



SQL Anywhere® Server SQL-Benutzerhandbuch

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 dieses Handbuch	ix
Tabellen, Ansichten und Indizes	1
Namen von Datenbankobjekten und Präfixe	1
Liste mit Systemobjekten anzeigen (Sybase Central)	3
Liste mit Systemobjekten anzeigen (SQL)	3
Tabellen	4
Temporäre Tabellen	10
Berechnete Spalten	13
Primärschlüssel	17
Fremdschlüssel	21
Indizes	27
Ansichten	41
Materialisierte Ansichten	56
Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen	81
Vorteile von Prozeduren, Triggern und benutzerdefinierten Funktionen	82
Prozeduren	82
Benutzerdefinierte Funktionen	93
Trigger	98
Batchanweisungen	112
Aufbau von Prozeduren, Triggern und benutzerdefinierten Funktionen ...	115
Steueranweisungen	118
Ergebnismengen	121
Cursor in Prozeduren, Triggern, benutzerdefinierten Funktionen und Batches	128
Behandlung von Fehlern und Warnungen	131
EXECUTE IMMEDIATE in Prozeduren, Triggern, benutzerdefinierten Funktionen und Batches	143
Transaktionen und Savepoints in Prozeduren, Triggern und benutzerdefinierten Funktionen	146

Tipps zum Schreiben von Prozeduren, Triggern, benutzerdefinierten Funktionen und Batches	146
Zulässige Anweisungen in Prozeduren, Triggern, Ereignissen und Batches	148
Verbergen des Inhalts von Prozeduren, Funktionen, Triggern, Ereignissen oder Ansichten	149
Performanceverbesserungen, Diagnosen und Monitoring	153
Tools für Performanceüberwachung und Diagnose	153
Tipps zum Verbessern der Performance	219
Praktische Einführungen in die Anwendungsprofilerstellung	264
Abfragen und Datenänderung	291
Abfragen	291
Volltextsuche	379
Abfrageergebnisse zusammenfassen, gruppieren und sortieren	470
Joins: Daten aus mehreren Tabellen abrufen	492
Allgemeine Tabellenausdrücke	536
OLAP-Unterstützung	552
Verwendung von Unterabfragen	600
Datenmanipulationsanweisungen	623
SQL-Dialekte und Kompatibilität	641
Testen der SQL-Konformität mit dem SQL Flagger	641
SQL Anywhere-Funktionen, die sich von anderen SQL-Implementierungen unterscheiden	643
Watcom SQL	648
Transact-SQL-Kompatibilität	648
Adaptive Server Enterprise-Architekturen	651
Transact-SQL-kompatible Datenbanken	656
Kompatible SQL-Anweisungen	663
Transact-SQL-Prozedursprache	669
Automatische Konvertierung von gespeicherten Prozeduren	671
Aus Transact-SQL-Prozeduren zurückgegebene Ergebnismengen	672

Variable in Transact-SQL-Prozeduren	673
Fehlerbehandlung in Transact-SQL-Prozeduren	673
XML in der Datenbank	677
Speichern von XML-Dokumenten in relationalen Datenbanken	677
Als XML exportierte relationale Daten	678
Methoden zum Importieren von XML-Dokumenten als relationale Daten ..	679
Abfrageergebnisse als XML	687
Verwendung von Interactive SQL zum Anzeigen von Ergebnissen	705
Verwendung von SQL/XML zum Abrufen von Abfrageergebnissen als XML	706
JSON in der Datenbank	715
Abfrageergebnisse mit der FOR JSON-Klausel als JSON abrufen	715
FOR JSON RAW	716
FOR JSON AUTO	716
FOR JSON EXPLICIT	717
Datenimport und -export	721
Performance-Aspekte von Massenvorgängen	721
Die Datenwiederherstellung bei Massenvorgängen	722
Datenimport	723
Datenexport	743
Zugriff auf Daten auf Clientcomputern	760
Neuaufbau von Datenbanken	763
Datenbankextraktion	774
Datenbankmigration nach SQL Anywhere	774
SQL-Skriptdateien	780
Adaptive Server Enterprise-Kompatibilität	784
Ferndatenzugriff	785
Zuordnungen entfernter Tabellen	786
Fremdserver	787

Verzeichniszugriffsserver	795
Externe Logins	804
Proxy-Tabellen	805
Native Anweisungen und Fremdserver	813
Entfernte Prozeduraufrufe	814
Transaktionsverwaltung und entfernte Daten	817
Interne Vorgänge	818
Fehlerbehandlung beim Ferndatenzugriff	822
Serverklassen für den Ferndatenzugriff	824
Datenintegrität	849
Wie Ihre Daten ungültig werden können	849
Integritätsregeln	849
Wie sich der Inhalt Ihrer Datenbank ändert	850
Werkzeuge zur Aufrechterhaltung der Datenintegrität	850
SQL-Anweisungen für die Implementierung von Integritätsregeln	851
Spalten-Standardwerte	852
Tabellen- und Spalten-Integritätsregeln	859
Domänen	865
Entitätsintegrität und referenzielle Integrität	868
Integritätsregeln in den Systemtabellen	877
Transaktionen und Isolationsstufen	881
Transaktionen	882
Parallelität	884
Savepoints innerhalb von Transaktionen	884
Isolationsstufen und Konsistenz	885
Transaktion blockieren und Deadlock	901
Funktionsweise von Sperren	906
Richtlinien zum Auswählen der Isolationsstufen	922
Praktische Einführung in Isolationsstufen	926
Primärschlüssel-Generierung und Parallelität	949
Datendefinitionsanweisung und Parallelität	954
Zusammenfassung	954

LDAP-Benutzerauthentifizierung	957
LDAP-Benutzerauthentifizierungsumgebung erstellen (SQL)	958
LDAP-Benutzerauthentifizierungsumgebung erstellen (Sybase Central) ..	962
 SQL Anywhere-Debugger	 971
Voraussetzungen für die Benutzung des Debuggers	971
Praktische Einführung: Erste Schritte mit dem Debugger	972
Breakpoints	977
Variablen	981
Fehlersuche in Verbindungen	983
 Index	 985

Über dieses Handbuch

In diesem Handbuch wird beschrieben, wie Objekte einer Datenbank hinzugefügt, Daten importiert, exportiert, geändert bzw. abgerufen und gespeicherte Prozeduren und Trigger erstellt werden.

Tabellen, Ansichten und Indizes

Die SQL-Anweisungen für das Erstellen, Ändern und Löschen von Datenbankobjekten werden als **Datendefinitionssprache** (DDL = Data Definition Language) bezeichnet. Die Definitionen der Datenbankobjekte bilden das Datenbankschema. Ein Schema ist der logische Rahmen der Datenbank.

Siehe auch

- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ auf Seite 81
- „Datenintegrität“ auf Seite 849
- „Datenbankerstellung“ [*SQL Anywhere Server - Datenbankadministration*]
- „Sybase Central“ [*SQL Anywhere Server - Datenbankadministration*]
- „Interactive SQL“ [*SQL Anywhere Server - Datenbankadministration*]

Namen von Datenbankobjekten und Präfixe

Der Name eines jeden Datenbankobjekts ist ein Bezeichner. In Beispielabfragen, die in dieser Dokumentation verwendet werden, wird auf Datenbankobjekte aus der Beispieldatenbank im Allgemeinen nur mit ihrem Bezeichner verwiesen. Zum Beispiel:

```
SELECT * FROM Employees;
```

Tabellen, Prozeduren und Ansichten haben alle einen Eigentümer. Der Benutzer GROUPO hat die Tabellen in der Beispieldatenbank erstellt. In einigen Fällen müssen Sie dem Objektnamen den Namen des Eigentümers voranstellen, wie im folgenden Beispiel:

```
SELECT * FROM GROUPO.Employees;
```

Der Bezug auf die Tabelle "Employees" ist **qualifiziert**. In anderen Fällen ist es ausreichend, den Objektnamen anzugeben. In diesem Abschnitt wird beschrieben, wann Sie das Eigentümerpräfix verwenden müssen, um Tabellen, Ansichten und Prozeduren zu bezeichnen, und wann nicht.

Wenn Sie Bezug auf ein Datenbankobjekt nehmen, müssen Sie ein Präfix angeben, es sei denn:

- Sie sind der Eigentümer des Datenbankobjekts.
- Das Datenbankobjekt ist Eigentum einer Rolle, die Ihnen erteilt wurde.

Beispiel

Beachten Sie das folgende Beispiel für eine Unternehmensdatenbank der Firma Acme. Eine Benutzer-ID "Admin" wird mit vollständigen Administrationsrechten für die Datenbank erstellt. Zwei weitere Benutzer-IDs, Joe und Sally, werden für Mitarbeiter erstellt, die in der Verkaufsabteilung arbeiten.

```
CREATE USER Admin IDENTIFIED BY secret;  
GRANT ROLE SYS_AUTH_DBA_ROLE TO Admin;  
CREATE USER Sally IDENTIFIED BY xxxxxx;  
CREATE USER Joe IDENTIFIED BY xxxxxx;
```

Der Admin-Benutzer erstellt die Tabellen in der Datenbank und weist die Eigentumsrechte der Acme-Rolle zu.

```
CREATE ROLE Acme;  
CREATE TABLE Acme.Customers ( ... );  
CREATE TABLE Acme.Products ( ... );  
CREATE TABLE Acme.Orders ( ... );  
CREATE TABLE Acme.Invoices ( ... );  
CREATE TABLE Acme.Employees ( ... );  
CREATE TABLE Acme.Salaries ( ... );
```

Nicht jeder im Unternehmen sollte Zugriff auf alle Daten erhalten. Joe und Sally arbeiten in der Verkaufsabteilung und sollen Zugriff auf die Tabellen Customers, Products und Orders erhalten. Dazu müssen Sie eine Rolle namens SalesForce erstellen, dieser Rolle die Privilegien zuordnen, die für den Zugriff auf eine beschränkte Menge von Tabellen erforderlich sind, und diese Rolle diesen beiden Mitarbeitern zuordnen.

```
CREATE ROLE SalesForce;  
GRANT ALL ON Acme.Customers TO SalesForce;  
GRANT ALL ON Acme.Orders TO SalesForce;  
GRANT SELECT ON Acme.Products TO SalesForce;  
GRANT ROLE SalesForce TO Sally;  
GRANT ROLE SalesForce TO Joe;
```

Joe und Sally haben zwar die erforderlichen Privilegien, um diese Tabellen zu verwenden, aber sie müssen immer noch ihre Tabellenreferenzen qualifizieren, weil der Tabelleneigentümer Acme ist.

```
SELECT * FROM Acme.Customers;
```

Um diese Situation zu korrigieren, gewähren Sie der Sales-Rolle die Acme-Rolle.

```
GRANT ROLE Acme TO SalesForce;
```

Da Joe und Sally jetzt die Sales-Rolle haben, erhalten sie indirekt auch die Acme-Rolle und können ihre Tabellen ohne Qualifizierer referenzieren. Die SELECT-Anweisung kann wie folgt vereinfacht werden:

```
SELECT * FROM Customers;
```

Hinweis

Die benutzerdefinierte Rolle Acme erteilt keine Privilegien auf Objektebene. Diese Rolle ermöglicht es einfach einem Benutzer, die Objekte zu referenzieren, die der Rolle gehören, ohne dass der Eigentümer qualifiziert werden muss. Joe und Sally haben keine zusätzlichen Privilegien aufgrund der Acme-Rolle. Der Acme-Rolle wurden keine weiteren speziellen Privilegien explizit erteilt. Der Admin-Benutzer verfügt über das implizite Privileg, Tabellen wie "Salaries" anzuzeigen, weil er die Tabellen erstellt hat und die SYS_AUTH_DBA_ROLE-Systemrolle hat. Daher erhalten Joe und Sally immer noch eine Fehlermeldung, wenn sie eine der folgenden Anweisungen ausführen:

```
SELECT * FROM Acme.Salaries;  
SELECT * FROM Salaries;
```

In beiden Fällen verfügen Joe und Sally nicht über die Privilegien zum Anzeigen der Tabelle "Salaries".

