung,493
- CONNECTION RETRIES-Klausel
 - ALTER SERVER-Anweisung,470
 - CREATE LDAP SERVER-Anweisung,644
- CONNECTION TIMEOUT-Klausel
 - ALTER SERVER-Anweisung,470
 - CREATE LDAP SERVER-Anweisung,644
- CONNECTION_EXTENDED_PROPERTY-Funktion
 - Syntax,197
- CONNECTION_PROPERTY-Funktion
 - Syntax,199
- connection_string-Option
 - ALTER MIRROR SERVER-Anweisung,481
- CREATE MIRROR SERVER-Anweisung,659
- ConnsDisabled-Eigenschaft
 - mit sa_server_option einstellen,1306
- ConnsDisabledForDB-Eigenschaft
 - mit sa_server_option einstellen,1306
- ConsoleLogFile-Eigenschaft
 - mit sa_server_option einstellen,1306
- ConsoleLogMaxSize-Eigenschaft
 - mit sa_server_option einstellen,1306
- CONSOLIDATE-Privileg
 - erteilen,889
 - REVOKE CONSOLIDATE-Anweisung,1006
- CONSOLIDATED-Klausel
 - CREATE EVENT-Anweisung,606
- CONSTRAINT-Klausel
 - CREATE TABLE-Anweisung,737
- CONTAINS-Klausel
 - FROM-Klausel,868
- CONTAINS-Suchbedingung
 - annähernde Übereinstimmung,59
 - Suchbedingungen,42
 - Syntax,59
 - Verwendung des BEFORE-Schlüsselworts nicht unterstützt,63
 - zulässige Syntax für Sonderzeichen,64
- Content-Length
 - HTTP_RESPONSE_HEADER-Funktion,286
- Content-Type
 - HTTP_RESPONSE_HEADER-Funktion,286
- CONTINUE-Anweisung
 - Transact-SQL-Syntax,581
- CONTINUE-Klausel
 - WHENEVER-Anweisung [ESQL],1122
- CONVERT USING-Klausel
 - CREATE SPATIAL UNIT OF MEASURE-Anweisung,728
- CONVERT-Funktion
 - Datentypkonvertierung,146
 - Syntax,200
- Cookie
 - HTTP_HEADER-Funktion,284
- COORDINATE-Klausel
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung,508
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung,723
- Coordinated Universal Time
 - CURRENT UTC TIMESTAMP,75

UTC TIMESTAMP,84	CREATE ENCRYPTED TABLE DATABASE-
COPY-Klausel	Anweisung
BACKUP-Anweisung,552	Syntax,601
Copyright	CREATE EVENT-Anweisung
abrufen,1422	Syntax,606
CORR-Funktion	CREATE EXISTING TABLE-Anweisung
Syntax,203	sp_remote_columns-Systemprozedur,1386
COS-Funktion	sp_remote_tables-Systemprozedur,1396
Syntax,204	Syntax,613
COT-Funktion	CREATE EXTERNLOGIN-Anweisung
Syntax,204	Syntax,616
COUNT-Funktion	CREATE FUNCTION-Anweisung
Syntax,205	Syntax,633
COUNT-Klausel	Syntax für die Erstellung von
GET DESCRIPTOR-Anweisung [ESQL],878	Webdienstfunktionen,624
SET DESCRIPTOR-Anweisung [ESQL],1033	Syntax für eine Schnittstelle zu nativen
COUNT_BIG-Funktion	Funktionen,617
Syntax,206	Transact-SQL-Beispiel,638
COUNT_SET_BITS-Funktion	CREATE INDEX-Anweisung
Syntax,208	Syntax,638
CounterClockwise-Format	Tabellenverwendung,641
CREATE SPATIAL REFERENCE SYSTEM-	CREATE LDAP SERVER-Anweisung
Anweisung,725	Syntax,642
COVAR_POP-Funktion	CREATE LOCAL TEMPORARY TABLE-
Syntax,209	Anweisung
COVAR_SAMP-Funktion	Syntax,645
Syntax,210	CREATE LOGIN POLICY-Anweisung
CREATE CERTIFICATE-Anweisung	Syntax,647
Syntax,582	CREATE MATERIALIZED VIEW-Anweisung
CREATE DATABASE-Anweisung	Syntax,652
Syntax,583	CREATE MESSAGE-Anweisung
CREATE DATATYPE-Anweisung	Transact-SQL-Syntax,655
Syntax,598	CREATE MIRROR SERVER-Anweisung
CREATE DBSPACE-Anweisung	Syntax,656
Syntax,593	CREATE PROCEDURE-Anweisung
CREATE DECRYPTED DATABASE-Anweisung	sp_remote_pcols-Systemprozedur,1391
Syntax,594	sp_remote_procedures-Systemprozedur,1395
CREATE DECRYPTED FILE-Anweisung	Syntax,681
Syntax,596	Syntax für die Erstellung von
CREATE DOMAIN-Anweisung	Webdienstprozeduren,670
Syntax,598	Syntax für eine Schnittstelle zu nativen
verwenden,138	Prozeduren,661
CREATE ENCRYPTED DATABASE-Anweisung	Transact-SQL-Syntax,679
Syntax,601	CREATE PUBLICATION-Anweisung
CREATE ENCRYPTED FILE-Anweisung	MobiLink-Syntax,690
Beispiel zum Ändern eines Chiffrierschlüssels,606	SQL Remote-Syntax,690
Syntax,604	CREATE REMOTE [MESSAGE] TYPE-Anweisung
	SQL Remote-Syntax,694

- CREATE ROLE-Anweisung
 - Syntax,696
- CREATE SCHEMA-Anweisung
 - Syntax,697
- CREATE SEQUENCE-Anweisung
 - Syntax,699
- CREATE SERVER-Anweisung
 - Syntax,701
- CREATE SERVICE-Anweisung [SOAP-Webdienst]
 - Syntax,706,713
- CREATE SPATIAL REFERENCE SYSTEM-Anweisung
 - Syntax,719
- CREATE SPATIAL UNIT OF MEASURE-Anweisung
 - Syntax,727
- CREATE STATISTICS-Anweisung
 - Syntax,729
- CREATE SUBSCRIPTION-Anweisung
 - SQL Remote-Syntax,730
- CREATE SYNCHRONIZATION PROFILE-Anweisung
 - MobiLink-Syntax,732
- CREATE SYNCHRONIZATION SUBSCRIPTION-Anweisung
 - MobiLink-Syntax,733
- CREATE SYNCHRONIZATION USER-Anweisung
 - MobiLink-Syntax,735
- CREATE TABLE-Anweisung
 - entfernte Tabellen,739
 - Syntax,737
 - Transact-SQL,750
- CREATE TABLE-Klausel
 - INPUT-Anweisung,911
 - OUTPUT-Anweisung,969
- CREATE TEMPORARY PROCEDURE-Anweisung
 - Syntax,681
- CREATE TEMPORARY TRACE EVENT SESSION-Anweisung
 - Syntax,755
- CREATE TEMPORARY TRACE EVENT-Anweisung
 - Syntax,753
- CREATE TEXT CONFIGURATION-Anweisung
 - Syntax,757
- CREATE TEXT INDEX-Anweisung
 - Syntax,759
- CREATE TRIGGER-Anweisung
 - Syntax,762
 - Transact-SQL-Syntax,769
 - Triggerbedingungen,765
- CREATE USER-Anweisung
 - Kennwort, maximale Länge,770
 - Kennwörter, Zeichensatzkonvertierung,770
 - Syntax,770
 - Zeichensatzkonvertierung für Kennwörter,770
- CREATE VARIABLE-Anweisung
 - Syntax,771
- CREATE VIEW-Anweisung
 - Syntax,773
- CREATEDIRS-Klausel
 - CREATE SERVER-Anweisung,703
- CROSS APPLY-Klausel
 - FROM-Klausel,868
- CROSS JOIN-Klausel
 - FROM-Klausel, SQL-Syntax,863
- CSCONVERT-Funktion
 - Syntax,211
- CUBE, Vorgang
 - GROUP BY-Klausel,904
 - WITH CUBE-Klausel,904
- CUME_DIST-Funktion
 - Syntax,213
- CURRENT DATABASE-Klausel
 - CREATE TABLE-Anweisung,737
- CURRENT DATABASE-Spezialwert
 - Syntax,70
- CURRENT DATE-Funktion
 - TODAY-Funktion,411
- CURRENT DATE-Klausel
 - CREATE TABLE-Anweisung,737
- CURRENT DATE-Spezialwert
 - Syntax,70
- CURRENT PUBLISHER-Klausel
 - CREATE TABLE-Anweisung,737
- CURRENT PUBLISHER-Spezialwert
 - setzen,894
 - Syntax,71
- CURRENT REMOTE USER-Klausel
 - CREATE TABLE-Anweisung,737
- CURRENT REMOTE USER-Spezialwert
 - Syntax,71
- CURRENT TIME-Spezialwert
 - Syntax,72
- CURRENT TIMESTAMP-Klausel
 - CREATE TABLE-Anweisung,737

-
- CURRENT TIMESTAMP-Spezialwert
 - Syntax,73
 - CURRENT TIMESTAMP-Standardwert
 - CREATE TABLE-Anweisung,519,742
 - CURRENT USER-Klausel
 - CREATE TABLE-Anweisung,737
 - CURRENT USER-Spezialwert
 - Syntax,74
 - CURRENT UTC TIMESTAMP-Klausel
 - CREATE TABLE-Anweisung,737
 - CURRENT UTC TIMESTAMP-Spezialwert
 - Syntax,75
 - CURRENT UTC TIMESTAMP-Standardwert
 - CREATE TABLE-Anweisung,519,742
 - CURRENT-Klausel
 - DISCONNECT-Anweisung [ESQL] [Interactive SQL]-Anweisung,802
 - CURRENT_TIMESTAMP-Spezialwert
 - Syntax,73
 - CURRENT_USER-Spezialwert
 - Syntax,74
 - CurrentMultiProgrammingLevel-Eigenschaft
 - mit sa_server_option einstellen,1306
 - Cursor
 - Aktualisierbarkeit festgelegt in SELECT-Anweisung,1025
 - Anweisungen vorbereiten,976
 - Auflisten pro Verbindung,1248
 - CLOSE-Anweisung [ESQL] [SP],571 deklarieren,778
 - DESCRIBE-Anweisung [ESQL],794
 - Einstellungen für Datenbankoptionen ändern,1041
 - EXPLAIN-Anweisung, Syntax,851
 - in Transact-SQL deklarieren, DECLARE CURSOR-Anweisung,782
 - in Transact-SQL deklarieren, SELECT-Anweisung,1031
 - öffnen,964
 - OPEN-Anweisung,965
 - Schleifendurchlauf,857
 - Verhalten beim Beschreiben in benutzerdefinierten Prozeduren,684
 - Verhalten beim Beschreiben in externen Prozeduren,664
 - Wiederbeschreiben in benutzerdefinierten Prozeduren,684
 - Wiederbeschreiben in externen Prozeduren,664
 - Zeilen abrufen,853
 - Zeilen einfügen,981
 - Zeilen löschen,789
 - CYCLE-Klausel
 - ALTER SEQUENCE-Anweisung,491
 - CREATE SEQUENCE-Anweisung,700
- ## D
- DATA-Klausel
 - GET DESCRIPTOR-Anweisung [ESQL],878
 - SET DESCRIPTOR-Anweisung [ESQL],1033
 - DATABASE SIZE-Klausel
 - CREATE DATABASE-Anweisung,587
 - DATABASE-Klausel
 - CONNECT-Anweisung,578
 - DatabaseCleaner-Eigenschaft
 - mit sa_server_option einstellen,1306
 - DataLastModified-Eigenschaft
 - sa_materialized_view_info-Systemprozedur,1258
 - DATALength-Funktion
 - Syntax,214
 - DataStatus-Eigenschaft
 - sa_materialized_view_info-Systemprozedur,1258
 - DATATYPE-Klausel
 - ALTER SERVICE-Anweisung [SOAP über HTTP],500
 - CREATE SERVICE-Anweisung [SOAP-Webdienst],502,714
 - Date
 - HTTP_RESPONSE_HEADER-Funktion,286
 - DATE-Datentyp
 - Datums- und Zeitangaben an die Datenbank senden,117
 - ISO 8601-Kalenderdatum,117
 - ISO 8601-Ordinaldatum,117
 - ISO 8601-Wochendatum,117
 - mit Datums- und Uhrzeitangaben vergleichen,143
 - Syntax,123
 - zweideutige Datumsangaben lösen,118
 - DATE-Funktion
 - Syntax,216
 - DATEADD-Funktion
 - Syntax,217
 - DATEDIFF-Funktion
 - Syntax,218
 - datefirst-Option
 - SET-Anweisung, Syntax,1056
 - DATEFORMAT-Funktion

- Syntax,220
- Dateien
 - Abfrage in Dateien,867
 - auf Clientcomputer schreiben,432
 - auf einem Clientcomputer lesen,348
 - Daten in Tabellen importieren,910
 - Daten von Tabellen exportieren,968
 - Datenbanken mit der CREATE DBSPACE-Anweisung erstellen,593
 - mit der CREATE DECRYPTED DATABASE-Anweisung entschlüsseln,594
 - mit der CREATE DECRYPTED FILE-Anweisung entschlüsseln,596
 - mit der CREATE ENCRYPTED FILE-Anweisung verschlüsseln,604
 - Speicher für Datenbank zuweisen,457
 - SQL-Anweisungen lesen,984
 - xp_read_file-Systemprozedur,1424
 - xp_write_file-Systemprozedur,1435
- Dateigröße
 - Ereignisse mit der CREATE EVENT-Anweisung erstellen,606
- Dateiname
 - allgemeines Element der SQL-Syntax,445
- Daten
 - aus Tabellen in Dateien exportieren,968
 - importieren, in Tabellen aus Dateien,910
 - Zeilen auswählen,1020
- Daten entladen
 - Mehrbyte-Zeichensätze,1100
- Daten exportieren
 - aus Tabellen in Dateien,968
- Daten importieren
 - Binärdateien,1424
 - Grafiken,1424
 - in Tabellen aus Dateien,910
- Daten laden
 - Mehrbyte-Zeichensätze,940
- DATENAME-Funktion
 - Syntax,221
- Datenbank-Kennnummern
 - DBID_Funktion,230
- Datenbankaufräumvorgang
 - Info,1174
 - sa_clean_database-Systemprozedur,1174
- Datenbankdateien
 - Indizes speichern,640
 - mit der CREATE DECRYPTED DATABASE-Anweisung entschlüsseln,594
 - mit der CREATE DECRYPTED FILE-Anweisung entschlüsseln,596
 - mit der CREATE ENCRYPTED DATABASE-Anweisung verschlüsseln,601
 - mit der CREATE ENCRYPTED FILE-Anweisung verschlüsseln,604
 - mit der DROP DATABASE-Anweisung löschen,804
- Datenbanken
 - aus Archiven wiederherstellen,1001
 - Checkpoints mit der CHECKPOINT-Anweisung setzen,570
 - Dateien mit der CREATE DBSPACE-Anweisung erstellen,593
 - Daten mit UNLOAD-Anweisung entladen,1098
 - Massendaten einfügen,931
 - migrieren,1265
 - mit der BACKUP-Anweisung sichern,548
 - mit der CONNECT-Anweisung verbinden,578
 - mit der CREATE DATABASE-Anweisung erstellen,583
 - mit der DROP DATABASE-Anweisung löschen,804
 - mit der sa_validate-Systemprozedur validieren,1352
 - Rückgabe des Speicherorts der aktuellen Datenbank,230
 - Schema (Liste der Ansichten),1129
 - Schema (Liste der Tabellen),1437
 - Standard-Systemansichten,1437
 - starten,1062
 - SYSFILE-Systemansicht,1536
 - Systemprozeduren,1161
 - Systemtabellen,1129
 - Upgrade von jConnect mit der ALTER DATABASE-Anweisung,451
 - Verbindungen deaktivieren,1306
 - verschlüsselte Kopie einer Datenbank erstellen,601
- Datenbanken erstellen
 - CREATE DATABASE-Anweisung,583
- Datenbanken migrieren
 - sa_migrate-Systemprozedur,1265
- Datenbanken stoppen
 - STOP DATABASE-Anweisung,1074
- Datenbankextraktion
 - SQL Remote REMOTE RESET-Anweisung,995

Datenbanknamen
 Rückgabe mit der DB_NAME-Funktion,231

Datenbankobjekte
 identifizieren,4
 Kommentare mit der COMMENT-Anweisung
 hinzufügen,573

Datenbankoptionen
 Anfangseinstellungen und die
 sp_login_environment-Systemprozedur,1376
 Anfangseinstellungen und die
 sp_tsql_environment-Systemprozedur,1409
 benutzerdefinierte Optionen,1040
 in Transact-SQL einstellen,1056
 quoted_identifizier und T-SQL-Kompatibilität,41
 Transact-SQL-Kompatibilität,1409
 Zeilen aus Cursor abrufen,1041

Datenbankschemata
 Systemansichten,1437
 Systemtabellen,1129

Datenbankserver
 Datenbankspiegelung, ALTER MIRROR
 SERVER-Anweisung,480
 Datenbankspiegelung, CREATE MIRROR
 SERVER-Anweisung,656
 Optionen mit der sa_server_option-
 Systemprozedur setzen,1306
 Scale-Out mit Schreibschutz, ALTER MIRROR
 SERVER-Anweisung,480
 Scale-Out mit Schreibschutz, CREATE MIRROR
 SERVER-Anweisung,656
 START SERVER-Anweisung,1066
 STOP SERVER-Anweisung,1077

Datenbankserver, Meldungsfenster
 Meldungen anzeigen,959

Datenbankspiegelung
 ALTER MIRROR SERVER-Anweisung,480
 CREATE MIRROR SERVER-Anweisung,656
 Failover initiieren,453
 LOAD TABLE-Anweisung, Einschränkungen,944
 Server löschen,815
 SET MIRROR OPTION-Anweisung,1034

Datenbankvalidierung
 VALIDATE CHECKSUM-Anweisung,1118
 VALIDATE INDEX-Anweisung,1118

Datenbankverschlüsselung
 CREATE ENCRYPTED DATABASE-
 Anweisung,601

Datendekodierung
 BASE64_DECODE-Funktion,177
 HTML_DECODE-Funktion,278
 HTTP_DECODE-Funktion,281

Datenkodierung
 BASE64_ENCODE-Funktion,177
 HTML_ENCODE-Funktion,279

Datentypen
 abrufen,262
 benutzerdefinierte Domänen,138
 benutzerdefinierte mit der DROP DATATYPE-
 Anweisung löschen,805
 BIGINT-Datentyp,104
 BINARY-Datentyp,133
 BIT-Datentyp,105
 CHAR-Datentyp,95
 CREATE DOMAIN-Anweisung,598
 DATE-Datentyp,123
 DATETIME-Datentyp,124
 DATETIMEOFFSET-Datentyp,125
 DECIMAL-Datentyp,106
 DOUBLE-Datentyp,107
 FLOAT-Datentyp,108
 IMAGE-Datentyp,134
 INTEGER-Datentyp,109
 Java und SQL konvertieren,150
 Kompatibilität,145
 LONG BINARY-Datentyp,134
 LONG NVARCHAR-Datentyp,97
 LONG VARBIT-Datentyp,114
 LONG VARCHAR-Datentyp,97
 mit der ALTER DOMAIN-Anweisung ändern,460
 NCHAR,98
 NCHAR (NATIONAL CHAR),98
 NTEXT-Datentyp,99
 NUMERIC-Datentyp,109
 NVARCHAR-Datentyp,99
 REAL-Datentyp,111
 Rundungsfehler,103
 SMALLDATETIME-Datentyp,127
 SMALLINT-Datentyp,112
 SMALLMONEY-Datentyp,114
 Spezialwerte,70
 SQL-Konvertierungsfunktionen,156
 SYSDOMAIN-Systemansicht,1446
 SYSEXTERNLOGIN-Systemansicht,1451
 SYSUSERTYPE-Systemansicht,1510
 TEXT-Datentyp,100
 TIME-Datentyp,128

- TIMESTAMP WITH TIME ZONE-Datentyp,131
- TIMESTAMP-Datentyp,129
- TINYINT-Datentyp,112
- Unicode,95
- UNIQUEIDENTIFIER-Datentyp,134
- UNIQUEIDENTIFIERSTR-Datentyp,101
- VARBINARY-Datentyp,135
- VARBIT-Datentyp,115
- VARCHAR-Datentyp,101
- Werte vergleichen,140
- XML-Datentyp,103
- Zeichen,95
- zusammengesetzte Datentypen,136
- Datentypen für Datum und Uhrzeit
 - DATETIMEOFFSET,125
 - Info,116
 - TIME,128
 - TIMESTAMP,129
 - TIMESTAMP WITH TIME ZONE,131
- Datentypkonvertierung
 - beim Auswerten von Ausdrücken,140
 - CHAR- und NCHAR-Werte vergleichen,141
 - für Vergleichsoperatoren,140
 - Info,146
- Datentypkonvertierungen
 - CAST,186
 - DOUBLE in NUMERIC konvertieren,149
 - Info,146
 - Java in SQL,150
 - NCHAR in CHAR konvertieren,147
 - SQL in Java,150,151
 - Vergleichsoperatoren,140
- Datentypkonvertierungsfunktionen
 - Info,156
- Datenyp
 - allgemeines Element der SQL-Syntax,445
- Datenzugriffspläne
 - Textspezifikationen abrufen,851
- DATEPART-Funktion
 - Syntax,221
- DATETIME-Datentyp
 - Datums- und Zeitangaben an die Datenbank senden,117
 - Syntax,124
- DATETIME-Funktion
 - Syntax,222
- DATETIMEOFFSET-Datentyp
 - Datums- und Zeitangaben an die Datenbank senden,117
 - Syntax,125
- Datum/Zeit
 - Konvertierungsfunktionen,156
- Datums- und Zeitangaben aus der Datenbank abrufen
 - Info,122
- Datums- und Zeitangaben vergleichen
 - Info,143
- Datums- und Zeitdatentypen
 - DATE,123
 - DATETIME,124
 - Datumsformate,117
 - Datumsformate mit Uhrzeit,120
 - ISO 8601,117
 - SMALLDATETIME,127
 - Zeitformate,119
 - Zeitzoneformate,121
- Datumsangaben
 - 29. Februar,122
 - Abfrage des aktuellen Systemdatums,269
 - Abfragen,122
 - abrufen,123
 - an die Datenbank senden,117
 - eine Tabelle generieren,1301
 - einfügen,123
 - Interpretation,123
 - konvertieren von Zeichenfolgen,145
 - Konvertierungsfunktionen,156
 - Schaltjahre,122
 - speichern,116
 - SQL Anywhere,116
 - vergleichen,143
- Datumsfunktionen
 - alphabetische Liste,156
- Datumsteile
 - Info,156
- DAY-Funktion
 - Syntax,223
- DAYNAME-Funktion
 - Syntax,224
- DAYS-Funktion
 - Syntax,224
- DB2
 - nach SQL Anywhere mit sa_migrate-Systemprozedur migrieren,1266
- db_charset
 - CSCONVERT-Funktion,211

DB_EXTENDED_PROPERTY-Funktion
Syntax,226

DB_ID-Funktion
Syntax,230

DB_NAME-Funktion
Syntax,231

DB_PROPERTY-Funktion
Syntax,232

db_publisher-Option
mit der GRANT PUBLISH-Anweisung
festlegen,894

db_register_a_callback, Funktion
mit MESSAGE TO CLIENT verwenden,962

DBA PASSWORD-Klausel
CREATE DATABASE-Anweisung,587

DBA USER-Klausel
CREATE DATABASE-Anweisung,587

DBFILE ONLY-Klausel
BACKUP-Anweisung,550

DBFreePercent-Ereignisbedingung
Info,254

DBFreeSpace-Ereignisbedingung
Info,254

dbmlysync, Dienstprogramm
Speicherort mit der ALTER EXTERNAL
ENVIRONMENT-Anweisung angeben,463

dbname FORCE START-Klausel
ALTER DATABASE-Anweisung,453

dbo, Benutzer
RowGenerator-Systemtabelle,1159
Transact-SQL-Kompatibilitätsansichten,1546

dbo-Systemrolle
mit der GRANT-Anweisung erteilen,882
mit der REVOKE-Anweisung entziehen,1009

DBSize-Ereignisbedingung
Info,254

DBSpaces
Kommentare mit der COMMENT-Anweisung
hinzufügen,573
mit der ALTER DBSPACE-Anweisung ändern,457
mit der CREATE DBSPACE-Anweisung
erstellen,593
mit der CREATE DECRYPTED DATABASE-
Anweisung entschlüsseln,594
mit der CREATE DECRYPTED FILE-Anweisung
entschlüsseln,596
mit der CREATE ENCRYPTED DATABASE-
Anweisung verschlüsseln,601
mit der CREATE ENCRYPTED FILE-Anweisung
verschlüsseln,604
mit der DROP DBSPACE-Anweisung löschen,806
Seiten hinzufügen,458
SYSFILE-Systemansicht,1536
verfügbaren Platz bestimmen,1212
Verwendungsprivilegien erteilen,891

deadlock_logging-Eigenschaft
mit sa_server_option einstellen,1306

DeadlockLogging-Eigenschaft
mit sa_server_option einstellen,1306

Deadlocks
protokollieren,1306
sa_report_deadlocks-Systemprozedur,1296

Deadlocks, melden
sa_report_deadlocks-Systemprozedur,1296

DEALLOCATE DESCRIPTOR-Anweisung
Embedded SQL-Syntax,777

DEALLOCATE-Anweisung
Syntax,777

DEBUG ONLY-Klausel
MESSAGE-Anweisung,960

DebuggingInformation-Eigenschaft
mit sa_server_option einstellen,1306

DECIMAL-Datentyp
Syntax,106

DECLARE CURSOR-Anweisung
Embedded SQL-Syntax,778
Syntax,778
Transact-SQL-Syntax,782

DECLARE EXCEPTION
mit BEGIN-Anweisung verwendet,557
mit TRY-Anweisung verwendet,1093

DECLARE LOCAL TEMPORARY TABLE-
Anweisung
Syntax,784

DECLARE-Anweisung
mit BEGIN-Anweisung verwendet,557
mit BEGIN-Anweisung verwendet [TSQL],560
mit TRY-Anweisung verwendet,1093
Syntax,786

DECOMPRESS-Funktion
Syntax,233

DECRYPT-Funktion
Syntax,234

DEFAULT
in einer CALL-Anweisung angeben,564

DEFAULT LAST USER

- Spaltenreplikation in SQL Remote vermeiden,920
- DEFAULT TIMESTAMP, Spalten
 - TIMESTAMP-Spezialwert,82
- DEFAULT VALUES-Klausel
 - INSERT-Anweisung,918
- DEFAULT-Klausel
 - ALTER TABLE-Anweisung,518
 - CREATE DOMAIN-Anweisung,598
 - CREATE SERVICE-Anweisung [HTTP-Webdienst],706
 - CREATE SERVICE-Anweisung [SOAP-Webdienst],713
 - CREATE TABLE-Anweisung,741
 - Spezialwerte,519,742
- DEFAULTS-Klausel
 - LOAD TABLE-Anweisung,936
- DEFINITION-Klausel
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung,507
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung,720
- Definitionen
 - Tabellen mit der ALTER TABLE-Anweisung ändern,516
- Defragmentierung
 - REORGANIZE TABLE,998
- DEGREES-Funktion
 - Syntax,237
- Deklarieren
 - Ausnahmen mit DECLARE-Anweisung,786
 - Cursor,778
 - Cursor in Transact-SQL, DECLARE CURSOR-Anweisung,782
 - Cursor in Transact-SQL, SELECT-Anweisung,1031
 - Hostvariablen in Embedded SQL,778
 - Variablen mit DECLARE-Anweisung,786
 - zusammengesetzte Datentypen,136
- DELAY-Klausel
 - WAITFOR-Anweisung,1121
- DELETE ALL OPTION-Klausel
 - ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink] ,513
 - ALTER SYNCHRONIZATION USER-Anweisung [MobiLink] ,515
- DELETE OPTION-Klausel
 - ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink] ,513
 - ALTER SYNCHRONIZATION USER-Anweisung [MobiLink] ,515
- DELETE TABLE-Klausel
 - ALTER PUBLICATION-Anweisung,485
- DELETE TYPE-Klausel
 - ALTER EVENT-Anweisung,462
- DELETE-Anweisung
 - (positionsbasierte) Anweisung, Syntax,789
 - Datenbankoptionen einstellen,793
 - Syntax,791
- DELETE-Anweisung (positionsbasiert)
 - Embedded SQL-Syntax,789
- DELETE-Klausel
 - CREATE LOCAL TEMPORARY TABLE-Anweisung,645
 - CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote],690
 - CREATE TRIGGER-Anweisung,762
 - DECLARE LOCAL TEMPORARY TABLE-Anweisung,784
 - MERGE-Anweisung,956
- DELETE-Privileg
 - GRANT-Anweisung,883
 - REVOKE-Anweisung,1010
- DELETE-Privilegklausel
 - GRANT-Anweisung,885
- DELIMITED BY-Klausel
 - LOAD TABLE Anweisung,936
 - OUTPUT-Anweisung,969
 - UNLOAD-Anweisung,1099
- DELIMITED-Klausel
 - INPUT-Anweisung,912
- DENSE_RANK-Funktion
 - Syntax,237
- DESCRIBE CONNECTION-Anweisung
 - Interactive SQL-Syntax,798
- DESCRIBE-Anweisung
 - DT_DESCRIBE_INPUT,795
 - DT_HIDDEN_COLUMN,779
 - DT_KEY_COLUMN,779
 - DT_PROCEDURE_IN,795
 - DT_PROCEDURE_OUT,795
 - Embedded SQL-Syntax,794
 - Interactive SQL-Syntax,798
 - lange Spaltennamen,796
- DESCRIBE-Klausel
 - PREPARE-Anweisung,977
- DESCRIPTOR-Klausel

-
- EXECUTE-Anweisung,846
 - Deskriptorbereiche
 - einstellen,1033
 - EXECUTE-Anweisung,846
 - Informationen abrufen,878
 - Speicher zuweisen,449
 - UPDATE-Anweisung (positionsbasiert),1103
 - Zuweisungen aufheben,777
 - Deskriptoren
 - Anweisungen vorbereiten,976
 - DESCRIBE-Anweisung [ESQL],794
 - FETCH-Anweisung,853
 - DETACH TRACING-Anweisung
 - Diagnoseprotokollierung,801
 - Syntax,801
 - Deterministisches Verhalten
 - benutzerdefinierte Funktionen,634
 - externe Funktionen,619
 - Diagnosen
 - sa_performance_statistics-Systemprozedur,1285
 - Diagnoseprotokollierung
 - ATTACH TRACING-Anweisung,546
 - DETACH TRACING-Anweisung,801
 - REFRESH TRACING LEVEL-Anweisung,993
 - sa_diagnostic_auxiliary_catalog-Tabelle,1144
 - sa_diagnostic_blocking-Tabelle,1145
 - sa_diagnostic_cachecontents-Tabelle,1146
 - sa_diagnostic_connection-Tabelle,1147
 - sa_diagnostic_cursor-Tabelle,1148
 - sa_diagnostic_deadlock-Tabelle,1149
 - sa_diagnostic_hostvariable-Tabelle,1150
 - sa_diagnostic_internalvariable-Tabelle,1151
 - sa_diagnostic_query-Tabelle,1151
 - sa_diagnostic_request-Tabelle,1153
 - sa_diagnostic_statement-Tabelle,1154
 - sa_diagnostic_statistics-Tabelle,1155
 - sa_diagnostic_tracing_level-Tabelle,1156
 - sa_save_trace_data-Systemprozedur,1302
 - sa_set_tracing_level-Systemprozedur,1330
 - Tabellen, Info,1144
 - Diagnoseprotokollierungsstufe
 - auf der Befehlszeile einstellen,1330
 - DIAGNOSTICS-Systemrolle
 - mit der GRANT-Anweisung erteilen,882
 - mit der REVOKE-Anweisung entziehen,1009
 - Dienste
 - Ändern der SOAP-Dienste mit der ALTER SERVICE-Anweisung [SOAP-Webdienst],500
 - Ändern von HTTP-Diensten mit der ALTER SERVICE-Anweisungen [HTTP-Webdienst],494
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - Webdienste mit der DROP SERVICE-Anweisung löschen,824
 - DIFFERENCE, Differenzmenge
 - Syntax,239
 - DIRECTORY-Klausel
 - BACKUP-Anweisung,549
 - START DATABASE-Anweisung,1063
 - DISABLE USE IN OPTIMIZATION-Klausel
 - ALTER MATERIALIZED VIEW-Anweisung,477
 - DISABLE VIEW DEPENDENCIES-Klausel
 - ALTER TABLE-Anweisung,528
 - DISABLE-Klausel
 - ALTER EVENT-Anweisung,461
 - ALTER MATERIALIZED VIEW-Anweisung,477
 - ALTER SERVICE-Anweisung [SOAP über HTTP],500
 - ALTER SERVICE-Anweisungen [HTTP-Webdienst],494
 - ALTER VIEW-Anweisung,544
 - CREATE SERVICE-Anweisung [HTTP-Webdienst],497,709
 - CREATE SERVICE-Anweisung [SOAP-Webdienst],503,716
 - DISCONNECT-Anweisung
 - Embedded SQL-Syntax,802
 - Interactive SQL-Syntax,802
 - DISH
 - CREATE SERVICE-Anweisung [SOAP-Webdienst],501,714
 - DISH, Dienste
 - Schrägstriche im Namen nicht zulässig,501,714
 - DiskSandbox-Eigenschaft
 - mit sa_db_option einstellen,1199
 - mit sa_server_option einstellen,1306
 - DISKSANDBOX-Klausel
 - START DATABASE-Anweisung,1064
 - DISTINCT, Schlüsselwort
 - Info,1022
 - DISTINCT-Klausel
 - MEDIAN-Funktion,311
 - NULL,78
 - SELECT-Anweisung,1022
 - Distinguished Name (DN)
 - ALTER LDAP SERVER-Anweisung,468
-

- ALTER LOGIN POLICY-Anweisung,471
- ALTER USER-Anweisung,542
- CREATE LDAP SERVER-Anweisung,642
- sa_get_user_status-Systemprozedur,1233
- SYSLDAPSERVER-Systemansicht,1461
- SYSUSER-Systemansicht,1506
- DML
 - auswählen aus,868
- DO-Klausel
 - FOR-Anweisung,857
- Dokumentation
 - Konventionen für die SQL-Syntax,445
- DOMAIN | DATATYPE-Klausel
 - CREATE DOMAIN-Anweisung,598
- Domänen
 - CREATE DOMAIN-Anweisung,598
 - Info,138
 - mit der ALTER DOMAIN-Anweisung ändern,460
 - mit der DROP DOMAIN-Anweisung löschen,807
 - Nullwertfähigkeit,598
 - Transact-SQL,140
- Domänen erstellen
 - CREATE DOMAIN-Anweisung,598
- Doppel-Bindestrich
 - Kommentar-Bezeichner,92
- Doppel-Schrägstrich
 - Kommentar-Bezeichner,92
- DOUBLE-Datentyp
 - in NUMERIC konvertieren,149
 - mit Datums- und Uhrzeitangaben vergleichen,143
- Syntax,107
- DOW-Funktion
 - Syntax,240
- Drei-Werte-Logik
 - NULL,77
 - Syntax,67
- DRIVER-Klausel
 - INPUT-Anweisung,910
- DriveType-Eigenschaft
 - DB_EXTENDED_PROPERTY-Funktion,226
- DROP CERTIFICATE-Anweisung
 - Syntax,803
- DROP CONNECTION-Anweisung
 - Syntax,803
- DROP DATABASE-Anweisung
 - Syntax,804
- DROP DATATYPE-Anweisung
 - Syntax,805
- DROP DBSPACE-Anweisung
 - Syntax,806
- DROP DOMAIN-Anweisung
 - Syntax,807
- DROP EVENT-Anweisung
 - Syntax,808
- DROP EXTERNLOGIN-Anweisung
 - Syntax,808
- DROP FUNCTION-Anweisung
 - Syntax,809
- DROP INDEX-Anweisung
 - Syntax,810
- DROP LDAP SERVER-Anweisung
 - Syntax,811
- DROP LOGIN POLICY-Anweisung
 - Syntax,812
- DROP MATERIALIZED VIEW-Anweisung
 - Syntax,813
- DROP MESSAGE-Anweisung
 - Syntax,814
- DROP MIRROR SERVER-Anweisung
 - Syntax,815
- DROP PCTFREE-Klausel
 - ALTER MATERIALIZED VIEW-Anweisung,477
- DROP PREFILTER-Klausel
 - ALTER TEXT CONFIGURATION-Anweisung,534
- DROP PROCEDURE-Anweisung
 - Syntax,816
- DROP PUBLICATION-Anweisung
 - MobiLink-Syntax,817
 - SQL Remote-Syntax,817
- DROP REMOTE CONNECTION-Anweisung
 - Syntax,819
- DROP REMOTE MESSAGE TYPE-Anweisung
 - SQL Remote-Syntax,818
- DROP ROLE-Anweisung
 - Syntax,820
- DROP SEQUENCE-Anweisung
 - Syntax,822
- DROP SERVER-Anweisung
 - Syntax,823
- DROP SERVICE-Anweisung
 - Syntax,824
- DROP SPATIAL REFERENCE SYSTEM-Anweisung
 - Syntax,825
- DROP SPATIAL UNIT OF MEASURE-Anweisung

Syntax,825
 DROP STATEMENT-Anweisung
 Embedded SQL-Syntax,826
 DROP STATISTICS-Anweisung
 Syntax,827
 DROP STOPLIST-Klausel
 ALTER TEXT CONFIGURATION-
 Anweisung,532
 DROP SUBSCRIPTION-Anweisung
 SQL Remote-Syntax,828,1069
 DROP SYNCHRONIZATION PROFILE-Anweisung
 MobiLink-Syntax,829
 DROP SYNCHRONIZATION SUBSCRIPTION-
 Anweisung
 Syntax,830
 DROP SYNCHRONIZATION USER-Anweisung
 MobiLink-Syntax,831
 DROP TABLE-Anweisung
 Syntax,832
 DROP TABLE-Klausel
 ALTER PUBLICATION-Anweisung,485
 DROP TEXT CONFIGURATION-Anweisung
 Syntax,833
 DROP TEXT INDEX-Anweisung
 Syntax,834
 DROP TRACE EVENT SESSION-Anweisung
 Syntax,836
 DROP TRACE EVENT-Anweisung
 Syntax,835
 DROP TRIGGER-Anweisung
 Syntax,837
 DROP USER-Anweisung
 Syntax,838
 DROP VARIABLE-Anweisung
 Syntax,839
 DROP VIEW-Anweisung
 Syntax,840
 DROP-Klausel
 ALTER TABLE-Anweisung,527
 DROP EXTERNLOGIN-Anweisung,809
 DropBadStatistics-Eigenschaft
 mit sa_server_option einstellen,1306
 DropUnusedStatistics-Eigenschaft
 mit sa_server_option einstellen,1306
 Drucken
 Meldungen im Meldungsfenster,980
 DSN-Klausel
 INPUT-Anweisung,910
 DT_DESCRIBE_INPUT
 DESCRIBE-Anweisung,795
 DT_HIDDEN_COLUMN
 DESCRIBE-Anweisung,779
 DT_KEY_COLUMN
 DESCRIBE-Anweisung,779
 DT_PROCEDURE_IN
 DESCRIBE-Anweisung,795
 DT_PROCEDURE_OUT
 DESCRIBE-Anweisung,795
 DUMMY
 Systemtabelle,1129
 Zeilenkonstruktor-Algorithmus,1129
 Durchreichmodus
 FORWARD TO-Anweisung,861
 PASSTHROUGH-Anweisung [SQL Remote],975
 starten,975
 stoppen,975
 Durchschnittsfunktion
 AVG-Funktion,175
 DYNAMIC RESULT SETS-Klausel
 CREATE PROCEDURE-Anweisung [externer
 Aufruf],664
 DYNAMIC SCROLL-Cursor
 deklarieren,778
 DYNAMIC SCROLL-Klausel
 DECLARE CURSOR-Anweisung,779
 FOR-Anweisung,858
 Dynamic SQL
 Prozeduren ausführen,843
E
 E-Mail
 erweiterte Systemprozeduren,1162
 Systemprozeduren,1162
 xp_sendmail-Systemprozedur,1426
 Eckige Klammern
 Datenbankobjekte,4
 SQL-Bezeichner,4
 Eigenschaften
 CONNECTION_PROPERTY-Funktion,199
 DB_PROPERTY-Funktion,232
 PROPERTY-Funktion,339
 Eigentümer
 allgemeines Element der SQL-Syntax,445
 Eindeutig

- Integritätsregel in der CREATE TABLE-Anweisung, 745
- Eindeutige Indizes
 - Info, 638
- Einfügen
 - BLOBs mit der SET-Anweisung, 1054
 - BLOBs mit der xp_read_file-Systemprozedur, 1424
 - breite Einfügungen, 846
 - Daten mit LOAD TABLE-Anweisung, 931
 - mehrzeilig, 846
 - Zeilen in Massendaten, 931
 - Zeilen in Tabellen, 917
 - Zeilen mit Cursor, 981
- Einfügen, BLOBs
 - mit der xp_read_file-Systemprozedur, 1424
- Einrichten
 - Savepoints, 1019
- Einstellen
 - Benutzer, 1059
 - Deskriptorbereiche, 1033
 - entfernte Optionen, 1046
 - Optionen, 1039
 - Optionen in Interactive SQL, 1043
 - Optionen in Transact-SQL, 1056
 - SQLCA, 1053
 - Verbindungen, 1032
 - Werte von SQL-Variablen, 1054
- Elemente
 - SQL-Sprachsyntax, 1
- ELLIPSOID-Klausel
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung, 508
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung, 724
- ELSE
 - CASE-Ausdruck, 25
 - IF-Ausdrücke, 24
- ELSE-Klausel
 - IF-Anweisung, 906
 - IF-Anweisung [T-SQL], 908
- ELSEIF-Klausel
 - IF-Anweisung, 906
- Embedded SQL
 - ALLOCATE DESCRIPTOR, Syntax, 449
 - BEGIN DECLARE-Anweisung, Syntax, 778
 - CLOSE-Anweisung, Syntax, 571
 - CONNECT-Anweisung, Syntax, 578
 - DEALLOCATE DESCRIPTOR-Anweisung, SQL-Syntax, 777
 - DECLARE CURSOR-Anweisung, Syntax, 778
 - DELETE-Anweisung (positionsbasiert), Syntax, 789
 - DESCRIBE-Anweisung, Syntax, 794
 - DISCONNECT-Anweisung, Syntax, 802
 - DROP STATEMENT-Anweisung, Syntax, 826
 - END DECLARE-Anweisung, Syntax, 778
 - EXECUTE IMMEDIATE-Anweisung, Syntax, 843
 - EXECUTE-Anweisung, Syntax, 846
 - EXPLAIN-Anweisung, Syntax, 851
 - GET DATA-Anweisung, Syntax, 876
 - GET DESCRIPTOR-Anweisung, Syntax, 878
 - GET OPTION-Anweisung, Syntax, 880
 - INCLUDE-Anweisung, Syntax, 909
 - OPEN-Anweisung, Syntax, 964
 - PREPARE-Anweisung, Syntax, 976
 - PUT-Anweisung, Syntax, 981
 - SET CONNECTION-Anweisung, Syntax, 1032
 - SET DESCRIPTOR-Anweisung, Syntax, 1033
 - SET SQLCA-Anweisung, Syntax, 1053
 - UPDATE-Anweisung (positionsbasiert), 1103
 - WHENEVER-Anweisung, Syntax, 1122
- ENABLE USE IN OPTIMIZATION-Klausel
 - ALTER MATERIALIZED VIEW-Anweisung, 477
- ENABLE | DISABLE-Klausel
 - CREATE EVENT-Anweisung, 610
- ENABLE-Klausel
 - ALTER MATERIALIZED VIEW-Anweisung, 477
 - ALTER SERVICE-Anweisung [SOAP über HTTP], 500
 - ALTER SERVICE-Anweisungen [HTTP-Webdienst], 494
 - ALTER VIEW-Anweisung, 544
 - CREATE SERVICE-Anweisung [HTTP-Webdienst], 497, 709
 - CREATE SERVICE-Anweisung [SOAP-Webdienst], 503, 716
- ENCODING-Klausel
 - CREATE DATABASE-Anweisung, 588
 - INPUT-Anweisung, 912
 - LOAD TABLE-Anweisung, 937
 - OUTPUT-Anweisung, 969
 - READ-Anweisung [Interactive SQL], 984
 - UNLOAD-Anweisung, 1099
- ENCRYPT-Funktion
 - Syntax, 241

<p>ENCRYPTED DATABASE-Klausel CREATE ENCRYPTED DATABASE- Anweisung,601</p> <p>ENCRYPTED TABLE DATABASE-Klausel CREATE ENCRYPTED TABLE DATABASE- Anweisung,601</p> <p>ENCRYPTED TABLE-Klausel CREATE DATABASE-Anweisung,588</p> <p>ENCRYPTED-Klausel ALTER MATERIALIZED VIEW-Anweisung,477 CREATE DATABASE-Anweisung,588 CREATE TABLE-Anweisung,739 LOAD TABLE-Anweisung,937 UNLOAD-Anweisung,1100</p> <p>END CASE-Ausdruck,25</p> <p>END CASE CASE-Ausdruck,25</p> <p>END CATCH-Anweisung Syntax,1093</p> <p>END DECLARE-Anweisung Embedded SQL-Syntax,778</p> <p>END FOR-Klausel FOR-Anweisung,857</p> <p>END IF Ausdrücke,24</p> <p>END IF-Klausel IF-Anweisung,906</p> <p>END LOOP-Klausel LOOP-Anweisung,950</p> <p>END TRY-Anweisung Syntax,1093</p> <p>END-Anweisung mit BEGIN-Anweisung verwendet,557 mit BEGIN-Anweisung verwendet [TSQL],560</p> <p>END-Schlüsselwort TSQL-Kompatibilität,560</p> <p>ENDIF IF-Ausdrücke,24</p> <p>ENDIF-Klausel IF-Anweisung,906</p> <p>Engines Datenbank starten,1066 Datenbank stoppen,1077</p> <p>Entfernen Ansichten mit der DROP VIEW-Anweisung,840 Datenbankdateien mit der DROP DATABASE- Anweisung,804</p>	<p>DBSpaces mit der DROP DBSPACE- Anweisung,806</p> <p>Ereignisse mit der DROP EVENT-Anweisung,808</p> <p>Funktionen mit der DROP FUNCTION- Anweisung,809</p> <p>Indizes mit der DROP INDEX-Anweisung,810</p> <p>Java-Klassen,997</p> <p>materialisierte Ansichten mit der DROP MATERIALIZED VIEW-Anweisung,813</p> <p>Optimiererstatistiken mit der DROP STATISTICS- Anweisung,827</p> <p>Privilegien,1009</p> <p>Prozeduren mit der DROP PROCEDURE- Anweisung,816</p> <p>Spalten mit der ALTER TABLE-Anweisung,516</p> <p>SQL-Variablen mit der DROP VARIABLE- Anweisung,839</p> <p>Tabellen mit der DROP TABLE-Anweisung,832</p> <p>Textkonfigurationsobjekte,833</p> <p>Trigger mit der DROP FUNCTION- Anweisung,809</p> <p>Trigger mit der DROP-Anweisung,837</p> <p>vorbereitete Anweisungen mit der DROP STATEMENT-Anweisung,826</p> <p>Entfernte Benutzer SQL Remote REVOKE REMOTE- Anweisung,1008</p> <p>Entfernte Funktionen AT-Klausel,635</p> <p>Entfernte gespeicherte Prozeduren Ergebnismengen,1391 Parameter,1391</p> <p>Entfernte Nachrichtentypen löschen,818 SQL Remote, ändern,487 SQL Remote, erstellen,694</p> <p>Entfernte Optionen SET REMOTE OPTION-Anweisung [SQL Remote],1046</p> <p>Entfernte Prozeduren AT-Klausel,685 in Transact-SQL erstellen,679 mit der sp_remote_procedures-Systemprozedur auflisten,1395 Voraussetzungen für RESULT-Klausel,686</p> <p>Entfernte Tabellen CREATE TABLE-Anweisung,739</p>
--	--

- Fremdschlüssel und sp_remote_exported_keys-Systemprozedur,1388
- Fremdschlüssel und sp_remote_imported_keys-Systemprozedur,1390
- mit der sp_remote_tables-Systemprozedur auflisten,1396
- Primärschlüssel und sp_remote_exported_keys-Systemprozedur,1388
- Primärschlüssel und sp_remote_imported_keys-Systemprozedur,1390
- Spalten,1386
- Entladen
 - Daten mit UNLOAD-Anweisung entladen,1098
 - Ergebnismengen mit UNLOAD-Anweisung entladen,1098
 - Kostenmodelle,1348
- Entschlüsseln
 - Dateien mit der CREATE DECRYPTED DATABASE-Anweisung,594
 - Dateien mit der CREATE DECRYPTED FILE-Anweisung,596
 - materialisierte Ansichten mit Sybase Central,476
 - von Tabellen mit der ALTER TABLE-Anweisung,516
- Entziehen
 - REVOKE-Anweisung,1009
 - SQL Remote, CONSOLIDATE-Privileg,1006
 - SQL Remote, PUBLISH-Privileg,1007
 - SQL Remote, REMOTE-Privileg,1008
 - SQL Remote-Replikationsprivilegien,1013,1014
- ENVIRONMENT-Klausel
 - INSTALL EXTERNAL OBJECT-Anweisung,923
- Ereignisbedingungen
 - Liste,254
- Ereignisprotokollierung
 - ALTER TRACE EVENT SESSION-Anweisung,538
 - CREATE TEMPORARY TRACE EVENT SESSION-Anweisung,755
 - CREATE TEMPORARY TRACE EVENT-Anweisung,753
 - DROP TRACE EVENT SESSION-Anweisung,836
 - DROP TRACE EVENT-Anweisung,835
 - NOTIFY TRACE EVENT-Anweisung,963
- Ereignisse
 - ALTER EVENT-Anweisung,461
 - CREATE EVENT-Anweisung,606
 - deaktivieren,461
 - DROP EVENT-Anweisung,808
 - EVENT_PARAMETER,256
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - mit der ALTER EVENT-Anweisung ändern,461
 - mit der ALTER EVENT-Anweisung planen,461
 - mit der CREATE EVENT-Anweisung planen,606
 - mit der DROP EVENT-Anweisung löschen,808
 - TRIGGER EVENT-Anweisung,1088
 - Trigger-auslösend,1088
- Ergebnismengen
 - aus gespeicherten Prozeduren auswählen,865
 - Ausführung von Prozeduren wieder aufnehmen,1003
 - Daten mit UNLOAD-Anweisung entladen,1098
 - mehrere Ergebnismengen abrufen,1003
 - variabel, CREATE PROCEDURE-Anweisung,683
 - variabel, CREATE PROCEDURE-Anweisung (externer Aufruf],663
 - variabel, DESCRIBE-Anweisung [ESQL],796
 - variabel, PREPARE-Anweisung,977
- Erneut anzeigen
 - Ausnahmebedingungen,1000
- ERROR_LINE-Funktion
 - Syntax,244
- ERROR_MESSAGE-Funktion
 - Syntax,245
- ERROR_PROCEDURE-Funktion
 - Syntax,246
- ERROR_SQLCODE-Funktion
 - Syntax,247
- ERROR_SQLSTATE-Funktion
 - Syntax,248
- ERROR_STACK_TRACE-Funktion
 - Syntax,249
- ERRORMSG-Funktion
 - Syntax,251
- ErrorNumber-Ereignisbedingung
 - Info,254
- Ersetzungszeichen
 - Vergleiche zwischen CHAR und NCHAR,141
 - verlustreiche Konvertierungen,140
- Erstellen
 - Benutzer erstellen mit der CREATE USER-Anweisung,770
 - CREATE INDEX-Anweisung,638
 - CREATE MATERIALIZED VIEW-Anweisung,652

-
- CREATE PUBLICATION-Anweisung,690
 - CREATE SYNCHRONIZATION PROFILE-Anweisung,732
 - CREATE SYNCHRONIZATION SUBSCRIPTION-Anweisung,733
 - CREATE TABLE-Anweisung,737
 - CREATE TRIGGER-Anweisung,762
 - CREATE VIEW-Anweisung,773
 - Cursor,778
 - Datenbanken mit der CREATE DATABASE-Anweisung,583
 - Datenbankendateien mit der CREATE DBSPACE-Anweisung,593
 - gespeicherte Prozeduren,681
 - LDAP-Serverkonfigurationsobjekte,642
 - lokale temporäre Tabellen,784
 - lokale temporäre Tabellen mit der CREATE LOCAL TEMPORARY TABLE-Anweisung,645
 - Meldungen,655
 - Proxy-Prozeduren mit der sp_remote_procedures-Systemprozedur,1395
 - Proxytabellen,739
 - Proxytabellen mit der CREATE EXISTING TABLE-Anweisung,613
 - Proxytabellen mit der sp_remote_tables-Systemprozedur,1396
 - Savepoints,1019
 - Schemata,697
 - Server,701
 - Sicherungen von Datenbanken mit der BACKUP-Anweisung,548
 - SQL Remote, entfernte Nachrichtentypen,694
 - SQL-Variable mit DECLARE-Anweisung,786
 - SQL-Variablen mit der CREATE VARIABLE-Anweisung,771
 - Subskriptionen,730
 - Textindizes für die Volltextsuche,759
 - Textkonfigurationsobjekte für die Volltextsuche,757
 - Trigger in Transact-SQL,769
 - Zertifikate,582
 - Erstellen von Ansichten
 - CREATE VIEW-Anweisung,773
 - Erstellen von Indizes
 - CREATE INDEX-Anweisung,638
 - Erstellen von Login-Richtlinien
 - CREATE LOGIN POLICY-Anweisung,647
 - Erstellen von materialisierten Ansichten
 - CREATE MATERIALIZED VIEW-Anweisung,652
 - Erstellen von Synchronisationsprofilen
 - CREATE SYNCHRONIZATION PROFILE-Anweisung [MobiLink],732
 - Erstellen von Synchronisationssubskriptionen
 - CREATE SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink],733
 - Erteilen
 - CONSOLIDATE-Privileg,889
 - Impersonierung (SET USER-Systemprivileg),881
 - Privilegien,881
 - Privilegien auf Objektebene,881
 - PUBLISH-Privileg,894
 - REMOTE-Privileg,900
 - Erweiterte Prozeduren
 - Info,1162
 - Erweiterte Systemprozeduren
 - Info,1162
 - ESCAPE CHARACTER-Klausel
 - INPUT-Anweisung,912
 - LOAD TABLE-Anweisung,937
 - OUTPUT-Anweisung,969
 - UNLOAD-Anweisung,1100
 - ESCAPE-Klausel
 - LIKE-Suchbedingung,51
 - ESCAPES-Klausel
 - INPUT-Anweisung,912
 - LOAD TABLE-Anweisung,938
 - OUTPUT-Anweisung,970
 - UNLOAD-Anweisung,1100
 - Escapesequenzen
 - Apostrophe in SQL-Zeichenfolgen,8
 - Backslashes in SQL-Zeichenfolgen,8
 - hexadezimale Werte in SQL-Zeichenfolgen,8
 - Unicode,421
 - Unicode-Werte in SQL-Zeichenfolgen,8
 - Zeilenumbruchszeichen in SQL-Zeichenfolgen,8
 - Escapezeichen
 - binäre Literale,6
 - Datenbanken erstellen,583
 - Info ,8
 - INPUT-Anweisung,910
 - OUTPUT-Anweisung,968
 - ESQL
 - Anweisungsindikatoren,448
 - ESTIMATE-Funktion
 - Syntax,252

- ESTIMATE_SOURCE-Funktion
 - Syntax,253
- EvenOdd-Format
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung,725
- Event-Handler
 - verbergen mit der ALTER EVENT-Anweisung,461
- EVENT_CONDITION-Funktion
 - Syntax,254
- EVENT_CONDITION_NAME-Funktion
 - Syntax,256
- EVENT_PARAMETER-Funktion
 - Syntax,256
- EVERY-Klausel
 - CREATE EVENT-Anweisung,610
- EXCEPT-Anweisung
 - Syntax,841
- EXCEPT-Klausel
 - Datenbankoptionen einstellen,842
- EXCEPTION-Klausel
 - BEGIN-Anweisung,557
- EXEC-Anweisung
 - Transact-SQL-Syntax,848
- EXECUTE
 - EXECUTE für eine Prozedur mit der REVOKE-Anweisung entziehen,1010
- EXECUTE IMMEDIATE-Anweisung
 - Syntax,843
- EXECUTE-Anweisung
 - Embedded SQL-Syntax,846
 - Transact-SQL-Syntax,848
- EXISTS-Suchbedingung
 - Suchbedingungen,42
 - Syntax,66
- EXIT-Anweisung
 - Interactive SQL-Syntax,850
- Exklusiv-OR
 - Bit-Operatoren,20
- EXP-Funktion
 - Syntax,259
- EXPERIENCE_ESTIMATE-Funktion
 - Syntax,259
- Expires
 - HTTP_RESPONSE_HEADER-Funktion,286
- EXPLAIN-Anweisung
 - Embedded SQL-Syntax,851
- EXPLANATION, Funktion
 - Syntax,260
- Explizite Selektivitätsschätzungen
 - Info,68
- Exponentialfunktion
 - EXP-Funktion,259
- Exportieren
 - BLOBs,1435
 - Daten mit UNLOAD-Anweisung entladen,1098
- EXPRTYPE-Funktion
 - Syntax,262
- EXTERNAL NAME-Klausel
 - ALTER TEXT CONFIGURATION-Anweisung,532
 - CREATE FUNCTION-Anweisung [externe Prozeduren],620
 - CREATE PROCEDURE-Anweisung [externe Prozeduren],665
- Externe Begriffsegmentierer
 - ALTER TEXT CONFIGURATION-Anweisung,533
- Externe Funktionen
 - eine Schnittstelle zu nativen Funktionen mit der CREATE FUNCTION-Anweisung,617
 - Schnittstelle zu nativen Prozeduren mit CREATE PROCEDURE-Anweisung,661
- Externe Logins
 - für Fremdserver löschen,808
 - zuweisen für Fremdserver,616
- Externe Logins erstellen
 - CREATE EXTERNLOGIN-Anweisung,616
- Externe Objekte
 - mit der INSTALL EXTERNAL OBJECT-Anweisung erstellen,923
 - mit REMOVE EXTERNAL OBJECT-Anweisung entfernen,996
 - SYSEXTERNENVOBJECT-Systemansicht,1450
- Externe Prozeduren
 - Schnittstelle zu nativen Prozeduren mit CREATE PROCEDURE-Anweisung,661
- Externe Umgebungen
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - starten mit der START EXTERNAL ENVIRONMENT-Anweisung,1066
 - stoppen mit der STOP EXTERNAL ENVIRONMENT-Anweisung,1075

F

Fähigkeiten

- Fremdserver, 1439
- SYSCAPABILITY-Systemansicht, 1439

FALSE, Bedingungen

- Drei-Werte-Logik, 67
- IS FALSE-Suchbedingungen, 67

FASTFIRSTROW, Tabellen-Hint

- FROM-Klausel, 871

Fehler

- anzeigen, 1061
- benutzerdefinierte Meldungen, 1510
- Ereignisse mit der CREATE EVENT-Anweisung erstellen, 606
- in Embedded SQL verfolgen, 1122
- RAISERROR-Anweisung, 982

Fehler anzeigen

- RAISERROR-Anweisung, 982

Fehlerbehandlung

- Protokolliervorgänge, 1306
- Sperren, 1250
- ungewöhnliche Laufwerke, 454

Fehlermeldungen

- ERRORMSG-Funktion, 251

Fehlersuche

- TRACEBACK, 412
- Verhalten steuern, MESSAGE-Anweisung, 959

Fenster (OLAP)

- WINDOW-Klausel, 1124

Fensterfunktionen

- AVG-Funktion, 175
- COUNT-Funktion, 205
- COUNT_BIG-Funktion, 206
- COVAR_POP-Funktion, 209
- CUME_DIST-Funktion, 213
- DENSE_RANK-Funktion, 237
- MAX-Funktion, 310
- MEDIAN-Funktion, 311
- MIN-Funktion, 313
- PERCENT_RANK-Funktion, 335
- RANK-Funktion, 346
- REGR_AVGX-Funktion, 351
- REGR_AVGY-Funktion, 352
- REGR_COUNT-Funktion, 354
- REGR_INTERCEPT-Funktion, 355
- REGR_R2-Funktion, 357
- REGR_SLOPE-Funktion, 358

REGR_SXX-Funktion, 360

REGR_SXY-Funktion, 361

ROW_NUMBER-Funktion, 373

STDDEV-Funktion, 393

STDDEV_POP-Funktion, 393

STDDEV_SAMP-Funktion, 395

SUM-Funktion, 403

VAR_POP-Funktion, 425

VAR_SAMP-Funktion, 427

Fenstername

- allgemeines Element der SQL-Syntax, 446

Fensterspezifikation

- Syntax in Fensterfunktionen, 1124

Ferndatenzugriff

- FORWARD TO-Anweisung, 861
- Verbindung trennen, 493

Festgeschriebene Lesevorgänge

- FROM-Klausel, 869

Festschreiben

- Transaktionen mit der COMMIT-Anweisung, 575

FETCH-Anweisung

- Embedded SQL-Syntax, 853
- Syntax, 853

File-Eigenschaft

- DB_EXTENDED_PROPERTY-Funktion, 226

FILE-Klausel

- DROP REMOTE MESSAGE TYPE-Anweisung, 818
- GRANT CONSOLIDATE-Anweisung [SQL Remote], 889

FILE-Nachrichtentyp

- SQL Remote CREATE REMOTE MESSAGE TYPE-Anweisung, 694
- SQL Remote, ALTER REMOTE MESSAGE TYPE-Anweisung, 487

FileSize-Eigenschaft

- DB_EXTENDED_PROPERTY-Funktion, 226

filler() Spaltenname

- FROM-Klausel, 867

filler()-Spaltenname

- LOAD TABLE-Anweisung, 931

FIRST-Klausel

- DELETE-Anweisung, 791
- FETCH-Anweisung, 854
- SELECT-Anweisung, 1022
- UPDATE-Anweisung, 1111

FIRST_VALUE-Funktion

- Syntax, 263

- FLOAT-Datentyp
 - mit Datums- und Uhrzeitangaben vergleichen, 143
 - Syntax, 108
- FLOOR-Funktion
 - Syntax, 265
- FOLLOWING-Klausel
 - WINDOW-Klausel, 1126
- FOR DELETE-Klausel
 - CREATE TRIGGER-Anweisung [Transact-SQL], 769
- FOR DOWNLOAD ONLY-Klausel
 - CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote], 690
- FOR EACH-Klausel
 - CREATE TRIGGER-Anweisung, 764
- FOR INSERT-Klausel
 - CREATE TRIGGER-Anweisung [Transact-SQL], 769
- FOR JSON-Klausel
 - SELECT-Anweisung, 1027
- FOR OLAP WORKLOAD-Klausel
 - CREATE INDEX-Anweisung, 640
 - CREATE TABLE-Anweisung, 748
- FOR OLAP WORKLOAD-Option
 - ALTER TABLE-Anweisung, 516
- FOR READ ONLY-Klausel
 - DECLARE CURSOR-Anweisung [T-SQL], 782
 - FOR-Anweisung, 858
 - PREPARE-Anweisung [ESQL], 977
 - SELECT-Anweisung, 1025
 - START DATABASE-Anweisung, 1063
- FOR TABLES-Klausel
 - START SYNCHRONIZATION SCHEMA CHANGE-Anweisung, 1072
- FOR UPDATE BY LOCK-Klausel
 - SELECT-Anweisung, 1026
- FOR UPDATE BY TIMESTAMP-Klausel
 - SELECT-Anweisung, 1026
- FOR UPDATE BY VALUES-Klausel
 - SELECT-Anweisung, 1026
- FOR UPDATE BY-Klausel
 - SELECT-Anweisung, 1026
- FOR UPDATE-Klausel
 - CREATE TRIGGER-Anweisung [Transact-SQL], 769
 - DECLARE CURSOR-Anweisung [T-SQL], 782
 - FETCH-Anweisung, 855
 - FOR-Anweisung, 858
 - PREPARE-Anweisung [ESQL], 977
 - SELECT-Anweisung, 1025, 1027
- FOR UPLOAD-Klausel
 - ALTER PUBLICATION-Anweisung, 485
 - CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote], 690
- FOR XML-Klausel
 - INTERSECT-Anweisung, 927
 - SELECT-Anweisung, 841, 1027
- FOR-Anweisung
 - Syntax, 857
- FOR-Klausel
 - ALTER EVENT-Anweisung, 462
 - ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink] , 513
 - CREATE EVENT-Anweisung, 611
 - CREATE SUBSCRIPTION-Anweisung [SQL Remote], 730
 - CREATE SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink], 733
 - CREATE TRIGGER-Anweisung [Transact-SQL], 769
 - DECLARE CURSOR-Anweisung [T-SQL], 782
 - DESCRIBE-Anweisung [ESQL], 794
 - DROP SUBSCRIPTION-Anweisung [SQL Remote], 828
 - DROP SYNCHRONIZATION SUBSCRIPTION-Anweisung, 830
 - MESSAGE-Anweisung, 960
 - SELECT-Anweisung, 1025
 - START SUBSCRIPTION-Anweisung (SQL Remote), 1069
 - STOP SUBSCRIPTION-Anweisung (SQL Remote), 1078
 - SYNCHRONIZE SUBSCRIPTION-Anweisung (SQL Remote), 1086
- FORCE BUILD-Klausel
 - REFRESH MATERIALIZED VIEW-Anweisung, 988
 - REFRESH TEXT INDEX-Anweisung, 991
- FORCE INCREMENTAL-Klausel
 - REFRESH TEXT INDEX-Anweisung, 991
- FORCE INDEX
 - Index-Hints, 871
- FORCE NO OPTIMIZATION, Option
 - INSERT-Anweisung, 920
- FORCE NO OPTIMIZATION-Klausel
 - DELETE-Anweisung, 793

SELECT-Anweisung,1028
 UPDATE-Anweisung,1112
 FORCE OPTIMIZATION, Option
 INSERT-Anweisung,920
 UPDATE-Anweisung,1112
 FORCE OPTIMIZATION-Klausel
 DELETE-Anweisung,793
 EXCEPT-Anweisung,842
 INTERSECT-Anweisung,927
 MERGE-Anweisung,957
 SELECT-Anweisung,1027
 UNION-Anweisung,1097
 FORCE PASSWORD CHANGE-Klausel
 ALTER USER-Anweisung,541
 CREATE USER-Anweisung,770
 FORCE START-Klausel
 ALTER DATABASE-Anweisung,455
 FORCE-Klausel
 REFRESH TEXT INDEX-Anweisung,991
 FOREIGN KEY-Klausel
 ALTER INDEX-Anweisung,466
 CREATE TABLE-Anweisung,737
 REORGANIZE TABLE-Anweisung,998
 VALIDATE-Anweisung,1118
 FORMAT BCP-Klausel
 LOAD TABLE-Anweisung,938
 FORMAT FIXED-Klausel
 INPUT-Anweisung,913
 FORMAT SHAPEFILE-Klausel
 INPUT-Anweisung,913
 LOAD TABLE-Anweisung,938
 FORMAT TEXT-Klausel
 INPUT-Anweisung,913
 LOAD TABLE-Anweisung,938
 FORMAT XML-Klausel
 LOAD TABLE-Anweisung,939
 FORMAT-Klausel
 ALTER SERVICE-Anweisung [SOAP über
 HTTP],500
 CREATE SERVICE-Anweisung [SOAP-
 Webdienst],502,715
 INPUT-Anweisung,912
 LOAD TABLE-Anweisung,938
 OUTPUT-Anweisung,970
 UNLOAD-Anweisung,1100
 Formdateien
 mit der LOAD TABLE-Anweisung laden,938
 mit der st_geometry_load_shapefile-
 Systemprozedur laden,1416
 mit INPUT-Anweisung laden,910
 FORWARD TO-Anweisung
 Syntax,861
 Fragmentierung
 mit REORGANIZE TABLE defragmentieren,998
 sa_index_density-Systemprozedur,1240
 sa_table_fragmentation ,1338
 FREE PAGE ELIMINATION-Klausel
 BACKUP-Anweisung,553
 FreePages-Eigenschaft
 DB_EXTENDED_PROPERTY-Funktion,226
 Freigeben
 Savepoints,994
 Fremdschlüssel
 ALTER INDEX-Anweisung,466
 Integritätsregeln in der CREATE TABLE-
 Anweisung,745
 Kommentare mit der COMMENT-Anweisung
 hinzufügen,573
 konsolidierte Ansichten,1522
 mit der ALTER INDEX-Anweisung
 gruppieren,466
 mit der ALTER INDEX-Anweisung
 umbenennen,466
 sp_remote_exported_keys-Systemprozedur,1388
 sp_remote_imported_keys-Systemprozedur,1390
 Systemansichten,1452
 unbenannt, in der CREATE TABLE-
 Anweisung,745
 Fremdserver
 Attribute mit der ALTER SERVER-Anweisung
 ändern,491
 CREATE SERVER-Anweisung,701
 CREATE TABLE-Anweisung,737
 Fähigkeiten mit sp_servercaps bestimmen,1398
 Fähigkeiten, SYSCAPABILITY-
 Systemansicht,1439
 Logins löschen,808
 Logins zuweisen,616
 mit der DROP SERVER-Anweisung löschen,823
 SQL-Anweisungen senden,861
 SYSCAPABILITYNAME-Systemansicht,1439
 Verbindung trennen,493
 Fremdtabellen
 Systemansichten,1452
 FROM FILE-Klausel

- INSTALL EXTERNAL OBJECT-Anweisung,923
- INSTALL JAVA- Anweisung,925
- FROM SERVER-Klausel
 - ALTER MIRROR SERVER-Anweisung,481
 - CREATE MIRROR SERVER-Anweisung,658
- FROM VALUE-Klausel
 - INSTALL EXTERNAL OBJECT-Anweisung,923
- FROM, Klausel
 - , Syntax,863
- FROM-Klausel
 - aus DML auswählen,868
 - aus gespeicherten Prozeduren auswählen,865
 - CREATE CERTIFICATE-Anweisung,582
 - CREATE DECRYPTED DATABASE-Anweisung,595
 - CREATE DECRYPTED FILE-Anweisung,596
 - CREATE ENCRYPTED DATABASE-Anweisung,601
 - CREATE ENCRYPTED FILE-Anweisung,604
 - CREATE ENCRYPTED TABLE DATABASE-Anweisung,601
 - CREATE TEXT CONFIGURATION-Anweisung,758
 - DELETE-Anweisung,792
 - DELETE-Anweisung (positionsbasiert) [ESQL] [SP],789
 - INSTALL EXTERNAL OBJECT-Anweisung,923
 - INSTALL JAVA- Anweisung,926
 - LOAD TABLE-Anweisung,933
 - PUT-Anweisung [ESQL],981
 - RESTORE DATABASE-Anweisung,1001
 - REVOKE ROLE
 - SYS_REPLICATION_ADMIN_ROLE-Anweisung [MobiLink] [SQL Remote],1013
 - REVOKE ROLE
 - SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote],1014
 - SELECT-Anweisung,1024
 - UPDATE-Anweisung,1111
 - UPDATE-Anweisung (positionsbasiert),1103
- FTP-Klausel
 - GRANT CONSOLIDATE-Anweisung [SQL Remote],889
- FTP-Nachrichtentyp
 - SQL Remote CREATE REMOTE MESSAGE TYPE-Anweisung,694
 - SQL Remote, ALTER REMOTE MESSAGE TYPE-Anweisung,487
- Funktionen
 - Aggregat,153
 - alphabetische Liste,165
 - ALTER FUNCTION-Anweisung,465
 - Aufrufe nativer Funktionen,620
 - benannte Parametern,93
 - benutzerdefinierte,158
 - benutzerdefinierte beenden,1004
 - Bilddaten, SQL,166
 - Bit-Array,155
 - Datentypkonvertierung, SQL,156
 - Datum und Zeit,156
 - eine Schnittstelle zu nativen Funktionen mit der CREATE FUNCTION-Anweisung,617
 - Einführung,153
 - Funktionstypen,153
 - gespeicherte SQL-Funktionen erstellen,633
 - HTTP,162
 - im Vergleich zu Prozeduren,159
 - Indizes,640
 - Java,158
 - mit der CREATE FUNCTION-Anweisung erstellen [Webdienst],624
 - mit der DROP FUNCTION-Anweisung löschen,809
 - numerische,161
 - Rangfolge,155
 - Schnittstelle für externe Funktion ersetzen,619
 - Schnittstelle zu nativen Prozeduren mit CREATE PROCEDURE-Anweisung,661
 - SOAP,162
 - sperren,948
 - System,165
 - temporäre,634
 - Text, SQL,166
 - verschiedene,160
 - Werte von benutzerdefinierten zurückgeben,1004
 - Zeichenfolge,163
- Funktionen, Aggregat
 - AVG,175
 - BIT_AND,179
 - BIT_OR,180
 - BIT_XOR,182
 - COUNT,205
 - COUNT_BIG,206
 - FIRST_VALUE,263
 - GROUPING,271
 - Info,153

LAST_VALUE,297	MINUTE,315
LIST,302	MINUTES,315
MAX,310	MONTH,318
MEDIAN,311	MONTHNAME,318
MIN,313	MONTHS,319
SET_BITS,381	NOW,330
STDDEV,393	QUARTER,343
STDDEV_POP,393	SECOND,377
STDDEV_SAMP,395	SECONDS,378
SUM,403	SWITCHOFFSET,406
VAR_POP,425	SYSDATETIMEOFFSET,406
VAR_SAMP,427	TODATETIMEOFFSET,411
VARIANCE,429	TODAY,411
Funktionen, Bit	WEEKS,430
GET_BIT,266	YEAR,440
Funktionen, Bit-Array	YEARS,441
alphabetische Liste,155	YMD,442
BIT_LENGTH,180	Funktionen, HTTP
BIT_SUBSTR,181	HTTP_HEADER,283
COUNT_SET_BITS,208	HTTP_RESPONSE_HEADER,285
Info,155	HTTP_VARIABLE,287
SET_BIT,379	NEXT_HTTP_HEADER,325
Funktionen, Datentypkonvertierung	NEXT_HTTP_RESPONSE_HEADER,326
BINTOHEX,178	NEXT_HTTP_VARIABLE,327
CAST,186	Funktionen, Java und SQL, benutzerdefiniert
CONVERT,200	Info,158
HEXTOBIN,274	Funktionen, numerisch
HEXTOINT,275	ABS,166
Info,156	ACOS,167
INTTOHEX,292	ASIN,173
ISDATE,293	ATAN,173
ISNULL,295	ATAN2,174
Funktionen, Datum und Uhrzeit	ATN2,174
DATE,216	CEILING,187
DATEADD,217	COS,204
DATEDIFF,218	COT,204
DATEFORMAT,220	DEGREES,237
DATENAME,221	EXP,259
DATEPART,221	FLOOR,265
DATETIME,222	Info,161
DAY,223	LOG,307
DAYNAME,224	LOG10,307
DAYS,224	MOD,317
DOW,240	PI,337
GETDATE,268	POWER,338
HOUR,276	RADIANS,344
HOURS,276	RAND,345
Info,156	REMAINDER,364

- ROUND,371
- SIGN,381
- SIN,383
- SQRT,392
- TAN,407
- TRUNCATE,417
- TRUNCNUM,417
- Funktionen, Rangfolge
 - Info,155
- Funktionen, SOAP
 - NEXT_SOAP_HEADER,328
 - SOAP_HEADER,384
- Funktionen, System
 - CONNECTION_PROPERTY,199
 - DATALength,214
 - DB_EXTENDED_PROPERTY,226
 - DB_ID,230
 - DB_NAME,231
 - DB_PROPERTY,232
 - EVENT_CONDITION,254
 - EVENT_CONDITION_NAME,256
 - EVENT_PARAMETER,256
 - ISENCRYPTED,294
 - NEXT_CONNECTION,322
 - NEXT_DATABASE,324
 - PROPERTY,339
 - PROPERTY_DESCRIPTION,341
 - PROPERTY_NAME,342
 - PROPERTY_NUMBER,342
 - SUSER_ID,404
 - SUSER_NAME,405
 - TSEQUAL,418
 - USER_ID,423
 - USER_NAME,423
- Funktionen, Text und Bild
 - Info,166
 - TEXTPTR,408
- Funktionen, Verschiedene
 - Info,160
- Funktionen, verschiedene
 - ARGN,168
 - COALESCE,191
 - CONFLICT,194
 - ERRORMSG,251
 - ESTIMATE,252
 - ESTIMATE_SOURCE,253
 - EXPERIENCE_ESTIMATE,259
 - EXPLANATION,260
 - GET_IDENTITY,267
 - GRAPHICAL_PLAN,269
 - GREATER,271
 - IDENTITY,289
 - IFNULL,290
 - INDEX_ESTIMATE,290
 - ISNUMERIC,296
 - LESSER,302
 - NEWID,321
 - NULLIF,331
 - NUMBER,331
 - PLAN,337
 - REWRITE,368
 - SQLDIALECT,390
 - TRACEBACK,412
 - TRACED_PLAN,413
 - TRANSACTSQL,413
 - VAREXISTS,429
 - WATCOMSQL,430
- Funktionen, Zeichenfolge
 - ASCII,172
 - BYTE_LENGTH,183
 - BYTE_SUBSTR,184
 - CHAR,188
 - CHAR_LENGTH,189
 - CHARINDEX,189
 - COMPARE,192
 - COMPRESS-Funktion,193
 - CONNECTION_EXTENDED_PROPERTY,197
 - CSCONVERT,211
 - DECOMPRESS-Funktion,233
 - DECRYPT-Funktion,234
 - DIFFERENCE,239
 - ENCRYPT-Funktion,241
 - HASH-Funktion,272
 - Info,163
 - INSERTSTR,291
 - LCASE,299
 - LEFT,300
 - LENGTH,301
 - LOCATE,305
 - LOWER,308
 - LTRIM,309
 - NCHAR,321
 - PATINDEX,333
 - READ_CLIENT_FILE,348
 - REPEAT,365
 - REPLACE,365