Siehe auch

- „Gruppen“ [[SQL Anywhere Server - Datenbankadministration](#)]

Liste mit Systemobjekten anzeigen (Sybase Central)

Verwenden Sie Sybase Central, um Informationen über Systemobjekte wie Systemtabellen, Systemansichten, gespeicherte Prozeduren und Domänen anzuzeigen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Sie führen diese Aufgabe aus, wenn die Liste der Systemobjekte in der Datenbank und ihrer Definitionen angezeigt werden soll oder wenn Sie diese Definition verwenden möchten, um andere ähnliche Objekte zu erstellen.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Wählen Sie die Datenbank aus und klicken Sie auf **Datei » Eigentümerfilter konfigurieren**.
3. Wählen Sie **SYS** und **dbo** und klicken Sie auf **OK**.

Ergebnisse

Die Liste von Systemobjekten wird in Sybase Central angezeigt.

Siehe auch

- „SYSOBJECT-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSTAB-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSUSER-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Liste mit Systemobjekten anzeigen (SQL)

Fragen Sie die SYSOBJECT-Systemansicht ab, um Informationen über Systemobjekte wie Systemtabellen, Systemansichten, gespeicherte Prozeduren und Domänen anzuzeigen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Sie führen diese Aufgabe aus, wenn die Liste der Systemobjekte in der Datenbank und ihrer Definitionen angezeigt werden soll oder wenn Sie diese Definition verwenden möchten, um andere ähnliche Objekte zu erstellen.

Aufgabe

1. Stellen Sie in Interactive SQL eine Verbindung mit einer Datenbank her.
2. Führen Sie eine SELECT-Anweisung aus, welche die Systemansicht SYSOBJECT für eine Liste von Objekten abfragt.

Ergebnisse

Die Liste von Systemobjekten wird in Interactive SQL angezeigt.

Beispiel

Die folgende SELECT-Anweisung fragt die Systemansicht SYSOBJECT ab und liefert eine Liste aller Tabellen und Ansichten, die SYS und dbo gehören. Mit der Systemansicht SYSTAB wird ein Join hergestellt, um den Objektnamen abzurufen, und ein weiterer Join mit der Systemansicht SYSUSER liefert den Namen des Eigentümers.

```
SELECT b.table_name "Object Name",
       c.user_name "Owner",
       b.object_id "ID",
       a.object_type "Type",
       a.status "Status"
FROM ( SYSOBJECT a JOIN SYSTAB b
      ON a.object_id = b.object_id )
JOIN SYSUSER c
WHERE c.user_name = 'SYS'
      OR c.user_name = 'dbo'
ORDER BY c.user_name, b.table_name;
```

Siehe auch

- „SYSOBJECT-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSTAB-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSUSER-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Tabellen

Wenn eine Datenbank erstellt wird, sind die einzigen Tabellen in der Datenbank die Systemtabellen. Systemtabellen enthalten das Datenbankschema.

Sie sollten SQL-Skriptdateien zur Definition der Tabellen in Ihrer Datenbank erstellen, um sich das Neuerstellen des Datenbankschemas zu erleichtern, falls dies erforderlich wird. Die SQL-Skriptdateien sollten die Anweisungen CREATE TABLE und ALTER TABLE enthalten.

Siehe auch

- „Benutzererweiterte Rollen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Ergebnismengen in Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Namen von Datenbankobjekten und Präfixe“ auf Seite 1

Erstellen einer Tabelle

Sie können Sybase Central verwenden, um Tabellen in Ihrer Datenbank zu erstellen.

Voraussetzungen

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.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf **Tabellen** und klicken Sie auf **Neu » Tabelle**.
3. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen einer Tabelle**.
4. Klicken Sie im rechten Fensterausschnitt auf die Registerkarte **Spalten** und erstellen Sie neue Spalten für Ihre Tabelle.
5. Klicken Sie auf **Datei » Speichern**.

Ergebnisse

Die neue Tabelle wird in der Datenbank gespeichert.

Nächste Schritte

Sie können Daten in Ihre Tabelle eingeben oder laden.

Siehe auch

- „CREATE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Daten mit INSERT hinzufügen“ auf Seite 626
- „Importieren von Daten mit der INSERT-Anweisung“ auf Seite 731
- „INSERT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „LOAD TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Tabellenänderung

Sie können die Struktur oder Spaltendefinitionen einer Tabelle ändern, indem Sie Spalten hinzufügen, verschiedene Spaltenattribute ändern oder Spalten löschen.

Tabellenänderungen und Ansichtenabhängigkeiten

Bevor Sie eine Tabelle ändern, sollten Sie mit der `sa_dependent_views`-Systemprozedur ermitteln, ob tabellenabhängige Ansichten vorhanden sind.

Wenn Sie das Schema einer Tabelle mit abhängigen Ansichten ändern, sind je nach Art der Ansicht möglicherweise zusätzliche Schritte erforderlich:

- **Abhängige reguläre Ansichten** Wenn Sie das Schema einer Tabelle ändern, wird die Tabellendefinition in der Datenbank aktualisiert. Falls es abhängige reguläre Ansichten gibt, werden diese vom Datenbankserver automatisch neu kompiliert, nachdem Sie die Tabellenänderung durchgeführt haben. Wenn der Datenbankserver eine abhängige reguläre Ansicht nach der Änderung eines Tabellenschemas nicht neu kompilieren kann, liegt dies wahrscheinlich daran, dass die Ansichtsdefinition durch die Änderung ungültig geworden ist. In einem solchen Fall müssen Sie die Ansichtsdefinition korrigieren.
- **Abhängige materialisierte Ansichten** Wenn es abhängige materialisierte Ansichten gibt, müssen Sie diese deaktivieren, bevor Sie die Tabellenänderung durchführen, und dann wieder aktivieren, nachdem die Änderung abgeschlossen ist. Wenn Sie eine abhängige materialisierte Ansicht nach der Änderung eines Tabellenschemas nicht neu kompilieren können, liegt dies wahrscheinlich daran, dass die Ansichtsdefinition durch die Änderung ungültig geworden ist. In einem solchen Fall müssen Sie die materialisierte Ansicht löschen und dann mit einer gültigen Definition wieder neu erstellen, oder geeignete Änderungen an der zugrunde liegenden Tabelle durchführen, bevor Sie versuchen, die materialisierte Ansicht wieder zu aktivieren.

Änderungen der Tabelleneigentümer

Sie können den Eigentümer einer Tabelle mit der `ALTER TABLE`-Anweisung oder Sybase Central ändern. Wenn Sie den Tabelleneigentümer ändern, können Sie festlegen, ob bestehende Fremdschlüssel in den Tabellen und diejenigen, die auf die Tabelle verweisen, erhalten bleiben sollen. Wenn alle Fremdschlüssel gelöscht werden, wird die Tabelle isoliert, dadurch erhöht sich aber auch die Sicherheit. Außerdem können Sie angeben, ob vorhandene explizit erteilte Privilegien beibehalten werden sollen. Für Sicherheitszwecke können Sie alle explizit erteilten Privilegien löschen, die einem Benutzer den Zugriff auf die Tabelle ermöglichen. Implizit erteilte Privilegien für den Eigentümer der Tabelle werden dem neuen Eigentümer erteilt und aus dem alten Eigentümer gelöscht.

Siehe auch

- „Reguläre Ansichten ändern“ auf Seite 50
- „Materialisierte Ansichten erstellen“ auf Seite 65
- „Ansichtenabhängigkeiten“ auf Seite 43
- „`sa_dependent_views`-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Tabellen ändern

Sie können Sybase Central verwenden, um Tabellen in Ihrer Datenbank zu ändern, z.B. wenn Sie Spalten hinzufügen oder entfernen oder den Tabelleneigentümer ändern möchten.

Voraussetzungen

Sie müssen Eigentümer sein oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Tabelle und entweder das COMMENT ANY OBJECT-Systemprivileg, das CREATE ANY OBJECT-Systemprivileg oder das CREATE ANY TABLE-Systemprivileg.
- ALTER ANY TABLE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg
- Das ALTER ANY OBJECT OWNER-Privileg (beim Ändern des Tabelleneigentümers) und entweder das ALTER ANY OBJECT-Systemprivileg, das ALTER ANY TABLE-Systemprivileg oder das ALTER-Privileg für die Tabelle.

Das Ändern von Tabellen schlägt fehl, wenn abhängige materialisierte Ansichten vorhanden sind. Sie müssen die abhängigen materialisierten Ansichten zunächst deaktivieren. Verwenden Sie die sa_dependent_views-Systemprozedur, um zu ermitteln, ob abhängige materialisierte Ansichten vorhanden sind.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Wählen Sie eine der folgenden Optionen:

Option	Aktion
Spalten ändern	<ol style="list-style-type: none"> a. Doppelklicken Sie auf die Tabelle, die Sie ändern wollen. b. Klicken Sie im rechten Fensterausschnitt auf die Registerkarte Spalten und ändern Sie die Spalten für die Tabelle nach Wunsch. c. Klicken Sie auf Datei » Speichern.
Eigentümer der Tabelle ändern	<ul style="list-style-type: none"> • Rechtsklicken Sie auf eine Tabelle, klicken Sie auf Eigenschaften » Eigentümer jetzt ändern und ändern Sie den Tabelleneigentümer.

Ergebnisse

Die Tabellendefinition wird in der Datenbank aktualisiert.

Nächste Schritte

Wenn Sie zum Ändern der Tabelle materialisierte Ansichten deaktiviert haben, müssen Sie jede einzelne davon wieder aktivieren und initialisieren.

Siehe auch

- „sa_dependent_views-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Aktivieren oder Deaktivieren einer materialisierten Ansicht“ auf Seite 68
- „Datenintegrität“ auf Seite 849
- „Ansichtenabhängigkeiten“ auf Seite 43
- „ALTER TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Abhängigkeiten und Vorgänge, die ein Schema ändern“ auf Seite 43

Löschen einer Tabelle

Sie können Sybase Central verwenden, um eine Tabelle aus Ihrer Datenbank zu löschen, z.B. wenn sie nicht mehr benötigt wird.

Voraussetzungen

Sie müssen entweder Eigentümer sein oder das DROP ANY TABLE-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Sie können eine Tabelle nicht löschen, wenn sie als Artikel in einer Publikation verwendet wird. Wenn Sie dies in Sybase Central versuchen, tritt ein Fehler auf. Auch wenn Sie eine Tabelle löschen, die über abhängige Ansichten verfügt, sind möglicherweise zusätzliche Schritte erforderlich.