REPLICATE,367
 REVERSE,367
 RIGHT,370
 RTRIM,376
 SIMILAR,382
 SORTKEY,385
 SOUNDEX,388
 SPACE,389
 STR,397
 STRING,398
 STRTOUUID,399
 STUFF,400
 SUBSTRING,401
 TO_CHAR,409
 TO_NCHAR,410
 TRIM,415
 UCASE,419
 UNICODE,420
 UNISTR,421
 UPPER,422
 UUIDTOSTR,424
 WRITE_CLIENT_FILE,432
 XMLAGG,433
 XMLCONCAT,434
 XMLELEMENT,435
 XMLFOREST,438
 XMLGEN,439

G

Ganzzahlen
 eine Tabelle generieren ,1301
 Gemischtes Speicherformat
 ALTER SPATIAL REFERENCE SYSTEM-
 Anweisung,508
 CREATE SPATIAL REFERENCE SYSTEM-
 Anweisung,725
 Geometrien
 Fehlerbehandlung mit st_geometry_dump,1411
 Geplante Ereignisse
 Trigger-auslösend,1088
 WAITFOR-Anweisung,1120
 Gesicherte Funktionen
 mit sa_server_option ändern,1306
 Schlüssel ändern,1354
 Schlüssel erstellen,1360
 Schlüssel löschen,1366
 Gespeicherte Funktionen

 Aufrufe nativer Funktionen,620
 Gespeicherte Prozeduren
 Aufrufe nativer Funktionen,665
 auswählen aus,865
 erstellen,681
 in Dynamic SQL ausführen,843
 in Transact-SQL ausführen,848
 in Transact-SQL erstellen,679
 INPUT-Anweisung nicht verwendbar,916
 Systemprozeduren,1161
 T-SQL konvertieren,430
 Gesperrte Konten
 Ursache ermitteln,1233
 GET DATA-Anweisung
 Embedded SQL-Syntax,876
 GET DESCRIPTOR-Anweisung
 Embedded SQL-Syntax,878
 GET OPTION-Anweisung
 Embedded SQL-Syntax,880
 GET_BIT-Funktion
 Syntax,266
 GET_IDENTITY-Funktion
 Syntax,267
 GETDATE-Funktion
 Abfrage des aktuellen Systemdatums,269
 Syntax,268
 Gleich
 Vergleichsoperator,9
 GLOBAL AUTOINCREMENT
 Ereignisse mit der CREATE EVENT-Anweisung
 erstellen,606
 GLOBAL AUTOINCREMENT-Klausel
 CREATE TABLE-Anweisung,737
 GLOBAL AUTOINCREMENT-Standardwert
 CREATE TABLE-Anweisung,520,743
 global_database_id-Option
 CREATE TABLE-Anweisung,520,743
 Globale Prüfsummen
 Datenbankeinstellungen ändern,453
 mit Datenbanken erstellen,586
 validieren,1118
 Globale temporäre Tabellen
 CREATE TABLE-Anweisung,737
 Globale Variablen
 @@identity,91
 alphabetische Liste,88
 Definition,85
 Trigger und @@identity,92

- Globally Unique Identifiers (global eindeutige Bezeichner)
 - SQL-Syntax für die NEWID-Funktion,321
 - GOTO-Anweisung
 - Transact-SQL-Syntax,881
 - GOTO-Klausel
 - WHENEVER-Anweisung [ESQL],1122
 - GRANT CONNECT-Anweisung
 - Kennwort, maximale Länge,888
 - Kennwörter, Zeichensatzkonvertierung,888
 - Syntax,888
 - Zeichensatzkonvertierung für Kennwörter,888
 - GRANT CONSOLIDATE-Anweisung
 - SQL Remote-Syntax,889
 - GRANT CREATE-Anweisung
 - Syntax,891
 - GRANT EXECUTE-Anweisung
 - Syntax,892
 - GRANT INTEGRATED LOGIN-Anweisung
 - Syntax,892
 - GRANT KERBEROS LOGIN-Anweisung
 - Syntax,893
 - GRANT PUBLISH-Anweisung
 - SQL Remote-Syntax,894
 - GRANT REMOTE-Anweisung
 - SQL Remote-Syntax,900
 - GRANT ROLE
 - SYS_REPLICATION_ADMIN_ROLE-Anweisung
 - MobiLink-Syntax,896
 - SQL Remote-Syntax,896
 - GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung
 - MobiLink-Syntax,898
 - SQL Remote-Syntax,898
 - GRANT SET USER-Anweisung
 - Syntax,881
 - GRANT USAGE ON SEQUENCE-Anweisung
 - Syntax,902
 - GRANT-Anweisung
 - Privilegien überprüfen,1441
 - Syntax,881
 - GRAPHICAL_PLAN, Funktion
 - Syntax,269
 - GREATER-Funktion
 - Syntax,271
 - Groß- und Kleinschreibung berücksichtigen
 - Vergleichsoperatoren,9
 - Großbuchstaben
 - UPPER-Funktion,422
 - Großbuchstaben-Zeichenfolgen
 - UCASE-Funktion,419
 - UPPER-Funktion,422
 - Große binäre Objekte
 - von Spalten abrufen,876
 - Große Datenbanken
 - Indexspeicherung,640
 - Größer als
 - Vergleichsoperator,9
 - Größer gleich
 - Vergleichsoperator,9
 - GROUP BY, Klausel
 - Syntax,903
 - GROUP BY-Klausel
 - CUBE-Vorgang,904
 - GROUPING SETS-Vorgang,903
 - ROLLUP-Vorgang,904
 - SELECT-Anweisung,1025
 - GROUP-Klausel
 - ALTER SERVICE-Anweisung [SOAP über HTTP],500
 - CREATE SERVICE-Anweisung [SOAP-Webdienst],501,714
 - GROUPING SETS, Vorgang
 - GROUP BY-Klausel,903
 - GROUPING-Funktion
 - UltraLite-Syntax,271
 - Gruppieren
 - Anweisungen in einer BEGIN-Anweisung,557
 - Anweisungen in einer BEGIN-Anweisung [TSQL],560
 - Anweisungen in einer TRY-Anweisung,1093
 - Gruppierung
 - für reguläre Ausdrücke,28
 - GROUP BY-Klausel,903
 - GUIDs
 - SQL-Syntax für die NEWID-Funktion,321
 - SQL-Syntax für die STRTOUUID-Funktion,399
 - SQL-Syntax für die UUIDTOSTR-Funktion,424
 - UNIQUEIDENTIFIER-Datentyp,134
 - gzip-Dienstprogramm
 - COMPRESS-Funktion,194
 - DECOMPRESS-Funktion,233
- ## H
- HANDLER-Klausel

-
- CREATE EVENT-Anweisung,611
 - HASH-Funktion
 - Syntax,272
 - unterstützte Algorithmen,272
 - Häufigkeit
 - Nachrichten versenden, GRANT
 - CONSOLIDATE-Anweisung [SQL Remote],889
 - Nachrichten versenden, GRANT REMOTE-Anweisung [SQL Remote] ,900
 - HAVING-Klausel
 - SELECT-Anweisung,1025
 - Suchbedingungen,42
 - HEADER-Klausel
 - CREATE FUNCTION-Anweisung [Webclients],626
 - CREATE PROCEDURE-Anweisung [Webclients],674
 - HELP-Anweisung
 - Interactive SQL-Syntax,906
 - Herunterfahren
 - Datenbanken,1074
 - HEXADECIMAL-Klausel
 - LOAD TABLE-Anweisung,940
 - OUTPUT-Anweisung,970
 - UNLOAD-Anweisung,1100
 - Hexadezimal
 - Konvertierung in und aus hexadezimalen Werten,6
 - Kovertierung mit CAST, CONVERT, HEXTOINT und INTTOHEX,6
 - Hexadezimale Escapesequenzen
 - in SQL-Zeichenfolgen,8
 - Hexadezimale Konstanten
 - als binär behandelt,6
 - Konvertierung in und aus hexadezimalen Werten,6
 - Hexadezimale Zeichenfolgen
 - Info,178,274,275
 - HEXTOBIN-Funktion
 - Syntax,274
 - HEXTOINT-Funktion
 - Syntax,275
 - Hints
 - Index-Hints mit der FROM-Klausel,871
 - Hinzufügen
 - Indizes mit der CREATE INDEX-Anweisung,638
 - Java-Klassen,925
 - LDAP-Serverkonfigurationsobjekte,642
 - Meldungen,655
 - Server,701
 - Spalten mit der ALTER TABLE-Anweisung,516
 - Zertifikate,582
 - Histogramme
 - abrufen,1225
 - mit CREATE STATISTICS aktualisieren,729
 - mit CREATE STATISTICS erstellen,729
 - nur teilweise aktualisiert durch LOAD TABLE,945
 - Selektivitätsschätzungen,68
 - SYSCOLSTAT-Systemansicht,1441
 - HISTORY-Klausel
 - ATTACH TRACING-Anweisung,546
 - BACKUP-Anweisung,551
 - RESTORE DATABASE-Anweisung,1002
 - Hohe Verfügbarkeit
 - ALTER MIRROR SERVER-Anweisung,480
 - CREATE MIRROR SERVER-Anweisung,656
 - HOLDLOCK, Tabellen-Hint
 - FROM-Klausel,869
 - HOLDLOCK-Klausel
 - READTEXT-Anweisung [T-SQL],986
 - Host
 - HTTP_HEADER-Funktion,284
 - Hostvar
 - allgemeines Element der SQL-Syntax,445
 - Hostvariablen
 - allgemeines Element der SQL-Syntax,445
 - in Embedded SQL deklarieren,778
 - HOURL-Funktion
 - Syntax,276
 - HOURS-Funktion
 - Syntax,276
 - HTML
 - CREATE SERVICE-Anweisung [HTTP-Webdienst],495,707
 - HTML_DECODE-Funktion
 - Syntax,278
 - HTML_ENCODE-Funktion
 - Syntax,279
 - HTTP
 - Header einrichten,1323
 - Optionen einrichten,1324
 - Optionen für SOAP-Header einstellen,1329
 - HTTP-Anforderungs-Header
 - Rückgabe von Namen und Werte,1235
 - HTTP-Anforderungsheader
 - @HttpMethod,284
 - @HttpRequestString,284
 - @HttpURI,284
-

- @HttpVersion,284
- Accept-Encoding,284
- Cookie,284
- Host,284
- Referer,284
- User-Agent,284
- HTTP-Antwortheader
 - @HttpStatus,286
 - Connection,286
 - Content-Length,286
 - Content-Type,286
 - Date,286
 - Expires,286
 - Location,286
 - sa_set_http_header_info-Systemprozedur,1323
 - Server,286
 - Transfer-Encoding,286
 - User-Agent,286
 - WWW-Authenticate,286
- HTTP-Dienste
 - erstellen mit der CREATE SERVICE-Anweisung [HTTP-Webdienst],706
- HTTP-Funktionen
 - alphabetische Liste,162
- HTTP-Header
 - HTTP_HEADER-Funktion,284
 - HTTP_RESPONSE_HEADER-Funktion,286
- HTTP-Nachrichtentyp
 - SQL Remote-Anweisung CREATE REMOTE MESSAGE TYPE,694
- HTTP-Systemprozeduren
 - alphabetische Liste,1162
- HTTP_BODY-Funktion
 - Syntax,280
- HTTP_DECODE-Funktion
 - Syntax,281
- HTTP_ENCODE-Funktion
 - Syntax,282
- HTTP_HEADER-Funktion
 - Syntax,283
- HTTP_RESPONSE_HEADER-Funktion
 - Syntax,285
- HTTP_VARIABLE-Funktion
 - Syntax,287
- HttpMethod
 - HTTP-HEADER-Funktion,284
- HttpQueryString
 - HTTP-HEADER-Funktion,284

- HttpStatus header
 - HTTP_RESPONSE_HEADER-Funktion,286
- HttpStatus-Header
 - sa_set_http_header_info-Systemprozedur,1323
- HttpURI
 - HTTP-HEADER-Funktion,284
- HttpVersion
 - HTTP-HEADER-Funktion,284

I

- I/O
 - das I/O-Kostenmodell neu kalibrieren,454
- IDENTIFIED BY ENCRYPTED-Klausel
 - ALTER LDAP SERVER-Anweisung,469
 - CREATE LDAP SERVER-Anweisung,643
- IDENTIFIED BY Kennwort-Klausel
 - ALTER USER-Anweisung,541
 - CONNECT-Anweisung,578
- IDENTIFIED BY-Klausel
 - ALTER LDAP SERVER-Anweisung,469
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung,507
 - CREATE EXTERNLOGIN-Anweisung,616
 - CREATE LDAP SERVER-Anweisung,643
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung,720
 - CREATE USER-Anweisung,770
- IDENTITY, Spalte
 - @@identity,91
- IDENTITY-Funktion
 - Syntax,289
- IDENTITY-Klausel
 - ALTER TABLE-Anweisung,523
 - CREATE TABLE-Anweisung,744
- IdleTime-Ereignisbedingung
 - Info,254
- IdleTimeout-Eigenschaft
 - mit sa_server_option einstellen,1306
- IF EXISTS-Klausel
 - DROP EVENT-Anweisung,808
 - DROP EXTERNLOGIN-Anweisung,809
 - DROP INDEX-Anweisung,810
 - DROP MATERIALIZED VIEW-Anweisung,813
 - DROP PROCEDURE-Anweisung,816
 - DROP PUBLICATION-Anweisung [MobiLink] [SQL Remote],817

-
- DROP SPATIAL REFERENCE SYSTEM-Anweisung,825
 - DROP SPATIAL UNIT OF MEASURE-Anweisung,825
 - DROP SYNCHRONIZATION PROFILE-Anweisung,829
 - DROP TABLE-Anweisung,832
 - DROP TRIGGER-Anweisung,837
 - DROP VARIABLE-Anweisung,839
 - DROP VIEW-Anweisung,840
 - IF NOT EXISTS-Klausel
 - CREATE INDEX-Anweisung,639
 - CREATE LOCAL TEMPORARY TABLE-Anweisung,646
 - CREATE PUBLICATION-Anweisung [MobiLink]
[SQL Remote],691
 - CREATE TABLE-Anweisung,740
 - CREATE TEXT INDEX-Anweisung,759
 - IF UPDATE-Klausel
 - CREATE TRIGGER-Anweisung [Transact-SQL],769
 - in Trigger,762
 - in Trigger in Transact-SQL,769
 - IF-Anweisung
 - Syntax,906
 - Transact-SQL-Syntax,908
 - IF-Ausdrücke
 - Suchbedingungen,42
 - Syntax,24
 - IFNULL-Funktion
 - Syntax,290
 - IMAGE-Datentyp
 - Syntax,134
 - IMMEDIATE REFRESH-Klausel
 - ALTER MATERIALIZED VIEW-Anweisung,477
 - CREATE MATERIALIZED VIEW-Anweisung,652
 - CREATE TEXT INDEX-Anweisung,759
 - Impersonierung
 - SETUSER-Anweisung,1059
 - IN EXCLUSIVE MODE-Klausel
 - LOCK TABLE-Anweisung,949
 - IN SHARE MODE-Klausel
 - LOCK TABLE-Anweisung,949
 - IN | ON-Klausel
 - CREATE INDEX-Anweisung,640
 - IN-Klausel
 - CREATE FUNCTION-Anweisung,633
 - CREATE FUNCTION-Anweisung
[Webdienst],624
 - CREATE MATERIALIZED VIEW-Anweisung,652
 - CREATE PROCEDURE-Anweisung,683
 - CREATE TABLE-Anweisung,739
 - CREATE TEXT INDEX-Anweisung,759
 - LOCK TABLE-Anweisung,949
 - IN-Suchbedingung
 - Syntax,58
 - Inaktiver Server
 - Ereignisse mit der CREATE EVENT-Anweisung
erstellen,606
 - INCLUDE-Anweisung
 - Embedded SQL-Syntax,909
 - INCREMENT BY-Klausel
 - ALTER SEQUENCE-Anweisung,490
 - CREATE SEQUENCE-Anweisung,699
 - INDEX FOR-Klausel
 - DESCRIBE-Anweisung,798
 - INDEX ONLY -Klausel
 - FROM-Klausel,871
 - Index-Hints
 - FROM-Klausel,871
 - INDEX-Klausel
 - ALTER INDEX-Anweisung,466
 - ALTER TABLE-Anweisung,523
 - CREATE TABLE-Anweisung,741
 - FROM-Klausel,871
 - REORGANIZE TABLE-Anweisung,998
 - VALIDATE-Anweisung,1118
 - INDEX_COL-Funktion
 - Syntax,165
 - INDEX_ESTIMATE-Funktion
 - Syntax,290
 - INDICATOR-Klausel
 - GET DESCRIPTOR-Anweisung [ESQL],878
 - SET DESCRIPTOR-Anweisung [ESQL],1033
 - Indikatoren
 - Kommentare,92
 - Indizes
 - ALTER INDEX-Anweisung,466
 - Ansichten,641
 - automatisch erstellte,641
 - benennen,641
 - CREATE INDEX-Anweisung,638
 - DROP INDEX-Anweisung,810
 - Eigentümer,641

- eindeutige,638
- Fremdschlüssel,641
- Funktionen,640
 - für OLAP-Arbeitslasten optimieren,640
- Indexfragmentierung mit sa_index_density erkennen,1240
- integrierte Funktionen,638
- Kommentare mit der COMMENT-Anweisung hinzufügen,573
- komprimieren,998
- mit der ALTER INDEX-Anweisung gruppieren,466
- mit der ALTER INDEX-Anweisung umbenennen,466
- physische Indizes in SYSPHYSIDX-Systemansicht aufgezeichnet,1468
- Primärschlüssel,641
- Stufen,1243
- Systemansichten,1456
- Tabellenverwendung,641
- VALIDATE-Anweisung,1118
 - verschobene Indizes mit sa_index_density erkennen,1240
- virtuelle,638
- Initialisieren
 - Datenbanken mit der CREATE DATABASE-Anweisung,583
- INLINE-Klausel
 - ALTER TABLE-Anweisung,522
 - CREATE TABLE-Anweisung,740
- INNER JOIN-Klausel
 - FROM-Klausel, SQL-Syntax,863
- INOUT-Klausel
 - CREATE PROCEDURE-Anweisung,683
- INPUT INTO-Anweisung
 - Interactive SQL-Syntax,910
- INPUT-Anweisung
 - Interactive SQL-Syntax,910
 - nicht in gespeicherten Prozeduren verwendbar,916
 - Syntax,910
- INPUT-Klausel
 - DESCRIBE-Anweisung,795
- INSENSITIVE-Klausel
 - DECLARE CURSOR-Anweisung,780
 - FOR-Anweisung,858
- INSERT-Anweisung
 - Ansichten aktualisieren,921
 - Datenbankoptionen einstellen,920
 - mehrzeilige Einfügungen,918
 - Syntax,917
- INSERT-Klausel
 - CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote],690
 - CREATE TRIGGER-Anweisung,762
- INSERT-Privileg
 - GRANT-Anweisung,883
 - REVOKE-Anweisung,1010
- INSERT-Privilegklausel
 - GRANT-Anweisung,885
- INSERTSTR-Funktion
 - Syntax,291
- INSTALL EXTERNAL OBJECT-Anweisung
 - Syntax,923
- INSTALL JAVA-Anweisung
 - Java-Klassen installieren,925
 - Syntax,925
- Installieren
 - Java-Klassen,925
- INSTEAD OF-Klausel
 - CREATE TRIGGER-Anweisung,762
- INSTEAD OF-Trigger
 - Ansichten ändern,544
 - CREATE TRIGGER-Anweisung,762
 - durch CREATE OR REPLACE VIEW gelöscht,773
 - Ersetzen von Ansichten,773
 - Erstellen von Ansichten,773
- INTEGER UNSIGNED-Datentyp
 - Syntax,109
- INTEGER-Datentyp
 - mit Datums- und Uhrzeitangaben vergleichen,143
 - Syntax,109
- Integrierte Logins
 - GRANT-Anweisung,893
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - REVOKE-Anweisung,1009
- Integrität
 - Integritätsregeln in der CREATE TABLE-Anweisung,745
- Integritätsregeln
 - Spalte, CREATE TABLE-Anweisung,745
 - umbenennen,527
- Interactive SQL
 - BYE-Anweisung, Syntax,850
 - CLEAR-Anweisung, Syntax,571

CONFIGURE-Anweisung, Syntax,577
 CONNECT-Anweisung, Syntax,578
 DESCRIBE CONNECTION-Anweisung, Syntax,798
 DESCRIBE-Anweisung, Syntax,798
 DISCONNECT-Anweisung, Syntax,802
 EXIT-Anweisung, Syntax,850
 HELP-Anweisung, Syntax,906
 INPUT-Anweisung, Syntax,910
 Kodierung für INPUT-Anweisung festlegen,912
 Kodierung für OUTPUT-Anweisung festlegen,969
 Kodierung für READ-Anweisung festlegen,984
 OUTPUT-Anweisung,968
 PARAMETERS-Anweisung, Syntax,974
 Prozedurprofilinformationen,1306
 QUIT-Anweisung, Syntax,850
 READ-Anweisung, Syntax,984
 RESUME-Anweisung nicht unterstützt,1004
 Rückgabecodes,850
 SET CONNECTION-Anweisung, Syntax,1032
 SET OPTION-Anweisung, Syntax,1043
 START DATABASE-Anweisung,1062
 START LOGGING-Anweisung, Syntax,1068
 START SERVER-Anweisung, Syntax,1066
 STOP LOGGING-Anweisung, Syntax,1077
 SYSTEM-Anweisung, Syntax,1088
 Internationale Sprachen und Zeichensätze
 Ersetzungszeichen,140
 Internes Speicherformat
 ALTER SPATIAL REFERENCE SYSTEM-Anweisung,508
 CREATE SPATIAL REFERENCE SYSTEM-Anweisung,725
 INTERSECT ALL-Klausel
 INTERSECT-Anweisung,927
 INTERSECT DISTINCT-Klausel
 INTERSECT-Anweisung,927
 INTERSECT-Anweisung
 Syntax,927
 INTERSECT-Klausel
 Datenbankoptionen einstellen,927
 INTERSECT-Anweisung,927
 Interval-Ereignisbedingung
 Info,254
 INTO CLIENT FILE-Klausel
 UNLOAD-Anweisung,1099
 INTO DESCRIPTOR-Klausel
 PUT-Anweisung [ESQL],981
 INTO FILE-Klausel
 UNLOAD-Anweisung,1099
 INTO LOCAL TEMPORARY TABLE-Klausel
 SELECT-Anweisung,1024
 INTO VARIABLE-Klausel
 UNLOAD-Anweisung,1099
 INTO-Klausel
 EXECUTE-Anweisung,846
 EXPLAIN-Anweisung [ESQL],851
 FETCH-Anweisung,853
 GET DATA-Anweisung ,876
 GET OPTION-Anweisung [ESQL],880
 INPUT-Anweisung,914
 MERGE-Anweisung,953
 PUT-Anweisung [ESQL],981
 SELECT-Anweisung,968,1023
 INTTOHEX-Funktion
 Syntax,292
 Inverse Abplattungmmm
 Definition,1485
 IOParallelism-Eigenschaft
 DB_EXTENDED_PROPERTY-Funktion,226
 IPAddressMonitorPeriod-Eigenschaft
 mit sa_server_option einstellen,1306
 IS
 Drei-Werte-Logik,67
 logische Operatoren, Beschreibung,10
 IS DISTINCT FROM-Suchbedingung
 Syntax,47
 IS FALSE-Suchbedingung
 Syntax,67
 IS NOT DISTINCT FROM-Suchbedingung
 Syntax,47
 IS NOT NULL-Suchbedingung
 Syntax,67
 IS NOT OF-Ausdrücke
 Suchbedingungen,42
 IS NULL-Suchbedingung
 Syntax,67
 IS OF-Ausdrücke
 Suchbedingungen,42
 IS OF-Ausdruckstyp
 Suchbedingungen,43
 IS TRUE-Suchbedingung
 Syntax,67
 IS UNKNOWN-Suchbedingung
 Syntax,67
 ISDATE-Funktion

- Syntax,293
- ISENCRYPTED-Funktion
 - Syntax,294
- ISNULL-Funktion
 - Syntax,295
- ISNUMERIC-Funktion
 - Syntax,296
- ISO 8601
 - Unterstützung,117
- ISOLATION LEVEL
 - Klausel,965
- ISOLATION LEVEL-Klausel
 - OPEN-Anweisung,965
- isolation_level, Option
 - für INSERT-Anweisungen setzen,920
 - für UPDATE-Anweisungen setzen,1112
 - in einer MERGE-Anweisung außer Kraft setzen,957
- isolation_level-Option
 - für DELETE-Anweisungen setzen,793
 - für die EXCEPT-Anweisung setzen,842
 - für INTERSECT-Anweisung setzen,927
 - für UNION-Anweisung setzen,1097
- Isolationsstufen
 - ISOLATION LEVEL-Klausel,965
 - Tabellen-Hints,869
- ISYSARTICLE
 - Systemtabelle,1130
- ISYSARTICLECOL
 - Systemtabelle,1130
- ISYSATTRIBUTE
 - Systemtabelle,1130
- ISYSATTRIBUTENAME
 - Systemtabelle,1130
- ISYSCAPABILITY
 - Systemtabelle,1130
- ISYSCERTIFICATE
 - Systemtabelle,1130
- ISYSCHECK
 - Systemtabelle,1130
- ISYSCOLPERM
 - Systemtabelle,1131
- ISYSCOLSTAT
 - Statistiken laden,930
 - Systemtabelle,1131
 - verschlüsselt, wenn die Datenbank verschlüsselt ist,1131
- verschlüsselt, wenn die Tabellenverschlüsselung aktiviert ist,1131
- ISYSCONSTRAINT
 - Systemtabelle,1131
- ISYSDBFILE
 - Systemtabelle,1131
- ISYSDBSPACE
 - Systemtabelle,1132
- ISYSDBSPACEPERM
 - Systemtabelle,1132
- ISYSDEPENDENCY
 - Systemtabelle,1131
- ISYSDOMAIN
 - Systemtabelle,1132
- ISYSEVENT
 - Systemtabelle,1132
- ISYSEXTERNENV
 - Systemtabelle,1132
- ISYSEXTERNENVOBJECT
 - Systemtabelle,1133
- ISYSEXTERNLOGIN
 - Systemtabelle,1133
 - verschlüsselt, wenn die Datenbank verschlüsselt ist,1133
 - verschlüsselt, wenn die Tabellenverschlüsselung aktiviert ist,1133
- ISYSFKEY
 - Systemtabelle,1133
- ISYSHISTORY
 - Systemtabelle,1133
- ISYSIDX
 - Systemtabelle,1133
- ISYSIDXCOL
 - Systemtabelle,1134
- ISYSJAR
 - Systemtabelle,1134
- ISYSJARCOMPONENT
 - Systemtabelle,1134
- ISYSJAVACLASS
 - Systemtabelle,1134
- ISYSLDAPSERVER
 - Systemtabelle,1134
- ISYSLOGINMAP
 - Systemtabelle,1134
- ISYSLOGINPOLICY
 - Systemtabelle,1135
- ISYSLOGINPOLICYOPTION
 - Systemtabelle,1135

ISYSMIRROROPTION
 Systemtabelle, 1135
ISYSMIRRORSERVER
 Systemtabelle, 1135
ISYSMIRRORSERVEROPTION
 Systemtabelle, 1135
ISYSMVOPTION
 Systemtabelle, 1136
ISYSMVOPTIONNAME
 Systemtabelle, 1136
ISYSOBJECT
 Systemtabelle, 1136
ISYSOPTION
 Systemtabelle, 1136
ISYSOPTSTAT
 Systemtabelle, 1136
ISYSPHYSIDX
 Systemtabelle, 1137
ISYSPROCEDURE
 Systemtabelle, 1137
ISYSPROCPARM
 Systemtabelle, 1137
ISYSPROCPerm
 Systemtabelle, 1137
ISYSPROXYTAB
 Systemtabelle, 1137
ISYSPUBLICATION
 Systemtabelle, 1137
ISYSREMARK
 Systemtabelle, 1138
ISYSREMOTEOPTION
 Systemtabelle, 1138
ISYSREMOTEOPTIONTYPE
 Systemtabelle, 1138
ISYSREMOType
 Systemtabelle, 1138
ISYSREMOTEUSER
 Systemtabelle, 1138
ISYSROLEGRANT
 Systemtabelle, 1139
ISYSROLEGRANTTEXT
 Systemtabelle, 1139
ISYSSCHEDULE
 Systemtabelle, 1139
ISYSSEQUENCE
 Systemtabelle, 1139
ISYSSEQUENCEPerm
 Systemtabelle, 1139

ISYSSERVER
 Fremdserver für Component Integration
 Services, 701
 Server hinzufügen, 701
 Systemtabelle, 1140
ISYSSOURCE
 Systemtabelle, 1140
ISYSSPATIALREFERENCESYSTEM
 Systemtabelle, 1140
ISYSSQLSERVERTYPE
 Systemtabelle, 1140
ISYSSUBSCRIPTION
 Systemtabelle, 1140
ISYSSYNC
 Systemtabelle, 1141
ISYSSYNCPROFILE
 Systemtabelle, 1141
ISYSSYNCSRIPT
 Systemtabelle, 1141
ISYSTAB
 Systemtabelle, 1141
ISYSTABCOL
 Systemtabelle, 1141
ISYSTABLEPerm
 Systemtabelle, 1142
ISYSTETextCONFIG
 Systemtabelle, 1141
ISYSTETextIDX
 Systemtabelle, 1142
ISYSTETextIDXTAB
 Systemtabelle, 1142
ISYSTRIGGER
 Systemtabelle, 1142
ISYSTYPEMAP
 Systemtabelle, 1142
ISYSUNITOFMEASURE
 Systemtabelle, 1143
ISYSUSER
 Systemtabelle, 1143
 verschlüsselt, wenn die Datenbank verschlüsselt
 ist, 1143
 verschlüsselt, wenn die Tabellenverschlüsselung
 aktiviert ist, 1143
ISYSUSERMESSAGE
 Systemtabelle, 1143
ISYSUSERTYPE
 Systemtabelle, 1143
ISYSVIEW

- Systemtabelle,1143
- ISYSWEBSERVICE
 - Hinzufügen HTTP-Dienste mit CREATE SERVICE-Anweisung [HTTP-Webdienst],706
 - HTTP-Webdienste ändern,494
 - SOAP über HTTP-Dienste ändern,500
 - SOAP über HTTP-Dienste hinzufügen,713
 - Systemtabelle,1144

J

- JAR, Dateien
 - entfernen,997
- JAR-Dateien
 - INSTALL JAVA- Anweisung,925
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
- JAR-Klausel
 - INSTALL JAVA- Anweisung,925
 - REMOVE JAVA-Anweisung,997
- Java
 - benutzerdefinierte Funktionen,158
 - installieren,925
 - konvertieren von Java und SQL,150
 - Systemtabellen,1159
- Java VM
 - STOP JAVA-Anweisung,1076
- Java-Datentypen
 - aus SQL konvertieren,151
 - in SQL konvertieren,150
- Java-Klassen
 - Fehlerbehandlung,1247
 - in Datenbank geladen,1247
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
- Java/SQL, Datentypen konvertieren
 - Info,150
- Java/SQL-Datentypen konvertieren
 - Info,150
- JavaScript Object Notation
 - CREATE SERVICE-Anweisung [HTTP-Webdienst],495,707
- jConnect
 - CREATE DATABASE-Anweisung,589
- JCONNECT-Klausel
 - ALTER DATABASE-Anweisung,451
 - CREATE DATABASE-Anweisung,589
- JDBC

- Datentypkonvertierung,150
- Java in SQL, Datentypkonvertierung,150
- SQL in Java, Datentypkonvertierung,151
- Upgrade von Datenbankkomponenten,451
- Join-Operatoren
 - Kompatibilität mit ASE,21
- Joins
 - ANSI-Äquivalenz,368
 - darauf basierende Aktualisierungen,1112
 - darauf basierende SQL Remote-Aktualisierungen,1107
 - FROM-Klausel, SQL-Syntax,863
 - Zeilen basierend auf Joins löschen,791
- JSON
 - CREATE SERVICE-Anweisung [HTTP-Webdienst],495,707

K

- Kalenderdatum
 - ISO 8601,117
- Kalibrieren
 - Datenbankserver mit der ALTER DATABASE-Anweisung,451
 - Kostenmodelle mit der ALTER DATABASE-Anweisung,451
 - Laden und Entladen von Kostenmodellen mit der sa_load_cost_model-Systemprozedur,1249
 - Laden und Entladen von Kostenmodellen mit der sa_transactions-Systemprozedur,1348
 - parallele I/O-Fähigkeiten,454
- Katalog
 - Daten in Tabellen anzeigen,1129
 - Standard-Systemansichten,1437
 - Systemtabellen,1129
- Katalogdaten
 - anzeigen,1129
- Katalogliste der Systemprozeduren
 - Info,1161
- Katalogprozeduren
 - alphabetische Liste ,1168
 - Transact-SQL,1167
- Katalogprozeduren (ASE)
 - Transact-SQL-Liste,1166
- Kein Plattenspeicher
 - Ereignisse mit der CREATE EVENT-Anweisung erstellen,606
- Kennwörter

-
- maximale Länge, ALTER USER-Anweisung,542
 - maximale Länge, CREATE USER-Anweisung,770
 - maximale Länge, GRANT CONNECT-Anweisung,888
 - sa_verify_password-Systemprozedur,1353
 - Zeichensatzkonvertierung, CREATE USER-Anweisung,770
 - Zeichensatzkonvertierung, GRANT CONNECT-Anweisung,888
 - Kennwörter für die Doppelkontrolle
 - sa_get_user_status-Systemprozedur,1233
 - Kennwörter, Zeichensatzkonvertierung
 - ALTER USER-Anweisung,542
 - Kerberos
 - KERBEROS LOGIN entziehen,1009
 - Kerberos-Login erteilen,893
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - Prinzipal, Groß- und Kleinschreibung berücksichtigen,894
 - Privileg erteilen,893
 - KERBEROS LOGIN
 - mit der REVOKE-Anweisung entziehen,1010
 - Kerberos-Logins
 - GRANT KERBEROS LOGIN-Anweisung,893
 - KEY JOIN-Klausel
 - FROM-Klausel, SQL-Syntax,863
 - KEY-Klausel
 - ALTER DATABASE-Anweisung,453
 - CREATE DECRYPTED DATABASE-Anweisung,595
 - CREATE DECRYPTED FILE-Anweisung,596
 - CREATE ENCRYPTED DATABASE-Anweisung,601
 - CREATE ENCRYPTED FILE-Anweisung,604
 - CREATE ENCRYPTED TABLE DATABASE-Anweisung,601
 - DROP DATABASE-Anweisung,804
 - RESTORE DATABASE-Anweisung,1002
 - START DATABASE-Anweisung,1063
 - Kind-Tests
 - vom OPENXML-Operator unterstützt,16
 - Klammern
 - Anführungszeichen,4
 - Datenbankobjekte,4
 - Klassen
 - Java entfernen,997
 - Java-Methoden,158
 - Klauseln
 - Reihenfolge, SQL-Konventionen,446
 - Klauselreihenfolge in SQL
 - Syntaxkonventionen,446
 - Kleinbuchstaben-Zeichenfolgen
 - LCASE-Funktion,299
 - LOWER-Funktion,308
 - Kleiner als
 - Vergleichsoperator,9
 - Kleiner gleich
 - Vergleichsoperator,9
 - Kodierung
 - INPUT-Anweisung,912
 - LOAD TABLE-Syntax,931
 - OUTPUT-Anweisung,969
 - READ-Anweisung,984
 - UNLOAD-Anweisung,1098
 - Kodierungen
 - CREATE DATABASE-Anweisung,587
 - Koeffizient der Bestimmtheit
 - Info,357
 - Kollationen
 - bei Datenbankerstellung einstellen,587
 - REGEXP-Suchbedingung,55
 - REGEXP_SUBSTR-Funktion,349
 - SIMILAR TO-Suchbedingung,57
 - SORTKEY-Funktion,385
 - Kollationsanpassung
 - COLLATION-Klausel, CREATE DATABASE-Anweisung,587
 - COMPARE-Funktion,192
 - NCHAR COLLATION-Klausel, CREATE DATABASE-Anweisung,589
 - SORTKEY-Funktion,385
 - Kollationssequenzen
 - CREATE DATABASE-Anweisung,587
 - LIKE-Suchbedingung,52
 - Kombinieren
 - Ergebnis von mehreren SELECT-Anweisungen,1096
 - Kommentare
 - einfügen,92
 - entfernen,92
 - mit der COMMENT-Anweisung zu Datenbankobjekten hinzufügen,573
 - Syntax,92
 - Kommunikationsprotokolle
 - mehrfache Einstellungen in MobiLink,736

- Kompatibilität
 - Ansichten,1534
 - Datum/Uhrzeit,145
 - NULL,78
 - T-SQL-Ausdrücke und die Option QUOTED IDENTIFIER,41
 - Transact-SQL, globale Variablen,88
 - Transact-SQL, lokale Variablen,86
 - Transact-SQL-Ansichten,1546
 - Transact-SQL-Ausdrücke,41
 - Transact-SQL-Vergleichsoperatoren,9
- Kompatibilität von Ausdrücken
 - Info,41
- Kompatibilitätsansichten
 - Info,1534
 - SYSCOLLATION,1535
 - SYSCOLLATIONMAPPINGS,1535
 - SYSCOLUMN,1536
 - SYSFKCOL,1537
 - SYSFOREIGNKEY,1537
 - SYSGROUP,1538
 - SYSGROUPS,1539
 - SYSINDEX,1539
 - SYSINFO,1540
 - SYSIXCOL,1540
 - SYSTABLE,1541
 - SYSUSERAUTH,1542
 - SYSUSERAUTHORITY,1543
 - SYSUSERLIST,1544
 - SYSUSERPERM,1544
 - SYSUSERPERMS,1545
- Kompatibilitätsrollen
 - mit ALTER ROLE migrieren,488
 - mit der GRANT-Anweisung erteilen,882
 - mit der REVOKE-Anweisung entziehen,1009
- Komprimieren
 - Tabellen mit der ALTER TABLE-Anweisung,516
- Komprimieren von Spalten
 - CREATE TABLE-Anweisung,740
- Komprimierte Spalten
 - ALTER TABLE-Anweisung,516
 - Komprimierungsstatistiken abrufen,1177
- Komprimierung
 - COMPRESS-Funktion,194
 - Statistiken,1181
- Konflikte
 - CONFLICT-Funktion für SQL Remote,194
- Konflikte lösen
 - CONFLICT-Funktion für SQL Remote,194
- Konsolidierte Ansichten
 - Info,1514
 - SYSARTICLECOLS,1518
 - SYSARTICLES,1519
 - SYSCAPABILITIES,1519
 - SYSCATALOG,1520
 - SYSCOLAUTH,1520
 - SYSCOLSTATS,1521
 - SYSCOLUMNS,1522
 - SYSFOREIGNKEYS,1522
 - SYSINDEXES,1523
 - SYSOPTIONS,1524
 - SYSPROCAUTH,1524
 - SYSPROCPARMS,1525
 - SYSPROCS,1525,1530
 - SYSPUBLICATIONS,1526
 - SYSREMOTEOPTION2,1526
 - SYSREMOTEOPTIONS,1527
 - SYSREMOTETYPES,1527
 - SYSREMOTEUSERS,1528
 - SYSSUBSCRIPTIONS,1528
 - SYSSYNC2,1529
 - SYSSYNCPUBLICATIONDEFAULTS,1529
 - SYSSYNCS,1530
 - SYSSYNCSUBSCRIPTIONS,1531
 - SYSSYNCSUSERS,1532
 - SYSTABAUTH,1532
 - SYSTRIGGERS,1533
 - SYSUSEROPTIONS,1534
 - SYSVIEWS,1534
- Konsolidierte Datenbanken
 - SQL Remote, Privilegien entziehen,1006
- Konsolidierte Privilegien entziehen
 - REVOKE-Anweisung,1009
- Konsolidierungsprivilegien entziehen
 - SQL Remote, CONSOLIDATE-Privilegien,1006
- Konstanten
 - Info,6
 - Syntax,24
 - Transact-SQL,41
- Konventionen
 - SQL-Sprachsyntax,1
 - Syntax,446
- Konvertieren
 - Bit-Arrays,148
 - Bits,148
 - Datentypen,146

-
- Datentypkonvertierungen,146
 - mit Vergleichsoperatoren,140
 - SQL und Java,150
 - Zeichenfolgen in Datumsangaben,145
 - zweideutige Datumsangaben und Zeichenfolgen,118
 - Konvertieren, NULL-Konstanten in NUMERIC- oder Zeichenfolgentypen
 - Info,148
 - Konvertierung
 - beim Auswerten von Ausdrücken,140
 - CAST,186
 - DOUBLE in NUMERIC konvertieren,149
 - NCHAR in CHAR,147
 - Konvertierung beim Einsatz von Vergleichsoperatoren
 - Info,140
 - Konvertierung zwischen Zeichensätzen
 - Info,141
 - Konvertierungen von Bit-Arrays
 - Info,148
 - Konvertierungsfunktionen
 - alphabetische Liste,156
 - Datentyp,156
 - Kopieknoten
 - Server löschen,815
 - Kopieserver
 - mit der ALTER MIRROR SERVER-Anweisung ändern,480
 - mit der CREATE MIRROR SERVER-Anweisung definieren,658
 - Korrelationsfunktion
 - CORR-Funktion,203
 - Korrelationsnamen
 - DELETE-Anweisung,792
 - UPDATE-Anweisung,1112
 - Kosinus-Funktion
 - COS-Funktion,204
 - Kostenbasierte Optimierung
 - bei Prozeduren erzwingen,1028
 - erzwingen mit FORCE OPTIMIZATION-Option,1027
 - Vermeiden mit FORCE NO OPTIMIZATION-Klausel,1028
 - Kostenmodelle
 - den Datenbankserver kalibrieren,451
 - entladen,1348
 - laden,1249
 - mit der ALTER DATABASE-Anweisung neu kalibrieren,451
 - Kotangens-Funktion
 - COT-Funktion,204
 - Kürzen
 - schlägt fehl, wenn die Tabelle durch eine Sofortansicht referenziert wird,1090
 - schlägt fehl, wenn die Tabelle einen sofortigen Textindex hat,1090
 - Tabellen,1089
 - Textindizes,1091
- ## L
- Label
 - Anweisungen,881
 - für Anweisungen,446
 - Laden
 - Daten aus einem angegebenen Wert laden,933
 - Daten aus einer Datei auf dem Clientsystem laden,933
 - Daten aus einer Datei auf dem Datenbankserversystem laden,933
 - Formdateien mit der st_geometry_load_shapefile-Systemprozedur,1416
 - Kostenmodelle,1249
 - LOAD TABLE-Anweisung,931
 - Masseneinfügungen,931
 - schlägt fehl, wenn die Tabelle einen sofortigen Textindex hat,944
 - schlägt fehl, wenn Tabelle von Sofortansicht referenziert wird,944
 - Lange Spaltennamen
 - abrufen,796
 - LANGUAGE C_ESQL32-Klausel
 - CREATE FUNCTION-Anweisung [externer Aufruf],620
 - CREATE PROCEDURE-Anweisung [externer Aufruf],666
 - LANGUAGE C_ESQL64-Klausel
 - CREATE FUNCTION-Anweisung [externer Aufruf],620
 - CREATE PROCEDURE-Anweisung [externer Aufruf],666
 - LANGUAGE C_ODBC32-Klausel
 - CREATE FUNCTION-Anweisung [externer Aufruf],620

- CREATE PROCEDURE-Anweisung [externer Aufruf],666
- LANGUAGE C_ODBC64-Klausel
 - CREATE FUNCTION-Anweisung [externer Aufruf],620
 - CREATE PROCEDURE-Anweisung [externer Aufruf],666
- LANGUAGE CLR, Klausel
 - CREATE PROCEDURE-Anweisung [externer Aufruf],666
- LANGUAGE CLR-Klausel
 - CREATE FUNCTION-Anweisung [externer Aufruf],621
- LANGUAGE JAVA-Klausel
 - CREATE FUNCTION-Anweisung [externer Aufruf],622,667
- LANGUAGE PERL, Klausel
 - CREATE PROCEDURE-Anweisung [externer Aufruf],666
- LANGUAGE PERL-Klausel
 - CREATE FUNCTION-Anweisung [externer Aufruf],621
- LANGUAGE PHP-Klausel
 - CREATE FUNCTION-Anweisung [externer Aufruf],621
 - CREATE PROCEDURE-Anweisung [externe Prozeduren],667
- LAST USER-Klausel
 - CREATE TABLE-Anweisung,737
- LAST USER-Spezialwert
 - Syntax,76
- LAST USER-Standardwert
 - CREATE TABLE-Anweisung,521,744
- LAST-Klausel
 - FETCH-Anweisung,854
- LAST_VALUE-Funktion
 - Syntax,297
- Lateral abgeleitete Tabellen
 - FROM-Klausel, äußere Referenzen,866
- LCASE-Funktion
 - Syntax,299
- LDAP SERVER
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
- LDAP-Benutzerauthentifizierung
 - LDAP-Serverkonfigurationsobjekte ändern,468
 - LDAP-Serverkonfigurationsobjekte erstellen,642
 - sa_get_ldapserver_status-Systemprozedur,1227
- LDAP-Server
 - LDAP-Serverkonfigurationsobjekte ändern,468
 - LDAP-Serverkonfigurationsobjekte erstellen,642
 - LDAP-Serverkonfigurationsobjekte löschen,811
 - LDAP-Serverkonfigurationsobjekte validieren,1116
 - Status abrufen,1227
- LDAP-Serverkonfigurationsobjekte
 - SYSLDAPSERVER-Systemansicht,1461
- ldap_auto_failback_period-Option
 - ALTER LOGIN POLICY-Anweisung,472
 - CREATE LOGIN POLICY-Anweisung,647
- ldap_failover_to_std-Option
 - ALTER LOGIN POLICY-Anweisung,472
 - CREATE LOGIN POLICY-Anweisung,647
- ldap_primary_server-Option
 - ALTER LOGIN POLICY-Anweisung,472
 - CREATE LOGIN POLICY-Anweisung,647
- ldap_refresh_dn-Option
 - ALTER LOGIN POLICY-Anweisung,472
 - CREATE LOGIN POLICY-Anweisung,647
- ldap_secondary_server-Option
 - ALTER LOGIN POLICY-Anweisung,472
 - CREATE LOGIN POLICY-Anweisung,647
- LEAVE-Anweisung
 - Syntax,929
- LEFT OUTER JOIN-Klausel
 - FROM-Klausel, SQL-Syntax,863
- LEFT-Funktion
 - Syntax,300
- Leistung
 - SQL Remote-Aktualisierungen,1108
- LEN-Funktion
 - Syntax,301
- LENGTH-Funktion
 - Syntax,301
- LENGTH-Klausel
 - GET DESCRIPTOR-Anweisung [ESQL],878
 - SET DESCRIPTOR-Anweisung [ESQL],1033
- Lesen
 - Text- und Bildwerte aus der Datenbank,986
- Lesen von Dateien
 - mit xp_read_file,1424
- LesenPS-Sperren
 - sa_locks-Systemprozedur,1251
- LESSER-Funktion
 - Syntax,302

-
- LIKE-, REGEXP- und SIMILAR TO-Suchbedingungen
 - Info,48
 - LIKE-Suchbedingung
 - Kollationen,52
 - Musterlänge,52
 - Syntax,51
 - und Berücksichtigung von Groß- und Kleinschreibung,52
 - verglichen mit REGEXP und SIMILAR TO,48
 - LIMIT-Klausel
 - ATTACH TRACING-Anweisung,547
 - SELECT-Anweisung,1022
 - LINEAR UNIT OF MEASURE-Klausel
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung,508
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung,721
 - Linieninterpretation, räumliche Daten
 - PLANAR-Typ,722
 - ROUND EARTH-Typ,722
 - LIST-Funktion
 - Syntax,302
 - Listen
 - LIST-Funktion, Syntax,302
 - sa_split_list-Systemprozedur,1332
 - speichern als Zeilen oder Arrays,136
 - Literale
 - Info,6
 - LivenessTimeout-Eigenschaft
 - mit sa_server_option einstellen,1306
 - LOAD STATISTICS-Anweisung
 - Syntax,930
 - LOAD TABLE-Anweisung
 - bei Datenbankspiegelung verwenden,944
 - Syntax,931
 - LOAD-Privileg
 - GRANT-Anweisung,883
 - REVOKE-Anweisung,1010
 - LOAD-Privilegklausel
 - GRANT-Anweisung,885
 - LOCAL DATABASE-Klausel
 - ATTACH TRACING-Anweisung,546
 - LOCATE-Funktion
 - Syntax,305
 - Location
 - HTTP_RESPONSE_HEADER-Funktion,286
 - LOCATION-Klausel
 - ALTER EXTERNAL ENVIRONMENT-Anweisung,464
 - LOCK FEATURE-Anweisung
 - Syntax,948
 - LOCK TABLE-Anweisung
 - Syntax,949
 - locked-Option
 - ALTER LOGIN POLICY-Anweisung,472
 - CREATE LOGIN POLICY-Anweisung,647
 - LOG-Funktion
 - Syntax,307
 - LOG10-Funktion
 - Syntax,307
 - Logdateien
 - Anforderungslog mit sa_get_request_profile analysieren,1228
 - Anforderungslog mit sa_get_request_times analysieren,1229
 - Plattenspeicher mit ALTER DBSPACE zuweisen,457
 - verfügbaren Platz bestimmen,1212
 - LogDiskSpace, Systemereignis
 - Beispiel,255
 - logfile-Option
 - ALTER MIRROR SERVER-Anweisung,481
 - CREATE MIRROR SERVER-Anweisung,659
 - LogFreePercent-Ereignisbedingung
 - Info,254
 - LogFreeSpace-Ereignisbedingung
 - Info,254
 - LOGIN POLICY-Klausel
 - ALTER USER-Anweisung,540
 - CREATE USER-Anweisung,770
 - Login-Richtlinien
 - ALTER LOGIN POLICY-Anweisung,471
 - ALTER USER-Anweisung,541
 - CREATE LOGIN POLICY-Anweisung,647
 - CREATE USER-Anweisung,770
 - DROP LOGIN POLICY-Anweisung,812
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - Richtlinienoptionen,647
 - Standard-Login-Richtlinie,647
 - Logins
 - für Fremdserver löschen,808
 - Status ermitteln,1233
 - Verbindungen zu einem Server deaktivieren,1306
 - zuweisen für Fremdserver,616
-

- Logische Operatoren
 - Drei-Werte-Logik,67
 - Syntax,10
- LogSize-Ereignisbedingung
 - Info,254
- Lokale temporäre Tabellen
 - erstellen,784
 - mit der CREATE LOCAL TEMPORARY TABLE-Anweisung erstellen,645
- Lokale Variablen
 - Definition,85
 - Syntax,86
- LONG BINARY-Datentyp
 - Syntax,134
- LONG NAMES-Klausel
 - DESCRIBE-Anweisung,796
- LONG NVARCHAR-Datentyp
 - Beschreibung,97
 - Syntax,97
- LONG VARBIT-Datentyp
 - Syntax,114
- LONG VARCHAR-Datentyp
 - Syntax,97
- LOOP-Anweisung
 - Syntax,950
- LOOP-Klausel
 - LOOP-Anweisung,950
- Löschen
 - alle Zeilen aus einer Tabelle,1089
 - Ansichten mit der DROP VIEW-Anweisung,840
 - Benutzer mit REVOKE-Anweisung,1009
 - Datenbankdateien mit der DROP DATABASE-Anweisung,804
 - DBSpaces mit der DROP DBSPACE-Anweisung,806
 - Domänen,807
 - Domänen mit der DROP DOMAIN-Anweisung,807
 - DROP PUBLICATION-Anweisung,817
 - DROP SUBSCRIPTION-Anweisung,828
 - DROP SYNCHRONIZATION SUBSCRIPTION-Anweisung,830
 - DROP SYNCHRONIZATION USER-Anweisung,831
 - entfernte Nachrichtentypen,818
 - Ereignisse mit der DROP EVENT-Anweisung,808
 - Fremdserver mit der DROP SERVER-Anweisung,823
 - Funktionen mit der DROP FUNCTION-Anweisung,809
 - Indizes mit der DROP INDEX-Anweisung,810
 - Interactive SQL-Bereich,571
 - Java-Klassen,997
 - LDAP-Serverkonfigurationsobjekte,811
 - Login-Richtlinien mit der DROP LOGIN POLICY-Anweisung,812
 - Login-Richtlinien mit der DROP USER-Anweisung,838
 - Logins für Fremdserver,808
 - materialisierte Ansichten mit der DROP MATERIALIZED VIEW-Anweisung,813
 - Nachrichten mit der DROP MESSAGE-Anweisung,814
 - Optimiererstatistiken mit der DROP STATISTICS-Anweisung,827
 - Privilegien erteilen,1009
 - Prozeduren mit der DROP PROCEDURE-Anweisung,816
 - räumliches Bezugssystem mit der DROP SPATIAL REFERENCE SYSTEM-Anweisung,825
 - Rollen mit der DROP ROLE-Anweisung,820
 - Spalten mit der ALTER TABLE-Anweisung,516
 - SQL-Variablen mit der DROP VARIABLE-Anweisung,839
 - START SYNCHRONIZATION DELETE-Anweisung,1071
 - STOP SYNCHRONIZATION DELETE-Anweisung,1079
 - Tabellen mit der DROP TABLE-Anweisung,832
 - Textindizes für die Volltextsuche,834
 - Textkonfigurationsobjekte,833
 - Trigger mit der DROP FUNCTION-Anweisung,809
 - Trigger mit der DROP-Anweisung,837
 - Verbindungen in Interactive SQL,802
 - Verbindungen mit der DROP CONNECTION-Anweisung,803
 - vorbereitete Anweisungen mit der DROP STATEMENT-Anweisung,826
 - Webdienste mit der DROP SERVICE-Anweisung,824
 - Zeilen aus Cursor,789
 - Zeilen aus Datenbanken,791
 - Zertifikate mit der DROP CERTIFICATE-Anweisung,803

-
- Löschen von Ansichten
 - DROP VIEW-Anweisung,840
 - Löschen von entfernten Prozeduren
 - DROP PROCEDURE-Anweisung,816
 - Löschen von Indizes
 - DROP-Anweisung,810
 - Löschen von Synchronisationsprofilen
 - DROP SYNCHRONIZATION PROFILE-Anweisung [MobiLink],829
 - Löschen von Tabellen
 - DROP TABLE-Anweisung,832
 - Löschen, materialisierte Ansichten
 - DROP MATERIALIZED VIEW-Anweisung,813
 - LOWER-Funktion
 - Syntax,308
 - LTRIM-Funktion
 - Syntax,309
 - M**
 - MANUAL REFRESH-Klausel
 - ALTER MATERIALIZED VIEW-Anweisung,478
 - CREATE MATERIALIZED VIEW-Anweisung,652
 - Manuelle Ansichten
 - ALTER MATERIALIZED VIEW-Anweisung,476
 - MAPI
 - E-Mail-Sitzungen starten,1431
 - E-Mail-Sitzungen stoppen,1434
 - erweiterte Systemprozeduren,1162
 - MAPI- und SMTP-Systemprozeduren,1426
 - Rückgabecodes für MAPI- und SMTP-Systemprozeduren,1163
 - MAPI- und SMTP-Systemprozeduren
 - Rückgabecodes,1163
 - Marken-Informationen
 - abrufen,1422
 - Maßeinheit
 - vordefinierte installieren,1245
 - Maßeinheiten
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung,506
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung,719
 - CREATE SPATIAL UNITS OF MEASURE-Anweisung,727
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - ST_UNITS_OF_MEASURE-Systemansicht,1518
 - Massenimport von Daten
 - LOAD TABLE-Anweisung,931
 - Massenvorgänge
 - Daten mit UNLOAD-Anweisung entladen,1098
 - MATCH FULL-Klausel
 - CREATE TABLE-Anweisung,747
 - MATCH SIMPLE-Klausel
 - CREATE TABLE-Anweisung,747
 - MATCH UNIQUE FULL-Klausel
 - CREATE TABLE-Anweisung,747
 - MATCH UNIQUE SIMPLE-Klausel
 - CREATE TABLE-Anweisung,747
 - MATCH-Klausel
 - ALTER TABLE-Anweisung,522
 - CREATE TABLE-Anweisung,747
 - Materialisierte Ansicht, Eigenschaften
 - RefreshType-Eigenschaft,1258
 - Materialisierte Ansichten
 - Abfrage mit Index-Hints,871
 - Abfrage mit Tabellen-Hints,869
 - aktivieren mit SQL,476
 - Aktualisieren mit Sybase Central,987
 - alle materialisierten Ansichten in der Datenbank auflisten,1258
 - ändern mit SQL,476
 - Änderung von materialisierten Ansichten eines anderen Eigentümers,478
 - CREATE MATERIALIZED VIEW-Anweisung,652
 - Daten mit UNLOAD-Anweisung entladen,1098
 - DROP MATERIALIZED VIEW-Anweisung,813,987
 - Eignung als Sofortansicht testen,1256
 - Einstellung der Isolationsstufe zum Aktualisieren,987
 - entschlüsseln mit Sybase Central,476
 - für Sofortansichten erforderliche Privilegien,478
 - in SYSMVOPTION gespeicherte Erstellungsoptionen,1465
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - löschen mit Sybase Central,813
 - mit SQL erstellen,652
 - Status bestimmen,1258
 - verschlüsseln mit Sybase Central,476
 - Verwendung bei Optimierung mit SQL aktivieren,476

- Verwendung bei Optimierung mit SQL
 - deaktivieren,476
- materialisierte Ansichten
 - ALTER INDEX-Anweisung,466
 - ALTER MATERIALIZED VIEW-Anweisung,476
 - deaktivieren mit SQL,476
 - Indizes validieren,1118
- Materialisierte Ansichten aktualisieren
 - REFRESH MATERIALIZED VIEWS-Anweisung,987
 - REFRESH-Klausel, ALTER MATERIALIZED VIEW-Anweisung,477
- MATERIALIZED VIEW OPTIMIZATION, Option
 - MERGE-Anweisung,957
- MATERIALIZED VIEW OPTIMIZATION-Klausel
 - DELETE-Anweisung,793
 - EXCEPT-Anweisung,842
 - INSERT-Anweisung,920
 - INTERSECT-Anweisung,927
 - SELECT-Anweisung,1027
 - UPDATE-Anweisung,1112
- MATERIALIZED VIEW-Klausel
 - TRUNCATE-Anweisung,1089
 - VALIDATE-Anweisung,1118
- materialized_view_optimization, Option
 - für INSERT-Anweisungen setzen,920
 - für UPDATE-Anweisungen setzen,1112
 - in SELECT-Anweisung außer Kraft setzen,1027
- materialized_view_optimization-Option
 - für DELETE-Anweisungen setzen,793
 - für EXCEPT-Klausel einstellen,842
 - für INTERSECT-Anweisung setzen,927
 - für INTERSECT-Klauseln einstellen,927
 - für UNION-Klausel setzen,1097
- Mathematische Ausdrücke
 - arithmetische Operatoren,11
- MAX WRITE-Klausel
 - BACKUP-Anweisung,552
- MAX-Funktion
 - Syntax,310
- max_connections-Option
 - ALTER LOGIN POLICY-Anweisung,472
 - CREATE LOGIN POLICY-Anweisung,647
- max_days_since_login-Option
 - ALTER LOGIN POLICY-Anweisung,472
 - CREATE LOGIN POLICY-Anweisung,647
- max_disconnected_time-Option
 - SET MIRROR OPTION-Anweisung,1035
- max_failed_login_attempts-Option
 - ALTER LOGIN POLICY-Anweisung,472
 - CREATE LOGIN POLICY-Anweisung,647
- max_non_dba_connections-Option
 - ALTER LOGIN POLICY-Anweisung,472
 - CREATE LOGIN POLICY-Anweisung,647
- max_query_tasks, Option
 - für INSERT-Anweisungen setzen,920
 - für UPDATE-Anweisungen setzen,1112
 - in einer MERGE-Anweisung außer Kraft setzen,957
- max_query_tasks-Option
 - für DELETE-Anweisungen setzen,793
 - für die EXCEPT-Anweisung setzen,842
 - für INTERSECT-Anweisung setzen,927
 - für UNION-Anweisung setzen,1097
- max_retry_connect_time-Option
 - SET MIRROR OPTION-Anweisung,1035
- Maximum
 - Datumsbereiche,123
- MAXIMUM TERM LENGTH-Klausel
 - ALTER TEXT CONFIGURATION-Anweisung,533
- MaxMultiProgrammingLevel-Eigenschaft
 - mit sa_server_option einstellen,1306
- MAXVALUE-Klausel
 - ALTER SEQUENCE-Anweisung,491
 - CREATE SEQUENCE-Anweisung,700
- MEDIAN-Funktion
 - Syntax,311
- Mehrbyte-Zeichensätze
 - Daten entladen,1100
 - Daten laden,940
- Mehrfache Ergebnismengen
 - abrufen,1003
- Mehrzeilenabrufe
 - FETCH-Anweisung,854
 - OPEN-Anweisung,965
- Mehrzeilige Einfügungen
 - Info,846
- Meldungen
 - anzeigen,959
 - erstellen,655
 - MESSAGE-Anweisung,959
 - mit der DROP MESSAGE-Anweisung löschen,814
- Meldungsfenster
 - Meldungen drucken,980
- Mengen

für reguläre Ausdrücke,28
 Mengenoperatoren
 EXCEPT-Anweisung,841
 INTERSECT-Anweisung,927
 Mengendifferenz,841
 NULL,78
 Schnittmenge setzen,927
 UNION setzen,1096
 UNION-Anweisung,1096
 MERGE-Anweisung
 Syntax,952
 MERGE-Klausel
 ALTER SYNCHRONIZATION PROFILE-
 Anweisung,511
 MESSAGE LOG-Klausel
 LOAD TABLE-Anweisung,940
 MESSAGE-Anweisung
 Syntax,959
 MessageCategoryLimit-Eigenschaft
 mit sa_server_option einstellen,1306
 Meta-Eigenschaften
 LOAD TABLE-Anweisung,939
 OPENXML-Operator,13
 Metazeichen
 Liste der in regulären Ausdrücken verwendeten
 Metazeichen,28
 METHODS-Klausel
 ALTER SERVICE-Anweisung [SOAP über
 HTTP],500
 ALTER SERVICE-Anweisungen [HTTP-
 Webdienst],494
 CREATE SERVICE-Anweisung [HTTP-
 Webdienst],497,709
 CREATE SERVICE-Anweisung [SOAP-
 Webdienst],504,716
 Migrieren
 Kompatibilitätsrollen mit ALTER ROLE,488
 MIME base64
 Datendekodierung,177
 Datenkodierung,177
 MIN-Funktion
 Syntax,313
 Minimum
 Datumsbereiche,123
 MINIMUM TERM LENGTH-Klausel
 ALTER TEXT CONFIGURATION-
 Anweisung,533
 MinMultiProgrammingLevel-Eigenschaft
 mit sa_server_option einstellen,1306
 MINUTE-Funktion
 Syntax,315
 MINUTES-Funktion
 Syntax,315
 MINVALUE-Klausel
 ALTER SEQUENCE-Anweisung,491
 CREATE SEQUENCE-Anweisung,700
 MIRROR ON-Klausel
 START DATABASE-Anweisung,1064
 MIRROR-Klausel
 CREATE DATABASE-Anweisung,590
 MirrorServerState-Eigenschaft
 DB_EXTENDED_PROPERTY-Funktion,226
 MirrorState-Eigenschaft
 DB_EXTENDED_PROPERTY-Funktion,226
 MobiLink
 ALTER PUBLICATION-Anweisung,485
 ALTER SYNCHRONIZATION PROFILE-
 Anweisung,511
 ALTER SYNCHRONIZATION
 SUBSCRIPTION-Anweisung,512
 ALTER SYNCHRONIZATION USER-
 Anweisung,515
 CREATE PUBLICATION-Anweisung,690
 CREATE SYNCHRONIZATION PROFILE-
 Anweisung,732
 CREATE SYNCHRONIZATION
 SUBSCRIPTION-Anweisung,733
 CREATE SYNCHRONIZATION USER-
 Anweisung,735
 DROP PUBLICATION-Anweisung,817
 DROP SYNCHRONIZATION PROFILE-
 Anweisung,829
 DROP SYNCHRONIZATION SUBSCRIPTION-
 Anweisung,830
 START SYNCHRONIZATION DELETE-
 Anweisung,1071
 START SYNCHRONIZATION SCHEMA
 CHANGE-Anweisung,1072
 STOP SYNCHRONIZATION DELETE-
 Anweisung,1079
 STOP SYNCHRONIZATION SCHEMA
 CHANGE-Anweisung,1085
 MobiLink-Benutzer
 ALTER SYNCHRONIZATION USER-
 Anweisung,515

- CREATE SYNCHRONIZATION USER-Anweisung, 735
- DROP SYNCHRONIZATION USER-Anweisung, 831
- MOD-Funktion
 - Syntax, 317
- MONEY-Datentyp
 - Syntax, 113
- MONTH-Funktion
 - Syntax, 318
- MONTHNAME-Funktion
 - Syntax, 318
- MONTHS-Funktion
 - Syntax, 319
- Multiprogramming-Stufe
 - aktuellen Wert einstellen, 1306
 - automatische Optimierung steuern, 1306
 - Höchstwert festlegen, 1306
 - Mindestwert festlegen, 1306
 - Statistiken anzeigen, 1306
- Musterlänge
 - LIKE-Suchbedingung, 52
- Musterübereinstimmung
 - Kollationen, 52
 - LIKE-Suchbedingung, 51
 - Musterlänge, 52
 - PATINDEX-Funktion, 333
 - REGEXP-Suchbedingung, 55
 - SIMILAR TO-Suchbedingung, 56
 - und Berücksichtigung von Groß- und Kleinschreibung, 52
- N**
- N-gram
 - ALTER TEXT CONFIGURATION-Anweisung, 533
- Nachbarschaftssuchen
 - CONTAINS-Suchbedingung, 59
- Nachrichten
 - entfernte Typen löschen, 818
 - SQL Remote, entfernte Typen ändern, 487
 - SQL Remote, entfernte Typen erstellen, 694
- Nachrichten-Steuerungsparameter
 - einstellen, 1046
- NAME-Klausel
 - GET DESCRIPTOR-Anweisung [ESQL], 878
- Name_materialisierte_Ansicht
 - allgemeines Element der SQL-Syntax, 445
- Namen
 - Spaltennamen, 24
- NAMESPACE-Klausel
 - CREATE FUNCTION-Anweisung [Webclients], 628
 - CREATE PROCEDURE-Anweisung [Webclients], 677
- Native Funktionen, Aufrufe
 - Prozeduren, 665
- Native Prozeduren
 - Schnittstelle zu nativen Prozeduren mit CREATE PROCEDURE-Anweisung, 661
- NATURAL JOIN-Klausel
 - FROM-Klausel, SQL-Syntax, 863
- NCHAR COLLATION-Klausel
 - CREATE DATABASE-Anweisung, 589
 - Kollationsanpassung, 589
- NCHAR-Datentyp
 - Beschreibung, 98
 - DESCRIBE bei einer NCHAR-Spalte verwenden, 98
 - mit der LIKE-Suchbedingung verwenden, 54
 - mit der REGEXP-Suchbedingung verwenden, 55
 - mit der SIMILAR TO-Suchbedingung verwenden, 57
 - Syntax, 98
 - vergleichen mit CHAR-Datentyp, 141
- NCHAR-Funktion
 - Syntax, 321
- NcharCollation-Eigenschaft
 - DB_EXTENDED_PROPERTY-Funktion, 226
- NEAR-Klausel
 - CONTAINS-Suchbedingung, 59
- NEW SUBSCRIBE BY-Klausel
 - UPDATE-Anweisung, 1109
 - UPDATE-Anweisung [SQL Remote], 1106
- NEW-Klausel
 - INSTALL EXTERNAL OBJECT-Anweisung, 923
 - INSTALL JAVA-Anweisung, 925
- NEWID-Funktion
 - Syntax, 321
- NEXT-Klausel
 - FETCH-Anweisung, 854
- NEXT_CONNECTION-Funktion
 - Syntax, 322
- NEXT_DATABASE-Funktion
 - Syntax, 324

NEXT_HTTP_HEADER-Funktion	CREATE FUNCTION-Anweisung [externer Aufruf],617
Syntax,325	
NEXT_HTTP_RESPONSE_HEADER-Funktion	NOT ENCRYPTED-Klausel
Syntax,326	ALTER MATERIALIZED VIEW-Anweisung,477
NEXT_HTTP_VARIABLE-Funktion	LOAD TABLE-Anweisung,937
Syntax,327	NOT NULL-Klausel
NEXT_SOAP_HEADER-Funktion	ALTER TABLE-Anweisung,518
Syntax,328	CREATE EXISTING TABLE-Anweisung,613
NextScheduleTime-Eigenschaft	CREATE TABLE-Anweisung,741
DB_EXTENDED_PROPERTY-Funktion,226	NOT TRANSACTIONAL-Klausel
Nicht festgeschriebene Lesevorgänge	CREATE LOCAL TEMPORARY TABLE-Anweisung,646
FROM-Klausel,869	CREATE TABLE-Anweisung,739
Nicht größer als	DECLARE LOCAL TEMPORARY TABLE-Anweisung,785
Vergleichoperator,9	NOTFOUND-Klausel
Nicht kleiner als	WHENEVER-Anweisung [ESQL],1122
Vergleichsoperator,9	NOTIFY TRACE EVENT-Anweisung
NO COPY-Klausel	Syntax,963
BACKUP-Anweisung,552	NOW-Funktion
NO INDEX-Klausel	Syntax,330
ALTER TABLE-Anweisung,523,741	NTEXT-Datentyp
FROM-Klausel,871	Syntax,99
NO RESULT SET-Klausel	NULL
CREATE PROCEDURE-Anweisung,684	ASE-Kompatibilität,78
CREATE PROCEDURE-Anweisung [externer Aufruf],664	Domänen,598
CREATE PROCEDURE-Anweisung [T-SQL],679	Drei-Werte-Logik,77
NO SCROLL-Cursor	Info,76
deklarieren,778	ISNULL-Funktion,295
NO SCROLL-Klausel	NULL,76
DECLARE CURSOR-Anweisung,779	Platzbedarf für NULL-Werte,77
FOR-Anweisung,857	zurückgegeben von Funktionen, wenn ein NULL-Argument angegeben wird,153
NOLOCK, Tabellen-Hint	NULL-Klausel
FROM-Klausel,869	ALTER TABLE-Anweisung,518
NONCLUSTERED-Klausel	CASE-Anweisung,566
ALTER INDEX-Anweisung,466	CASE-Anweisung [T-SQL],568
NOSTRIP-Klausel	CREATE DOMAIN-Anweisung,598
INPUT-Anweisung,914	CREATE TABLE-Anweisung,741
NOT	NULL-Konstanten
Bit-Operatoren,20	in NUMERIC konvertieren,148
Drei-Werte-Logik,67	in Zeichenfolgentypen konvertieren,148
logische Operatoren, Beschreibung,10	NULLABLE-Klausel
NOT COMPRESSED-Klausel	GET DESCRIPTOR-Anweisung [ESQL],878
LOAD TABLE-Anweisung,936	NULLIF-Funktion
NOT DETERMINISTIC-Klausel	Info,331
CREATE FUNCTION-Anweisung [benutzerdefiniert],634	mit CASE-Ausdrücken verwenden,26
CREATE FUNCTION-Anweisung [externe Prozeduren],619	NUMBER-Funktion

- SQL Remote-Aktualisierungen,1107
- Syntax,331
- NUMERIC-Datentyp
 - in DOUBLE konvertieren,149
 - mit Datums- und Uhrzeitangaben vergleichen,143
 - Syntax,109
- Nummerische Datentypen
 - BIGINT,104
 - BIT,105
 - DECIMAL,106
 - DOUBLE,107
 - DOUBLE in NUMERIC konvertieren,149
 - FLOAT,108
 - Info,103
 - INTEGER,109
 - NUMERIC,109
 - REAL,111
 - SMALLINT,112
 - TINYINT,112
- Nummerische Funktionen
 - alphabetische Liste,161
- NVARCHAR-Datentyp
 - Beschreibung,99
 - DESCRIBE bei einer NVARCHAR-Spalte,99
 - Syntax,99
- O**
- OBJECT_ID-Funktion
 - Syntax,165
- OBJECT_NAME-Funktion
 - Syntax,165
- Objekte
 - Benutzerprivilegien auflisten,1363,1379
- Objekte ersetzen
 - sa_make_object,1254
- ODBC
 - statische Cursor deklarieren,778
- Öffnen, Cursor
 - OPEN-Anweisung,964
- OFFSET-Klausel
 - GET DATA-Anweisung,877
 - SELECT-Anweisung,1022
- OLAP
 - CUBE-Vorgang,904
 - GROUP BY-Klausel,903
 - GROUPING SETS-Vorgang,903
 - GROUPING-Funktion,271
 - ROLLUP-Vorgang,904
 - WINDOW-Klausel,1124
- OLAP-Funktionen
 - AVG-Funktion,175
 - COUNT-Funktion,205
 - COUNT_BIG-Funktion,206
 - COVAR_POP-Funktion,209
 - CUME_DIST-Funktion,213
 - DENSE_RANK-Funktion,237
 - MAX-Funktion,310
 - MEDIAN-Funktion,311
 - MIN-Funktion,313
 - PERCENT_RANK-Funktion,335
 - RANK-Funktion,346
 - REGR_AVGX-Funktion,351
 - REGR_AVGY-Funktion,352
 - REGR_COUNT-Funktion,354
 - REGR_INTERCEPT-Funktion,355
 - REGR_R2-Funktion,357
 - REGR_SLOPE-Funktion,358
 - REGR_SXX-Funktion,360
 - REGR_SXY-Funktion,361
 - ROW_NUMBER-Funktion,373
 - STDDEV-Funktion,393
 - STDDEV_POP-Funktion,393
 - STDDEV_SAMP-Funktion,395
 - SUM-Funktion,403
 - VAR_POP-Funktion,425
 - VAR_SAMP-Funktion,427
- OLD KEY-Klausel
 - CREATE ENCRYPTED FILE-Anweisung,604
 - CREATE ENCRYPTED TABLE DATABASE-Anweisung,601
- OLD SUBSCRIBE BY-Klausel
 - UPDATE-Anweisung,1109
 - UPDATE-Anweisung [SQL Remote] ,1106
- ON COMMIT-Klausel
 - CREATE LOCAL TEMPORARY TABLE-Anweisung,646
 - CREATE TABLE-Anweisung,739
 - DECLARE LOCAL TEMPORARY TABLE-Anweisung,784
- ON DATABASE-Klausel
 - STOP DATABASE-Anweisung,1074
- ON EXCEPTION RESUME-Klausel
 - CREATE FUNCTION-Anweisung [benutzerdefiniert],634
 - CREATE PROCEDURE-Anweisung,685

- Info,685
- ON EXISTING -Klausel
 - INSERT-Anweisung,919
- ON EXISTING ERROR-Klausel
 - BACKUP-Anweisung,551
 - Verhalten mit DEFAULT-Spalten,920
- ON EXISTING SKIP-Klausel
 - Verhalten mit DEFAULT-Spalten,920
- ON-Formulierung
 - Suchbedingungen,42
- ON-Klausel
 - ALTER STATISTICS-Anweisung,510
 - ALTER TRIGGER-Anweisung,539
 - CREATE EVENT-Anweisung,610
 - CREATE INDEX-Anweisung,638
 - CREATE TEXT INDEX-Anweisung,759
 - CREATE TRIGGER-Anweisung [Transact-SQL],769
 - DROP STATISTICS-Anweisung,827
 - DROP TEXT INDEX-Anweisung,834
 - MERGE-Anweisung,954
 - TRUNCATE TEXT INDEX-Anweisung,1092
 - VALIDATE-Anweisung,1118
- on_tsq_error-Option
 - und ON EXCEPTION RESUME-Klausel,685
- OPEN-Anweisung
 - Embedded SQL-Syntax,964
 - Syntax,964
 - WITH HOLD-Cursor,964
- OPENSTRING-Klausel
 - Beispiel,876
 - FROM-Klausel,867
- OPENXML-Operator
 - Liste der unterstützten Meta-Eigenschaften,13
 - Syntax,12
 - unterstützte Kind-Tests,16
- Operator-Vorrang
 - Syntax,21
- Operatoren
 - arithmetische Operatoren,11
 - Array-Operatoren,19
 - Bit-Operatoren,20
 - Info,9
 - logische Operatoren, Beschreibung,10
 - OPENXML-Operator,12
 - Vergleichsoperatoren,9
 - Vorrang,21
 - Zeichenfolgenoperatoren,11
- Optimierer
 - CREATE STATISTICS-Anweisung,729
 - explizite Selektivitätsschätzungen,68
 - Tabellen,1144
- Optimiererpläne
 - Textspezifikationen abrufen,851
- Optimiererstatistiken
 - mit der DROP STATISTICS-Anweisung löschen,827
- Optimierung
 - bestehende Tabellen definieren,614
 - erzwingen mit FORCE OPTIMIZATION-Option,1027
 - Vermeiden mit FORCE NO OPTIMIZATION-Klausel,1028
- optimization_goal, Option
 - für INSERT-Anweisungen setzen,920
 - für UPDATE-Anweisungen setzen,1112
 - in einer MERGE-Anweisung außer Kraft setzen,957
- optimization_goal-Option
 - für DELETE-Anweisungen setzen,793
 - für die EXCEPT-Anweisung setzen,842
 - für INTERSECT-Anweisung setzen,927
 - für UNION-Anweisung setzen,1097
- optimization_level, Option
 - für INSERT-Anweisungen setzen,920
 - für UPDATE-Anweisungen setzen,1112
 - in einer MERGE-Anweisung außer Kraft setzen,957
- optimization_level-Option
 - für DELETE-Anweisungen setzen,793
 - für die EXCEPT-Anweisung setzen,842
 - für INTERSECT-Anweisung setzen,927
 - für UNION-Anweisung setzen,1097
- optimization_workload, Option
 - für INSERT-Anweisungen setzen,920
 - für UPDATE-Anweisungen setzen,1112
 - in einer MERGE-Anweisung außer Kraft setzen,957
- optimization_workload-Option
 - für DELETE-Anweisungen setzen,793
 - für die EXCEPT-Anweisung setzen,842
 - für INTERSECT-Anweisung setzen,927
 - für UNION-Anweisung setzen,1097
- OPTION-Klausel
 - CREATE SYNCHRONIZATION
 - SUBSCRIPTION-Anweisung [MobiLink],734