Das Löschen von Tabellen schlägt fehl, wenn abhängige materialisierte Ansichten vorhanden sind. Sie müssen die abhängigen materialisierten Ansichten zunächst deaktivieren. Verwenden Sie die sa_dependent_views-Systemprozedur, um zu ermitteln, ob abhängige materialisierte Ansichten vorhanden sind.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf **Tabellen**.
3. Rechtsklicken Sie auf die Tabelle und klicken Sie auf **Löschen**.
4. Klicken Sie auf **Ja**.

Ergebnisse

Wenn Sie eine Tabelle löschen, wird ihre Definition aus der Datenbank entfernt. Falls es abhängige reguläre Ansichten gibt, versucht der Datenbankserver, sie neu zu kompilieren und zu aktivieren, nachdem Sie die Tabellenänderung durchgeführt haben. Falls dies nicht gelingt, liegt es wahrscheinlich daran, dass die Definition der Ansicht durch das Löschen der Tabelle ungültig wurde. In einem solchen Fall müssen Sie die Ansichtsdefinition korrigieren.

Wenn es abhängige materialisierte Ansichten gab, schlägt eine nachfolgende Aktualisierung fehl, weil deren Definitionen nicht mehr gültig sind. In diesem Fall müssen Sie die materialisierte Ansicht löschen und dann mit einer gültigen Definition neu erstellen.

Alle Indizes für die Tabelle werden gelöscht.

Das Löschen einer Tabelle bewirkt, dass eine COMMIT-Anweisung ausgeführt wird. Damit werden alle Änderungen der Datenbank seit dem letzten COMMIT oder ROLLBACK festgeschrieben.

Nächste Schritte

Abhängige reguläre oder materialisierte Ansichten müssen gelöscht oder Referenzen auf die gelöschte Tabelle aus ihren Definitionen entfernt werden.

Siehe auch

- „sa_dependent_views-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Aktivieren oder Deaktivieren einer materialisierten Ansicht“ auf Seite 68
- „Ansichtenabhängigkeiten“ auf Seite 43
- „Reguläre Ansichten ändern“ auf Seite 50
- „Abhängigkeiten und Vorgänge, die ein Schema ändern“ auf Seite 43
- „DROP TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Daten in Tabellen anzeigen (Sybase Central)

Sie können Sybase Central verwenden, um die Daten in Tabellen zu durchsuchen.

Voraussetzungen

Sie müssen das SELECT-Privileg für die Tabelle oder das SELECT ANY TABLE-Systemprivileg haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf **Tabellen**.
3. Klicken Sie im rechten Fensterausschnitt auf das Register **Daten**.

Ergebnisse

Die Daten für die Tabelle werden auf der Registerkarte **Daten** angezeigt.

Nächste Schritte

Sie können die Daten auf der Registerkarte **Daten** bearbeiten.

Wenn Sie in Interactive SQL arbeiten, führen Sie eine Anweisung ähnlich der folgenden aus, wobei *table-name* die Daten enthält, die Sie anzeigen möchten:

```
SELECT * FROM table-name;
```

Siehe auch

- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Daten in einer regulären Ansicht durchsuchen“ auf Seite 55

Daten in Tabellen anzeigen (SQL)

Sie können Interactive SQL verwenden, um die Daten in Tabellen anzuzeigen.

Voraussetzungen

Sie müssen das SELECT-Privileg für die Tabelle oder das SELECT ANY TABLE-Systemprivileg haben.

Aufgabe

- Führen Sie eine Anweisung ähnlich der folgenden aus, wobei *table-name* für die Tabelle steht, deren Daten Sie anzeigen möchten.

```
SELECT * FROM table-name;
```

Ergebnisse

Die Daten für die Tabelle werden im Fensterausschnitt **Ergebnisse** angezeigt.

Nächste Schritte

Sie können die Daten im Fensterausschnitt **Ergebnisse** bearbeiten.

Siehe auch

- „Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Temporäre Tabellen

Temporäre Tabellen werden in der temporären Datei gespeichert. Seiten der temporären Datei können im Cache abgelegt werden, ebenso wie Seiten jedes anderen Dbspaces. Vorgänge in temporären Tabellen werden nie in das Transaktionslog geschrieben. Es gibt zwei Arten von Tabellen: **lokale temporäre** Tabellen und **globale temporäre** Tabellen.

- **Lokale temporäre Tabellen** Eine lokale temporäre Tabelle existiert nur für die Dauer einer Verbindung bzw. (wenn sie für eine zusammengesetzte Anweisung definiert wurde) für die Dauer der zusammengesetzten Anweisung.

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.

Wenn Sie einen Index für eine lokale temporäre Tabelle erstellen und die Option `auto_commit_on_create_local_temp_index` deaktiviert ist, wird kein Festschreiben durchgeführt, bevor ein Index für die Tabelle erstellt wird.

- **Globale temporäre Tabellen** Eine globale temporäre Tabelle verbleibt in der Datenbank, bis sie explizit durch eine `DROP TABLE`-Anweisung gelöscht wird. Mehrere Verbindung von derselben oder von verschiedenen Anwendungen können eine globale temporäre Tabelle gleichzeitig verwenden. Globale temporäre Tabellen haben folgende Merkmale:
 - Die Definition der Tabelle wird im Katalog aufgezeichnet und bleibt bestehen, bis die Tabelle explizit gelöscht wird.
 - Einfügungen, Aktualisierungen und Löschungen in der Tabelle werden nicht im Transaktionslog aufgezeichnet.
 - Spaltenstatistiken für die Tabelle werden vom Datenbankserver im Speicher geführt.

Es gibt zwei Arten von globalen temporären Tabellen: **nicht gemeinsam genutzte Tabellen** und **gemeinsam genutzte Tabellen**. Normalerweise wird eine globale temporäre Tabelle nicht gemeinsam genutzt, d.h., jede Verbindung sieht nur die eigenen Zeilen in der Tabelle. Wenn eine Verbindung beendet wird, werden die Zeilen für diese Verbindung aus der Tabelle gelöscht.

Wenn eine globale temporäre Tabelle gemeinsam genutzt wird, werden alle Daten der Tabelle von allen Verbindungen gemeinsam genutzt. Um eine gemeinsam genutzte globale temporäre Tabelle zu erstellen, geben Sie beim Erstellen die Klausel `SHARE BY ALL` an. Für gemeinsam genutzte globale temporäre Tabellen gelten, zusätzlich zu den allgemeinen Merkmalen globaler temporärer Tabellen, die folgenden Merkmale:

- Der Inhalt der Tabelle bleibt bestehen, bis sie explizit gelöscht wird oder bis die Datenbank heruntergefahren wird.
 - Beim Start der Datenbank ist die Tabelle leer.
 - Zeilensperren funktionieren bei der Tabelle genau so wie bei einer Basistabelle.
- **Nicht-transaktionale temporäre Tabellen** Temporäre Tabellen können mit der Klausel `NOT TRANSACTIONAL` in der Anweisung `CREATE TABLE` als nicht-transaktional deklariert werden. Die `NOT TRANSACTIONAL`-Klausel bietet Performanceverbesserungen unter bestimmten Umständen, da Vorgänge in nicht-transaktionalen temporären Tabellen keine Einträge im Rollback-Log bewirken. Beispiel: `NOT TRANSACTIONAL` kann sinnvoll sein, wenn Prozeduren, die die temporäre Tabelle verwenden, wiederholt ohne dazwischenliegende `COMMITs` oder `ROLLBACKs` aufgerufen werden oder wenn die Tabelle viele Zeilen enthält. Änderungen nicht-transaktionaler temporärer Tabellen werden durch `COMMIT` oder `ROLLBACK` nicht beeinflusst.

Siehe auch

- [„Transaktionen und Isolationsstufen“ auf Seite 881](#)
- [„Funktionsweise von Sperren“ auf Seite 906](#)
- [„CREATE TABLE-Anweisung“ \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- [„DECLARE LOCAL TEMPORARY TABLE-Anweisung“ \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)

Erstellen einer globalen temporären Tabelle

Erstellen Sie eine globale temporäre Tabelle mit Sybase Central. Führen Sie diese Aufgabe aus, um globale temporäre Tabellen zu erstellen, wenn Sie Daten bearbeiten möchten, ohne Zeilensperren berücksichtigen zu müssen, und um unnötige Aktivität im Transaktions- und Wiederherstellungslog zu reduzieren.

Voraussetzungen

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.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie auf **Tabellen** und klicken Sie auf **Neu » Globale temporäre Tabelle**.
3. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen einer globalen temporären Tabelle**.
4. Im rechten Fensterausschnitt klicken Sie auf der Registerkarte **Spalten** und konfigurieren die Tabelle.
5. Klicken Sie auf **Datei » Speichern**.

Ergebnisse

Eine globale temporäre Tabelle wird erstellt. Die Definition der globalen temporären Tabelle wird in der Datenbank gespeichert, bis sie gelöscht wird, und ist zur Verwendung durch andere Verbindungen verfügbar.

Siehe auch

- „Temporäre Tabellen“ auf Seite 10
- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DECLARE LOCAL TEMPORARY TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Referenzen auf temporäre Tabellen innerhalb von Prozeduren

Die gemeinsame Nutzung einer temporären Tabelle durch Prozeduren kann zu Problemen führen, wenn die Tabellendefinitionen inkonsistent sind. Nehmen wir z.B. an, dass Sie zwei Prozeduren haben (procA und procB), die beide eine temporäre Tabelle definieren (temp_table) und eine weitere Prozedur (sharedProc) aufrufen. Weder procA noch procB wurde bisher aufgerufen, daher existiert die temporäre Tabelle noch nicht.

Nun nehmen wir an, dass sich die procA-Definition für temp_table etwas von der Definition in procB unterscheidet: Während beide dieselben Spaltennamen und -typen verwenden, ist die Spaltenreihenfolge eine andere.

Wenn Sie procA aufrufen, gibt sie das erwartete Ergebnis zurück. Wenn Sie allerdings procB aufrufen, gibt sie ein anderes Ergebnis zurück.

Der Grund dafür ist, dass procA nach ihrem Aufruf temp_table erstellt und dann sharedProc aufgerufen hat. Als sharedProc aufgerufen wurde, wurde die darin enthaltene SELECT-Anweisung syntaktisch analysiert und validiert. Anschließend wurde eine syntaktisch analysierte Darstellung der Anweisung im Cache abgelegt, für den Fall, dass die SELECT-Anweisung erneut ausgeführt wird. Die im Cache abgelegte Version spiegelt die Spaltensortierfolge aus der Tabellendefinition in procA wider.

Ein Aufruf von procB bewirkt, dass temp_table neu erstellt wird, aber mit einer anderen Spaltensortierfolge. Wenn procB sharedProc aufruft, verwendet der Datenbankserver die im Cache abgelegte Darstellung der SELECT-Anweisung. Daher sind die Ergebnisse unterschiedlich.

Sie können diese Situation vermeiden, indem Sie eine der folgenden Aktionen ausführen:

- Stellen Sie sicher, dass die auf diese Art verwendeten temporären Tabellen konsistent sind.
- Verwenden Sie anstatt dessen eine globale temporäre Tabelle.