- CREATE SYNCHRONIZATION USER,736
- DELETE-Anweisung,793
- EXCEPT-Anweisung,842
- INSERT-Anweisung,920
- INTERSECT-Anweisung,927
- MERGE-Anweisung,957
- SELECT-Anweisung,1027
- UNION-Anweisung,1097
- UPDATE-Anweisung,1112
- Optionen
 - Anfangseinstellungen für die sp_login_environment-Systemprozedur,1376
 - Anfangseinstellungen für die sp_tsql_environment-Systemprozedur,1409
 - außer Kraft setzen,1306
 - einstellen in Interactive SQL,1043
 - entfernte einstellen,1046
 - in Transact-SQL einstellen,1056
 - mit sp_tsql_environment-Systemprozedur setzen,1409
 - quoted_identifizier und T-SQL-Kompatibilität,41
 - SET OPTION-Anweisung,1039
 - SYSOPTIONS, konsolidierte Ansicht,1524
 - Systemansichten,1467
 - SYSUSEROPTIONS, konsolidierte Ansicht,1534
 - Werte abrufen,880
- Optionen, Überwachungsliste
 - mit sa_server_option konfigurieren,1306
- OptionWatchAction-Eigenschaft
 - mit sa_server_option einstellen,1306
- OptionWatchList-Eigenschaft
 - mit sa_server_option konfigurieren,1306
- OR
 - Bit-Operatoren,20
 - Drei-Werte-Logik,67
 - logische Operatoren, Beschreibung,10
- OR REPLACE-Klausel
 - CREATE FUNCTION-Anweisung [benutzerdefiniert],633
 - CREATE FUNCTION-Anweisung [externe Prozeduren],619
 - CREATE FUNCTION-Anweisung [externer Aufruf],617
 - CREATE FUNCTION-Anweisung [Webdienst],624
 - CREATE MIRROR SERVER-Anweisung,657
 - CREATE PROCEDURE-Anweisung,681
 - CREATE PROCEDURE-Anweisung [externer Aufruf],661
 - CREATE PROCEDURE-Anweisung [T-SQL],679
 - CREATE PROCEDURE-Anweisung [Webclients],671
 - CREATE SEQUENCE-Anweisung,699
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung,719
 - CREATE SYNCHRONIZATION PROFILE-Anweisung,732
 - CREATE TRIGGER-Anweisung,762
 - CREATE VARIABLE-Anweisung,771
 - CREATE VIEW-Anweisung,773
- Oracle-Datenbanken
 - nach SQL Anywhere mit sa_migrate-Systemprozedur migrieren,1266
- ORDER BY-Klausel
 - DELETE-Anweisung,792
 - Info,1025
 - INTERSECT-Anweisung,927
 - SELECT-Anweisung,1020
 - UPDATE-Anweisung,841,1112
 - UPDATE-Anweisung [SQL Remote] ,1106
 - WINDOW-Klausel,1125
- ORDER-Klausel
 - CREATE TRIGGER-Anweisung,763
 - LOAD TABLE-Anweisung,940
 - UNLOAD-Anweisung,1100
- Ordinaldatum
 - ISO 8601,117
- ORGANIZATION-Klausel
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung,507
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung,721
- OUT-Klausel
 - CREATE PROCEDURE-Anweisung,683
- OUTER APPLY-Klausel
 - FROM-Klausel,868
- OUTPUT-Anweisung
 - Syntax,968
- OUTPUT-Klausel
 - CREATE PROCEDURE-Anweisung [T-SQL],679
 - DESCRIBE-Anweisung,795
- output_log_send_limit
 - SQL Remote-Syntax,1046
- output_log_send_now
 - SQL Remote-Syntax,1046

output_log_send_on_error
 SQL Remote-Syntax,1046
OWNER.TABLE.COLUMN-Klausel
 DESCRIBE-Anweisung [ESQL],794
OwnerName-Eigenschaft
 sa_materialized_view_info-Systemprozedur,1258

P

PAGE SIZE-Klausel
 CREATE DATABASE-Anweisung,589
page_timeout-Option
 SET MIRROR OPTION-Anweisung,1035
Pakete
 Java-Klassen entfernen,997
 Java-Klassen installieren,925
Parallele Sicherungen
 BACKUP-Anweisung,548
Parallelität
 Tabellen sperren,949
Parameter
 benannt,93
 Interactive SQL-Skriptdateien,974
PARAMETERS-Anweisung
 Interactive SQL-Syntax,974
Parametrisierte Ansichten
 Info,774
PARTITION BY-Klausel
 WINDOW-Klausel,1125
Partner-Server
 ALTER MIRROR SERVER-Anweisung,480
Partnerserver
 CREATE MIRROR SERVER-Anweisung,656
 mit der ALTER MIRROR SERVER-Anweisung
 ändern,480
 mit der CREATE MIRROR SERVER-Anweisung
 definieren,657
Passgenauigkeit
 Regressionszeilen,357
PASSTHROUGH FOR SUBSCRIPTION-Klausel
 PASSTHROUGH-Anweisung [SQL Remote],975
PASSTHROUGH FOR-Klausel
 PASSTHROUGH-Anweisung (SQL Remote),975
PASSTHROUGH ONLY-Klausel
 PASSTHROUGH-Anweisung (SQL Remote),975
PASSTHROUGH STOP-Klausel
 PASSTHROUGH-Anweisung (SQL Remote),975
PASSTHROUGH-Anweisung
 SQL Remote-Syntax,975
password_expiry_on_next_login-Option
 ALTER LOGIN POLICY-Anweisung,472
 CREATE LOGIN POLICY-Anweisung,647
password_grace_time-Option
 ALTER LOGIN POLICY-Anweisung,472
 CREATE LOGIN POLICY-Anweisung,647
password_life_time-Option
 ALTER LOGIN POLICY-Anweisung,472
 CREATE LOGIN POLICY-Anweisung,647
PATINDEX-Funktion
 Syntax,333
PCTFREE-Einstellung
 ALTER TABLE-Anweisung,516
 CREATE LOCAL TEMPORARY TABLE-
 Syntax,645
 CREATE TABLE-Anweisung,737
 DECLARE LOCAL TEMPORARY TABLE-
 Syntax,784
 LOAD TABLE-Syntax,931
PCTFREE-Klausel
 CREATE LOCAL TEMPORARY TABLE-
 Anweisung,645
 CREATE TABLE-Anweisung,748
 DECLARE LOCAL TEMPORARY TABLE-
 Anweisung,784
 LOAD TABLE,940
PERCENT_RANK-Funktion
 Syntax,335
Performance
 das I/O-Kostenmodell neu kalibrieren,454
 den Datenbankserver neu kalibrieren,451
 Komprimierungsstatistiken,1181
 Speicherplatz vorab zuweisen,458
Performance überwachen
 Bestimmung der Ausführungszeit,1229
PERL-Klausel
 INSTALL EXTERNAL OBJECT-Anweisung,923
PHP-Klausel
 INSTALL EXTERNAL OBJECT-Anweisung,923
Physische Indizes
 in SYSPHYSIDX-Systemansicht
 aufgezeichnet,1468
PI-Funktion
 Syntax,337
PLAN-Funktion
 Syntax,337
PLANAR-Modell

- ALTER SPATIAL REFERENCE SYSTEM-
Anweisung,508
- CREATE SPATIAL REFERENCE SYSTEM-
Anweisung,722
- Pläne
 - Beispiel für das Speichern eines Plans in einer
Datei,270
 - Cursor,337
 - EXPLANATION-Funktion,260
 - GRAPHICAL_PLAN-Funktion,269
 - PLAN-Funktion,337
 - TRACED_PLAN-Funktion,413
- Planen
 - Ereignisse mit der ALTER EVENT-
Anweisung,461
 - Ereignisse mit der CREATE EVENT-
Anweisung,606
 - Ereignisse mit der CREATE EVENT-Anweisung
erstellen,606
 - Textspezifikationen abrufen,851
 - WAITFOR,1120
- Plattenspeicher
 - Ereignisse für fehlenden Plattenspeicher
erstellen,606
 - Ereignisse mit der CREATE EVENT-Anweisung
erstellen,606
 - verfügbaren Speicher ermitteln,1212
- Plattenspeicher-Zeitmodell
 - aktueller Wert,1223
- Platzhalter
 - PATINDEX-Funktion,333
- Platzhalterzeichen
 - für reguläre Ausdrücke,28
 - LIKE-Suchbedingung,51
 - REGEXP-Suchbedingung,55
 - SIMILAR TO-Suchbedingung,56
- POLYGON FORMAT-Klausel
 - ALTER SPATIAL REFERENCE SYSTEM-
Anweisung,508
 - CREATE SPATIAL REFERENCE SYSTEM-
Anweisung,725
- Polygonausrichtung
 - Clockwise,725
 - CounterClockwise,725
 - EvenOdd,725
- Polygonformate
 - Clockwise,725
 - CounterClockwise,725
- EvenOdd,725
- Pooling
 - Verbindungspooling aktivieren,1059
- Populationskovarianz
 - Info,209
- Populationsvarianz
 - Info,425
- Positionsbasierte DELETE-Anweisung
 - Syntax,789
- Positionssperren
 - sa_locks-Systemprozedur,1251
- POWER-Funktion
 - Syntax,338
- Prädikate
 - ALL-Suchbedingung,45
 - ANY-Suchbedingung,46
 - BETWEEN-Suchbedingung,48
 - CONTAINS-Suchbedingung,59
 - Drei-Werte-Logik,67
 - EXISTS-Suchbedingung,66
 - explizite Selektivitätsschätzungen,68
 - IN-Suchbedingung,58
 - Info,42
 - IS DISTINCT FROM,47
 - IS NOT DISTINCT FROM,47
 - IS NOT NULL-Suchbedingung,67
 - IS NULL-Suchbedingung,67
 - IS TRUE- oder FALSE-Suchbedingungen,67
 - IS UNKNOWN-Suchbedingung,67
 - LIKE-Suchbedingung,51
 - REGEXP-Suchbedingung,55
 - SIMILAR TO-Suchbedingung,56
 - SOME-Suchbedingung,46
 - SQL-Unterabfragen in,44
 - Syntax,42
 - Vergleichsoperatoren,9
- Präfixsuchen
 - CONTAINS-Suchbedingung,59
- PRECEDING-Klausel
 - WINDOW-Klausel,1126
- PRECISION-Klausel
 - GET DESCRIPTOR-Anweisung [ESQL],878
 - SET DESCRIPTOR-Anweisung [ESQL],1033
- preferred-Option
 - ALTER MIRROR SERVER-Anweisung,482
 - CREATE MIRROR SERVER-Anweisung,659
- PREFILTER EXTERNAL NAME-Klausel

ALTER TEXT CONFIGURATION-Anweisung,534
 PREFIX-Klausel
 ALTER TABLE-Anweisung,522
 CREATE TABLE-Anweisung,740
 PREPARE TO COMMIT-Anweisung
 Syntax,979
 PREPARE-Anweisung
 Embedded SQL-Syntax,976
 PRESERVE-Klausel
 CREATE LOCAL TEMPORARY TABLE-Anweisung,645
 DECLARE LOCAL TEMPORARY TABLE-Anweisung,784
 Primärschlüssel
 ALTER INDEX-Anweisung,466
 CREATE SEQUENCE-Anweisung,699
 eindeutige Werte generieren,321
 eindeutige Werte mit UUIDs generieren,321
 entfernte Tabellen und sp_remote_exported_keys-Systemprozedur ,1388
 entfernte Tabellen und sp_remote_imported_keys-Systemprozedur,1390
 Integritätsregeln in der CREATE TABLE-Anweisung,745
 mit der ALTER INDEX-Anweisung gruppieren,466
 mit der ALTER INDEX-Anweisung umbenennen,466
 Spaltenreihenfolge in der CREATE TABLE-Anweisung,745
 UUIDs und GUIDs,321
 Primärschlüsselsperren
 sa_locks-Systemprozedur,1251
 Primärserver
 mit der CREATE MIRROR SERVER-Anweisung definieren,657
 Primärtabellen
 Systemansichten,1452
 PRIMARY KEY-Integritätsregelklausel
 CREATE TABLE-Anweisung,745
 PRIMARY KEY-Klausel
 ALTER INDEX-Anweisung,466
 ALTER TABLE-Anweisung ,522
 CREATE TABLE-Anweisung,745
 REORGANIZE TABLE-Anweisung,998
 VALIDATE-Anweisung,1118
 PRINT-Anweisung
 Transact-SQL-Syntax,980
 Privilegien
 Berichte generieren,1363,1379
 CONSOLIDATE erteilen,889
 dazugehörige Systemprozeduren,1162
 DBSpaces,891
 entziehen,1009
 für eine Tabelle löschen,528
 für Sequenzen entziehen,1011
 Privilegien auf Objektebene mit der GRANT-Anweisung erteilen,883
 Privilegien auf Objektebene mit der REVOKE-Anweisung entziehen,1010
 Prozeduren,892
 PUBLISH erteilen,894
 REMOTE erteilen,900
 REVOKE REMOTE-Anweisung [SQL Remote],1008
 Sequenzen,902
 sp_auth_sys_role_info-Systemprozedur,1355
 sp_has_role-Systemprozedur,1371
 sp_proc_priv-Systemprozedur,1384
 sp_sys_priv_role_info-Systemprozedur,1399
 SQL Remote, CONSOLIDATE entziehen,1006
 SQL Remote, entziehen,1013,1014
 SQL Remote, PUBLISH entziehen,1007
 SYSCOLPERM-Systemansicht,1441
 SYSTABAUTH, konsolidierte Ansicht,1532
 SYSTABLEPERM-Systemansicht,1498
 Systemprivilegien mit der GRANT-Anweisung erteilen,883
 Systemprivilegien mit der REVOKE-Anweisung entziehen,1010
 Privilegien auf Objektebene
 mit der GRANT-Anweisung erteilen,883
 mit der REVOKE-Anweisung entziehen,1010
 PROCEDURE-Klausel
 ALTER DATABASE-Anweisung,451
 CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote],690
 DESCRIBE-Anweisung,798
 ProcedureProfiling-Eigenschaft
 mit sa_server_option einstellen,1306
 ProcessorAffinity-Eigenschaft
 mit sa_server_option einstellen,1306
 Produktname
 abrufen,1422
 ProfileFilterConn-Eigenschaft

- mit sa_server_option einstellen,1306
- ProfileFilterUser-Eigenschaft
 - mit sa_server_option einstellen,1306
- promotion_time-Option
 - SET MIRROR OPTION-Anweisung,1035
- PROMPT-Klausel
 - INPUT-Anweisung,914
- Properties-Eigenschaft
 - DB_EXTENDED_PROPERTY-Funktion,226
- PROPERTY-Funktion
 - Syntax,339
- PROPERTY_DESCRIPTION-Funktion
 - Syntax,341
- PROPERTY_NAME-Funktion
 - Syntax,342
- PROPERTY_NUMBER-Funktion
 - Syntax,342
- Protokollierung
 - ATTACH TRACING-Anweisung,546
 - Deadlocks,1306
 - DETACH TRACING-Anweisung,801
 - REFRESH TRACING LEVEL-Anweisung,993
 - Spalten aktualisieren ohne,1127
 - START LOGGING-Anweisung,1068
 - STOP LOGGING-Anweisung,1077
- Protokollierung zur Diagnose
 - Datensätze löschen,1296
- Protokollierungsdaten
 - mit sa_save_trace_data-Systemprozedur speichern,1302
- Protokollierungsstufen
 - sa_set_tracing_level-Systemprozedur einstellen,1330
- PROXY-Klausel
 - CREATE FUNCTION-Anweisung [Webclients],629
 - CREATE PROCEDURE-Anweisung [Webclients],675
- Proxytabellen
 - CREATE TABLE-Anweisung,739
 - mit der CREATE EXISTING TABLE-Anweisung erstellen,613
- Prozedur
 - Privilegien mit sp_proc_priv auflisten,1384
- Prozeduraufrufe
 - mit der CALL-Anweisung,564
- Prozeduren
 - alphabetische Liste ,1168
 - alphabetische Liste der Systemprozeduren,1168
 - Aufrufe nativer Funktionen,665
 - Ausführung wieder aufnehmen,1003
 - auswählen aus,865
 - beenden,1004
 - benannte Parameter,93
 - benannte Parameter beim Aufrufen zulässig,564
 - benutzerdefinierte Prozeduren ersetzen,683
 - CREATE PROCEDURE-Anweisung,679
 - EXECUTE mit der REVOKE-Anweisung entziehen,1010
 - gespeicherte in Transact-SQL ausführen,848
 - gespeicherte SQL-Prozeduren erstellen,681
 - GRANT EXECUTE-Anweisung,892
 - im Vergleich zu Funktionen,159
 - in Dynamic SQL ausführen,843
 - in Transact-SQL erstellen,679
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - Liste der erweiterten,1162
 - mit der ALTER PROCEDURE-Anweisung ändern,483
 - mit der ALTER PROCEDURE-Anweisung replizieren,483
 - mit der CALL-Anweisung aufrufen,564
 - mit der DROP PROCEDURE-Anweisung löschen,816
 - RAISERROR-Anweisung,982
 - Schnittstelle zu nativen Prozeduren mit CREATE PROCEDURE-Anweisung,661
 - Schnittstelle zur externen Prozedur ersetzen,663
 - System,1161
 - Transact-SQL-Liste,1166
 - variable Ergebnismengen, CREATE PROCEDURE-Anweisung ,683
 - variable Ergebnismengen, CREATE PROCEDURE-Anweisung [externer Aufruf],663
 - variable Ergebnismengen, DESCRIBE-Anweisung [ESQL],796
 - variable Ergebnismengen, PREPARE-Anweisung,977
 - Webdienste ersetzen,671
 - Webdienste erstellen,670
 - Werte zurückgeben,1004
- Prozeduren aufrufen
 - CALL-Anweisung,564
- Prozedurparameter
 - in Interactive SQL auflisten,798

Prozedurprofile
 in Interactive SQL,1306
 in Interactive SQL anzeigen,1290
 von Interactive SQL aus aktivieren,1306
 von Interactive SQL aus deaktivieren,1306
Prozedurprofilerstellung
 sa_procedure_profile-Systemprozedur,1288
Prozedurprofilinformationen
 Zusammenfassung der Prozeduren,1290
Prozentzeichen
 Kommentar-Bezeichner,92
Prüf-Integritätsregeln
 in Domänen verwenden,600
Prüfsummen
 Datenbankeinstellungen ändern,453
 Datenbanken starten,1063
 mit Datenbanken erstellen,586
 VALIDATE CHECKSUM-Anweisung,1118
 validieren,1118
PUBLIC-Klausel
 SET OPTION-Anweisung,1039
PUBLIC-Systemrolle
 mit der GRANT-Anweisung erteilen,882
 mit der REVOKE-Anweisung entziehen,1009
PUBLICATION-Klausel
 UPDATE-Anweisung,1109
 UPDATE-Anweisung [SQL Remote] ,1106
Publikationen
 ALTER PUBLICATION-Anweisung,485
 CREATE PUBLICATION-Anweisung,690
 DROP PUBLICATION-Anweisung,817
 SQL Remote UPDATE-Anweisung,1108
 UPDATE-Anweisung,1113
Publikationseigentümer
 Adresse,818
 entfernter,900
 GRANT PUBLISH-Anweisung,894
 SQL Remote-Adresse,694
 SQL Remote-Adressen,487
PUBLISH-Privileg
 erteilen,894
 SQL Remote, entziehen,1007
PunctuationSensitivity-Eigenschaft
 DB_EXTENDED_PROPERTY-Funktion,226
PURGE-Klausel
 FETCH-Anweisung,855
PUT-Anweisung
 Embedded SQL-Syntax,981

Q

Quadratwurzelfunktion
 SQRT-Funktion,392
Quantifizierer
 für reguläre Ausdrücke,28
QUARTER-Funktion
 Syntax,343
QUIT-Anweisung
 Interactive SQL-Syntax,850
QuittingTime-Eigenschaft
 mit sa_server_option einstellen,1306
QUOTE-Klausel
 LOAD TABLE-Anweisung,941
 OUTPUT-Anweisung,971
 UNLOAD-Anweisung,1101
quoted_identifizier-Option
 mit Transact-SQL SET-Anweisung setzen,1056
T-SQL-Ausdruckskompatibilität,41
QUOTES-Klausel
 LOAD TABLE-Anweisung,941
 UNLOAD-Anweisung,1101

R

R-Quadrat
 Regressionszeilen,357
RADIANS-Funktion
 Syntax,344
RAISERROR-Anweisung
 Syntax,982
RAISERROR-Klausel
 MERGE-Anweisung,956
RAND-Funktion
 Syntax,345
RANGE-Klausel
 WINDOW-Klausel,1125
Rangfolgefunktionen
 alphabetische Liste,155
 CUME_DIST-Funktion,213
 DENSE_RANK-Funktion,237
 PERCENT_RANK-Funktion,335
 RANK-Funktion,346
RANK-Funktion
 Syntax,346
Rastergröße
 ALTER SPATIAL REFERENCE SYSTEM-
 Anweisung,508

- CREATE SPATIAL REFERENCE SYSTEM-Anweisung,724
- Rate-Eigenschaften
 - Werte einstellen,1350
 - Werte inkrementieren,1349
- Räumliche Bezugssysteme
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - vordefinierte installieren,1245
- Räumliche Daten
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung,506
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung,719
 - CREATE SPATIAL UNIT OF MEASURE-Anweisung,727
 - DROP SPATIAL REFERENCE SYSTEM-Anweisung,825
 - DROP SPATIAL UNIT OF MEASURE-Anweisung,825
 - ESRI-Formdateien importieren,938
 - Fehlerbehandlung bei ungültigen Geometrien,1411
 - Geometrien zerlegen,1411
 - ISYSSPATIALREFERENCESYSTEM-Systemtabelle,1140
 - ISYSUNITOFMEASURE-Systemtabelle,1143
 - st_geometry_load_shapefile-Systemprozedur,1416
 - TREAT Funktion,414
- RAW
 - CREATE SERVICE-Anweisung [HTTP-Webdienst],495,707
- Raw-Eigenschaften
 - Werte einstellen,1350
 - Werte inkrementieren,1349
- READ COMMITTED-Klausel
 - REFRESH MATERIALIZED VIEW-Anweisung,988
- READ ONLY-Klausel
 - CREATE SERVER-Anweisung,703
- READ UNCOMMITTED-Klausel
 - REFRESH MATERIALIZED VIEW-Anweisung,988
- READ-Anweisung
 - Syntax,984
- READ_CLIENT_FILE-Funktion
 - Syntax,348
- READCOMMITTED, Tabellen-Hint
 - FROM-Klausel,869
- READONLY-Klausel
 - CREATE SERVER-Anweisung,703
- READPAST, Tabellen-Hint
 - FROM-Klausel,869
- READTEXT-Anweisung
 - Transact-SQL-Syntax,986
- READUNCOMMITTED, Tabellen-Hint
 - FROM-Klausel,869
- REAL-Datentyp
 - mit Datums- und Uhrzeitangaben vergleichen,143
 - Syntax,111
- REBUILD-Klausel
 - ALTER INDEX-Anweisung,467
- RECOMPILE-Klausel
 - ALTER FUNCTION-Anweisung,465
 - ALTER PROCEDURE-Anweisung,483
 - ALTER VIEW-Anweisung,544
- RECOVER-Klausel
 - BACKUP-Anweisung,552
- REFERENCES-Klausel
 - ALTER TABLE-Anweisung,522
 - CREATE TABLE-Anweisung,746
- REFERENCES-Privileg
 - GRANT-Anweisung,883
 - REVOKE-Anweisung,1010
- REFERENCES-Privilegklausel
 - GRANT-Anweisung,885
- REFERENCING-Klausel
 - CREATE TRIGGER-Anweisung,764
- Referenzielle Integrität
 - FROM-Klausel,865
 - MATCH-Klausel in der CREATE TABLE-Anweisung,747
- Referer
 - HTTP_HEADER-Funktion,284
- REFRESH DN-Klausel
 - ALTER USER-Anweisung,542
- REFRESH MATERIALIZED VIEW-Anweisung
 - Syntax,987
- REFRESH TEXT INDEX-Anweisung
 - Syntax,991
- REFRESH TRACING LEVEL-Anweisung
 - Diagnoseprotokollierung,993
 - Syntax,993
- REFRESH TRACING LEVELS-Anweisung
 - sa_diagnostic_tracing_level-Tabelle füllen,1156
- REFRESH-Klausel
 - ALTER TEXT INDEX-Anweisung,536

CREATE TEXT INDEX-Anweisung,759
 RefreshType-Eigenschaft
 sa_materialized_view_info-Systemprozedur,1258
 Regeln
 SQL-Sprachsyntax,1
 REGEXP-Suchbedingung
 Datenbankkollation und Suche nach
 Übereinstimmungen,55
 reguläre Ausdrücke,28
 Syntax,55
 Übereinstimmung mit Teilzeichenklassen
 suchen,55
 verglichen mit LIKE und SIMILAR TO,48
 REGEXP_SUBSTR-Funktion
 Datenbankkollation und Übereinstimmungen,349
 reguläre Ausdrücke,28
 Syntax,349
 REGR_AVGX-Funktion
 Syntax,351
 REGR_AVGY-Funktion
 Syntax,352
 REGR_COUNT-Funktion
 Syntax,354
 REGR_INTERCEPT-Funktion
 Syntax,355
 REGR_R2-Funktion
 Syntax,357
 REGR_SLOPE-Funktion
 Syntax,358
 REGR_SXX-Funktion
 Syntax,360
 REGR_SXY-Funktion
 Syntax,361
 REGR_SYY-Funktion
 Syntax,362
 Regressionsfunktionen
 REGR_AVGX-Funktion,351
 REGR_AVGY-Funktion,352
 REGR_COUNT-Funktion,354
 REGR_INTERCEPT-Funktion,355
 REGR_R2-Funktion,357
 REGR_SLOPE-Funktion,358
 REGR_SXX-Funktion,360
 REGR_SXY-Funktion,361
 REGR_SYY-Funktion,362
 Reguläre Ausdrücke
 Beispiele,39
 Beispiele für Assertierungen,36
 Datenbankkollationen und Suchmuster,49
 Info,27
 Liste der Assertierungen,36
 Metazeichen,28
 Platzhalterzeichen, gruppieren, Mengen,28
 REGEXP-Suchbedingung,28
 REGEXP-Syntax,55
 REGEXP_SUBSTR-Funktion,349
 SIMILAR TO-Suchbedingung,56
 spezielle Zeichenklassen,31
 Syntax,28
 unterstützte Quantifizierer,28
 Reihenfolge der Vorgänge
 SQL-Operator-Vorrang,21
 Reihenfolge von SQL-Klauseln
 Syntaxkonventionen,446
 Reiner Download
 CREATE PUBLICATION-Syntax,690
 Relative Pfade
 INPUT-Anweisung,910
 READ-Anweisung,984
 RELATIVE-Klausel
 FETCH-Anweisung,854
 RELEASE SAVEPOINT-Anweisung
 Syntax,994
 REMAINDER-Funktion
 Syntax,364
 RememberLastPlan-Eigenschaft
 mit sa_server_option einstellen,1306
 RememberLastStatement-Eigenschaft
 mit sa_server_option einstellen,1306
 REMOTE DBA-Berechtigung (nicht mehr empfohlen)
 stattdessen REVOKE ROLE
 SYS_RUN_REPLICATION_ROLE
 verwenden,1009
 REMOTE LOGIN-Klausel
 CREATE EXTERNLOGIN-Anweisung,616
 REMOTE RESET-Anweisung
 SQL Remote-Syntax,995
 REMOTE-Klausel
 CREATE EVENT-Anweisung,606
 REMOTE-Privileg
 SQL Remote, entziehen,1008
 SQL REMOTE, erteilen,900
 remoteoption, Ansicht
 Info,1475
 remoteoptiontype, Ansicht

- Info, 1475
- REMOVE EXTERNAL OBJECT-Anweisung
 - Syntax, 996
- REMOVE JAVA-Anweisung
 - Syntax, 997
- RENAME-Klausel
 - ALTER DBSPACE-Anweisung, 458
 - ALTER DOMAIN-Anweisung, 460
 - ALTER INDEX-Anweisung, 466
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung, 506
 - ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink], 513
 - ALTER TABLE-Anweisung, 527
 - ALTER TEXT INDEX-Anweisung, 536
 - RESTORE DATABASE-Anweisung, 1002
- REORGANIZE TABLE-Anweisung
 - Syntax, 998
- REPEAT-Funktion
 - Syntax, 365
- REPEATABLE READ-Klausel
 - REFRESH MATERIALIZED VIEW-Anweisung, 988
- REPEATABLE READ, Tabellen-Hint
 - FROM-Klausel, 869
- REPLACE-Funktion
 - Syntax, 365
- REPLICATE-Funktion
 - Syntax, 367
- Replikation
 - ALTER TABLE-Anweisung, 516
- Replikationsrollen
 - SYS_REPLICATION_ADMIN_ROLE-Systemrolle, 896
 - SYS_RUN_REPLICATION_ROLE-Systemrolle, 898
- RequestFilterConn-Eigenschaft
 - mit sa_server_option einstellen, 1306
- RequestFilterDB-Eigenschaft
 - mit sa_server_option einstellen, 1306
- RequestLogFile-Eigenschaft
 - clientseitiges Caching von Anweisungen, 1306
 - mit sa_server_option einstellen, 1306
- RequestLogging-Eigenschaft
 - mit sa_server_option einstellen, 1306
- RequestLogMaxSize, Eigenschaft
 - mit sa_server_option einstellen, 1306
- RequestLogNumFiles, Eigenschaft
 - mit sa_server_option einstellen, 1306
- RequestTiming, Eigenschaft
 - mit sa_server_option einstellen, 1306
- Reservierte Wörter
 - als Bezeichner verwenden, 41
 - Liste der SQL-Schlüsselwörter, 1
 - sa_reserved_words-Systemprozedur, 1298
 - SQL Anywhere Server, 1
 - Verwendung in der Syntax, 1
- RESET LOGIN POLICY-Klausel
 - ALTER USER-Anweisung, 541
- RESIGNAL-Anweisung
 - Syntax, 1000
- RESOLVE-Klausel
 - CREATE TRIGGER-Anweisung, 762
- RESTART WITH-Klausel
 - ALTER SEQUENCE-Anweisung, 490
- RESTART-Klausel
 - ALTER DATABASE-Anweisung, 452
- RESTORE DATABASE-Anweisung
 - Syntax, 1001
- RESTORE DEFAULT CALIBRATION-Klausel
 - ALTER DATABASE-Anweisung, 452
- RESULT-Klausel
 - CREATE PROCEDURE-Anweisung, 683
 - CREATE PROCEDURE-Anweisung [externer Aufruf], 663
- RESUME-Anweisung
 - in Interactive SQL nicht unterstützt, 1004
 - Syntax, 1003
- RETURN-Anweisung
 - Syntax, 1004
- RETURNED_LENGTH-Klausel
 - GET DESCRIPTOR-Anweisung [ESQL], 878
- RETURNS-Klausel
 - CREATE FUNCTION-Anweisung, 633
 - CREATE FUNCTION-Anweisung [externer Aufruf], 617
 - CREATE FUNCTION-Anweisung [Webclients], 625
- REVERSE-Funktion
 - Syntax, 367
- REVOKE CONSOLIDATE-Anweisung
 - Remote SQL-Syntax, 1006
- REVOKE PUBLISH-Anweisung
 - Remote SQL-Syntax, 1007
- REVOKE REMOTE DBA-Anweisung

stattdessen REVOKE ROLE
SYS_RUN_REPLICATION_ROLE
verwenden,1013

REVOKE REMOTE DBA-Anweisung (nicht mehr empfohlen)
stattdessen REVOKE ROLE
SYS_RUN_REPLICATION_ROLE
verwenden,1009

REVOKE REMOTE-Anweisung
Remote SQL-Syntax,1008

REVOKE ROLE
SYS_REPLICATION_ADMIN_ROLE-Anweisung
MobiLink-Syntax,1013
SQL Remote-Syntax,1013

REVOKE ROLE
SYS_RUN_REPLICATION_ROLE-Anweisung
MobiLink-Syntax,1014
SQL Remote-Syntax,1014

REVOKE ROLE-Anweisung
Syntax,1009

REVOKE-Anweisung
Syntax,1009

REWRITE-Funktion
Syntax,368

RI-Integritätsregeln
ALTER TABLE-Anweisung,737
mit der ALTER TABLE-Anweisung hinzufügen,
löschen oder ändern,516
mit der ALTER TABLE-Anweisung
umbenennen,528
nicht umbenannt, wenn der zugrunde liegende
Index umbenannt wird,467

Richtlinienoptionen
ändern mit der ALTER LOGIN POLICY-
Anweisung,471
ändern mit der ALTER USER-Anweisung,540

Richtungskoeffizient
Regressionszeilen,358

RIGHT OUTER JOIN-Klausel
FROM-Klausel, SQL-Syntax,863

RIGHT-Funktion
Syntax,370

ROLLBACK TO SAVEPOINT-Anweisung
Syntax,1015

ROLLBACK TRANSACTION-Anweisung
Transact-SQL-Syntax,1016

ROLLBACK TRIGGER-Anweisung
Syntax,1017

ROLLBACK-Anweisung
Syntax,1015

Rollen
benutzerdefinierte Rollen mit der GRANT-
Anweisung erteilen,882
benutzerdefinierte Rollen mit der REVOKE-
Anweisung entziehen,1009
benutzererweiterte Rollen erstellen,696
benutzererweiterte Rollen in Benutzer
zurückkonvertieren,820
dazugehörige Systemprozeduren,1162
Kompatibilitätsrollen mit ALTER ROLE
migrieren,488
Kompatibilitätsrollen mit der GRANT-Anweisung
erteilen,882
Kompatibilitätsrollen mit der REVOKE-
Anweisung entziehen,1009
mit der ALTER ROLE-Anweisung ändern (außer
umbenennen),696
mit der CREATE ROLE-Anweisung erstellen,696
mit der DROP ROLE-Anweisung löschen,820
Systemrollen mit der GRANT-Anweisung
erteilen,882
Systemrollen mit der REVOKE-Anweisung
entziehen,1009
zugrunde liegenden Systemprivilegien
ermitteln,1363

Rollennamen
allgemeines Element der SQL-Syntax,445
Fremdschlüssel in der CREATE TABLE-
Anweisung,746

ROLLUP, Vorgang
GROUP BY-Klausel,904
GROUPING-Funktion,271
WITH ROLLUP-Klausel,904

ROOT-Klausel
CREATE SERVER-Anweisung,703

root_auto_unlock_time-Option
ALTER LOGIN POLICY-Anweisung,472
CREATE LOGIN POLICY-Anweisung,647

ROUND EARTH-Modell
CREATE SPATIAL REFERENCE SYSTEM-
Anweisung,722

ROUND Earth-Modell
ALTER SPATIAL REFERENCE SYSTEM-
Anweisung,508

ROUND-Funktion
Syntax,371

- ROW DELIMITED BY, Option
 - LOAD TABLE-Anweisung, 941
 - ROW DELIMITED BY-Klausel
 - UNLOAD-Anweisung, 1101
 - ROW LOG-Klausel
 - LOAD TABLE-Anweisung, 941
 - ROW-Datentyp
 - deklarieren, 136
 - ROW-Konstruktor
 - Syntax, 371
 - ROW_NUMBER-Funktion
 - Syntax, 373
 - rowcount-Option
 - mit Transact-SQL SET-Anweisung setzen, 1056
 - RowGenerator
 - Systemtabelle, 1159
 - ROWID-Funktion
 - Syntax, 375
 - ROWS-Klausel
 - WINDOW-Klausel, 1125
 - rs_systabgroup-Systemrolle
 - mit der GRANT-Anweisung erteilen, 882
 - mit der REVOKE-Anweisung entziehen, 1009
 - RTRIM-Funktion
 - Syntax, 376
 - Rückgabecodes
 - EXIT-Anweisung [Interactive SQL], 850
 - für MAPI- und SMTP-Systemprozeduren, 1163
 - Rückgängig machen
 - Änderungen durch Zurücksetzen von Transaktionen, 1015
 - Rundungsfehler
 - DOUBLE, 107
 - FLOAT, 108
 - Info, 103
 - REAL, 111
- S**
- sa_ansi_standard_packages-Systemprozedur
 - Info, 1168
 - sa_audit_string-Systemprozedur
 - Syntax, 1170
 - sa_certificate_info-Systemprozedur
 - Syntax, 1170
 - sa_char_terms-Systemprozedur
 - Syntax, 1172
 - sa_check_commit-Systemprozedur
 - Syntax, 1173
 - sa_clean_database-Systemprozedur
 - Syntax, 1174
 - sa_column_stats-Systemprozedur
 - Syntax, 1177
 - sa_conn_activity-Systemprozedur
 - Syntax, 1179
 - sa_conn_compression_info-Systemprozedur
 - Syntax, 1181
 - sa_conn_info-Systemprozedur
 - Syntax, 1183
 - sa_conn_list-Systemprozedur
 - Syntax, 1186
 - sa_conn_options-Systemprozedur
 - Syntax, 1188
 - sa_conn_properties-Systemprozedur
 - Syntax, 1189
 - sa_convert_ml_progress_to_timestamp-Systemprozedur
 - Syntax, 1191
 - sa_convert_timestamp_to_ml_progress-Systemprozedur
 - Syntax, 1192
 - sa_copy_cursor_to_temp_table-Systemprozedur
 - Syntax, 1193
 - sa_cpu_topology-Systemprozedur
 - Syntax, 1194
 - sa_db_info-Systemprozedur
 - Syntax, 1196
 - sa_db_list-Systemprozedur
 - Syntax, 1197
 - sa_db_option-Systemprozedur
 - Syntax, 1199
 - sa_db_properties-Systemprozedur
 - Syntax, 1200
 - SA_DEBUG-Systemrolle
 - mit der GRANT-Anweisung erteilen, 882
 - mit der REVOKE-Anweisung entziehen, 1009
 - sa_dependent_views-Systemprozedur
 - Syntax, 1202
 - sa_describe_cursor-Systemprozedur
 - Syntax, 1203
 - sa_describe_query-Systemprozedur
 - Syntax, 1206
 - sa_describe_shapefile-Systemprozedur
 - Syntax, 1209
 - sa_diagnostic_auxiliary_catalog, Tabelle
 - Info, 1144

sa_diagnostic_blocking, Tabelle Info, 1145	sa_get_info-Systemprozedur Syntax, 1225
sa_diagnostic_cachecontents, Tabelle Info, 1146	sa_get_ldapserver_status-Systemprozedur Syntax, 1227
sa_diagnostic_connection, Tabelle Info, 1147	sa_get_request_profile-Systemprozedur Syntax, 1228
sa_diagnostic_cursor, Tabelle Info, 1148	sa_get_request_times-Systemprozedur Syntax, 1229
sa_diagnostic_deadlock, Tabelle Info, 1149	sa_get_server_messages-Systemprozedur Syntax, 1231
sa_diagnostic_hostvariable, Tabelle Info, 1150	sa_get_table_definition-Systemprozedur Syntax, 1232
sa_diagnostic_internalvariable, Tabelle Info, 1151	sa_get_user_status-Systemprozedur Syntax, 1233
sa_diagnostic_query, Tabelle Info, 1151	sa_http_header_info-Systemprozedur Syntax, 1235
sa_diagnostic_request, Tabelle Info, 1153	sa_http_php_page-Systemprozedur Syntax, 1236
sa_diagnostic_statement, Tabelle Info, 1154	sa_http_php_page_interpreted-Systemprozedur Syntax, 1237
sa_diagnostic_statistics, Tabelle Info, 1155	sa_http_variable_info-Systemprozedur Syntax, 1239
sa_diagnostic_tracing_level, Tabelle Info, 1156	sa_index_density-Systemprozedur Syntax, 1240
sa_disable_auditing_type-Systemprozedur Syntax, 1211	sa_index_levels-Systemprozedur Syntax, 1243
sa_disk_free_space-Systemprozedur Syntax, 1212	sa_install_feature-Systemprozedur Syntax, 1245
sa_enable_auditing_type-Systemprozedur Syntax, 1214	sa_list_cursors-Systemprozedur Syntax, 1248
sa_eng_properties-Systemprozedur Syntax, 1215	sa_load_cost_model-Systemprozedur Syntax, 1249
sa_error_stack_trace-Systemprozedur Syntax, 1216	sa_locks-Systemprozedur Syntax, 1250
sa_event_schedules-Systemprozedur Syntax, 1218	sa_make_object-Systemprozedur Syntax, 1254
sa_external_library_unload-Systemprozedur Syntax, 1219	sa_materialized_view_can_be_immediate-Systemprozedur Ergebnisse mit sa_materialized_view_info kombinieren, 1262 Syntax, 1256
sa_flush_activity-Systemprozedur Syntax, 1220	sa_materialized_view_info-Systemprozedur AvailForOptimization-Eigenschaft, 1258 Beispiele, 1262 DataLastModified-Eigenschaft, 1258 DataStatus-Eigenschaft, 1258
sa_flush_statistics-Systemprozedur Syntax, 1220	
sa_get_bits-Systemprozedur Syntax, 1221	
sa_get_dtt-Systemprozedur Syntax, 1223	
sa_get_dtt_groupreads-Systemprozedur Syntax, 1224	

- Ergebnisse mit
- sa_materialized_view_can_be_immediate
kombinieren, 1262
- OwnerName-Eigenschaft, 1258
- Status-Eigenschaft, 1258
- Syntax, 1258
- ViewLastRefreshed-Eigenschaft, 1258
- ViewName-Eigenschaft, 1258
- sa_migrate-Systemprozedur
Syntax, 1265
- sa_migrate_create_fks-Systemprozedur
Syntax, 1268
- sa_migrate_create_remote_fks_list-Systemprozedur
Syntax, 1269
- sa_migrate_create_remote_table_list-Systemprozedur
Syntax, 1270
- sa_migrate_create_tables-Systemprozedur
Syntax, 1272
- sa_migrate_data-Systemprozedur
Syntax, 1273
- sa_migrate_drop_proxy_tables-Systemprozedur
Syntax, 1274
- sa_mirror_server_status-Systemprozedur
Syntax, 1275
- sa_nchar_terms-Systemprozedur
Syntax, 1278
- sa_performance_diagnostics-Systemprozedur
Syntax, 1279
- sa_performance_statistics-Systemprozedur
Syntax, 1285
- sa_post_login_procedure-Systemprozedur
Syntax, 1287
- sa_procedure_profile-Systemprozedur
Syntax, 1288
- sa_procedure_profile_summary-Systemprozedur
Syntax, 1290
- sa_recompile_views-Systemprozedur
Syntax, 1293
- sa_refresh_materialized_views-Systemprozedur
Syntax, 1294
- sa_refresh_text_indexes-Systemprozedur
Syntax, 1295
- sa_remove_tracing_data-Systemprozedur
Syntax, 1296
- sa_report_deadlocks-Systemprozedur
Syntax, 1296
- sa_reserved_words-Systemprozedur
Syntax, 1298
- sa_reset_identity-Systemprozedur
Syntax, 1299
- sa_rowgenerator-Systemprozedur
Syntax, 1300
- sa_save_trace_data-Systemprozedur
Syntax, 1302
- sa_send_udp-Systemprozedur
Syntax, 1303
- sa_server_messages-Systemprozedur
Syntax, 1304
- sa_server_option-Systemprozedur
Syntax, 1306
- sa_set_http_header-Systemprozedur
Syntax, 1323
- sa_set_http_option-Systemprozedur
Syntax, 1324
- sa_set_soap_header-Systemprozedur
Syntax, 1329
- sa_set_tracing_level-Systemprozedur
Syntax, 1330
- sa_snapshots-Systemprozedur
Syntax, 1331
- sa_split_list-Systemprozedur
Syntax, 1332
- sa_stack_trace-Systemprozedur
Syntax, 1335
- sa_statement_text-Systemprozedur
Syntax, 1337
- sa_table_fragmentation-Systemprozedur
Syntax, 1338
- sa_table_page_usage-Systemprozedur
Syntax, 1339
- sa_table_stats-Systemprozedur
Syntax, 1340
- sa_text_index_stats-Systemprozedur
Syntax, 1342
- sa_text_index_vocab-Systemprozedur
Syntax, 1343
- sa_text_index_vocab_nchar-Systemprozedur
Syntax, 1345
- sa_transactions-Systemprozedur
Syntax, 1347
- sa_unload_cost_model-Systemprozedur
Syntax, 1348
- sa_user_defined_counter_add-Systemprozedur
Syntax, 1349
- sa_user_defined_counter_set-Systemprozedur
Syntax, 1350

sa_validate-Systemprozedur
 Syntax,1352
 sa_verify_password-Systemprozedur
 Syntax,1353
 Sandboxing
 Datenbanken starten,1064
 SAVE OPTION VALUES-Klausel
 ALTER TEXT CONFIGURATION-
 Anweisung,534
 SAVE TRANSACTION-Anweisung
 Transact-SQL-Syntax,1018
 SAVEPOINT-Anweisung
 Syntax,1019
 Savepoint-Name
 allgemeines Element der SQL-Syntax,446
 Savepoints
 auf Savepoints zurücksetzen,1015
 erstellen,1019
 freigeben,994
 SCALE-Klausel
 GET DESCRIPTOR-Anweisung [ESQL],878
 SET DESCRIPTOR-Anweisung [ESQL],1033
 Scale-Out
 Server löschen,815
 SET MIRROR OPTION-Anweisung,1034
 Scale-Out mit Schreibschutz
 ALTER MIRROR SERVER-Anweisung,480
 CREATE MIRROR SERVER-Anweisung,656
 Server löschen,815
 SET MIRROR OPTION-Anweisung,1034
 Schaltjahre
 Info,122
 Schätzungen
 explizite Selektivitätsschätzungen,68
 SCHEDULE-Klausel
 CREATE EVENT-Anweisung,609
 Schemasperren
 sa_locks-Systemprozedur,1251
 Schemata
 erstellen,697
 Standard-Systemansichten,1437
 Systemtabellen,1129
 Schiefe
 in Indizes mit sa_index_density erkennen,1240
 Schleifendurchlauf
 über Cursor,857
 Schließen
 Cursor mit der CLOSE-Anweisung schließen
 [ESQL] [SP],571
 Interactive SQL,850
 Verbindungen mit der DROP CONNECTION-
 Anweisung,803
 Schlüssel
 CREATE SEQUENCE-Anweisung,699
 Schlüsselwörter
 Liste der reservierten Wörter,1
 sa_reserved_words-Systemprozedur,1298
 SQL, reservierte Wörter,1
 Verwendung von SQL-reservierten Wörtern in der
 Syntax,1
 Schnittmenge
 Ergebnis von mehreren SELECT-
 Anweisungen,927
 Schrägstrich-Stern
 Kommentar-Bezeichner,92
 Schreib-Prüfsummen
 validieren,1118
 Schreiben von Dateien
 mit xp_write_file,1435
 SchreibenKeinPS-Sperren
 sa_locks-Systemprozedur,1251
 Schreibgeschützt
 schreibgeschützte Spiegelserver definieren,657
 Schreibschutz
 Sicherungen, die keine Wiederherstellung
 erfordern,553
 Tabellen sperren,949
 SCRIPT VERSION-Klausel
 CREATE SYNCHRONIZATION
 SUBSCRIPTION-Anweisung [MobiLink],734
 SCROLL-Cursor
 deklarieren,778
 SCROLL-Klausel
 DECLARE CURSOR-Anweisung,780
 FOR-Anweisung,858
 SEARCH DN-Klausel
 ALTER LDAP SERVER-Anweisung,469
 CREATE LDAP SERVER-Anweisung,643
 SECOND-Funktion
 Syntax,377
 SECONDS, Sekunden
 Syntax,378
 SECURE-Klausel
 ALTER SERVICE-Anweisung [SOAP über
 HTTP],500

- ALTER SERVICE-Anweisungen [HTTP-Webdienst],494
- CREATE SERVICE-Anweisung [HTTP-Webdienst],498,710
- CREATE SERVICE-Anweisung [SOAP-Webdienst],505,717
- SecureFeatures, Eigenschaft
 - mit sa_server_option einstellen,1306
- Seitenbelegung
 - Tabellen,1339
- Seitengrößen
 - Datenbanken erstellen,589
- SELECT
 - Konvertieren von T-SQL,430
- SELECT LIST FOR-Klausel
 - DESCRIBE-Anweisung [ESQL],794
- SELECT list-Klausel
 - SELECT-Anweisung,1023
- SELECT-Anweisung
 - aus gespeicherten Prozeduren auswählen,865
 - Syntax,1020
 - Transact-SQL-Syntax,1031
- SELECT-Liste
 - Cursor beschreibende,794
- SELECT-Privileg
 - GRANT-Anweisung,883
 - REVOKE-Anweisung,1010
- SELECT-Privilegklausel
 - GRANT-Anweisung,885
- Selektivitätsschätzungen
 - benutzerdefiniert,68
 - Quelle der Schätzung,253
- self_recursion-Option
 - mit Transact-SQL SET-Anweisung setzen,1056
- SEND AT-Klausel
 - GRANT CONSOLIDATE-Anweisung [SQL Remote],889
 - GRANT REMOTE-Anweisung [SQL Remote],900
 - Info,889
- SEND EVERY-Klausel
 - GRANT CONSOLIDATE-Anweisung [SQL Remote],889
 - GRANT REMOTE-Anweisung [SQL Remote],900
 - Info,889
- Senden
 - SQL-Anweisungen an Fremdserver,861
- Senden von Datums- und Zeitangaben an die Datenbank
 - Info,117
- SENSITIVE-Klausel
 - DECLARE CURSOR-Anweisung,780
 - FOR-Anweisung,858
- Sequenz
 - SELECT-Anweisung,1028
- Sequenzen
 - ALTER SEQUENCE-Anweisung,490
 - CREATE SEQUENCE-Anweisung,699
 - CREATE TABLE-Anweisung,519,742
 - DROP SEQUENCE-Anweisung,822
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - USAGE-Privilegien entziehen,1011
 - Verwendungsprivilegien erteilen,902
- Sequenzgenerator
 - ALTER SEQUENCE-Anweisung,490
 - CREATE SEQUENCE-Anweisung,699
 - DROP SEQUENCE-Anweisung,822
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - USAGE-Privilegien entziehen,1011
- Serialisierbar
 - FROM-Klausel,869
- SERIALIZABLE, Tabellen-Hint
 - FROM-Klausel,869
- SERIALIZABLE-Klausel
 - REFRESH MATERIALIZED VIEW-Anweisung,988
- Server
 - Datenbank starten,1066
 - Datenbank stoppen,1077
 - entfernte Attribute mit der ALTER SERVER-Anweisung ändern,491
 - Ereignisse für inaktive Server mit der CREATE EVENT-Anweisung erstellen,606
 - erstellen,701
 - Fremdserver löschen ,823
 - HTTP_RESPONSE_HEADER-Funktion,286
- server-option-Klausel
 - ALTER MIRROR SERVER-Anweisung,481
 - CREATE MIRROR SERVER-Anweisung,659
- ServerIdle, Systemereignis
 - Beispiel,961
- Serveroptionen
 - mit der sa_server_option-Systemprozedur setzen,1306
- SessionTimeout-Option

-
- sa_set_http_option-Systemprozedur,1324
 - SET CONNECTION-Anweisung
 - Embedded SQL-Syntax,1032
 - Interactive SQL-Syntax,1032
 - SET DESCRIPTOR-Anweisung
 - Embedded SQL-Syntax,1033
 - SET HIDDEN-Klausel
 - ALTER EVENT-Anweisung,462
 - ALTER FUNCTION-Anweisung,465
 - ALTER MATERIALIZED VIEW-Anweisung,477
 - ALTER PROCEDURE-Anweisung,483
 - ALTER VIEW-Anweisung,544
 - SET MIRROR OPTION-Anweisung
 - Syntax,1034
 - SET OPTION-Anweisung
 - Interactive SQL-Syntax,1043
 - Syntax,1039
 - Transact-SQL-Syntax,1056
 - SET PARTNER FAILOVER-Klausel
 - ALTER DATABASE-Anweisung,453
 - SET PERMANENT-Anweisung
 - Interactive SQL-Syntax,1043
 - SET REMOTE OPTION-Anweisung
 - Dateioptionen,1048
 - FTP-Optionen,1049
 - gemeinsame Optionen,1048
 - HTTP-Optionen,1049
 - SMTP-Optionen,1051
 - SQL Remote-Syntax,1046
 - SET SCRIPT VERSION ... ON SUBSCRIPTION-Klausel
 - START SYNCHRONIZATION SCHEMA CHANGE-Anweisung,1072
 - SET SCRIPT VERSION-Klausel
 - ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink] ,513
 - START SYNCHRONIZATION SCHEMA CHANGE-Anweisung,1072
 - SET SESSION AUTHORIZATION-Anweisung
 - Syntax,1059
 - SET SQLCA-Anweisung
 - Embedded SQL-Syntax,1053
 - SET TEMPORARY OPTION-Anweisung
 - Interactive SQL-Syntax,1043
 - Syntax,1039
 - SET USER-Systemprivileg
 - GRANT-Anweisung,881
 - mit der GRANT-Anweisung erteilen,883
 - SET, Klausel
 - UPDATE-Anweisung,1111
 - SET-Anweisung
 - Interactive SQL-Syntax,1043
 - Syntax,1054
 - Transact-SQL-Syntax,1056
 - SET-Klausel
 - CREATE PROCEDURE-Anweisung [Webclients],629,675
 - Umleitung, URIs,675
 - UPDATE-Anweisung (positionsbasiert),1104
 - UPDATE-Anweisung [SQL Remote] ,1106
 - SET_BIT-Funktion
 - Syntax,379
 - SET_BITS-Funktion
 - Syntax,381
 - SETUSER
 - Benutzer einrichten,1059
 - SETUSER-Anweisung
 - Syntax,1059
 - Setzen
 - Zeilen in Cursor,981
 - SHARE BY ALL-Klausel
 - CREATE TABLE-Anweisung,740
 - Sicherheit
 - SQL Remote-Replikation,1013,1014
 - Sicherungen
 - auf Band mit der BACKUP-Anweisung,548
 - auf einem schreibgeschützten Datenbankserver starten,553
 - BACKUP-Anweisung,548
 - Datenbanken wiederherstellen,1001
 - Ereignisse mit der CREATE EVENT-Anweisung erstellen,606
 - mit der BACKUP-Anweisung erstellen,548
 - Sicherungskopien
 - mit der BACKUP-Anweisung erstellen,548
 - SIGN-Funktion
 - Syntax,381
 - SIGNAL-Anweisung
 - Syntax,1061
 - SIMILAR TO-Suchbedingung
 - Datenbankkollation und Suche nach Übereinstimmungen,57
 - reguläre Ausdrücke,28
 - Syntax,56
 - übereinstimmende Teilzeichenklassen,56
 - verglichen mit REGEXP und LIKE,48

- SIMILAR-Funktion
 - Syntax, 382
- SIN-Funktion
 - Syntax, 383
- SIZE-Klausel
 - ATTACH TRACING-Anweisung, 546
- SKIP-Klausel
 - LOAD TABLE-Anweisung, 941
 - MERGE-Anweisung, 956
- Skriptgesteuerter Upload
 - Angaben zum Verarbeitungsfortschritt in Zeitstempel umwandeln, 1191
 - CREATE PUBLICATION-Syntax, 690
 - Umwandlung von Verarbeitungsfortschrittwerten in UNSIGNED BIGINT, 1192
- SMALLDATETIME-Datentyp
 - Datums- und Zeitangaben an die Datenbank senden, 117
 - Syntax, 127
- SMALLINT UNSIGNED-Datentyp
 - Syntax, 112
- SMALLINT-Datentyp
 - mit Datums- und Uhrzeitangaben vergleichen, 143
 - Syntax, 112
- SMALLMONEY-Datentyp
 - Syntax, 114
- SMTP
 - E-Mail-Beispiel, 1429
 - E-Mail-Sitzungen starten, 1431
 - E-Mail-Sitzungen stoppen, 1434
 - erweiterte Systemprozeduren, 1162
 - Rückgabecodes, 1163
- SMTP-Klausel
 - GRANT CONSOLIDATE-Anweisung [SQL Remote], 889
- SMTP-Nachrichtentyp
 - SQL Remote-Anweisung CREATE REMOTE MESSAGE TYPE, 694
- Snapshot-Isolation
 - BEGIN SNAPSHOT-Anweisung, 556
 - mit Volltextsuche verwenden, 760
 - sa_snapshots-Systemprozedur, 1331
 - sa_transactions-Systemprozedur, 1347
- SNAPSHOT-Klausel
 - REFRESH MATERIALIZED VIEW-Anweisung, 988
- Snapshots
 - BEGIN SNAPSHOT-Anweisung, 556
- SOAP
 - CREATE SERVICE-Anweisung [SOAP-Webdienst], 501, 714
 - SOAP über HTTP-Dienste
 - CREATE SERVICE-Anweisung [SOAP-Webdienst], 713
 - SOAP, Header
 - Einstellung, 1329
 - SOAP-Dienste
 - Datentypisierung, 502, 714
 - SOAP-Funktionen
 - alphabetische Liste, 162
 - SOAP-Systemprozeduren
 - alphabetische Liste, 1162
 - SOAP_HEADER-Funktion
 - Syntax, 384
 - SOAPHEADER-Klausel
 - CREATE FUNCTION-Anweisung [Webclients], 626
 - CREATE PROCEDURE-Anweisung [Webclients], 677
- Sofortansichten
 - ALTER MATERIALIZED VIEW-Anweisung, 476
 - nicht zum Aktualisieren von Basistabellen erforderliche Privilegien, 478
- SOME-Suchbedingung
 - Syntax, 46
- Sonderzeichen
 - in Binärwerten verwendet, 6
 - in SQL-Zeichenfolgen, 8
 - in Zeichenfolgen ,8
 - zulässige Syntax in einer Volltext-Abfragezeichenfolge, 64
- Sortieren
 - SORTKEY-Funktion, 385
- Sortierschlüssel
 - mit der SORTKEY-Funktion generieren, 385
- SORTKEY-Funktion
 - Kollationsanpassung, 385
 - Syntax, 385
- SOUNDEX-Funktion
 - Syntax, 388
- SP
 - Anweisungsindikatoren, 448
- sp_addgroup-Systemprozedur
 - Adaptive Server Enterprise-Systemprozeduren, 1166
- sp_addlogin-Systemprozedur

Adaptive Server Enterprise-Systemprozeduren,1166	Adaptive Server Enterprise-Systemprozeduren,1166
sp_addmessage-Systemprozedur Adaptive Server Enterprise-Systemprozeduren,1166 Info,655	sp_droptype-Systemprozedur Adaptive Server Enterprise-Systemprozeduren,1166
sp_addtype-Systemprozedur Adaptive Server Enterprise-Systemprozeduren,1166	sp_dropuser-Systemprozedur Adaptive Server Enterprise-Systemprozeduren,1166
sp_adduser-Systemprozedur Adaptive Server Enterprise-Systemprozeduren,1166	sp_fkeys-Katalogprozedur Adaptive Server Enterprise-Katalogprozeduren,1167
sp_alter_secure_feature_key-Systemprozedur Syntax,1354	sp_forward_to_remote_server-Systemprozedur Syntax,1367
sp_auth_sys_role_info-Systemprozedur Syntax,1355	sp_get_last_synchronize_result-Systemprozedur Syntax,1368
sp_changegroup-Systemprozedur Adaptive Server Enterprise-Systemprozeduren,1166	sp_getmessage-Systemprozedur Adaptive Server Enterprise-Systemprozeduren,1166
sp_column_privileges-Katalogprozedur Adaptive Server Enterprise-Katalogprozeduren,1167	sp_has_role-Systemprozedur Syntax,1371
sp_columns-Katalogprozedur Adaptive Server Enterprise-Katalogprozeduren,1167	sp_helptext-Systemprozedur Adaptive Server Enterprise-Systemprozeduren,1166
sp_copy_directory-Systemprozedur Syntax,1356	sp_list_directory-Systemprozedur Syntax,1373
sp_copy_file-Systemprozedur Syntax,1358	sp_list_secure_feature_keys-Systemprozedur Syntax,1375
sp_create_directory-Systemprozedur Syntax,1359	sp_login_environment-Systemprozedur Syntax,1376
sp_create_secure_feature_key-Systemprozedur Syntax,1360	sp_move_directory-Systemprozedur Syntax,1377
sp_delete_directory-Systemprozedur Syntax,1361	sp_move_file-Systemprozedur Syntax,1378
sp_delete_file-Systemprozedur Syntax,1362	sp_objectpermission-Systemprozedur Syntax,1379
sp_displayroles-Systemprozedur Syntax,1363	sp_parse_json-Systemprozedur Syntax,1383
sp_drop_secure_feature_key-Systemprozedur Syntax,1366	sp_password-Systemprozedur Adaptive Server Enterprise-Systemprozeduren,1166
sp_dropgroup-Systemprozedur Adaptive Server Enterprise-Systemprozeduren,1166	sp_pkeys-Katalogprozedur Adaptive Server Enterprise-Katalogprozeduren,1167
sp_droplogin-Systemprozedur Adaptive Server Enterprise-Systemprozeduren,1166	sp_proc_priv-Systemprozedur Syntax,1384
sp_dropmessage-Systemprozedur	sp_remote_columns-Systemprozedur Syntax,1386
	sp_remote_exported_keys-Systemprozedur

- Syntax, 1388
- sp_remote_imported_keys-Systemprozedur
 - Syntax, 1390
- sp_remote_pcols-Systemprozedur
 - Syntax, 1391
- sp_remote_primary_keys-Systemprozedur
 - Syntax, 1393
- sp_remote_procedures-Systemprozedur
 - Syntax, 1395
- sp_remote_tables-Systemprozedur
 - Syntax, 1396
- sp_servercaps-Systemprozedur
 - Syntax, 1398
- sp_special_columns-Katalogprozedur
 - Adaptive Server Enterprise-Katalogprozeduren, 1167
- sp_sproc_columns-Katalogprozedur
 - Adaptive Server Enterprise-Katalogprozeduren, 1167
- sp_statistics-Katalogprozedur
 - Adaptive Server Enterprise-Katalogprozeduren, 1167
- sp_stored_procedures-Katalogprozedur
 - Adaptive Server Enterprise-Katalogprozeduren, 1167
- sp_sys_priv_role_info-Systemprozedur
 - Syntax, 1399
- sp_tables-Katalogprozedur
 - Adaptive Server Enterprise-Katalogprozeduren, 1167
- sp_trace_event_fields-Systemprozedur
 - Syntax, 1401
- sp_trace_event_session_events-Systemprozedur
 - Syntax, 1402
- sp_trace_event_session_target_options-Systemprozedur
 - Syntax, 1403
- sp_trace_event_session_targets-Systemprozedur
 - Syntax, 1405
- sp_trace_event_sessions-Systemprozedur
 - Syntax, 1406
- sp_trace_events-Systemprozedur
 - Syntax, 1407
- sp_tsql_environment-Systemprozedur
 - Syntax, 1409
- sp_use_secure_feature_key-Systemprozedur
 - Syntax, 1410
- SPACE-Funktion
 - Syntax, 389
- Spalten
 - Aktualisieren in SQL Remote, 1106
 - Aliase, 1023
 - benutzerdefinierte Datentypen, 139
 - Binärdaten abrufen, 876
 - Domänen, 139
 - Integritätsregeln in der CREATE TABLE-Anweisung, 745
 - Integritätsregeln und Standardwerte bei Domänen, 139
 - mit der ALTER TABLE-Anweisung ändern, 516
 - ohne Protokollierung aktualisieren, 1127
 - Privilegien, 1441
 - SYSTABCOL-Ansicht, 1495
 - umbenennen, 527
- Spalten-Integritätsregeln
 - mit ALTER TABLE-Anweisung ändern, 525
 - mit der ALTER TABLE-Anweisung hinzufügen, 521
- Spaltendefinition
 - CREATE TABLE-Anweisung, 740
- Spaltenkomprimierung
 - ALTER TABLE-Anweisung, 516
 - CREATE TABLE-Anweisung, 737
 - Komprimierungsstatistiken abrufen, 1177
- Spaltennamen
 - allgemeines Element der SQL-Syntax, 445
 - SQL-Syntax, 24
 - Syntax, 24
- Spaltenstatistiken
 - mit CREATE STATISTICS aktualisieren, 729
 - nur teilweise aktualisiert durch LOAD TABLE, 945
 - Selektivitätsschätzungen, 68
 - SYSCOLSTAT-Systemansicht, 1441
 - SYSCOLSTATS, konsolidierte Ansicht, 1521
- Specification-Eigenschaft
 - DB_EXTENDED_PROPERTY-Funktion, 226
- Speicher
 - für Deskriptorbereiche zuweisen, 449
- Speicherformate
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung, 508
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung, 725
- Sperren
 - anzeigen, 1250
 - Blocks, 1183

- Funktionen,948
- Tabellen,949
- Spezialansichten
 - Info,1437
- Spezialtabellen
 - Info,1129
- Spezialwerte
 - allgemeines Element der SQL-Syntax,446
 - CREATE TABLE-Anweisung,519,742
 - CURRENT DATABASE,70
 - CURRENT DATE,70
 - CURRENT PUBLISHER,71
 - CURRENT REMOTE USER,71
 - CURRENT TIME,72
 - CURRENT TIMESTAMP,73
 - CURRENT USER,74
 - CURRENT UTC TIMESTAMP,75
 - CURRENT_TIMESTAMP,73
 - CURRENT_USER,74
 - Info,70
 - LAST USER,76
 - NULL,76
 - SQLCODE,79
 - SQLSTATE,80
 - TIMESTAMP,82
 - USER,83
 - UTC TIMESTAMP,84
- Spezielle Zeichenklassen
 - reguläre Ausdrücke,31
- Spiegelserver
 - ALTER MIRROR SERVER-Anweisung,480
 - CREATE MIRROR SERVER-Anweisung,656
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - mit der ALTER MIRROR SERVER-Anweisung ändern,480
 - mit der CREATE MIRROR SERVER-Anweisung definieren,657
 - mit der CREATE MIRROR SERVER-Anweisung erstellen ,656
 - mit der DROP MIRROR SERVER-Anweisung löschen,815
- Spiegelung
 - Server löschen,815
- Sprachelemente
 - Ausdrücke,22
 - benannte Parameter,93
 - Bezeichner,4
 - Kommentare,92
 - Konstanten,6
 - Operatoren,9
 - Schlüsselwörter,1
 - Spezialwerte,70
 - Suchbedingungen,42
 - Syntax,1
 - Variablen,85
 - Zeichenfolgen,6
- SQL
 - alphabetische Liste der SQL Anywhere Server-Anweisungen,449
- SQL Anywhere Native
 - ODBC-Treiber-Referenz,703
- SQL Flagger
 - eine SQL-Anweisung auf Nicht-Kern-Erweiterungen testen,1168
 - SQLFLAGGER-Funktion,390
- SQL in Java, Datentypkonvertierung
 - Info,151
- SQL Remote
 - Artikel, SYSARTICLE,1438
 - Artikel, SYSARTICLECOL,1438
 - entfernte Optionen einstellen,1046
 - Subskriptionen erstellen,730
 - SYSARTICLE-Systemansicht,1438
 - SYSARTICLECOL-Systemansicht,1438
 - SYSPUBLICATION-Systemansicht,1473
 - SYSPUBLICATIONS, konsolidierte Ansicht,1526
 - SYSREMOTEOPTION-Systemansicht,1475
 - SYSREMOTEOPTIONS, konsolidierte Ansicht,1527
 - SYSREMOTEOPTIONTYPE-Systemansicht,1475
 - SYSREMOTETYPE-Systemansicht,1476
 - SYSREMOTETYPES, konsolidierte Ansicht,1527
 - SYSREMOTEUSER-Systemansicht,1476
 - SYSREMOTEUSERS, konsolidierte Ansicht,1528
- SQL Remote-Replikationsprivilegien
 - entziehen,1013,1014
- SQL Remote-Systemansichten
 - Artikel-Systemansicht,1438
 - SYSARTICLECOL,1438
 - SYSPUBLICATION-Systemansicht,1473
 - SYSPUBLICATIONS, konsolidierte Ansicht,1526
 - SYSREMOTEOPTION,1475
 - SYSREMOTEOPTIONS, konsolidierte Ansicht,1527
 - SYSREMOTEOPTIONTYPE,1475

- SYSREMOETYPES, konsolidierte Ansicht,1527
- SYSREMOEUSER ,1476
- SYSREMOEUSERS, konsolidierte Ansicht,1528
- SQL SECURITY-Klausel
 - ALTER PROCEDURE-Anweisung,483,681
 - CREATE FUNCTION-Anweisung [benutzerdefiniert],634
 - CREATE FUNCTION-Anweisung [externer Aufruf],619
 - CREATE PROCEDURE-Anweisung,684
 - CREATE PROCEDURE-Anweisung [externer Aufruf],665
- SQL Server
 - nach SQL Anywhere mit sa_migrate-Systemprozedur migrieren,1266
- SQL-Anweisungen
 - alphabetische Liste der SQL Anywhere Server-Anweisungen,449
 - an Fremdserversenden,861
 - aus Dateien lesen,984
 - Java-Klassen installieren,925
 - Konventionen in der Dokumentation,445
- SQL-Deskriptorbereiche
 - DESCRIBE-Anweisung [ESQL],794
 - INCLUDE-Anweisung,909
 - Zeilen mit Cursor einfügen,981
- SQL-Funktionen
 - Aggregat,153
 - benutzerdefinierte,158
 - Bilddaten,166
 - Bit-Array,155
 - Datentypenvertierung,156
 - Datum und Zeit,156
 - Einführung,153
 - Funktionstypen,153
 - HTTP,162
 - NULL zurückgeben, wenn NULL-Argument angegeben wird,153
 - numerische,161
 - Rangfolge,155
 - SOAP,162
 - System,165
 - Text,166
 - verschiedene,160
 - Zeichenfolge,163
- SQL-Schlüsselwörter
 - sa_reserved_words-Systemprozedur,1298
- SQL-Skriptdateien
 - Parameter für Interactive SQL,974
 - SQL-Anweisungen lesen,984
- SQL-Sprachelemente
 - Info,1
- SQL-Standards
 - Konformität testen,390
- SQL-Syntax
 - ALL-Suchbedingung,45
 - alphabetische Liste der SQL Anywhere Server-Anweisungen,449
 - alphabetische Liste der Systemprozeduren,1168
 - ANY-Suchbedingung,46
 - arithmetische Operatoren,11
 - Array-Operatoren,19
 - Ausdrücke,22
 - benannte Parametern,93
 - BETWEEN-Suchbedingung,48
 - Bezeichner,4
 - Bit-Operatoren,20
 - CASE-Ausdruck,25
 - CONTAINS-Suchbedingung,59
 - CURRENT DATABASE-Spezialwert,70
 - CURRENT DATE-Spezialwert,70
 - CURRENT PUBLISHER-Spezialwert,71
 - CURRENT REMOTE USER-Spezialwert,71
 - CURRENT TIME-Spezialwert,72
 - CURRENT TIMESTAMP-Spezialwert,73
 - CURRENT USER-Spezialwert,74
 - CURRENT UTC TIMESTAMP-Spezialwert,75
 - CURRENT_TIMESTAMP-Spezialwert,73
 - CURRENT_USER-Spezialwert,74
 - Drei-Werte-Logik,67
 - EXISTS-Suchbedingung,66
 - Funktionen,153
 - IF-Ausdrücke,24
 - IN-Suchbedingung,58
 - IS NOT NULL-Suchbedingung,67
 - IS NULL-Suchbedingung,67
 - IS TRUE- oder FALSE-Suchbedingungen,67
 - Kommentare,92
 - Konstanten ,6
 - Konstanten in Ausdrücken,24
 - Konventionen in der Dokumentation,445
 - LAST USER-Spezialwert,76
 - LIKE-Suchbedingung,51
 - logische Operatoren,10
 - lokale Variablen,86
 - NULL,76

- Operator-Vorrang,21
- Operatoren,9
- Prädikate,42
- REGEXP-Suchbedingung,55
- Schlüsselwörter,1
- SIMILAR TO-Suchbedingung,56
- SOME-Suchbedingung,46
- Spaltennamen,24
- Spezialwerte,70
- SQLCODE-Spezialwert,79
- SQLSTATE-Spezialwert,80
- Suchbedingungen,42
- TIMESTAMP-Spezialwert,82
- Transact-SQL-Ausdruckskompatibilität,41
- Unterabfragen,24
- Unterabfragen in Suchbedingungen,44
- USER-Spezialwert,83
- UTC TIMESTAMP-Spezialwert,84
- Variablen,85
- Verbindungsebenen-Variablen,87
- Vergleichsoperatoren,9
- Zeichenfolgen,6
- Zeichenfolgenoperatoren,11
- SQL-Variable
 - Werte einstellen,1054
- SQL-Variablen
 - deklarieren,786
 - erstellen,771
 - mit der DROP VARIABLE-Anweisung löschen,839
- SQL/1992
 - Testen der SQL-Konformität,390
- SQL/1999
 - Testen der SQL-Konformität,390
- SQL/2003
 - Testen der SQL-Konformität,390
- SQL/2008
 - Testen der SQL-Konformität,390
- SQLCA
 - einstellen,1053
 - INCLUDE-Anweisung,909
- SQLCODE
 - Spezialwert,79
- SQLDA
 - DESCRIBE-Anweisung [ESQL],794
 - einstellen,1033
 - EXECUTE-Anweisung,846
 - INCLUDE-Anweisung,909
 - Informationen abrufen,878
 - Speicher zuweisen,449
 - UPDATE-Anweisung (positionsbasiert),1103
 - Zeilen mit Cursor einfügen,981
 - Zuweisungen aufheben,777
- SQLDescribeCol
 - Beschreiben der CHAR-Datentypen,95
 - Beschreiben der NCHAR-Datentypen,98
 - Beschreiben der NVARCHAR-Datentypen,99
 - Beschreiben der VARCHAR-Datentypen,101
- SQLDIALECT-Funktion
 - Syntax,390
- SQLERROR-Klausel
 - WHENEVER-Anweisung [ESQL],1122
- SQLFLAGGER-Funktion
 - Syntax,390
- SQLSetConnectAttr
 - mit MESSAGE TO CLIENT verwenden,962
- SQLSTATE
 - Konformität mit ISO/ANSI-Standard,80
 - Spezialwert,80
- SQLWARNING-Klausel
 - WHENEVER-Anweisung [ESQL],1122
- SQRT-Funktion
 - Syntax,392
- SRIDs
 - SRID bei der Erstellung eines räumlichen Bezugssystems auswählen,720
- ST_GEOMETRY_COLUMNS
 - konsolidierte Ansicht,1514
- st_geometry_dump-Systemprozedur
 - Syntax,1411
- st_geometry_load_shapefile-Systemprozedur
 - Syntax,1416
- ST_SPATIAL_REFERENCE_SYSTEMS
 - konsolidierte Ansicht,1515
- ST_UNITS_OF_MEASURE
 - konsolidierte Ansicht,1518
- STACK_TRACE-Funktion
 - Syntax,392
- Standard-Login-Richtlinie
 - Info,647
- Standardabweichung
 - STDDEV-Funktion,393
 - STDDEV_POP-Funktion,393
 - STDDEV_SAMP-Funktion,395
- Standardwerte
 - CREATE TABLE-Anweisung,741

- CURRENT REMOTE USER,71
- CURRENT_TIMESTAMP,73
- CURRENT_USER,74
 - in Domänen verwenden,600
- INSERT-Anweisung,918
 - mit DEFAULT für optionale Parameter,564
- SQLCODE,79
- SQLSTATE,80
- TIMESTAMP,82
- USER,83
- UTC TIMESTAMP,84
- Starke Verschlüsselung
 - CREATE DATABASE-Anweisung,588
- START AT-Klausel
 - DELETE-Anweisung,791
 - SELECT-Anweisung,1022
 - UPDATE-Anweisung,1111
- START DATABASE-Anweisung
 - Syntax,1062
- START DATE-Klausel
 - ALTER EVENT-Anweisung,462
 - CREATE EVENT-Anweisung,610
- START EXTERNAL ENVIRONMENT-Anweisung
 - Syntax,1066
- START JAVA-Anweisung
 - Syntax,1067
- START LOGGING-Anweisung
 - Interactive SQL-Syntax,1068
- START SERVER-Anweisung
 - Interactive SQL-Syntax,1066
- START SYNCHRONIZATION DELETE-Anweisung
 - MobiLink-Syntax,1071
- START SYNCHRONIZATION SCHEMA CHANGE-Anweisung
 - MobiLink-Syntax,1072
- START TIME-Klausel
 - ALTER EVENT-Anweisung,462
 - CREATE EVENT-Anweisung,609
- START WITH-Klausel
 - CREATE SEQUENCE-Anweisung,699
- Starten
 - Datenbanken,1062
 - Datenbankserver,1066
 - Durchreichmodus,975
 - Ereignisse mit der CREATE EVENT-Anweisung erstellen,606
 - externe Umgebungen mit der START EXTERNAL ENVIRONMENT-Anweisung,1066
 - Java VM, mit der START JAVA-Anweisung,1067
 - Protokollierung in Interactive SQL,1068
 - SQL Remote-Subskriptionen während der Datenbankextraktion,995
 - Subskriptionen,1069
 - state_file-Option
 - ALTER MIRROR SERVER-Anweisung,482
 - CREATE MIRROR SERVER-Anweisung,659
 - Statische Cursor
 - deklarieren,778
 - STATISTICS-Klausel
 - LOAD TABLE-Anweisung,943
 - StatisticsCleaner-Eigenschaft
 - mit sa_server_option einstellen,1306
 - Statistiken
 - auf Festplatte leeren,1220
 - CREATE STATISTICS-Anweisung,729
 - laden,930
 - mit der ALTER SERVICE-Anweisung aktualisieren,510
 - mit der DROP STATISTICS-Anweisung löschen,827
 - mit sa_get_histogram-Systemprozedur abrufen,1225
 - nur teilweise aktualisiert durch LOAD TABLE,945
 - SYSCOLSTAT-Systemansicht,1441
 - Statistikenaufräumvorgang
 - mit sa_server_option steuern,1306
 - Status-Eigenschaft
 - sa_materialized_view_info-Systemprozedur,1258
 - Statusinformationsdateien
 - ALTER MIRROR SERVER-Anweisung,482
 - CREATE MIRROR SERVER-Anweisung,659
 - STDDEV-Funktion
 - Syntax,393
 - STDDEV_POP-Funktion
 - Syntax,393
 - STDDEV_SAMP-Funktion
 - Syntax,395
 - Sternchen
 - in Unterabfragen verwenden,774
 - zulässige Syntax in einer CONTAINS-Klausel,63
 - zulässige Syntax in einer Volltext-Abfragezeichenfolge,63
 - Steueranweisungen
 - BREAK, Syntax,563
 - CALL-Anweisung,564
 - CASE-Anweisung,566