Berechnete Spalten

Eine berechnete Spalte ist eine Spalte, deren Wert ein Ausdruck ist, der die Werte von anderen Spalten, die als **abhängige Spalten** bezeichnet werden, in derselben Zeile referenzieren kann. Berechnete Spalten sind vor allem dann nützlich, wenn Sie einen Index für einen komplexen Ausdruck bilden wollen, der möglicherweise die Werte von einer oder mehreren abhängigen Spalten umfasst. Der Datenbankserver verwendet die berechnete Spalte immer dann, wenn er einen Ausdruck erkennt, der mit dem COMPUTE-Ausdruck der berechneten Spalte übereinstimmt. Dies gilt sowohl für die SELECT-Liste als auch für Prädikate. Wenn der Abfrageausdruck jedoch einen Spezialwert wie etwa CURRENT_TIMESTAMP enthält, tritt diese Übereinstimmung nicht auf.

Verwenden Sie für berechnete Spalten keine TIMESTAMP WITH TIME ZONE-Spalten. Der Wert der time_zone_adjustment-Option variiert zwischen Verbindungen, abhängig von ihrem Standort und der Jahreszeit. Dies führt zu falschen Ergebnissen und unerwartetem Verhalten, wenn die Werte berechnet werden.

Während der Abfrageoptimierung versucht der SQL Anywhere-Optimierer automatisch, ein Prädikat, das einen komplexen Ausdruck betrifft, in eines umzuwandeln, das sich einfach auf die Definition der berechneten Spalte bezieht. Nehmen wir z.B. an, dass Sie eine Tabelle abfragen wollen, die Zusammenfassungsinformationen über Produktlieferungen enthält:

```
CREATE TABLE Shipments(  
  ShipmentID INTEGER NOT NULL PRIMARY KEY,  
  ShipmentDate TIMESTAMP,  
  ProductCode CHAR(20) NOT NULL,  
  Quantity INTEGER NOT NULL,
```

```
TotalPrice DECIMAL(10,2) NOT NULL
);
```

Im Einzelnen sollte die Abfrage jene Lieferungen zurückgeben, deren Durchschnittskosten zwischen zwei und vier Dollar liegen. Die Abfrage könne folgendermaßen aussehen:

```
SELECT *
FROM Shipments
WHERE ( TotalPrice / Quantity ) BETWEEN 2.00 AND 4.00;
```

In der obenstehenden Abfrage ist das Prädikat in der WHERE-Klausel jedoch nicht "sargable" (als Suchargument nutzbar), da es nicht eine einzelne Basisspalte referenziert.

Wenn die Tabelle "Shipments" relativ groß ist, ist möglicherweise ein Abruf mit Index günstiger als ein sequenzielles Durchsuchen. Um einen indizierten Abruf zu nutzen, erstellen Sie eine berechnete Spalte namens "AverageCost" für die Tabelle "Shipments" und dann einen Index für die Spalte:

```
ALTER TABLE Shipments
ADD AverageCost DECIMAL(21,13)
COMPUTE( TotalPrice / Quantity );
CREATE INDEX IDX_average_cost
ON Shipments( AverageCost ASC );
```

Die Auswahl des Typs für die berechnete Spalte ist wichtig, denn der SQL Anywhere-Optimierer ersetzt nur dann komplexe Ausdrücke durch eine berechnete Spalte, wenn der Datentyp des Ausdrucks in der Abfrage genau dem Datentyp der berechneten Spalte entspricht. Um den Typ eines Ausdrucks festzustellen, können Sie die integrierte EXPRTYPE-Funktion verwenden, die den Typ des Ausdrucks in SQL-Format zurückgibt:

```
SELECT EXPRTYPE(
'SELECT ( TotalPrice/Quantity ) AS X FROM Shipments', 1 )
FROM DUMMY;
```

Für die Tabelle "Shipments" gibt die vorherige Abfrage decimal(21,13) zurück. Während der Optimierung schreibt der SQL Anywhere-Optimierer die vorige Abfrage folgendermaßen um:

```
SELECT *
FROM Shipments
WHERE AverageCost
BETWEEN 2.00 AND 4.00;
```

In diesem Fall ist das Prädikat in der WHERE-Klausel nun sargable. Dadurch erhält der Optimierer die Möglichkeit, eine indizierte Suche zu verwenden, in dem der neue Index "IDX_average_cost" für den Zugriffsplan der Abfrage benutzt wird.

Siehe auch

- „Spezialwerte“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Abfrageprädikate“ auf Seite 292

Berechnete Spalten ändern

Sie können den in einer berechneten Spalte verwendeten Ausdruck ändern oder entfernen.

Voraussetzungen

Sie müssen Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Tabelle sowie entweder das COMMENT ANY OBJECT-Systemprivileg, das CREATE ANY OBJECT-Systemprivileg oder das CREATE ANY TABLE-Systemprivileg.
- ALTER ANY TABLE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Führen Sie eine ALTER TABLE-Anweisung ähnlich der folgenden aus, um den Ausdruck für eine berechnete Spalte zu ändern:

```
ALTER TABLE table-name
ALTER column-name
SET COMPUTE ( new-expression );
```

3. Um eine Spalte in eine normale (nicht berechnete) Spalten zu konvertieren, müssen Sie eine ALTER TABLE-Anweisung ähnlich der folgenden ausführen:

```
ALTER TABLE
table-name
ALTER column-name
DROP COMPUTE;
```

Ergebnisse

Im Fall einer Änderung der Berechnung für die Spalte wird die Spalte beim Ausführen dieser Anweisung neu berechnet.

Falls eine berechneten Spalte in eine normale (nicht berechnete) Spalte konvertiert wird, werden vorhandene Werte in der Spalte weder beim Ausführen der Anweisung geändert noch danach automatisch aktualisiert.

Beispiel

Erstellen Sie eine Tabelle namens alter_compute_test, füllen Sie sie mit Daten und führen Sie eine Auswahlabfrage in der Tabelle durch, indem Sie die folgenden Anweisungen ausführen:

```
CREATE TABLE alter_compute_test (
    c1 INT,
    c2 INT
) ;
INSERT INTO alter_compute_test (c1) VALUES(100);
SELECT * FROM alter_compute_test ;
```

Beachten Sie, dass die Spalte c2 einen NULL-Wert zurückgibt. Ändern Sie die Spalte c2, damit daraus eine berechnete Spalte wird, füllen Sie die Spalte mit Daten und führen Sie eine weitere SELECT-Anweisung in der Tabelle alter_compute_test aus.

```
ALTER TABLE alter_compute_test
ALTER c2
```

```
SET COMPUTE ( DAYS ( '2001-01-01' , CURRENT DATE ) )  
INSERT INTO alter_compute_test (c1) VALUES(200) ;  
SELECT * FROM alter_compute_test ;
```

Die Spalte c2 enthält nun die Anzahl der Tage seit 2001-01-01. Danach ändern Sie Spalte c2 so, dass sie nicht mehr eine berechnete Spalte ist:

```
ALTER TABLE alter_compute_test  
ALTER c2  
DROP COMPUTE ;
```

Siehe auch

- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- COMPUTE-Klausel [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Neuberechnung von berechneten Spalten“ auf Seite 17

Einfügungen in und Aktualisierungen von berechneten Spalten

Im Hinblick auf das Einfügen von Informationen in berechnete Spalten und ihre Aktualisierungen ist Folgendes zu beachten:

- **Direkte Einfügungen und Aktualisierungen** Eine INSERT- oder UPDATE-Anweisung kann einen Wert für eine berechnete Spalte angeben. Dieser Wert wird jedoch ignoriert. Der Server berechnet den Wert für berechnete Spalten basierend auf der COMPUTE-Spezifikation und verwendet den berechneten Wert an Stelle des in der INSERT- oder UPDATE-Anweisung angegebenen Werts.
- **Spaltenabhängigkeiten** Es wird ausdrücklich davon abgeraten, Trigger zu verwenden, um den Wert einer Spalte zu setzen, die in der Definition einer berechneten Spalte referenziert wird (z.B. einen NULL-Wert in einen Nicht-NULL-Wert ändern), weil dies dazu führen kann, dass der Wert in der berechneten Spalte nicht die gewünschte Berechnung widerspiegelt.
- **Auflisten von Spaltennamen** Sie müssen in INSERT-Anweisungen für Tabellen mit berechneten Spalten immer explizit die Spaltennamen angeben.
- **Trigger** Sie können Trigger für eine berechnete Spalte definieren, damit eine INSERT- oder UPDATE-Anweisung in diesen Spalten die Trigger auslöst.

Die LOAD TABLE-Anweisung ermöglicht die *optionale* Berechnung von berechneten Spalten. Das Unterdrücken von Berechnungen während eines Ladevorgangs beschleunigt möglicherweise die Durchführung von komplexen Entlade-/Aktualisierungssequenzen. Das kann auch nützlich sein, wenn der Wert einer berechneten Spalte konstant bleiben muss, selbst wenn sich der COMPUTE-Ausdruck auf einen nicht-deterministischen Wert bezieht, wie z.B. CURRENT TIMESTAMP.

Vermeiden Sie es, die Werte von abhängigen Spalten in Triggern zu ändern, weil durch die Wertänderung der Wert in der berechneten Spalte möglicherweise inkonsistent in Bezug zur Spaltendefinition wird.

Wenn eine berechnete Spalte x von einer Spalte y abhängt, die als nicht-NULL deklariert ist, wird der Versuch, y auf NULL zu setzen, mit einem Fehler verworfen, bevor Trigger ausgelöst werden.

Neuberechnung von berechneten Spalten

Berechnete Spaltenwerte werden automatisch vom Datenbankserver aufrechterhalten, während Zeilen eingefügt und aktualisiert werden. Bei den meisten Anwendungen sollte es niemals erforderlich sein, berechnete Spaltenwerte direkt zu aktualisieren oder einzufügen.

Berechnete Spalten werden unter den folgenden Umständen neu berechnet:

- Eine Spalte wird gelöscht, hinzugefügt oder umbenannt.
- Die Tabelle wird durch eine ALTER TABLE-Anweisung geändert, die den Datentyp oder die COMPUTE-Klausel einer Spalte ändert.
- Eine Zeile wird eingefügt.
- Eine Zeile wird aktualisiert.

Berechnete Spalten werden unter den folgenden Umständen *nicht* neu berechnet:

- Die Tabelle wird umbenannt.
- Die berechnete Spalte wird abgefragt.
- Die berechnete Spalte ist abhängig von den Werten in anderen Zeilen (mit einer Unterabfrage oder einer benutzerdefinierten Funktion) und diese Zeilen werden geändert.

Primärschlüssel

Alle Tabellen einer relationalen Datenbank müssen über einen **Primärschlüssel** verfügen. Bei einem Primärschlüssel handelt es sich um eine Spalte bzw. eine Gruppe von Spalten, die alle Zeilen in einer Tabelle eindeutig kennzeichnet. Innerhalb einer Tabelle kann jeder Primärschlüsselwert jeweils nur einmal vorkommen und keine Spalte in einem Primärschlüssel darf den Wert NULL enthalten.

Nur Basistabellen und globale temporäre Tabellen können Primärschlüssel haben. Bei deklarierten temporären Tabellen können Sie einen eindeutigen Index für eine Gruppe von NOT NULL-Spalten erstellen, um die Semantik eines Primärschlüssels zu imitieren.

Es wird empfohlen, keine angenäherten Datentypen wie FLOAT und DOUBLE für Primärschlüssel oder Spalten mit Eindeutigkeits-Integritätsregeln zu verwenden. Bei angenäherten numerischen Datentypen können nach arithmetischen Vorgängen Rundungsfehler auftreten.

Außerdem können Sie mithilfe der CLUSTERED-Klausel angeben, ob der Primärschlüsselindex als Clustered-Index erstellt werden soll.

Spaltenreihenfolge bei Primärschlüsseln mit mehreren Spalten

Die Spaltenreihenfolge für Primärschlüssel wird durch die Reihenfolge der Spalten bestimmt, die in der Primärschlüsseldeklaration der CREATE TABLE-Anweisung (oder ALTER TABLE) festgelegt ist. Sie können auch die Sortierreihenfolge (aufsteigend oder absteigend) für jede einzelne Spalte angeben. Diese Spezifikationen der Sortierreihenfolge werden vom Datenbankserver bei der Erstellung des Primärschlüsselindexes verwendet.

Die Reihenfolge der Spalten in einem Primärschlüssel bestimmt nicht die Reihenfolge der Spalten in Integritätsregeln zur Erhaltung der referenziellen Integrität. Mithilfe einer Fremdschlüsseldeklaration können Sie eine andere Spaltenreihenfolge und unterschiedliche Sortierreihenfolgen angeben.

Beispiele

In der SQL Anywhere-Beispieldatenbank enthält die Employees-Tabelle personenbezogene Daten der Mitarbeiter. In dieser Tabelle gibt es eine Primärschlüsselspalte mit der Bezeichnung EmployeeID, in der eine jedem Mitarbeiter eindeutig zugewiesene Identifikationsnummer gespeichert wird. Im Allgemeinen werden ID-Nummern in speziellen Spalten als Primärschlüssel gespeichert. Dieses Verfahren hat Vorteile gegenüber Spalten mit Namen bzw. anderen Bezeichnungen, da diese nicht immer eindeutig sind.

Ein umfangreicherer Primärschlüssel ist in der Tabelle SalesOrderItems der SQL Anywhere-Beispieldatenbank enthalten. Die Tabelle enthält Daten über einzelne vom Unternehmen bestellte Artikel und besteht aus den folgenden Spalten:

- **ID** Eine Bestellnummer, mit der die Bestellung identifiziert wird, zu welcher der Artikel gehört
- **LineID** Eine Zeilennummer, mit der jedes Element einer Bestellung gekennzeichnet wird
- **ProductID** Eine Produktnummer, mit der jedes bestellte Produkt gekennzeichnet wird
- **Quantity** Eine Menge, mit der angezeigt wird, wie viele Artikel bestellt wurden
- **ShipDate** Ein Lieferdatum, mit dem angegeben wird, wann die Artikel ausgeliefert wurden

Ein bestimmter Artikel einer Bestellung wird durch die Bestellung, zu der er gehört, und eine Zeilennummer auf dieser Bestellung gekennzeichnet. Diese beiden Nummern werden in den Spalten ID und LineID gespeichert. So haben manche Artikel unter Umständen den gleichen Wert für ID (beispielsweise bei einer Bestellung mit mehr als einem Artikel) und mit hoher Wahrscheinlichkeit jeweils den gleichen Wert für LineID (alle Artikel in der ersten Zeile einer Bestellung haben eine LineID von 1). Bei keinem der Artikel jedoch sind beide Werte gleich. Daher wird der Primärschlüssel aus diesen beiden Spalten zusammengesetzt.

Siehe auch

- „Clustered-Indizes“ auf Seite 31
- „Primärschlüssel erzwingen Entitätsintegrität“ auf Seite 868

Primärschlüssel verwalten (Sybase Central)

Sie können Primärschlüssel in Sybase Central verwalten, um die Abfrageperformance für die Tabelle zu verbessern.

Voraussetzungen

Sie müssen Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- ALTER ANY OBJECT-Systemprivileg
- Systemprivilegien ALTER ANY INDEX *und* ALTER ANY TABLE-
- ALTER-*und* REFERENCES-Privileg für die Tabelle sowie entweder das COMMENT ANY OBJECT-Systemprivileg, das CREATE ANY OBJECT-Systemprivileg oder das CREATE ANY TABLE-Systemprivileg.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Tabellen**.
3. Rechtsklicken Sie auf die Tabelle und wählen Sie eine der folgenden Optionen:

Option	Aktion
Primärschlüssel erstellen oder ändern	Klicken Sie auf Primärschlüssel definieren und befolgen Sie die Anweisungen des Assistenten zum Definieren eines Primärschlüssels .
Primärschlüssel löschen	Entfernen Sie im Fensterausschnitt Spalten der Tabelle das Häkchen aus der Spalte PSch und klicken Sie dann auf Speichern .

Ergebnisse

Ein Primärschlüssel wird hinzugefügt, geändert oder gelöscht.

Siehe auch

- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ALTER INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Primärschlüssel verwalten (SQL)“ auf Seite 19
- „Primärschlüssel erzwingen Entitätsintegrität“ auf Seite 868

Primärschlüssel verwalten (SQL)

Sie können Primärschlüssel in SQL verwalten, um die Abfrageperformance für die Tabelle zu verbessern.

Voraussetzungen

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

Spalten im Primärschlüssel dürfen nicht NULL enthalten.

Aufgabe

- Stellen Sie eine Verbindung mit der Datenbank her.

Option	Aktion
Primärschlüssel erstellen	Führen Sie die Anweisung <code>ALTER TABLE <i>table-name</i> ADD PRIMARY KEY (<i>column-name</i>)</code> aus.
Primärschlüssel löschen	Führen Sie die Anweisung <code>ALTER TABLE <i>table-name</i> DROP PRIMARY KEY</code> aus.
Primärschlüssel ändern	Löschen Sie den bestehenden Primärschlüssel, bevor Sie einen neuen Primärschlüssel für die Tabelle erstellen.

Ergebnisse

Ein Primärschlüssel wird hinzugefügt, gelöscht oder geändert.

Beispiel

Die folgende Anweisung erstellt eine Tabelle namens "Skills" und ordnet die SkillID-Spalte als Primärschlüssel zu:

```
CREATE TABLE Skills (  
    SkillID INTEGER NOT NULL,  
    SkillName CHAR( 20 ) NOT NULL,  
    SkillType CHAR( 20 ) NOT NULL,  
    PRIMARY KEY( SkillID )  
);
```

Die Primärschlüsselwerte müssen für jede Zeile der Tabelle eindeutig sein. In diesem Fall bedeutet dies, dass es nicht mehr als eine Zeile mit einer bestimmten Skill-ID geben darf. Jede Zeile in einer Tabelle wird durch ihren Primärschlüssel eindeutig gekennzeichnet.

Zum Ändern des Primärschlüssels, damit die Spalten "SkillID" und "Skillname" gemeinsam für den Primärschlüssel verwendet werden, müssen Sie zuerst den Primärschlüssel löschen, den Sie erstellt haben, und dann den neuen Primärschlüssel hinzufügen:

```
ALTER TABLE Skills DELETE PRIMARY KEY;  
ALTER TABLE Skills ADD PRIMARY KEY ( SkillID, SkillName );
```

Siehe auch

- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ALTER INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Primärschlüssel verwalten (Sybase Central)“ auf Seite 18
- „Primärschlüssel erzwingen Entitätsintegrität“ auf Seite 868

Fremdschlüssel

Ein Fremdschlüssel besteht aus einer Spalte oder einer Gruppe von Spalten. Er stellt eine Referenz auf eine Zeile in der Primärtabelle mit dem angegebenen Schlüsselwert dar. Fremdschlüssel können nur für Basistabellen verwendet werden, nicht jedoch für temporäre Tabellen, globale temporäre Tabellen, Ansichten oder materialisierte Ansichten. Ein Fremdschlüssel wird manchmal auch als **Integritätsregel zur Erhaltung der referenziellen Integrität** bezeichnet, da die Basistabelle, die den Fremdschlüssel enthält, als **referenzierende Tabelle** bezeichnet wird und die Tabelle, die den Primärschlüssel enthält, als **referenzierte Tabelle**.

Wenn der Fremdschlüssel nullwertfähig ist, ist die Beziehung optional, da die Fremdschlüsselzeile bestehen kann, ohne dass ein übereinstimmender Primärschlüsselwert in der referenzierten Tabelle vorhanden ist, weil weder Primärschlüssel noch Spalten mit UNIQUE-Integritätsregeln NULL sein können. Wenn Fremdschlüsselspalten als NOT NULL deklariert sind, ist die Beziehung obligatorisch: Jede Zeile in der referenzierenden Tabelle muss einen Fremdschlüsselwert enthalten, der als Primärschlüssel in der referenzierten Tabelle vorhanden ist.

Fremdschlüssel und verwaiste Zeilen

Um die referenzielle Integrität zu gewährleisten, darf die Datenbank keine Nicht-NULL-Fremdschlüsselwerte enthalten, für die keine Übereinstimmung vorhanden ist. Eine Fremdschlüsselzeile, die die referenzielle Integrität verletzt, wird als **Waise** bezeichnet, weil sie mit keinem Primärschlüsselwert in der referenzierten Tabelle übereinstimmt. Eine Waise kann auf folgende Arten erstellt werden:

- Durch Einfügen oder Aktualisieren einer Zeile in der referenzierenden Tabelle mit einem Nicht-NULL-Wert für die Fremdschlüsselspalte, der mit keinem Primärschlüsselwert in der referenzierten Tabelle übereinstimmt.
- Durch Aktualisieren oder Löschen einer Zeile in der Primärtabelle, wenn dies dazu führt, dass mindestens eine Zeile in der referenzierenden Tabelle nicht länger einen übereinstimmenden Primärschlüsselwert enthält.

SQL Anywhere verhindert Verletzungen der referenziellen Integrität, indem es die Erstellung von Waisenzeilen verhindert.

Zusammengesetzte Fremdschlüssel

SQL Anywhere unterstützt außerdem Primär- und Fremdschlüssel mit mehreren Spalten, sogenannte **zusammengesetzte Schlüssel**. Bei einem zusammengesetzten Fremdschlüssel bedeutet NULL weiterhin, dass keine Übereinstimmung vorhanden ist, aber wie eine Waise identifiziert wird, hängt davon ab, wie Integritätsregeln zur Erhaltung der referenziellen Integrität in der MATCH-Klausel festgelegt sind.

Fremdschlüsselindizes und Sortierreihenfolge

Wenn Sie einen Fremdschlüssel erstellen, wird automatisch ein Index für den Schlüssel erstellt. Dabei muss weder die Reihenfolge der Fremdschlüsselspalten die Reihenfolge der Spalten im Primärschlüssel widerspiegeln noch die Sortierung des Primärschlüsselindexes mit der Sortierung des Fremdschlüsselindexes übereinstimmen. Die Sortierung (aufsteigend oder absteigend) jeder indizierten Spalte kann angepasst werden, um sicherzustellen, dass die Sortierung des Fremdschlüsselindexes mit der für bestimmte SQL-Abfragen in Ihrer Anwendung erforderlichen Sortierung übereinstimmt, die in den

ORDER BY-Klauseln der betreffenden Anweisungen angegeben ist. Sie können die Sortierung für jede Spalte angeben, wenn Sie die Fremdschlüssel-Integritätsregel festlegen.