CASE-Anweisung [T-SQL],568
 CONTINUE-Anweisung, Syntax,581
 GOTO, Transact-SQL-Anweisung,881
 IF-Anweisung,906
 LEAVE-Anweisung,929
 LOOP-Anweisung,950
 Transact-SQL CONTINUE-Anweisung,1123
 Transact-SQL IF-Anweisung,908
 Transact-SQL WHILE-Anweisung,1123
 WHILE-Anweisung,950
 Stichproben-Kovarianz
 Info,210
 Stichprobenvarianz
 Info,427
 STOP DATABASE-Anweisung
 Syntax,1074
 STOP DATABASE-Klausel
 STOP DATABASE-Anweisung,1074
 STOP EXTERNAL ENVIRONMENT-Anweisung
 Syntax,1075
 STOP JAVA-Anweisung
 Syntax,1076
 STOP LOGGING-Anweisung
 Interactive SQL-Syntax,1077
 STOP SERVER-Anweisung
 Syntax,1077
 STOP SUBSCRIPTION-Anweisung
 SQL Remote-Syntax,1078
 STOP SYNCHRONIZATION DELETE-Anweisung
 MobiLink-Syntax,1079
 STOP SYNCHRONIZATION SCHEMA CHANGE-Anweisung
 MobiLink-Syntax,1085
 STOP-Klausel
 WHENEVER-Anweisung [ESQL],1122
 STOPLIST-Klausel
 ALTER TEXT CONFIGURATION-Anweisung,532
 Stoppen
 Datenbankserver,1077
 Durchreichmodus,975
 externe Umgebungen mit der STOP EXTERNAL ENVIRONMENT-Anweisung,1075
 Java VM,1076
 Protokollierung in Interactive SQL,1077
 Stopplisten
 CREATE STOPLIST-Klausel,532
 STOPLIST-Klausel,532
 STORAGE FORMAT-Klausel
 ALTER SPATIAL REFERENCE SYSTEM-Anweisung,508
 CREATE SPATIAL REFERENCE SYSTEM-Anweisung,725
 STR-Funktion
 Syntax,397
 STRING-Funktion
 Syntax,398
 string_truncation-Option
 mit Transact-SQL SET-Anweisung setzen,1056
 STRIP-Klausel
 LOAD TABLE-Anweisung,942
 STRTOUUID-Funktion
 Syntax,399
 STUFF-Funktion
 Syntax,400
 SUBDIRS-Klausel
 CREATE SERVER-Anweisung,703
 SUBSCRIBE BY-Klausel
 ALTER PUBLICATION-Anweisung,485
 CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote],692
 UPDATE-Anweisung,1109
 UPDATE-Anweisung [SQL Remote] ,1106
 Subskriptionen
 ALTER SYNCHRONIZATION
 SUBSCRIPTION-Anweisung,512
 CREATE SUBSCRIPTION-Anweisung [SQL Remote],730
 CREATE SYNCHRONIZATION
 SUBSCRIPTION-Anweisung,733
 DROP SUBSCRIPTION-Anweisung,828
 DROP SYNCHRONIZATION SUBSCRIPTION-Anweisung,830
 SQL Remote REMOTE RESET-Anweisung,995
 SQL Remote UPDATE-Anweisung,1108
 SQL Remote-Erstellung,730
 START SUBSCRIPTION-Anweisung [SQL Remote],1069
 STOP SUBSCRIPTION-Anweisung [SQL Remote],1078
 SYNCHRONIZE SUBSCRIPTION-Anweisung [SQL Remote],1086
 UPDATE-Anweisung,1113
 Subskriptionen stoppen
 STOP SUBSCRIPTION-Anweisung,1078
 Subskriptionen, synchronisieren

- SYNCHRONIZE SUBSCRIPTION-Anweisung
 - [SQL Remote],1086
- SUBSTR-Funktion
 - Syntax,401
- SUBSTRING-Funktion
 - Syntax,401
- Suchbedingungen
 - ALL,45
 - allgemeines Element der SQL-Syntax,446
 - ANY,46
 - BETWEEN,48
 - CONTAINS,59
 - Drei-Werte-Logik,67
 - EXISTS,66
 - explizite Selektivitätsschätzungen,68
 - IN,58
 - Info,42
 - IS DISTINCT FROM,47
 - IS NOT DISTINCT FROM,47
 - IS NOT NULL,67
 - IS NULL,67
 - IS TRUE- oder FALSE-Suchbedingungen,67
 - IS UNKNOWN-Suchbedingung,67
 - LIKE,51
 - REGEXP,55
 - SIMILAR TO,56
 - SOME,46
 - Syntax,42
 - Unterabfragen in,44
 - Wahrwert,67
- SUM-Funktion
 - Syntax,403
- SUSER_ID-Funktion
 - Syntax,404
- SUSER_NAME-Funktion
 - Syntax,405
- SWITCHOFFSET-Funktion
 - Syntax,406
- synchronization_mode-Option
 - SET MIRROR OPTION-Anweisung,1035
- SYNCHRONIZE SUBSCRIPTION-Anweisung
 - SQL Remote-Syntax,1086
- SYNCHRONIZE-Anweisung
 - MobiLink-Syntax,1081
- Syntax
 - arithmetische Operatoren,11
 - Array-Operatoren,19
 - benannte Parameter,93
 - Bit-Operatoren,20
 - CASE-Ausdruck,25
 - CURRENT DATABASE-Spezialwert,70
 - CURRENT DATE-Spezialwert,70
 - CURRENT PUBLISHER-Spezialwert,71
 - CURRENT TIMESTAMP-Spezialwert,73
 - CURRENT USER-Spezialwert,74
 - CURRENT UTC TIMESTAMP-Spezialwert,75
 - CURRENT_TIMESTAMP-Spezialwert,73
 - CURRENT_USER-Spezialwert,74
 - Drei-Werte-Logik,67
 - IF-Ausdrücke,24
 - IS NULL-Suchbedingung,67
 - IS TRUE- oder FALSE-Suchbedingungen,67
 - Kommentare,92
 - Konstanten ,6
 - Konstanten in Ausdrücken,24
 - Konventionen,446
 - Konventionen in der Dokumentation,445
 - LAST USER-Spezialwert,76
 - LEN-Funktion,301
 - Liste der SQL-reservierten Wörter,1
 - logische Operatoren,10
 - lokale Variablen,86
 - NULL,76
 - Prädikate,42
 - Spezialwerte,70
 - SQL CURRENT REMOTE USER-Spezialwert,71
 - SQL CURRENT TIME-Spezialwert,72
 - SQL-Anweisungen,449
 - SQL-Ausdrücke,22
 - SQL-Bezeichner,4
 - SQL-Funktionen,153
 - SQL-Operator-Vorrang,21
 - SQL-Operatoren,9
 - SQL-Schlüsselwörter,1
 - SQL-Unterabfragen,24
 - SQL-Unterabfragen in Suchbedingungen,44
 - SQL-Variablen,85
 - SQLCODE-Spezialwert,79
 - SQLSTATE-Spezialwert,80
 - Suchbedingungen,42
 - Testen der Konformität mit einem Standard,390
 - TIMESTAMP-Spezialwert,82
 - Transact-SQL-Ausdruckskompatibilität,41
 - USER-Spezialwert,83
 - UTC TIMESTAMP-Spezialwert,84
 - Verbindungsebenen-Variablen,87

Vergleichsoperatoren,9
 Zeichenfolgen,6
 Zeichenfolgenoperatoren,11
 Syntaxkonventionen
 SQL-Anweisungen,446
 SYS
 Standard-Systemansichten,1437
 Systemtabellen,1129
 SYS-Systemrolle
 mit der GRANT-Anweisung erteilen,882
 mit der REVOKE-Anweisung entziehen,1009
 SYS_AUTH_BACKUP_ROLE-Kompatibilitätsrolle
 Kompatibilitätsrollen mit der GRANT-Anweisung erteilen,882
 Kompatibilitätsrollen mit der REVOKE-Anweisung entziehen,1009
 SYS_AUTH_DBA_ROLE-Kompatibilitätsrolle
 Kompatibilitätsrollen mit der GRANT-Anweisung erteilen,882
 Kompatibilitätsrollen mit der REVOKE-Anweisung entziehen,1009
 SYS_AUTH_PROFILE_ROLE-Kompatibilitätsrolle
 Kompatibilitätsrollen mit der GRANT-Anweisung erteilen,882
 Kompatibilitätsrollen mit der REVOKE-Anweisung entziehen,1009
 SYS_AUTH_READCLIENTFILE_ROLE-Kompatibilitätsrolle
 Kompatibilitätsrollen mit der GRANT-Anweisung erteilen,882
 Kompatibilitätsrollen mit der REVOKE-Anweisung entziehen,1009
 SYS_AUTH_READFILE_ROLE-Kompatibilitätsrolle
 Kompatibilitätsrollen mit der GRANT-Anweisung erteilen,882
 Kompatibilitätsrollen mit der REVOKE-Anweisung entziehen,1009
 SYS_AUTH_RESOURCE_ROLE-Kompatibilitätsrolle
 Kompatibilitätsrollen mit der GRANT-Anweisung erteilen,882
 Kompatibilitätsrollen mit der REVOKE-Anweisung entziehen,1009
 SYS_AUTH_SA_ROLE-Kompatibilitätsrolle
 Kompatibilitätsrollen mit der GRANT-Anweisung erteilen,882
 Kompatibilitätsrollen mit der REVOKE-Anweisung entziehen,1009
 SYS_AUTH_SSO_ROLE-Kompatibilitätsrolle
 Kompatibilitätsrollen mit der GRANT-Anweisung erteilen,882
 Kompatibilitätsrollen mit der REVOKE-Anweisung entziehen,1009
 SYS_AUTH_VALIDATE_ROLE-Kompatibilitätsrolle
 Kompatibilitätsrollen mit der GRANT-Anweisung erteilen,882
 Kompatibilitätsrollen mit der REVOKE-Anweisung entziehen,1009
 SYS_AUTH_WRITECLIENTFILE_ROLE-Kompatibilitätsrolle
 Kompatibilitätsrollen mit der GRANT-Anweisung erteilen,882
 Kompatibilitätsrollen mit der REVOKE-Anweisung entziehen,1009
 SYS_AUTH_WRITEFILE_ROLE-Kompatibilitätsrolle
 Kompatibilitätsrollen mit der GRANT-Anweisung erteilen,882
 Kompatibilitätsrollen mit der REVOKE-Anweisung entziehen,1009
 SYS_REPLICATION_ADMIN_ROLE-Systemrolle
 erteilen,896
 mit der GRANT-Anweisung erteilen,882
 mit der REVOKE-Anweisung entziehen,1009
 SYS_RUN_REPLICATION_ROLE-Systemrolle
 erteilen,898
 mit der GRANT-Anweisung erteilen,882
 mit der REVOKE-Anweisung entziehen,1009
 SYS_SAMONITOR_ROLE-Systemrolle
 mit der GRANT-Anweisung erteilen,882
 mit der REVOKE-Anweisung entziehen,1009
 SYS_SPATIAL_ADMIN_ROLE-Systemrolle
 mit der GRANT-Anweisung erteilen,882
 mit der REVOKE-Anweisung entziehen,1009
 SYSARTICLE
 Systemansicht,1438
 SYSARTICLECOL
 Systemansicht,1438
 SYSARTICLECOLS
 konsolidierte Ansicht,1518
 SYSARTICLES
 konsolidierte Ansicht,1519
 SYSCAPABILITIES

- konsolidierte Ansicht,1519
- SYSCAPABILITY
 - Systemansicht,1439
- SYSCAPABILITYNAME
 - Systemansicht,1439
- SYSCATALOG
 - konsolidierte Ansicht,1520
- SYSCERTIFICATE
 - Systemansicht,1440
 - Zertifikate hinzufügen,582
- SYSCHECK
 - Systemansicht,1440
- SYSCOLAUTH
 - konsolidierte Ansicht,1520
- SYSCOLLATION
 - Kompatibilitätsansicht (nicht mehr empfohlen),1535
- SYSCOLLATIONMAPPINGS
 - Kompatibilitätsansicht (nicht mehr empfohlen),1535
- SYSCOLPERM
 - Systemansicht,1441
- SYSCOLSTAT
 - Systemansicht,1441
- SYSCOLSTATS
 - konsolidierte Ansicht,1521
- SYSCOLUMN
 - Kompatibilitätsansicht (nicht mehr empfohlen),1536
- SYSCOLUMNS
 - konsolidierte Ansicht,1522
- syscolumns
 - Ansicht mit Eigentümer dbo,1546
- syscomments
 - Ansicht mit Eigentümer dbo,1546
- SYSCONSTRAINT
 - Systemansicht,1442
- SYSDATETIMEOFFSET-Funktion
 - Syntax,406
- SYSDATABASE
 - Systemansicht,1443
- SYSDATABASESPACE
 - Systemansicht,1444
- SYSDATABASESPACEPERM
 - Systemansicht,1445
- SYSDependency
 - Systemansicht,1445
- SYSDOMAIN
 - Systemansicht,1446
- SYSEVENT
 - Systemansicht,1446
- SYSEVENTTYPE
 - Systemansicht,1448
- SYSEXTERNENV
 - Systemansicht,1448
- SYSEXTERNENVOBJECT-
 - Systemansicht,1450
- SYSEXTERNLOGIN
 - Systemansicht,1451
- SYSFILE
 - Kompatibilitätsansicht (nicht mehr empfohlen),1536
- SYSFKCOL
 - Kompatibilitätsansicht (nicht mehr empfohlen),1537
- SYSFKEY
 - Systemansicht,1452
- SYSFOREIGNKEY
 - Kompatibilitätsansicht (nicht mehr empfohlen),1537
- SYSFOREIGNKEYS
 - konsolidierte Ansicht,1522
- SYSGROUP
 - Kompatibilitätsansicht,1538
- SYSGROUPS
 - Kompatibilitätsansicht,1539
- SYSHISTORY
 - Systemansicht,1453
- SYSIDX
 - Systemansicht,1456
- SYSIDXCOL
 - Systemansicht,1457
- SYSINDEX
 - Kompatibilitätsansicht (nicht mehr empfohlen),1539
- SYSINDEXES
 - konsolidierte Ansicht,1523
- sysindexes
 - Ansicht mit Eigentümer dbo,1546
- SYSINFO
 - Kompatibilitätsansicht (nicht mehr empfohlen),1540
- SYSIXCOL
 - Kompatibilitätsansicht (nicht mehr empfohlen),1540
- SYSJAR

Systemansicht,1458
SYSJARCOMPONENT
Systemansicht,1459
SYSJAVACLASS
Systemansicht,1460
SYSLDAPSERVER
Systemansicht,1461
SYSLOGINMAP
Systemansicht,1462
SYSLOGINPOLICY
Systemansicht,1463
SYSLOGINPOLICYOPTION
Systemansicht,1463
syslogins
Ansicht mit Eigentümer dbo,1546
SYSMIRROROPTION
Systemansicht,1463
SYSMIRRORSERVER
Systemansicht,1464
SYSMIRRORSERVEROPTION
Systemansicht,1465
SYSMVOPTION
Systemansicht,1465
SYSMVOPTIONNAME
Systemansicht,1466
SYSOBJECT
Systemansicht,1466
sysobjects
Ansicht mit Eigentümer dbo,1546
SYSOPTION
Systemansicht,1467
SYSOPTIONS
konsolidierte Ansicht,1524
SYSOPTSTAT
Systemansicht,1468
SYSPHYSIDX
Systemansicht,1468
SYSPROCAUTH
konsolidierte Ansicht,1524
SYSPROCEDURE
Systemansicht,1469
SYSPROCPARM
Systemansicht,1470
SYSPROCPARMS
konsolidierte Ansicht,1525
SYSPROCPERM
Systemansicht,1472
SYSPROCS

konsolidierte Ansicht,1525
SYSPROXYTAB
Systemansicht,1472
SYSPUBLICATION
Systemansicht,1473
SYSPUBLICATIONS
konsolidierte Ansicht,1526
SYSREMARK
Systemansicht,1474
SYSREMOTEOPTION
Systemansicht,1475
SYSREMOTEOPTION2
konsolidierte Ansicht,1526
SYSREMOTEOPTIONS
konsolidierte Ansicht,1527
SYSREMOTEOPTIONTYPE
Systemansicht,1475
SYSREMOTETYPE
Systemansicht,1476
SYSREMOTETYPES
konsolidierte Ansicht,1527
SYSREMOTEUSER
Systemansicht,1476
SYSREMOTEUSERS
konsolidierte Ansicht,1528
SYSROLEGRANT
Systemansicht,1478
SYSROLEGRANTTEXT
Systemansicht,1479
SYSROLEGRANTS
Systemansicht,1480
SYSSCHEDULE
Systemansicht,1481
SYSSEQUENCE
Systemansicht,1483
SYSSEQUENCEPERM
Systemansicht,1483
SYSSERVER
Systemansicht,1484
SYSSOURCE
Systemansicht,1485
SYSSPATIALREFERENCESYSTEM
Systemansicht,1485
SYSSQLSERVERTYPE
Systemansicht,1488
SYSSUBSCRIPTION
Systemansicht,1489
SYSSUBSCRIPTIONS

- konsolidierte Ansicht,1528
- SYSSYNC
 - Systemansicht,1489
- SYSSYNC2
 - konsolidierte Ansicht,1529
- SYSSYNCPROFILE
 - Systemansicht,1491
- SYSSYNCPUBLICATIONDEFAULTS
 - konsolidierte Ansicht,1529
- SYSSYNCS
 - konsolidierte Ansicht,1530
- SYSSYNCSCRIPT
 - Systemansicht,1491
- SYSSYNCSCRIPTS
 - konsolidierte Ansicht,1530
- SYSSYNCSUBSCRIPTIONS
 - konsolidierte Ansicht,1531
- SYSSYNCUSERS
 - konsolidierte Ansicht,1532
- SYSTAB
 - Kompatibilität, Ansichtsinformationen,1129
 - Systemansicht,1492
- SYSTABAUTH
 - konsolidierte Ansicht,1532
- SYSTABCOL
 - Systemansicht,1495
- SYSTABLE
 - Kompatibilitätsansicht (nicht mehr empfohlen),1541
- SYSTABLEPERM
 - Systemansicht,1498
- SYSTEM PROCEDURE AS DEFINER-Klausel
 - ALTER DATABASE-Anweisung,452
 - CREATE DATABASE-Anweisung,590
- SYSTEM PROCEDURE AS-Klausel
 - ALTER DATABASE-Anweisung,451
 - CREATE DATABASE-Anweisung,681
- System- und Katalog-Systemprozeduren
 - Info,1168
- SYSTEM-Anweisung
 - Interactive SQL-Syntax,1088
 - nicht unterstützt von dbisqlc,1088
- Systemansichten
 - Aktualisierungen,1437
 - alphabetische Liste von Systemansichten,1437
 - anzeigen,1437
 - Info,1437
 - SYSARTICLE,1438
 - SYSARTICLECOL,1438
 - SYSCAPABILITY,1439
 - SYSCAPABILITYNAME,1439
 - SYSCERTIFICATE,1440
 - SYSCHECK,1440
 - SYSCOLPERM,1441
 - SYSCOLSTAT,1441
 - SYSCONSTRAINT,1442
 - SYSDBFILE,1443
 - SYSDBSpace,1444
 - SYSDBSpacePERM,1445
 - SYSDEPENDENCY,1445
 - SYSDOMAIN,1446
 - SYSEVENT,1446
 - SYSEVENTTYPE,1448
 - SYSEXTERNENV,1448
 - SYSEXTERNENVOBJECT,1450
 - SYSEXTERNLOGIN,1451
 - SYSFILE ,1536
 - SYSFKEY,1452
 - SYSHISTORY,1453
 - SYSIDX,1456
 - SYSIDXCOL,1457
 - SYSJAR,1458
 - SYSJARCOMPONENT,1459
 - SYSJAVACLASS,1460
 - SYSLDAPSERVER,1461
 - SYSLOGINMAP,1462
 - SYSLOGINPOLICY,1463
 - SYSLOGINPOLICYOPTION,1463
 - SYSMIRROROPTION,1463
 - SYSMIRRORSERVER,1464
 - SYSMIRRORSERVEROPTION,1465
 - SYSMVOPTION,1465
 - SYSMVOPTIONNAME,1466
 - SYSOBJECT,1466
 - SYSOPTION,1467
 - SYSOPTSTAT,1468
 - SYSPHYSIDX,1468
 - SYSPROCEDURE,1469
 - SYSPROCPARM,1470
 - SYSPROCPERM,1472
 - SYSPROXYTAB,1472
 - SYSPUBLICATION,1473
 - SYSREMARK,1474
 - SYSREMOTEOPTION,1475
 - SYSREMOTEOPTIONTYPE,1475
 - SYSREMOOTETYPE,1476

SYSREMOTEUSER,1476
 SYSROLEGRANT,1478
 SYSROLEGRANTEXT,1479
 SYSROLEGRANTS,1480
 SYSSCHEDULE,1481
 SYSSEQUENCE,1483
 SYSSEQUENCEPERM,1483
 SYSSERVER,1484
 SYSSOURCE,1485
 SYSSPATIALREFERENCESYSTEM,1485
 SYSSQLSERVERTYPE,1488
 SYSSUBSCRIPTION,1489
 SYSSYNC,1489
 SYSSYNCPROFILE,1491
 SYSSYNCSRIPT,1491
 SYSTAB,1492
 SYSTABCOL,1495
 SYSTABLEPERM,1498
 SYSTEXTCONFIG,1499
 SYSTEXTIDX,1501
 SYSTEXTIDXTAB,1502
 SYSTRIGGER,1503
 SYSTYPEMAP,1505
 SYSUNITOFMEASURE,1505
 SYSUSER,1506
 SYSUSERMESSAGE,1510
 SYSUSERTYPE,1510
 SYSVIEW,1511
 SYSWEBSERVICE,1513
 Systemaufrufe
 von gespeicherten Prozeduren,1418
 xp_cmdshell-Systemprozedur,1418
 Systemfunktionen
 Kompatibilität,165
 Systemkatalog
 Ansichten,1437
 Tabellen,1129
 Systemprivilegien
 mit der GRANT-Anweisung erteilen,883
 mit der REVOKE-Anweisung entziehen,1010
 Systemprozeduren
 Adaptive Server Enterprise-
 Systemprozeduren,1166
 alphabetische Liste ,1168
 benannte Parameter beim Aufrufen zulässig,564
 Definitionen anzeigen,1161
 HTTP,1162
 Info,1161
 Liste der erweiterten,1162
 Meldungen erstellen,655
 sa_java_loaded_classes,1247
 SOAP,1162
 Sybase Central,1161
 Transact-SQL,1166
 Transact-SQL-Liste,1166
 Systemrollen
 mit der GRANT-Anweisung erteilen,882
 mit der REVOKE-Anweisung entziehen,1009
 nur MANAGE ROLES kann verwalten,883
 Systemtabellen
 Aktualisierungen,1129
 Ansichten,1129
 DUMMY,1129
 Info,1129
 Java,1159
 RowGenerator,1159
 Systemtabellendaten anzeigen,1129
 SYSTEXTCONFIG
 Systemansicht,1499
 SYSTEXTIDX
 Systemansicht,1501
 SYSTEXTIDXTAB
 Systemansicht,1502
 SYSTRIGGER
 Systemansicht,1503
 SYSTRIGGERS
 konsolidierte Ansicht,1533
 SYSTYPEMAP
 Systemansicht,1505
 systypes
 Ansicht mit Eigentümer dbo,1546
 SYSUNITOFMEASURE
 Systemansicht,1505
 SYSUSER
 Systemansicht,1506
 SYSUSERAUTH
 Kompatibilitätsansicht (nicht mehr
 empfohlen),1542
 SYSUSERAUTHORITY
 Kompatibilitätsansicht (nicht mehr
 empfohlen),1543
 SYSUSERLIST
 Kompatibilitätsansicht (nicht mehr
 empfohlen),1544
 SYSUSERMESSAGE
 Systemansicht,1510

SYSUSEROPTIONS

- konsolidierte Ansicht,1534

SYSUSERPERM

- Kompatibilitätsansicht (nicht mehr empfohlen),1544

SYSUSERPERMS

- Kompatibilitätsansicht (nicht mehr empfohlen),1545

sysusers

- Ansicht mit Eigentümer dbo,1546

SYSUSERTYPE

- Systemansicht,1510

SYSVIEW

- Systemansicht,1511

SYSVIEWS

- konsolidierte Ansicht,1534
- konsolidierte Ansichtsinformationen,1129

SYSWEBSERVICE

- Systemansicht,1513

T

Tabellen

- Abhängigkeiten ermitteln,1202
- Aktualisieren in SQL Remote,1106
- ALTER TABLE-Anweisung,516
- CREATE TABLE-Anweisung,737
- Daten aus Dateien importieren,910
- Daten mit UNLOAD-Anweisung entladen,1098
- Definition mit sa_get_table_definition-Systemprozedur generieren,1232
- exportieren, Daten in Dateien,968
- Kommentare mit der COMMENT-Anweisung hinzufügen,573
- kürzen,1089
- lokale temporäre erstellen,784
- lokale temporäre Tabellen mit der CREATE LOCAL TEMPORARY TABLE-Anweisung erstellen,645
- Massendateien laden,931
- mit der ALTER TABLE-Anweisung ändern,516
- mit der CREATE ENCRYPTED DATABASE-Anweisung verschlüsseln,601
- mit der DROP TABLE-Anweisung löschen,832
- neu organisieren,998
- Proxytabellen mit der CREATE EXISTING TABLE-Anweisung erstellen,613
- sperrern,949

- umbenennen,527

- Zeilen einfügen,917

Tabellen erstellen

- CREATE TABLE-Anweisung,737

Tabellen, neu organisieren

- REORGANIZE TABLE,998

Tabellen-Hints

- FROM-Klausel,869

Tabellen-Integritätsregeln

- CREATE TABLE-Anweisung,745
- mit ALTER TABLE-Anweisung ändern,527
- mit der ALTER TABLE-Anweisung hinzufügen,524
- mit der ALTER TABLE-Anweisung hinzufügen, löschen oder ändern,516

Tabellenentschlüsselung

- ALTER TABLE-Anweisung,516

Tabellenindizes

- in Interactive SQL auflisten,798

Tabellenliste

- FROM-Klausel,865

Tabellenlisten

- allgemeines Element der SQL-Syntax,446

Tabellennamen

- allgemeines Element der SQL-Syntax,446

Tabellennummer

- Systemansichten,1492

Tabellenseiten

- PCTFREE mit der ALTER TABLE-Anweisung setzen,516
- PCTFREE mit der CREATE LOCAL TEMPORARY TABLE-Anweisung festlegen,645
- PCTFREE mit der LOAD TABLE-Anweisung setzen,931
- PCTFREE setzen,737

Tabellenspalten

- in Interactive SQL auflisten,798

Tabellensperren

- sa_locks-Systemprozedur,1251

Tabellenverschlüsselung

- ALTER TABLE-Anweisung,516
- CREATE ENCRYPTED TABLE DATABASE-Anweisung,601

TABLE-Klausel

- ALTER PUBLICATION-Anweisung,485
- CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote],690
- DESCRIBE-Anweisung,798

TRUNCATE-Anweisung,1089
 VALIDATE-Anweisung,1118
 TABLE.COLUMN-Klausel
 DESCRIBE-Anweisung [ESQL],794
 Tag der Woche
 DOW-Funktion,240
 TAN-Funktion
 Syntax,407
 Teilzeichenfolgen
 ersetzen,365
 Info,401
 Teilzeichenklassen
 REGEXP-Suchbedingung,55
 reguläre Ausdrücke,31
 SIMILAR TO-Suchbedingung,56
 TempFreePercent-Ereignisbedingung
 Info,254
 TempFreeSpace-Ereignisbedingung
 Info,254
 Temporär
 DBSpace,452
 Temporäre Dateien
 verfügbaren Platz bestimmen,1212
 Temporäre Funktionen
 Info,634
 Temporäre gespeicherte Prozeduren
 erstellen,682
 Temporäre Optionen
 einstellen in Interactive SQL,1043
 SET OPTION-Anweisung,1039
 Temporäre Prozeduren
 CREATE PROCEDURE-Anweisung,681,682
 Temporäre Tabellen
 CREATE TABLE-Anweisung,737
 CREATE TABLE-Verwendung,749
 lokal nicht zulässige Ansichten,774
 lokale deklarieren,784
 lokale temporäre Dateien mit der CREATE
 LOCAL TEMPORARY TABLE-Anweisung
 erstellen,645
 Transact-SQL CREATE TABLE-Anweisung,750
 TEMPORARY, Schlüsselwort
 CREATE FUNCTION-Anweisung
 [benutzerdefiniert],634
 TEMPORARY-Klausel
 ALTER DBSPACE-Anweisung ,458
 CREATE PROCEDURE-Anweisung,681
 TempSize-Ereignisbedingung
 Info,254
 TERM BREAKER EXTERNAL NAME-Klausel
 ALTER TEXT CONFIGURATION-
 Anweisung,532
 TERM BREAKER-Klausel
 ALTER TEXT CONFIGURATION-
 Anweisung,533
 Text
 aus der Datenbank mit der READTEXT-
 Anweisung lesen,986
 TEXT-Datentyp
 Syntax,100
 Textfunktionen
 Info,166
 Textindizes
 aktualisieren mit REFRESH TEXT INDEX,991
 aktualisieren mit sa_refresh_text_indexes,1295
 ALTER TEXT INDEX-Anweisung,536
 auflisten mit sa_text_index_stats,1342
 CREATE TEXT INDEX-Anweisung,759
 in SYSMVOPTION gespeicherte
 Erstellungsoptionen,1465
 Kommentare mit der COMMENT-Anweisung
 hinzufügen,573
 kürzen,1091
 löschen,834
 validieren,1118
 Textkonfigurationsobjekte
 ALTER TEXT CONFIGURATION-
 Anweisung,532
 CREATE TEXT CONFIGURATION-
 Anweisung,757
 Kommentare mit der COMMENT-Anweisung
 hinzufügen,573
 löschen,833
 mit sa_char_terms die Aufteilung von
 Zeichenfolgen in Begriffe testen,1172
 mit sa_nchar_terms die Aufteilung von
 Zeichenfolgen in Begriffe testen,1278
 TEXTPTR-Funktion
 Syntax,408
 textsize-Option
 mit Transact-SQL SET-Anweisung setzen,1056
 THEN
 IF-Ausdrücke,24
 THEN-Klausel
 IF-Anweisung,906
 TIME-Datentyp

- Datums- und Zeitangaben an die Datenbank senden,117
- mit Datums- und Uhrzeitangaben vergleichen,143
- Syntax,128
- TIME-Klausel
 - WAITFOR-Anweisung,1121
- TIMESTAMP WITH TIME ZONE-Datentyp
 - Datums- und Zeitangaben an die Datenbank senden,117
 - mit Datums- und Uhrzeitangaben vergleichen,143
 - Syntax,131
- TIMESTAMP-Datentyp
 - Datums- und Zeitangaben an die Datenbank senden,117
 - mit Datums- und Uhrzeitangaben vergleichen,143
 - Syntax,129
- TIMESTAMP-Klausel
 - CREATE TABLE-Anweisung,737
- TIMESTAMP-Spezialwert
 - Syntax,82
- TIMESTAMP-Standardwert
 - CREATE TABLE-Anweisung,519,742
- TINYINT UNSIGNED-Datentyp
 - Syntax,112
- TINYINT-Datentyp
 - Syntax,112
- TLS Klausel
 - CREATE LDAP SERVER-Anweisung,644
- TLS-Klausel
 - ALTER LDAP SERVER-Anweisung,470
- TO-Klausel
 - ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink] ,513
 - CONNECT-Anweisung,578
 - CREATE EXTERNLOGIN-Anweisung,616
 - CREATE SUBSCRIPTION-Anweisung [SQL Remote],730
 - CREATE SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink],733
 - DROP EXTERNLOGIN-Anweisung,809
 - DROP SYNCHRONIZATION SUBSCRIPTION-Anweisung,830
 - GRANT CONSOLIDATE-Anweisung [SQL Remote],889
 - MESSAGE-Anweisung,960
 - PASSTHROUGH-Anweisung [SQL Remote],975
 - START SUBSCRIPTION-Anweisung [SQL Remote],1069
 - STOP SUBSCRIPTION-Anweisung (SQL Remote),1078
 - SYNCHRONIZE SUBSCRIPTION-Anweisung (SQL Remote),1086
 - UNLOAD-Anweisung,1099
- TO_CHAR-Funktion
 - Syntax,409
- TO_NCHAR-Funktion
 - Syntax,410
- TODATETIMEOFFSET-Funktion
 - Syntax,411
- TODAY-Funktion
 - Syntax,411
- TOLERANCE-Klausel
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung,508
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung,724
- Toleranz
 - für räumliche Berechnungen ändern,508
 - für räumliche Berechnungen einstellen,724
- TOP-Klausel
 - DELETE-Anweisung,791
 - SELECT-Anweisung,1022
 - UPDATE-Anweisung,1111
- TRACEBACK-Funktion
 - Syntax,412
- TRACED_PLAN-Funktion
 - Syntax,413
- Transact-SQL
 - alphabetische Liste der SQL Anywhere Server-Anweisungen,449
 - ANSI-Äquivalenz,368
 - Anweisungsindikatoren,448
 - benutzerdefinierte Datentypen,140
 - Bit-Operatoren,20
 - CASE-Anweisung [T-SQL],568
 - CONTINUE-Anweisung, Syntax,1123
 - CREATE FUNCTION-Anweisung,638
 - CREATE MESSAGE-Anweisung,655
 - CREATE PROCEDURE-Anweisung,679
 - CREATE SCHEMA-Anweisung, Syntax,697
 - CREATE TABLE-Anweisung, Syntax,750
 - CREATE TRIGGER-Anweisung, Syntax,769
 - Datum/Uhrzeit-Kompatibilität,145
 - DECLARE CURSOR-Anweisung, Syntax,782
 - Domänen,140
 - EXECUTE-Anweisung, Syntax,848

- gespeicherte Prozeduren konvertieren,430
- GOTO-Anweisung, Syntax,881
- IF-Anweisung, Syntax,908
- Katalogprozeduren,1167
- Konstanten,41
- Konvertieren von SELECT-Anweisungen,430
- lokale Variablen,86
- Outer-Join-Operatoren,21
- PRINT-Anweisung, Syntax,980
- quoted_identifizier-Option,41
- READTEXT-Anweisung, Syntax,986
- SELECT-Anweisung,1031
- SET OPTION-Anweisung, Syntax,1056
- SET-Anweisung, Syntax,1056
- SQL-Ausdruckskompatibilität,41
- System- und Katalogprozeduren,1166
- Systemfunktionen,165
- Systemprozeduren,1166
- Uhrzeit-Kompatibilität,145
- Vergleichsoperatoren,9
- Währungsdatentypen,113
- WHILE-Anweisung, Syntax,1123
- WRITETEXT-Anweisung, Syntax,1127
- Zeichenfolgen,41
- Transact-SQL - Konvertierung von Zeichenfolgen in Datums-/Zeitdatentypen
 - Info,145
- Transact-SQL, Kompatibilität
 - Ansichten,1546
 - globale Variablen,88
- Transact-SQL-Anweisungen
 - BEGIN TRANSACTION-Syntax,561
 - ROLLBACK TRANSACTION-Syntax,1016
 - SAVE TRANSACTION-Syntax,1018
- transaction isolation level-Option
 - mit Transact-SQL SET-Anweisung setzen,1056
- TRANSACTION LOG ONLY-Klausel
 - BACKUP-Anweisung,550
- TRANSACTION LOG RENAME [MATCH]-Klausel
 - BACKUP-Anweisung,550
- TRANSACTION LOG TRUNCATE-Klausel
 - BACKUP-Anweisung,550
- TRANSACTION LOG-Klausel
 - CREATE DATABASE-Anweisung,590
- TRANSACTSQL-Funktion
 - Syntax,413
- Transaktionen
 - benutzerdefinierte mit der BEGIN TRANSACTION-Anweisung beginnen,561
 - benutzerdefinierte Transaktionen mit der BEGIN TRANSACTION-Anweisung verschachteln,561
 - mit der COMMIT-Anweisung festschreiben,575
 - mit der ROLLBACK TO SAVEPOINT-Anweisung zurücksetzen,1015
 - mit der ROLLBACK-Anweisung zurücksetzen,1015
 - Savepoints erstellen,1019
 - zurücksetzen,1016
 - zurücksetzen und die SAVE TRANSACTION-Anweisung,1018
- Transaktionslog
 - mit der BACKUP-Anweisung sichern,548
 - mit der CREATE DECRYPTED DATABASE-Anweisung entschlüsseln,594
 - mit der CREATE DECRYPTED FILE-Anweisung entschlüsseln,596
 - mit der CREATE ENCRYPTED DATABASE-Anweisung verschlüsseln,601
 - mit der CREATE ENCRYPTED FILE-Anweisung verschlüsseln,604
 - ohne Sicherung umbenennen,550
 - Plattenspeicher mit ALTER DBSPACE zuweisen,457
 - TRUNCATE TABLE-Anweisung,1090
 - verfügbaren Platz bestimmen,1212
- Transaktionslog-Spiegeldatei
 - verfügbaren Platz bestimmen,1212
- Transaktionslogspiegel
 - mit der CREATE DECRYPTED DATABASE-Anweisung entschlüsseln,594
 - mit der CREATE DECRYPTED FILE-Anweisung entschlüsseln,596
 - mit der CREATE ENCRYPTED DATABASE-Anweisung verschlüsseln,601
 - mit der CREATE ENCRYPTED FILE-Anweisung verschlüsseln,604
- Transaktionsmodi
 - unkettet,561
 - verkettet,561
- Transaktionsverwaltung
 - BEGIN TRANSACTION-Anweisung,561
 - in Transact-SQL,561
 - Transact-SQL,575
- Transfer-Encoding
 - HTTP_RESPONSE_HEADER-Funktion,286