Beispiel 1

Die SQL Anywhere-Beispieldatenbank enthält eine Tabelle mit Mitarbeiterdaten und eine weitere mit Abteilungsdaten. Die Tabelle "Departments" umfasst die folgenden Spalten:

- **DepartmentID** Eine ID-Nummer für die Abteilung. Hierbei handelt es sich um den Primärschlüssel der Tabelle.
- **DepartmentName** Der Name der Abteilung
- **DepartmentHeadID** Die Mitarbeiter-ID des Abteilungsleiters

Um den Namen der Abteilung eines bestimmten Mitarbeiters zu finden, muss der Name der Mitarbeiterabteilung nicht in die Tabelle "Employees" eingegeben werden. Die Tabelle "Employees" enthält die Spalte "DepartmentID" mit einem Wert, der einem der "DepartmentID"-Werte der Tabelle "Departments" entspricht.

Die Spalte "DepartmentID" der Tabelle "Employees" stellt einen Fremdschlüssel für die Tabelle "Departments" dar. Ein Fremdschlüssel referenziert eine bestimmte Zeile in der Tabelle, die den entsprechenden Primärschlüssel enthält.

Die Tabelle "Employees" (die den Fremdschlüssel in der Beziehung enthält) wird daher als **Fremdtabelle** oder **referenzierende Tabelle** bezeichnet. Die Tabelle "Departments" (die den referenzierten Primärschlüssel enthält) wird als **Primärtabelle** oder **referenzierte Tabelle** bezeichnet.

Beispiel 2

Führen Sie die folgende Anweisung aus, um einen zusammengesetzten Primärschlüssel zu erstellen.

```
CREATE TABLE pt(  
    pk1 INT NOT NULL,  
    pk2 INT NOT NULL,  
    str VARCHAR(10),  
    PRIMARY KEY ( pk1, pk2 ));
```

Die folgenden Anweisungen erstellen einen Fremdschlüssel, der eine andere Spaltenreihenfolge aufweist als der Primärschlüssel, und eine andere Sortierung für die Fremdschlüsselspalten, die verwendet wird, um den Fremdschlüsselindex zu erstellen.

```
CREATE TABLE ft1(  
    fpk INT PRIMARY KEY,  
    ref1 INT,  
    ref2 INT );  
  
ALTER TABLE ft1 ADD FOREIGN KEY ( ref2 ASC, ref1 DESC)  
    REFERENCES pt ( pk2, pk1 ) MATCH SIMPLE;
```

Führen Sie die folgenden Anweisungen aus, um einen Fremdschlüssel zu erstellen, der dieselbe Spaltenreihenfolge aufweist wie der Primärschlüssel, aber eine andere Sortierung für den Fremdschlüsselindex. Außerdem wird im Beispiel die MATCH FULL-Klausel verwendet, um anzugeben, dass es zu verwaisten Zeilen kommt, wenn beide Spalten NULL sind. Die UNIQUE-Klausel erzwingt eine Eins-zu-Eins-Relation zwischen der **pt**-Tabelle und der **ft2**-Tabelle für Spalten, die nicht NULL sind.


```
CREATE TABLE ft2(  
    fpk INT PRIMARY KEY,  
    ref1 INT,  
    ref2 INT );  
  
ALTER TABLE ft2 ADD FOREIGN KEY ( ref1, ref2 DESC )  
    REFERENCES pt ( pk1, pk2 ) MATCH UNIQUE FULL;
```

Siehe auch

- [FOREIGN KEY-Klausel, CREATE TABLE-Anweisung \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- [„Referenzielle Integrität“ auf Seite 869](#)

Fremdschlüssel erstellen (Sybase Central)

Sie erstellen eine Fremdschlüsselbeziehung zwischen Tabellen. Die Fremdschlüsselbeziehung dient als Integritätsregel. Wenn Sie neue Zeilen in die untergeordnete Tabelle einfügen, prüft der Datenbankserver, ob der Wert, den Sie in die Fremdschlüsselspalte einfügen, mit einem Wert im Primärschlüssel der Primärtabelle übereinstimmt.

Voraussetzungen

Sie müssen das SELECT-Privileg für die Tabelle oder das SELECT ANY TABLE-Systemprivileg haben.

Außerdem müssen Sie Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Tabelle sowie entweder das COMMENT ANY OBJECT-Systemprivileg, das CREATE ANY OBJECT-Systemprivileg oder das CREATE ANY TABLE-Systemprivileg.
- ALTER ANY TABLE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Kontext und Bemerkungen

Es ist nicht erforderlich, beim Erstellen einer Fremdtabelle einen Fremdschlüssel zu erstellen. Der Fremdschlüssel wird automatisch erstellt.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Tabellen**.
3. Wählen Sie die Tabelle aus, für die Sie einen Fremdschlüssel erstellen möchten.
4. Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **Integritätsregeln**.
5. Erstellen Sie einen Fremdschlüssel:
 - a. Klicken Sie auf **Datei » Neu » Fremdschlüssel**.

- b. Befolgen Sie die Anweisungen des **Assistenten zum Definieren eines Fremdschlüssels**.

Ergebnisse

In Sybase Central erscheint der Fremdschlüssel einer Tabelle auf der Registerkarte **Integritätsregeln** im rechten Fensterausschnitt, wenn eine Tabelle ausgewählt ist. Die Definition der Tabelle wird aktualisiert, um die Fremdschlüsseldefinition einzubeziehen.

Nächste Schritte

Wenn Sie einen Fremdschlüssel mit dem Assistenten erstellen, können Sie die Eigenschaften für den Fremdschlüssel einstellen. Um Eigenschaften anzuzeigen, nachdem der Fremdschlüssel erstellt wurde, wählen Sie den Fremdschlüssel auf der Registerkarte **Integritätsregeln** aus und klicken Sie dann auf **Datei » Eigenschaften**.

Sie können die Eigenschaften eines referenzierenden Fremdschlüssels anzeigen, indem Sie die Tabelle auf der Registerkarte **Referenzierende Integritätsregeln** auswählen und dann auf **Datei » Eigenschaften** klicken.

Um eine Liste der Tabellen anzuzeigen, die eine bestimmte Tabelle referenzieren, wählen Sie die Tabelle unter **Tabellen** aus und klicken Sie anschließend im rechten Fensterausschnitt auf die Registerkarte **Referenzierende Integritätsregeln**.

Siehe auch

- „Fremdschlüssel erstellen (Sybase Central)“ auf Seite 23
- „CREATE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Fremdschlüssel erstellen (SQL)

In Interactive SQL können Sie Fremdschlüssel mit den Anweisungen CREATE TABLE und ALTER TABLE erstellen und bearbeiten. Mit diesen Anweisungen können Sie viele Tabellenattribute einstellen, einschließlich Spaltenintegritätsregeln und Prüfungen.

Voraussetzungen

Welche Privilegien zum Erstellen eines Fremdschlüssels erforderlich sind, hängt davon ab, wer Tabelleneigentümer ist. Es gilt Folgendes:

- **Sie sind Eigentümer der referenzierten (Primärschlüssel) und referenzierenden (Fremdschlüssel) Tabelle.** Sie benötigen keine Privilegien.
- **Sie sind Eigentümer der referenzierenden Tabelle, jedoch nicht der referenzierten Tabelle.** Sie benötigen entweder das REFERENCES-Privileg für die Tabelle oder das CREATE ANY INDEX-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg.

- **Sie sind Eigentümer der referenzierten Tabelle, jedoch nicht der referenzierenden Tabelle.**
 - Sie benötigen entweder das ALTER ANY OBJECT-Systemprivileg oder das ALTER ANY TABLE-Systemprivileg.
 - Oder Sie müssen das ALTER-Privileg für die Tabelle haben sowie das COMMENT ANY OBJECT-Systemprivileg, das CREATE ANY OBJECT-Systemprivileg oder das CREATE ANY TABLE-Systemprivileg.
 - Außerdem müssen Sie das SELECT-Privileg für die Tabelle oder das SELECT ANY TABLE-Systemprivileg haben.
- **Sie sind bei keiner Tabelle Eigentümer.**
 - Sie benötigen entweder das REFERENCES-Privileg für die Tabelle oder das CREATE ANY INDEX-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg.
 - Sie benötigen entweder das ALTER ANY OBJECT-Systemprivileg oder das ALTER ANY TABLE-Systemprivileg.
 - Oder Sie müssen das ALTER-Privileg für die Tabelle haben sowie das COMMENT ANY OBJECT-Systemprivileg, das CREATE ANY OBJECT-Systemprivileg oder das CREATE ANY TABLE-Systemprivileg.
 - Außerdem müssen Sie das SELECT-Privileg für die Tabelle oder das SELECT ANY TABLE-Systemprivileg haben.

Sie müssen das SELECT-Privileg für die Tabelle oder das SELECT ANY TABLE-Systemprivileg haben.

Außerdem müssen Sie Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Tabelle sowie entweder das COMMENT ANY OBJECT-Systemprivileg, das CREATE ANY OBJECT-Systemprivileg oder das CREATE ANY TABLE-Systemprivileg.
- ALTER ANY TABLE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Kontext und Bemerkungen

Es ist nicht erforderlich, beim Erstellen einer Fremdtabelle einen Fremdschlüssel zu erstellen. Der Fremdschlüssel wird automatisch erstellt.

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Führen Sie eine ALTER TABLE-Anweisung ähnlich der folgenden aus:

```
ALTER TABLE table-name ADD FOREIGN KEY foreign-key-name
( column-name ASC ) REFERENCES table-name ( column-name )
```

Ergebnisse

Die Definition der Tabelle wird aktualisiert, um die Fremdschlüsseldefinition einzubeziehen.

Beispiel

Im folgenden Beispiel erstellen Sie eine Tabelle namens "Skills", die eine Liste möglicher Kompetenzen enthält. Anschließend erstellen Sie eine Tabelle namens "EmployeeSkills", die eine Fremdschlüsselbeziehung zur Tabelle "Skills" hat. Beachten Sie, dass "EmployeeSkills.SkillID" eine Fremdschlüsselbeziehung mit der Primärschlüsselspalte (Id) der Tabelle "Skills" hat.

```
CREATE TABLE Skills (
    Id INTEGER PRIMARY KEY,
    SkillName CHAR(40),
    Description CHAR(100)
);
CREATE TABLE EmployeeSkills (
    EmployeeID INTEGER NOT NULL,
    SkillID INTEGER NOT NULL,
    SkillLevel INTEGER NOT NULL,
    PRIMARY KEY( EmployeeID ),
    FOREIGN KEY (SkillID) REFERENCES Skills ( Id )
);
```

Sie können einer Tabelle auch einen Fremdschlüssel hinzufügen, nachdem sie erstellt wurde, indem Sie die Anweisung ALTER TABLE benutzen. Im folgenden Beispiel erstellen Sie Tabellen ähnlich wie im vorherigen Beispiel, nur fügen Sie jetzt den Fremdschlüssel erst nach dem Erstellen der Tabelle hinzu.

```
CREATE TABLE Skills2 (
    ID INTEGER PRIMARY KEY,
    SkillName CHAR(40),
    Description CHAR(100)
);
CREATE TABLE EmployeeSkills2 (
    EmployeeID INTEGER NOT NULL,
    SkillID INTEGER NOT NULL,
    SkillLevel INTEGER NOT NULL,
    PRIMARY KEY( EmployeeID ),
);
ALTER TABLE EmployeeSkills2
    ADD FOREIGN KEY SkillFK ( SkillID )
    REFERENCES Skills2 ( ID );
```