- TRANSFORM DEFINITION-Klausel
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung, 507
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung, 721
- TRANSLOG-Klausel
 - ALTER DBSPACE-Anweisung , 458
- TREAT-Funktion
 - Syntax, 414
- Trennen von Verbindungen
 - DROP CONNECTION-Anweisung, 803
- Trennkommatalisten
 - LIST-Funktion, Syntax, 302
- Trigger
 - @@identity, globale Variable, 92
 - DELETING-Bedingung, 762
 - erstellen in Transact-SQL, 769
 - INSERTING-Bedingung, 762
 - Kommentare mit der COMMENT-Anweisung hinzufügen, 573
 - mit der ALTER TRIGGER-Anweisung ändern, 539
 - mit der CREATE TRIGGER-Anweisung erstellen, 762
 - mit der DROP FUNCTION-Anweisung löschen, 809
 - mit der DROP-Anweisung löschen, 837
 - TRUNCATE TABLE-Anweisung, 1089
 - UPDATING-Bedingung, 762
 - Vorgangsbedingungen, 762
 - Zeilebene, 765
 - zurücksetzen, 1017
- TRIGGER EVENT-Anweisung
 - Syntax, 1088
- Trigger-auslösend
 - Ereignisse, 1088
- TRIM-Funktion
 - Syntax, 415
- TRIM_ARRAY-Funktion
 - Syntax, 416
- TRUE, Bedingungen
 - Drei-Werte-Logik, 67
 - IS TRUE-Suchbedingungen, 67
- TRUNCATE MATERIALIZED VIEW-Anweisung
 - Syntax, 1089
- TRUNCATE TABLE-Anweisung
 - Syntax, 1089
- TRUNCATE TEXT INDEX-Anweisung
 - Syntax, 1091
- TRUNCATE-Anweisung
 - Syntax, 1089
- TRUNCATE-Funktion
 - Syntax, 417
- TRUNCATE-Privileg
 - GRANT-Anweisung, 883
 - REVOKE-Anweisung, 1010
 - TRUNCATE-Anweisung, 1089
- TRUNCATE-Privilegklausel
 - GRANT-Anweisung, 885
- TRUNCNUM-Funktion
 - Syntax, 417
- TRY-Anweisung
 - Syntax, 1093
- TSEQUAL-Funktion
 - Syntax, 418
- TYPE PLANAR-Klausel
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung, 508
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung, 722
- TYPE ROUND EARTH-Klausel
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung, 508
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung, 722
- TYPE-Klausel
 - ALTER SERVICE-Anweisung [SOAP über HTTP], 500
 - ALTER SERVICE-Anweisungen [HTTP-Webdienst], 494
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung, 508
 - ALTER SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink] , 513
 - ALTER SYNCHRONIZATION USER-Anweisung [MobiLink] , 515
 - CREATE EVENT-Anweisung, 608
 - CREATE FUNCTION-Anweisung [Webclients], 626
 - CREATE PROCEDURE-Anweisung [Webclients], 672
 - CREATE SERVICE-Anweisung [HTTP-Webdienst], 495, 707
 - CREATE SERVICE-Anweisung [SOAP-Webdienst], 501, 714
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung, 722

-
- CREATE SPATIAL UNIT OF MEASURE-Anweisung, 727
 - CREATE SYNCHRONIZATION SUBSCRIPTION-Anweisung [MobiLink], 734
 - CREATE SYNCHRONIZATION USER, 736
 - GET DESCRIPTOR-Anweisung [ESQL], 878
 - GRANT CONSOLIDATE-Anweisung [SQL Remote], 889
 - GRANT REMOTE-Anweisung [SQL Remote], 900
 - MESSAGE-Anweisung, 959
 - SET DESCRIPTOR-Anweisung [ESQL], 1033
- ## U
- Übereinstimmungstypen
 - referenzielle Integrität, 747
 - Übergeordnete Datentypen
 - Info, 140
 - Überlauffehler
 - AVG-Funktion, 176
 - SUM-Funktion, 404
 - Überwachungsliste
 - mit sa_server_option konfigurieren, 1306
 - UCASE-Funktion
 - Syntax, 419
 - UDF
 - benutzerdefinierte Funktion, Definition, 158
 - UDP-Pakete
 - senden, 1303
 - Uhrzeit
 - speichern, 116
 - Uhrzeiten
 - Abfragen, 122
 - an die Datenbank senden, 117
 - vergleichen, 143
 - UltraLite
 - NULL zurückgeben, wenn NULL-Argument angegeben wird, 153
 - Umbenennen
 - Integritätsregeln, 527
 - Spalten, 527
 - Spalten mit der ALTER TABLE-Anweisung, 516
 - Tabellen, 527
 - Tabellen mit der ALTER TABLE-Anweisung, 516
 - Umgebungen
 - mit der ALTER EXTERNAL ENVIRONMENT-Anweisung ändern, 463
 - Umgebungsvariable
 - xp_getenv-Systemprozedur, 1421
 - Umgebungsvariablen
 - abrufen, 1421
 - Umgehung der Optimierung
 - mit FORCE NO OPTIMIZATION-Klausel, 1028
 - vermeiden mit FORCE OPTIMIZATION-Option, 1027
 - Unabhängige Variable
 - Regressionszeile, 351
 - UNBOUNDED, Schlüsselwort
 - PRECEDING-Klausel der WINDOW-Klausel, 1126
 - UNBOUNDED-Schlüsselwort
 - FOLLOWING-Klausel der WINDOW-Klausel, 1126
 - UNCONDITIONALLY-Klausel
 - STOP DATABASE-Anweisung, 1074
 - STOP SERVER-Anweisung, 1077
 - Ungleich
 - Vergleichsoperator, 9
 - Unicode
 - Escapesequenzen, 421
 - Funktion UNISTR, 421
 - UNICODE Funktion, 420
 - Unicode-Daten
 - Speicherung, 95
 - Unicode-Datentypen
 - Info, 95
 - UNICODE-Funktion
 - Syntax, 420
 - UNION-Anweisung
 - Datenbankoptionen einstellen, 1097
 - Syntax, 1096
 - Unions
 - mehrere SELECT-Anweisungen, 1096
 - UNIQUE-Klausel
 - ALTER TABLE-Anweisung, 522
 - CREATE INDEX-Anweisung, 639
 - CREATE TABLE-Anweisung, 747
 - DECLARE CURSOR-Anweisung, 779
 - UNIQUEIDENTIFIER-Datentyp
 - Syntax, 134
 - UNIQUEIDENTIFIERSTR-Datentyp
 - Syntax, 101
 - UNISTR-Funktion
 - Syntax, 421
 - Universally Unique Identifiers (universell eindeutige Bezeichner)

- SQL-Syntax für die NEWID-Funktion,321
- Unix
 - Zeichenfolgen dekomprimieren,233
 - Zeichenfolgen komprimieren,194
- UNKNOWN, Bedingungen
 - IS UNKNOWN-Suchbedingungen,67
- UNLIMITED-Richtlinien
 - CREATE LOGIN POLICY-Anweisung,647
- UNLOAD MATERIALIZED VIEW-Anweisung
 - Syntax,1098
- UNLOAD TABLE-Anweisung
 - Syntax,1098
- UNLOAD-Anweisung
 - Syntax,1098
- UNNEST-Operator
 - Syntax,19
- UNSIGNED BIGINT-Datentyp
 - Syntax,104
- UNSIGNED INTEGER-Datentyp
 - Syntax,109
- UNSIGNED SMALLINT-Datentyp
 - Syntax,112
- UNSIGNED TINYINT-Datentyp
 - Syntax,112
- Unterabfragen
 - in SQL-Suchbedingungen,44
 - mit NULL auswerten, wenn keine übereinstimmenden Zeilen,24
- unzip, Dienstprogramm
 - DECOMPRESS-Funktion,233
- UPDATE SET-Klausel
 - MERGE-Anweisung,956
- UPDATE-Anweisung
 - Datenbankoptionen einstellen,1112
 - SQL Remote-Syntax,1106
 - Syntax,1109
- UPDATE-Klausel
 - CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote],690
 - CREATE TRIGGER-Anweisung,762
 - INSTALL EXTERNAL OBJECT-Anweisung,923
- UPDATE-Privileg
 - GRANT-Anweisung,883
 - REVOKE-Anweisung,1010
- UPDATE-Privilegklausel
 - GRANT-Anweisung,885
- UPDATE-Spaltenprivileg
- SYSCOLPERM-Systemansicht,1441
- UPDATEAnweisung
 - (positionsbasierte) Anweisung, Syntax,1103
- UPDLOCK, Tabellen-Hint
 - FROM-Klausel,869
- Upgrade von Datenbanken
 - ALTER DATABASE-Anweisung,451
- UPPER-Funktion
 - Syntax,422
- URI-Umleitung
 - CREATE PROCEDURE-Anweisung (Webclients),675
- URL-Klausel
 - ALTER LDAP SERVER-Anweisung,469
 - ALTER SERVICE-Anweisungen [HTTP-Webdienst],494
 - CREATE FUNCTION-Anweisung [Webclients],626
 - CREATE LDAP SERVER-Anweisung,643
 - CREATE PROCEDURE-Anweisung [Webclients],672
 - CREATE SERVICE-Anweisung [HTTP-Webdienst],495,707
- Ursprüngliches Speicherformat
 - ALTER SPATIAL REFERENCE SYSTEM-Anweisung,508
 - CREATE SPATIAL REFERENCE SYSTEM-Anweisung,725
- USE DEFAULT-Klausel
 - CREATE TABLE-Anweisung,737
- USER
 - CONNECT-Anweisung,578
- USER TYPES-Klausel
 - DESCRIBE-Anweisung,795
- User-Agent
 - HTTP_HEADER-Funktion,284
 - HTTP_RESPONSE_HEADER-Funktion,286
- USER-Klausel
 - ALTER SERVICE-Anweisung [SOAP über HTTP],500
 - ALTER SERVICE-Anweisungen [HTTP-Webdienst],494
 - CREATE SERVICE-Anweisung [HTTP-Webdienst],498,710
 - CREATE SERVICE-Anweisung [SOAP-Webdienst],505,717
 - CREATE TABLE-Anweisung,737
- USER-Spezialwert

- Syntax,83
- user_estimates, Option
 - Einstellung für INSERT-Anweisungen,920
 - Einstellung für UPDATE-Anweisungen,1112
 - in einer MERGE-Anweisung außer Kraft setzen,957
- user_estimates-Option
 - Einstellung für DELETE-Anweisungen,793
 - für die EXCEPT-Klausel setzen,842
 - für INTERSECT-Klausel setzen,927
 - für UNION-Klausel setzen,1097
- USER_ID-Funktion
 - Syntax,423
- USER_NAME-Funktion
 - Syntax,423
- USING AUTO PARENT-Klausel
 - ALTER MIRROR SERVER-Anweisung,481
 - CREATE MIRROR SERVER-Anweisung,658
- USING CLIENT FILE-Klausel
 - LOAD TABLE-Anweisung,933
- USING COLUMN-Klausel
 - LOAD TABLE-Anweisung,934
- USING DESCRIPTOR-Klausel
 - EXPLAIN-Anweisung [ESQL],851
 - GET DATA-Anweisung ,876
 - GET OPTION-Anweisung [ESQL],880
 - OPEN-Anweisung,965
 - PUT-Anweisung [ESQL],981
 - UPDATE-Anweisung (positionsbasiert),1104
- USING FILE-Klausel
 - LOAD TABLE-Anweisung,933
- USING VALUE-Klausel
 - LOAD TABLE-Anweisung,933
- USING-Klausel
 - ALTER PUBLICATION-Anweisung,485
 - ALTER SERVER-Anweisung,492
 - CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote],690
 - CREATE SERVER-Anweisung,702
 - DECLARE CURSOR-Anweisung,780
 - EXECUTE-Anweisung,846
 - FOR-Anweisung,857
 - INPUT-Anweisung,915
 - MERGE-Anweisung,954
 - OUTPUT-Anweisung,971
- UTC TIMESTAMP-Klausel
 - CREATE TABLE-Anweisung,737
- UTC TIMESTAMP-Spezialwert

- Syntax,84
- UTC TIMESTAMP-Standardwert
 - CREATE TABLE-Anweisung,519,742
- UTF-16-Kodierung
 - CSCONVERT-Funktion,211
 - LOAD TABLE-Anweisung,935,937
 - UNLOAD-Anweisung,1099
- UUIDs
 - SQL-Syntax für die NEWID-Funktion,321
 - SQL-Syntax für die STRTOUUID-Funktion,399
 - SQL-Syntax für die UUIDTOSTR-Funktion,424
 - UNIQUEIDENTIFIER-Datentyp,134
- UUIDTOSTR-Funktion
 - Syntax,424

V

- VALIDATE CHECKSUM-Anweisung
 - Syntax,1118
- VALIDATE DATABASE-Anweisung
 - Syntax,1118
- VALIDATE INDEX-Anweisung
 - Syntax,1118
- VALIDATE LDAP SERVER-Anweisung
 - Syntax,1116
- VALIDATE MATERIALIZED VIEW-Anweisung
 - Syntax,1118
- VALIDATE TABLE-Anweisung
 - Syntax,1118
- VALIDATE TEXT INDEX-Anweisung
 - Syntax,1118
- VALIDATE-Anweisung
 - Syntax,1118
- Validieren
 - Indizes mit VALIDATE-Anweisung,1118
 - LDAP-Serverkonfigurationsobjekte,1116
 - Prüfsummen,1118
 - sa_validate-Systemprozedur,1352
 - VALIDATE-Anweisung,1118
 - von Tabellen mit der VALIDATE TABLE-Anweisung,1118
- VALUE-Klausel
 - GET DESCRIPTOR-Anweisung [ESQL],878
 - SET DESCRIPTOR-Anweisung [ESQL],1033
- VALUES-Klausel
 - INSERT-Anweisung,918
- VAR_POP-Funktion
 - Syntax,425

- VAR_SAMP-Funktion
 - Syntax, 427
- VARBINARY-Datentyp
 - Syntax, 135
- VARBIT-Datentyp
 - Syntax, 115
- VARCHAR-Datentyp
 - Bytelänge-Semantik, 101
 - DESCRIBE bei einer VARCHAR-Spalte, 101
 - Syntax, 101
 - Zeichenlängensemantik, 101
- VAREXISTS-Funktion
 - Syntax, 429
- Variable
 - Anfangswert, 787
 - SQL-Variablen mit der DROP VARIABLE-Anweisung löschen, 839
 - Werte einstellen, 1054
- Variable Ergebnismengen
 - aus Prozeduren, CREATE PROCEDURE-Anweisung, 683
 - aus Prozeduren, CREATE PROCEDURE-Anweisung [externer Aufruf], 663
 - aus Prozeduren, DESCRIBE-Anweisung [ESQL], 796
 - von Prozeduren, PREPARE-Anweisung, 977
- Variablen
 - globale Variablen, 88
 - in Ansichtsdefinitionen verwenden, 774
 - lokale Variablen, 86
 - mit DECLARE-Anweisung deklarieren, 786
 - SQL erstellen, 771
 - Syntax, 85
 - Verbindungsebenen-Variablen, 87
 - von innerhalb eines Deskriptorbereichs abrufen, 878
- Variablennamen
 - allgemeines Element der SQL-Syntax, 446
- VARIANCE-Funktion
 - Syntax, 429
- Verarbeitung
 - RAISERROR-Anweisung, 982
- Verbinden
 - Datenbanken mit der CONNECT-Anweisung, 578
- Verbindung herstellen
 - Ereignisse mit der CREATE EVENT-Anweisung erstellen, 606
- Verbindung trennen
 - Ereignisse mit der CREATE EVENT-Anweisung erstellen, 606
- Verbindungen
 - Cursorliste, 1248
 - DROP CONNECTION-Anweisung, 803
 - einstellen, 1032
 - Ereignisse für fehlgeschlagene Verbindungen erstellen, 606
 - in Interactive SQL löschen, 802
 - Liste der Verbindungs-IDs erstellen, 1186
 - maximale Anzahl definieren, 982
 - mit RAISERROR verbieten, 982
 - Pool aktivieren, 1059
 - Verbindungen zu einer Datenbank deaktivieren, 1306
- Verbindungen deaktivieren
 - zu allen Datenbanken auf einem Server, 1306
 - zu einzelnen Datenbanken, 1306
- Verbindungen trennen
 - DROP CONNECTION-Anweisung, 803
- Verbindungsebenen-Variablen
 - Definition, 85
 - Syntax, 87
- Verbindungsnamen
 - allgemeines Element der SQL-Syntax, 445
- VERBOSE-Klausel
 - OUTPUT-Anweisung, 971
- Verfolgen
 - Fehler in Embedded SQL, 1122
- Vergleiche
 - CHAR- und NCHAR-Werte, 141
 - Datumsangaben, 143
 - numerische Datentypen, 142
 - Suchbedingungen, 42
 - Zeitangaben, 143
 - zusammengesetzte Datentypen, 145
- Vergleichen
 - CHAR und NCHAR, 141
 - COMPARE-Funktion, 192
- Vergleichsoperatoren
 - Datenkonvertierung, 140
 - Symbole, 9
 - Syntax, 9
 - Transact-SQL-Kompatibilität, 9
- Verifizieren
 - Kennwörter, 1353
- VERIFY-Klausel
 - UPDATE-Anweisung, 1109

-
- UPDATE-Anweisung [SQL Remote] ,1106
 - Verlustreiche Konvertierung
 - Info,140
 - Verschachteln
 - benutzerdefinierte Transaktionen mit der BEGIN TRANSACTION-Anweisung,561
 - Verschlüsseln
 - Datenbanken, CREATE ENCRYPTED DATABASE-Anweisung,601
 - materialisierte Ansichten mit Sybase Central,476
 - Verschlüsselung
 - CREATE DATABASE-Anweisung,588
 - CREATE DECRYPTED DATABASE-Anweisung,594
 - CREATE ENCRYPTED FILE-Anweisung,604
 - Verschlüsselung, Tabellen
 - ALTER TABLE-Anweisung,516
 - Verschlüsselungsalgorithmus
 - CREATE DATABASE-Anweisung,588
 - Versionsnummer
 - abrufen,1422
 - Verzeichniszugriffsserver
 - CREATE SERVER-Anweisung,701
 - Verzweigen
 - MERGE-Anweisung,957
 - Verzweigungen
 - MERGE-Anweisung,957
 - ViewLastRefreshed-Eigenschaft
 - sa_materialized_view_info-Systemprozedur,1258
 - ViewName-Eigenschaft
 - sa_materialized_view_info-Systemprozedur,1258
 - VIRTUAL-Klausel
 - CREATE INDEX-Anweisung,638
 - VM
 - START JAVA-Anweisung,1067
 - STOP JAVA-Anweisung,1076
 - Volltextsuche
 - ALTER TEXT CONFIGURATION-Anwendung,532
 - ALTER TEXT INDEX-Anweisung,536
 - Bindestrich, zulässige Syntax,63
 - CONTAINS-Klausel in der FROM-Klausel,863
 - CONTAINS-Suchbedingung,59
 - CREATE TEXT CONFIGURATION-Anweisung,757
 - CREATE TEXT INDEX-Anweisung,759
 - DROP TEXT CONFIGURATION-Anweisung,833
 - DROP TEXT INDEX-Anweisung,834
 - REFRESH TEXT INDEX-Anwendung,991
 - sa_char_terms-Systemprozedur,1172
 - sa_nchar_terms-Systemprozedur,1278
 - sa_refresh_text_indexes-Systemprozedur,1295
 - sa_text_index_stats-Systemprozedur,1342
 - sa_text_index_vocab-Systemprozedur,1343
 - sa_text_index_vocab_nchar-Systemprozedur,1345
 - Snapshot-Isolation-Kompatibilität,760
 - Sternchen, zulässige Syntax,63
 - Syntax für Sonderzeichen,64
 - TRUNCATE TEXT INDEX-Anwendung,1091
 - Vorrang von Operatoren,62
 - Warnung bei Verwendung von nicht-alphanumerischen Zeichen in der Abfragezeichenfolge,61
 - Vorbereiten
 - für Zwei-Phasen-Commit,979
 - Vorbereitete Anweisungen
 - ausführen,846
 - mit der DROP STATEMENT-Anweisung löschen,826
 - Vorfilter
 - ALTER TEXT CONFIGURATION-Anweisung,534
 - mit der ALTER TEXT CONFIGURATION-Anweisung löschen,534
 - Vorrang
 - SQL-Operator-Vorrang,21
 - Vorrang von Operatoren
 - Volltextsuche,62
 - W**
 - Währung
 - Währungsdatentypen,113
 - Währungsdatentypen
 - MONEY,113
 - SMALLMONEY,114
 - WAIT AFTER END-Klausel
 - BACKUP-Anweisung,550
 - WAIT BEFORE START-Klausel
 - BACKUP-Anweisung,550
 - WAITFOR-Anweisung
 - Syntax,1120
 - Wald
 - in nicht syntaktisch analysiertem XML-Dokument,438
 - Watcom-SQL
-

- DECLARE-Anweisung,786
- Watcom-SQL-Anweisungen
 - in Transact-SQL neu schreiben,413
- WATCOMSQL-Funktion
 - Syntax,430
- WebClientLogFile-Eigenschaft
 - mit sa_server_option einstellen,1306
- WebClientLogging-Eigenschaft
 - mit sa_server_option einstellen,1306
- Webclients
 - Erstellen von Clientfunktionen mit der CREATE FUNCTION-Anweisung [Webdienst],624
- Webdienst-Clientlogdatei
 - Namen einstellen,1306
- Webdienste
 - alphabetische Liste der Funktionen,162
 - alphabetische Liste der Systemprozeduren,1162
 - Ändern der SOAP-Dienste mit der ALTER SERVICE-Anweisung [SOAP-Webdienst],500
 - Ändern von HTTP-Diensten mit der ALTER SERVICE-Anweisungen [HTTP-Webdienst],494
 - erstellen mit der CREATE SERVICE-Anweisung [HTTP-Webdienst],706
 - erstellen mit der CREATE SERVICE-Anweisung [SOAP-Webdienst],713
 - HTML_DECODE-Funktion,278
 - HTML_ENCODE-Funktion,279
 - HTTP_BODY-Funktion,280
 - HTTP_DECODE-Funktion,281
 - HTTP_ENCODE-Funktion,282
 - HTTP_HEADER-Funktion,283
 - HTTP_RESPONSE_HEADER-Funktion,285
 - HTTP_VARIABLE-Funktion,287
 - Kommentare mit der COMMENT-Anweisung hinzufügen,573
 - Liste der Systemprozeduren für Webdienste,1162
 - NEXT_HTTP_HEADER-Funktion,325
 - NEXT_HTTP_RESPONSE_HEADER-Funktion,326
 - NEXT_HTTP_VARIABLE-Funktion,327
 - NEXT_SOAP_HEADER-Funktion,328
 - sa_http_header_info-Systemprozedur,1235
 - sa_http_php_page-Systemprozedur,1236
 - sa_http_php_page_interpreted-Systemprozedur,1237
 - sa_http_variable_info-Systemprozedur,1239
 - sa_set_http_header_info-Systemprozedur,1323
 - sa_set_http_option-Systemprozedur,1324
 - sa_set_soap_option-Systemprozedur,1329
 - SOAP_HEADER-Funktion,384
 - Systemansicht,1513
- WEEKS-Funktion
 - Syntax,430
- Weite Einfügungen
 - Info,846
- Werte
 - von Prozeduren zurückgeben,1004
- WHEN
 - CASE-Ausdruck,25
- WHEN MATCHED-Klausel
 - MERGE-Anweisung,955
- WHEN NOT MATCHED-Klausel
 - MERGE-Anweisung,955
- WHEN-Klausel
 - CASE-Anweisung ,566
 - CASE-Anweisung [T-SQL],568
 - CREATE TRIGGER-Anweisung,765
- WHENEVER-Anweisung
 - Embedded SQL-Syntax,1122
- WHERE CURRENT OF-Klausel
 - DELETE-Anweisung (positionsbasiert) [ESQL] [SP]-Anweisung],789
- WHERE-Klausel
 - ALTER EVENT-Anweisung,462
 - ALTER PUBLICATION-Anweisung,485
 - CREATE EVENT-Anweisung,609
 - CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote],691
 - DELETE-Anweisung,792
 - SELECT-Anweisung,1024
 - Suchbedingungen,42
 - UPDATE-Anweisung,1112
 - UPDATE-Anweisung [SQL Remote] ,1107
- WHILE-Anweisung
 - Transact-SQL-Syntax,1123
- WHILE-Klausel
 - LOOP-Anweisung,950
- Wieder aufnehmen
 - Ausführung von Prozeduren,1003
- Wiederbeschreiben, Cursor
 - CREATE PROCEDURE-Anweisung,684
 - CREATE PROCEDURE-Anweisung [externer Aufruf],664
- Wiederherstellen
 - Datenbanken aus Archiven,1001
- Wiederholbare Lesevorgänge

FROM-Klausel,869
 Wiederholen
 über Cursor,857
 WINDOW, Klausel
 Syntax,1124
 WINDOW-Klausel
 SELECT-Anweisung,1025
 WITH ACTIVATE-Klausel
 ALTER LDAP SERVER-Anweisung,470
 CREATE LDAP SERVER-Anweisung,644
 WITH ADMIN ONLY OPTION-Klausel
 nicht anwendbar für Systemrollen,883
 WITH ADMIN OPTION-Klausel
 nicht anwendbar für Systemrollen,883
 WITH AUTO NAME-Klausel
 INSERT-Anweisung,918
 MERGE-Anweisung,954
 WITH BATCH-Klausel
 EXECUTE IMMEDIATE-Anweisung,844
 WITH CHECK OPTION-Klausel
 ALTER VIEW-Anweisung,544
 CREATE VIEW-Anweisung,774
 WITH CHECKPOINT LOG-Klausel
 BACKUP-Anweisung,551
 WITH CHECKPOINT-Klausel
 LOAD TABLE-Anweisung,942
 WITH COMMENT-Klausel
 BACKUP-Anweisung,551
 WITH CONTENT LOGGING-Klausel
 bei Datenbankspiegelung erforderlich,944
 LOAD TABLE-Anweisung,943
 WITH DROP ALL REFERENCES-Klausel
 DROP LDAP SERVER-Anweisung,811
 WITH ESCAPES-Klausel
 EXECUTE IMMEDIATE-Anweisung,844
 WITH EXCLUSIVE MODE-Klausel
 REFRESH MATERIALIZED VIEW-Anweisung,988
 REFRESH TEXT INDEX-Anweisung,991
 WITH EXECUTE-Klausel
 PREPARE-Anweisung,977
 WITH EXPRESS CHECK-Klausel
 VALIDATE-Anweisung,1118
 WITH FILE NAME LOGGING-Klausel
 LOAD TABLE-Anweisung,942
 WITH HOLD-Cursor
 DECLARE CURSOR-Anweisung [ESQL]
 [SP],778
 sperren,965
 WITH HOLD-Klausel
 LOCK TABLE-Anweisung,949
 OPEN-Anweisung,965
 WITH ISOLATION LEVEL-Klausel
 REFRESH MATERIALIZED VIEW-Anweisung,988
 REFRESH TEXT INDEX-Anweisung,991
 WITH LOG-Klausel
 WRITETEXT-Anweisung [T-SQL],1127
 WITH MAX-Klausel
 ALLOCATE DESCRIPTOR-Anweisung
 [ESQL],449
 WITH NO ADMIN OPTION-Klausel
 nicht anwendbar für Systemrollen,883
 WITH NULLS NOT DISTINCT-Klausel
 CREATE INDEX-Anweisung,640
 WITH OPTION-Klausel
 SETUSER-Anweisung,1059
 WITH QUOTES-Klausel
 EXECUTE IMMEDIATE-Anweisung,843
 WITH RECOMPILE-Klausel
 CREATE PROCEDURE-Anweisung [T-SQL],679
 WITH RECURSIVE-Klausel
 SELECT-Anweisung,1022
 WITH REFRESH-Klausel
 ALTER LDAP SERVER-Anweisung,470
 WITH RESULT SET-Klausel
 EXECUTE IMMEDIATE-Anweisung,844
 WITH ROW LOGGING-Klausel
 bei Datenbankspiegelung erforderlich,944
 LOAD TABLE-Anweisung,943
 WITH SAVE-Klausel
 DETACH TRACING-Anweisung,801
 WITH SCRIPTED UPLOAD-Klausel
 CREATE PUBLICATION-Anweisung [MobiLink]
 [SQL Remote],690
 WITH SERVER NAME-Klausel
 START DATABASE-Anweisung,1063
 WITH SHAE MODE-Klausel
 REFRESH MATERIALIZED VIEW-Anweisung,988
 WITH SHARE MODE-Klausel
 REFRESH TEXT INDEX-Anweisung,991
 WITH SUSPEND-Klausel
 ALTER LDAP SERVER-Anweisung,470
 DROP LDAP SERVER-Anweisung,811
 WITH TEXTPTR-Klausel

- GET DATA-Anweisung,877
- WITH TRUNCATE AT CHECKPOINT-Klausel
 - START DATABASE-Anweisung,1063
- WITH VARIABLE RESULT-Klausel
 - DESCRIBE-Anweisung [ESQL],796
 - PREPARE-Anweisungen,977
- WITH-Klausel
 - ALTER LDAP SERVER-Anweisung,470
 - INTERSECT-Anweisung,927
 - REFRESH MATERIALIZED VIEW-Anweisung,987
 - REFRESH TEXT INDEX-Anweisung,991
 - SELECT-Anweisung,1022
- WITHOUT SAVE-Klausel
 - DETACH TRACING-Anweisung,801
- Wochendatum
 - ISO 8601,117
- WORK-Klausel
 - COMMIT-Anweisung,575
- WRITE_CLIENT_FILE-Funktion
 - Syntax,432
- WRITETEXT-Anweisung
 - Transact-SQL-Syntax,1127
- WSDL
 - CREATE FUNCTION-Anweisung [Webclients],628
 - CREATE PROCEDURE-Anweisung [Webclients],677
 - CREATE SERVICE-Anweisung [SOAP-Webdienst],501,714
- WWW-Authenticate
 - HTTP_RESPONSE_HEADER-Funktion,286

X

- XLOCK, Tabellen-Hint
 - FROM-Klausel,869
- XML
 - CREATE SERVICE-Anweisung [HTTP-Webdienst],495,707
 - laden aus, mit der LOAD TABLE-Anweisung,939
 - OPENXML-Operator,12
 - XML-Datentyp,103
 - XMLAGG-Funktion,433
 - XMLCONCAT-Funktion,434
 - XMLEMENT-Funktion,435
 - XMLFOREST-Funktion,438
 - XMLGEN-Funktion,439
 - XML-Datentyp
 - Syntax,103
 - XMLAGG-Funktion
 - Syntax,433
 - XMLATTRIBUTES, Parameter
 - XMLEMENT-Funktion,435
 - XMLCONCAT-Funktion
 - Syntax,434
 - XMLEMENT-Funktion
 - Syntax,435
 - XMLFOREST-Funktion
 - Syntax,438
 - XMLGEN-Funktion
 - Syntax,439
 - xp_cmdshell-Systemprozedur
 - Syntax,1418
 - xp_get_mail_error_code-Systemprozedur
 - Syntax,1419
 - xp_get_mail_error_text-Systemprozedur
 - Syntax,1420
 - xp_getenv-Systemprozedur
 - Syntax,1421
 - xp_msver-Systemprozedur
 - Syntax,1422
 - xp_read_file-Systemprozedur
 - Syntax,1424
 - xp_scanf-Systemprozedur
 - Syntax,1425
 - xp_sendmail-Systemprozedur
 - Syntax,1426
 - xp_sprintf-Systemprozedur
 - Syntax,1430
 - xp_startmail-Systemprozedur
 - Syntax,1431
 - xp_startsmtp-Systemprozedur
 - in McAfee VirusScan aktivieren,1433
 - mögliche Konflikte mit Virenschutzprogrammeinstellungen,1433
 - Syntax,1431
 - xp_stopmail-Systemprozedur
 - Syntax,1434
 - xp_stopsmtp-Systemprozedur
 - Syntax,1434
 - xp_write_file-Systemprozedur
 - Syntax,1435

Y

- YEAR-Funktion
 - Syntax,440
- YEARS-Funktion
 - Syntax,441
- YMD-Funktion
 - Syntax,442

Z

- Zahlen
 - allgemeines Element der SQL-Syntax,445
- Zeichendaten
 - Speicherung,95
 - Zeichenfolgen,6
- Zeichendatentypen
 - CHAR,95
 - Info,95
 - LONG NVARCHAR,97
 - LONG VARCHAR,97
 - NCHAR,98
 - NTEXT,99
 - NVARCHAR,99
 - TEXT,100
 - UNIQUEIDENTIFIERSTR,101
 - VARCHAR,101
 - XML,103
- Zeichenersetzung
 - Vergleiche zwischen CHAR und NCHAR,141
 - verlustreiche Zeichensatzkonvertierungen,140
- Zeichenfolgelänge
 - LENGTH-Funktion,301
- Zeichenfolgen
 - Anführungszeichen,41
 - Begrenzer,41
 - ersetzen,365
 - Escapezeichen,8
 - Info,6
 - Interpretation von begrenzten Zeichenfolgen
 - ändern,41
 - konvertieren in Datumsangaben,145
 - nachgestellte Leerzeichen entfernen,376
 - SQL-Funktionen,163
 - Transact-SQL,41
 - zweideutige Konvertierungen von zweistelligen Jahresangaben,118
- Zeichenfolgen in Unix komprimieren
 - COMPRESS-Funktion,194
- Zeichenfolgen konvertieren
 - Info,163
- Zeichenfolgen verketteten
 - Array-Operatoren,19
 - Zeichenfolgenoperatoren,11
- Zeichenfolgenausdrücke
 - allgemeines Element der SQL-Syntax,446
- Zeichenfolgenfunktionen
 - alphabetische Liste,163
- Zeichenfolgenlitterale
 - Escapesequenzen,8
 - Info,8
 - Sonderzeichen,8
- Zeichenfolgenoperatoren
 - Syntax,11
- Zeichenfolgeposition
 - LOCATE-Funktion,305
- Zeichenfunktionen
 - alphabetische Liste,163
- Zeichenklassen
 - Teilzeichenklassen,31
 - Unterstützung spezieller Teilzeichenklassen,31
- Zeichenlängensemantik
 - CHAR-Datentyp,95
 - VARCHAR-Datentyp,101
- Zeichensätze
 - beim Auswerten von Ausdrücken konvertieren,141
 - COMPARE-Funktion,192
 - Ersetzungszeichen,140
 - SORTKEY-Funktion,385
 - verlustreiche Zeichensatzkonvertierungen,140
- Zeichensatzkonvertierung
 - Ersetzungszeichen,140
 - Kennwörter, CREATE USER-Anweisung,770
 - Kennwörter, GRANT CONNECT-Anweisung,888
 - NCHAR in CHAR umwandeln,147
 - Vergleiche zwischen CHAR und NCHAR,141
 - Vergleiche zwischen Datentypen,140
 - Vergleiche zwischen numerischen Datentypen,142
 - verlustreiche Konvertierungen,140
- Zeichensatzkonvertierung für Kennwörter
 - ALTER USER-Anweisung,542
- Zeilen (*Siehe* zusammengesetzte Datentypen)
 - aktualisieren,1109
 - alle aus einer Tabelle löschen,1089
 - aus Cursor löschen,789
 - aus Datenbanken löschen,791

- Begrenzungswert zurückgegeben, 1020
- Daten mit UNLOAD-Anweisung entladen, 1098
- in Tabellen einfügen, 917
- Massendaten einfügen, 931
- mit Cursor einfügen, 981
- SELECT-Anweisung, 1020
- von Cursor abrufen, 853
- Zeilenbegrenzungen
 - Info, 1020
- Zeilenbeschränkungsklausel
 - DELETE-Anweisung, 791
 - SELECT-Anweisung, 1022
 - UPDATE-Anweisung, 1111
- Zeilengenerator
 - RowGenerator-Tabelle (dbo), 1159
 - sa_rowgenerator-Systemprozedur, 1300
- Zeilenkonstruktor, Algorithmus
 - DUMMY-Systemtabelle, 1129
- Zeilensperren
 - sa_locks-Systemprozedur, 1251
- Zeilenumbruchszeichen
 - in SQL-Zeichenfolgen, 8
- Zeitangaben
 - Konvertierungsfunktionen, 156
- Zeitdatentypen
 - DATETIME, 124
 - DATETIMEOFFSET, 125
 - SMALLDATETIME, 127
 - TIMESTAMP, 129
 - TIMESTAMP WITH TIME ZONE, 131
 - Übersicht, 116
- Zeitfunktionen
 - alphabetische Liste, 156
- Zeitmodell für Festplattenübertragungen
 - mit der ALTER DATABASE-Anweisung kalibrieren, 451
 - Standardwert mit der ALTER DATABASE-Anweisung wiederherstellen, 451
- Zeitstempel
 - Konvertierungen mit TIMESTAMP WITH TIMEZONE, 132
 - Vergleiche mit TIMESTAMP WITH TIMEZONE, 132
 - vergleichen, 143
- Zeitzone
 - CURRENT UTC TIMESTAMP, 75
 - UTC TIMESTAMP, 84
- Zertifikate
 - CREATE CERTIFICATE-Anweisung, 582
 - Informationen, 1171
 - Kommentare mit der COMMENT-Anweisung hinzufügen, 573
 - mit der DROP CERTIFICATE-Anweisung löschen, 803
 - sa_certificate_info-Systemprozedur, 1170
 - speichern, 582
 - SYSCERTIFICATE-Systemansicht, 1440
 - Zertifikate löschen, 803
- ZIP-Datei
 - E-Mail-Beispiel, 1429
- zip-Dienstprogramm
 - COMPRESS-Funktion, 194
- Zufallszahlen
 - RAND-Funktion, 345
- Zurückgeben
 - Werte von Prozeduren, 1004
- Zurückgegebene Zeilenanzahl begrenzen
 - Info, 1020
- Zurücksetzen
 - Transaktionen, 1016
 - Transaktionen auf Savepoints, 1015
 - Trigger, 1017
- Zusammenführung
 - MERGE-Anweisung, 952
- Zusammengesetzte Anweisungen
 - BEGIN-Anweisung, 557
 - BEGIN-Anweisung [TSQL], 560
 - TRY-Anweisung, 1093
 - TSQL-Kompatibilität, 560
- Zusammengesetzte Datentypen
 - Info, 136
 - vergleichen, 145
- Zuweisen
 - Logins für Fremdserver, 616
 - Plattenspeicher mit der ALTER DBSPACE-Anweisung, 457
 - Speicher für Deskriptorbereiche, 449
 - Werte an SQL-Variable, 1054
- Zuweisungen aufheben
 - Deskriptorbereiche, 777