Sie können Eigenschaften für den Fremdschlüssel festlegen, wenn Sie ihn erstellen. Mit der folgenden Anweisung wird z.B. derselbe Fremdschlüssel erstellt wie in Beispiel 2, aber er wird als NOT NULL definiert und erhält Einschränkungen für die Aktualisierung und das Löschen von Daten.

```
ALTER TABLE Skills2
    ADD NOT NULL FOREIGN KEY SkillFK ( SkillID )
    REFERENCES Skills2 ( ID )
    ON UPDATE RESTRICT
    ON DELETE RESTRICT;
```

Namen für Fremdschlüsselspalten werden den Namen für Primärschlüsselspalten immer entsprechend der Position in den beiden Listen zugeordnet. Wenn die Spaltennamen der Primärtabelle bei der Festlegung des Fremdschlüssels nicht angegeben werden, dann werden die Primärschlüsselspalten verwendet. Nehmen wir beispielsweise an, dass Sie zwei Tabellen folgendermaßen erstellen:

```
CREATE TABLE Table1( a INT, b INT, c INT, PRIMARY KEY ( a, b ) );
CREATE TABLE Table2( x INT, y INT, z INT, PRIMARY KEY ( x, y ) );
```

Anschließend erstellen Sie einen Fremdschlüssel fk1, indem Sie genau angeben, wie die Spalten zwischen den Tabellen gepaart werden sollen, und zwar folgendermaßen:

```
ALTER TABLE Table2 ADD FOREIGN KEY fk1( x,y ) REFERENCES Table1( a, b );
```

Unter Verwendung der folgenden Anweisung erstellen Sie einen zweiten Fremdschlüssel fk2, indem Sie nur die Fremdtabellenspalten angeben. Der Datenbankserver paart automatisch diese zwei Spalten mit den ersten zwei Spalten im Primärschlüssel der Primärtabelle.

```
ALTER TABLE Table2 ADD FOREIGN KEY fk2( x, y ) REFERENCES Table1;
```

Unter Verwendung der folgenden Anweisung erstellen Sie einen Fremdschlüssel, ohne die Primär- oder die Fremdtabelle anzugeben:

```
ALTER TABLE Table2 ADD FOREIGN KEY fk3 REFERENCES Table1;
```

Da Sie keine referenzierenden Spalten angegeben haben, sucht der Datenbankserver nach Spalten in der Fremdtabelle (Table2) mit demselben Namen wie Spalten in der Primärtabelle (Table1). Wenn sie vorhanden sind, stellt der Datenbankserver sicher, dass die Datentypen übereinstimmen und erstellt dann den Fremdschlüssel unter Verwendung dieser Spalten. Wenn keine Spalten vorhanden sind, werden sie in Table2 erstellt. In diesem Beispiel hat Table2 *keine* Spalten namens a und b, daher werden sie mit denselben Datentypen wie Table1.a und Table1.b erstellt. Diese automatisch erstellten Spalten können nicht Teil des Primärschlüssels der Fremdtabelle werden.

Siehe auch

- „Fremdschlüssel erstellen (Sybase Central)“ auf Seite 23
- „CREATE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Indizes

Ein **Index** bietet eine Sortierfolge für die Zeilen nach einer oder mehreren Spalten einer Tabelle. Ein Index funktioniert wie ein Telefonbuch, in dem Teilnehmer zunächst nach Nachnamen sortiert werden, dann nach Vornamen, wenn gleiche Nachnamen auftauchen. Diese Sortierung beschleunigt die Suche nach Telefonnummern zu bestimmten Nachnamen, aber sie hilft nicht bei der Suche nach Telefonnummern unter einer bestimmten Adresse. Genauso ist ein Datenbankindex nur für Suchen nach einer oder mehreren Spalten hilfreich.

Indizes sind besonders sinnvoll, wenn die Tabellen größer werden. Die durchschnittliche Zeit zur Suche nach einer Telefonnummer mit einer bestimmten Adresse nimmt mit wachsender Größe des Telefonbuchs zu, während die Suche z.B. nach einer Telefonnummer von K. Kaminski unabhängig von der Größe des Telefonbuchs gleich lange dauert.

Der Optimierer verwendet automatisch Indizes, um wo immer möglich die Performance von Datenbankabfragen zu steigern. Der Index wird außerdem automatisch aktualisiert, wenn Zeilen gelöscht, aktualisiert oder eingefügt werden. Auch wenn Sie während des Erstellens Ihrer Abfrage Indizes referenzieren können, indem Sie Index-Hints verwenden, so gibt es dafür keine Notwendigkeit.

Es gibt einige Nachteile bei der Erstellung von Indizes. Vor allem müssen Indizes mit der Tabelle selbst gepflegt werden, wenn Daten in einer Spalte geändert werden, sodass die Performance von Einfügungen, Aktualisierungen und Löschvorgängen durch Indizes beeinträchtigt sein kann. Daher sollten unnötige Indizes gelöscht werden. Verwenden Sie zur Identifizierung unnötiger Indizes den Indexberater.

Ermitteln der zu erstellenden Indizes

Die Auswahl eines entsprechenden Satzes von Indizes für eine Datenbank ist ein wichtiger Bestandteil bei der Optimierung der Performance. Das Festlegen der entsprechenden Indizes kann aber auch ein schwieriges Problem darstellen.

Es gibt keine einfache Formel, mit der Sie herausfinden, ob ein Index erstellt werden sollte. Sie müssen die Vorteile eines Abrufs mit Index und die Nachteile durch die gestiegenen Wartungskosten für diesen Index gegeneinander abwägen. Die folgenden Faktoren können Ihnen bei der Entscheidungsfindung behilflich sein:

- **Schlüssel und eindeutige Spalten** SQL Anywhere erstellt automatisch Indizes für Primärschlüssel, Fremdschlüssel und eindeutige Spalten. Sie sollten für diese Spalten keine zusätzlichen Indizes erstellen. Die Ausnahme hierbei sind zusammengesetzte Schlüssel, die manchmal durch zusätzliche Indizes verbessert werden.
- **Häufigkeit der Suche** Wenn eine bestimmte Spalte häufig durchsucht wird, können Sie mit einem Index für diese Spalte eine Verbesserung der Performance erreichen. Die Erstellung eines Indexes für eine Spalte, die selten durchsucht wird, ist möglicherweise nicht sinnvoll.
- **Größe der Tabelle** Indizes für relativ große Tabellen mit vielen Zeilen bringen mehr Vorteile als Indizes für relativ kleine Tabellen. Beispielsweise wird ein Index für eine Tabelle mit nur 20 Zeilen wahrscheinlich keinen Nutzen bringen, da sequenzielles Durchsuchen nicht länger dauern würde als eine Indexsuche.
- **Anzahl der Aktualisierungen** Ein Index wird jedesmal aktualisiert, wenn eine Zeile in die Tabelle eingefügt bzw. aus der Tabelle gelöscht und eine Spalte mit Index aktualisiert wird. Ein Index für eine Spalte verlangsamt die Performance von Einfügungen, Änderungen und Löschvorgängen. Eine Datenbank, die häufig aktualisiert wird, sollte weniger Indizes haben als eine, die nur gelesen wird.
- **Hinweise zum Speicherplatz** Indizes erfordern Speicherplatz in der Datenbank. Wenn die Datenbankgröße ein primäres Anliegen ist, sollten sie Indizes zurückhaltend erstellen.
- **Datenverteilung** Wenn eine Indexsuche zu viele Werte zurückgibt, ist sie kostenträchtiger als sequenzielles Durchsuchen. SQL Anywhere verwendet den Index nicht, wenn diese Bedingung erkannt wird. SQL Anywhere würde beispielsweise keinen Index für eine Spalte mit nur zwei Werten verwenden, wie die Tabelle "Employees.Sex" in der SQL Anywhere-Beispieldatenbank. Aus diesem Grund sollten Sie keinen Index für eine Tabelle erstellen, die lediglich einige unterschiedliche Werte enthält.

Beim Erstellen von Indizes bestimmt die Reihenfolge, in der Sie die Spalten angeben, auch die Reihenfolge, in der die Spalten im Index erscheinen. Doppelte Verweise auf Spaltennamen in der Indexdefinition sind nicht erlaubt.

Hinweis

Der Indexberater leitet Sie bei der richtigen Auswahl von Indizes. Er analysiert entweder eine einzelne Abfrage oder eine Reihe von Vorgängen und empfiehlt die Indizes, die Sie Ihrer Datenbank hinzufügen sollten. Er benachrichtigt Sie auch, welche Indizes nicht verwendet werden.

Indizes für temporäre Tabellen

Sie können Indizes sowohl für lokale als auch globale temporäre Tabellen erstellen. Es kann sinnvoll sein, einen Index für eine temporäre Tabelle zu erstellen, wenn die Tabelle voraussichtlich groß wird und mehrmals in sortierter Folge oder in einem Join auf sie zugegriffen werden soll. Andernfalls wird jegliche Performance-Verbesserung bei Abfragen von den Kosten des Erstellens und Löschens von Indizes aufgewogen.

Siehe auch

- „Zusammengesetzte Indizes“ auf Seite 29
- „Indizes“ auf Seite 27
- Die richtige Auswahl der Indizes kann sich entscheidend auf die Performance auswirken auf Seite 232
- „Abrufen von Empfehlungen des Indexberaters für eine Abfrage“ auf Seite 165
- „Tipp: Indizes effizient verwenden“ auf Seite 231
- „Abrufen von Empfehlungen des Indexberaters für eine Abfrage“ auf Seite 165
- WITH (Tabellen-Hint)-Klausel, FROM-Klausel [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Zusammengesetzte Indizes

Ein Index kann eine, zwei oder mehr Spalten enthalten. Ein Index für eine oder mehr Spalten wird **zusammengesetzter Index** genannt. Mit der folgenden Anweisung wird beispielsweise ein zweispaltiger zusammengesetzter Index erstellt.

```
CREATE INDEX name  
ON Employees ( Surname, GivenName );
```

Ein zusammengesetzter Index ist sinnvoll, wenn die erste Spalte allein keine hohe Selektivität bietet. Ein zusammengesetzter Index für Surname und GivenName ist beispielsweise nützlich, wenn viele Angestellte den gleichen Nachnamen ("Surname") haben. Für EmployeeID und Surname wäre ein zusammengesetzter Index nicht sinnvoll, da jeder Mitarbeiter eine eindeutige Kennung besitzt, sodass die Spalte Surname keine zusätzliche Selektivität bringt.

Durch zusätzliche Spalten in einem Index können Sie zwar Ihre Suche eingrenzen, doch ist ein zweispaltiger Index nicht dasselbe wie zwei separate Indizes. Ein zusammengesetzter Index ist wie ein Adressbuch aufgebaut: Zuerst werden die Adressaten nach Nachnamen und dann alle Personen mit demselben Nachnamen nach ihrem Vornamen sortiert. Ein Adressbuch ist nützlich, wenn Sie den Nachnamen kennen. Es ist sogar noch nützlicher, wenn Sie den Vornamen und den Nachnamen kennen - doch es nützt Ihnen nichts, wenn Sie nur den Vornamen, aber nicht den Nachnamen kennen.

Spaltenreihenfolge

Wenn Sie zusammengesetzte Indizes erstellen, sollten Sie sich die Reihenfolge der Spalten genau überlegen. Zusammengesetzte Indizes sind sinnvoll bei Suchen in allen Spalten des Indexes oder nur in der ersten Spalte, nicht aber für die Suche nur in nachfolgenden Spalten.

Wenn Sie wahrscheinlich viele Suchvorgänge nur für eine Spalte durchführen, dann sollte diese Spalte die erste im zusammengesetzten Index sein. Wenn Sie wahrscheinlich einzelne Suchvorgänge für beide Spalten eines zweispaltigen Indexes durchführen, ist es eventuell ratsam, einen zweiten Index zu erstellen, der nur die zweite Spalte enthält.

Angenommen, Sie erstellen beispielsweise einen zusammengesetzten Index für zwei Spalten. Eine Spalte enthält die Vornamen der Mitarbeiter, die andere ihre Nachnamen. Sie könnten einen Index erstellen, der die Vornamen und dann die Nachnamen enthält. Oder Sie indizieren die Nachnamen und dann die Vornamen. Obwohl diese beiden Indizes die Informationen in beiden Spalten ordnen, haben sie unterschiedliche Funktionen.

```
CREATE INDEX IX_GivenName_Surname
ON Employees ( GivenName, Surname );
CREATE INDEX IX_Surname_GivenName
ON Employees ( Surname, GivenName );
```

Angenommen, Sie möchten nach dem Vornamen John suchen. Der einzige nützliche Index ist der, der den Vornamen in der ersten Spalte des Indexes enthält. Der Index, der den Nachnamen vor dem Vornamen enthält, ist unnütz, weil jemand mit dem Vornamen John an einer beliebigen Stelle im Index auftreten kann.

Wenn Sie voraussichtlich nach Mitarbeitern ausschließlich anhand ihres Vornamens oder ausschließlich anhand ihres Nachnamens suchen werden, ist es sinnvoll, beide Indizes zu erstellen.

Alternativ können Sie zwei Indizes erstellen, die jeweils nur eine der Spalten indizieren. Bedenken Sie allerdings, dass SQL Anywhere nur einen Index verwendet, um auf eine Tabelle zuzugreifen, während eine einzelne Abfrage verarbeitet wird. Auch wenn Sie beide Namen kennen, ist es wahrscheinlich, dass SQL Anywhere zusätzliche Zeilen lesen muss, um die Zeilen mit dem korrekten zweiten Namen zu finden.

Wenn Sie wie im Beispiel oben einen Index mit der CREATE INDEX-Anweisung erstellen, ist die Reihenfolge der Spalten so, wie sie in Ihrer Anweisung angeführt ist.

Zusammengesetzte Indizes und ORDER BY

Standardmäßig werden die Spalten eines Indexes aufsteigend sortiert, sie können aber optional durch Angabe von DESC in der CREATE INDEX-Anweisung absteigend sortiert werden.

SQL Anywhere kann einen Index zur Optimierung einer ORDER BY-Abfrage verwenden, so lange die ORDER BY-Klausel nur Spalten enthält, die in diesen Index mit einbezogen sind. Außerdem müssen die Spalten im Index in genau derselben Weise (oder genau andersherum) sortiert sein wie in der ORDER BY-Klausel. Bei einspaltigen Indizes kann die Sortierreihenfolge stets optimiert werden, jedoch erfordern zusammengesetzte Indizes etwas mehr Überlegung. In der nachfolgenden Tabelle werden die Möglichkeiten für einen zweisepaltigen Index aufgeführt.

Indexspalten	Optimierbare ORDER BY-Abfragen	Nicht optimierbare ORDER BY-Abfragen
ASC, ASC	ASC, ASC oder DESC, DESC	ASC, DESC oder DESC, ASC
ASC, DESC	ASC, DESC oder DESC, ASC	ASC, ASC oder DESC, DESC
DESC, ASC	DESC, ASC oder ASC, DESC	ASC, ASC oder DESC, DESC
DESC, DESC	DESC, DESC oder ASC, ASC	ASC, DESC oder DESC, ASC

Ein Index mit mehr als zwei Spalten folgt der gleichen allgemeinen Regel wie oben. Angenommen, Sie haben beispielsweise den folgenden Index:

```
CREATE INDEX idx_example
ON table1 ( col1 ASC, col2 DESC, col3 ASC );
```

In diesem Fall können die folgenden Abfragen optimiert werden:

```
SELECT col1, col2, col3 FROM table1
ORDER BY col1 ASC, col2 DESC, col3 ASC;

SELECT col1, col2, col3 FROM example
ORDER BY col1 DESC, col2 ASC, col3 DESC;
```

Der Index wird nicht zum Optimieren einer Abfrage verwendet, die ein anderes Muster von ASC und DESC in der ORDER BY-Klausel aufweist. Die folgende Anweisung ist beispielsweise nicht optimiert:

```
SELECT col1, col2, col3 FROM table1
ORDER BY col1 ASC, col2 ASC, col3 ASC;
```

Clustered-Indizes

Sie können umfangreiche Index-Scans weiter verbessern, indem Sie den Index als **Clustered** deklarieren. Die Verwendung eines Clustered-Indexes erhöht die Chance, dass zwei Zeilen aus benachbarten Indexeinträgen auch auf derselben Seite der Datenbank erscheinen. Diese Strategie kann zu weiteren Performance-Verbesserungen führen, da die Häufigkeit des Einlesens einer Tabellenseite in den Pufferpool reduziert wird.

Das Vorhandensein eines Indexes mit einer Clustered-Eigenschaft veranlasst den Datenbankserver, zu versuchen, Tabellenzeilen in ungefähr der gleichen Reihenfolge zu speichern, in der sie im Clustered-Index erscheinen. Während der Datenbankserver zwar versucht, die Schlüsselreihenfolge einzuhalten, ist das Clustering ein Annäherungsverfahren, und ein vollständiges Clustering kann nicht garantiert werden. Der Datenbankserver ist daher nicht in der Lage, die Tabelle sequenziell zu durchsuchen und alle Zeilen in der Schlüsselreihenfolge eines Clustered-Indexes abzurufen. Um sicherzustellen, dass die Tabellenzeilen in sortierter Reihenfolge geliefert werden, ist ein Zugriffsplan erforderlich, der entweder über den Index auf die Zeilen zugreift oder der einen physischen Sortiervorgang durchführt.

Der Optimierer nutzt einen Index mit Clustered-Eigenschaft, indem er die erwarteten Kosten eines indextierten Abrufs so abwandelt, dass die erwartete physische Nachbarschaft von Tabellenzeilen mit übereinstimmenden oder angrenzenden Indexschlüsselwerten berücksichtigt wird.

Der Umfang des Clustering für eine bestimmte Tabelle kann im Laufe der Zeit zurückgehen, wenn immer mehr Zeilen eingefügt oder aktualisiert werden. Der Datenbankserver zeichnet automatisch den Umfang des Clustering für jeden einzelnen Clustered-Index in der Systemtabelle ISYSPHYSIDX auf. Falls der Datenbankserver feststellt, dass das Clustering der Zeilen in einer Tabelle erheblich abgenommen hat, passt der Optimierer seine erwarteten Indexabrufkosten an.

Wenn Sie vorhaben, einen der Indizes auf der Tabelle zu einem Clustered-Index zu machen, müssen Sie die erwartete Abfrage-Systemlast berücksichtigen. Dies wird üblicherweise ein Experimentieren erfordern. Im Allgemeinen kann der Datenbankserver einen Clustered-Index verwenden, um die Performance zu steigern, wenn die folgenden Bedingungen bei einer bestimmten Abfrage gelten:

- Viele der Tabellenseiten, die für die Erfüllung der Abfrage benötigt werden, befinden sich noch nicht im Speicher. Wenn sich Tabellenseiten bereits im Speicher befinden, muss der Datenbankserver diese Seiten nicht lesen und eine Clusterbildung wäre sinnlos.
- Die Abfrage kann durch einen Indexabruf erfüllt werden, von dem zu erwarten ist, dass er eine nicht-triviale Zeilenanzahl zurückgibt. Clusterbildung ist beispielsweise bei einfachen Primärschlüssel-Suchvorgängen meist irrelevant.
- Der Datenbankserver muss, im Gegensatz zu einem reinen Indexabruf, die Tabellenseiten tatsächlich lesen.

Deklarieren von Clustered-Indizes

Die Clustered-Eigenschaft eines Indexes kann jederzeit unter Verwendung von SQL-Anweisungen hinzugefügt oder entfernt werden. Alle Primärschlüsselindizes, Fremdschlüsselindizes, UNIQUE-Integritätsregel-Indizes und Sekundärindizes können mit der Eigenschaft CLUSTERED deklariert werden. Sie können jedoch nicht mehr als einen Clustered-Index pro Tabelle deklarieren. Dazu können Sie eine der folgenden Anweisungen benutzen:

- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ALTER DATABASE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DECLARE LOCAL TEMPORARY TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Einige Anweisungen arbeiten in Verbindung mit anderen Anweisungen. Auf diese Weise können Sie den Gruppierungseffekt (Clustering) beibehalten und wiederherstellen:

- Die UNLOAD TABLE-Anweisung ermöglicht das Entladen einer Tabelle in der Sortierung des Clustered-Index-Schlüssels.
- Die LOAD TABLE-Anweisung fügt Zeilen in der Sortierung des Clustered-Index-Schlüssels in die Tabelle ein.
- Die INSERT-Anweisung versucht, neue Zeilen auf derselben Tabellenseite hinzuzufügen, die aufgrund der Sortierung des Clustered-Index-Schlüssels benachbarte Zeilen enthält.
- Die REORGANIZE TABLE-Anweisung stellt die Gruppierung einer Tabelle durch Neuordnung der Zeilen gemäß dem Clustered-Index wieder her. Falls REORGANIZE TABLE für Tabellen benutzt wird, für die kein Clustering festgelegt wurde, werden die Tabellen entsprechend dem Primärschlüssel neu sortiert.

Sie können Clustered-Indizes in Sybase Central auch erstellen, indem Sie den **Assistenten zum Erstellen von Indizes** verwenden und auf **Clustered-Index erstellen** klicken, wenn Sie dazu aufgefordert werden.

Siehe auch

- „UNLOAD-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „LOAD TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „INSERT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REORGANIZE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Indizes erstellen“ auf Seite 33

Indizes erstellen

Erstellen Sie Indizes für Basistabellen, temporäre Tabellen und materialisierte Ansichten. Sie können auch Indizes für integrierte Funktionen unter Verwendung einer berechneten Spalte erstellen.

Voraussetzungen

Wenn Sie einen Index für eine Tabelle erstellen möchten, müssen Sie der Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- CREATE ANY INDEX-Systemprivileg
- CREATE ANY OBJECT-Systemprivileg
- REFERENCES-Privileg für die Tabelle und entweder das COMMENT ANY OBJECT-Systemprivileg, das ALTER ANY INDEX-Systemprivileg oder das ALTER 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

Sie können keinen Index für eine reguläre Ansicht erstellen. Sie können für eine deaktivierte materialisierte Ansicht keinen Index erstellen.

Kontext und Bemerkungen

Beim Erstellen von Indizes bestimmt die Reihenfolge, in der Sie die Spalten angeben, auch die Reihenfolge, in der die Spalten im Index erscheinen. Doppelte Verweise auf Spaltennamen in der Indexdefinition sind nicht erlaubt. Sie können den Indexberater verwenden, der Sie bei der Auswahl der richtigen Indizes für Ihre Datenbank unterstützt.

Es wird ein automatisches Festschreiben durchgeführt, wenn ein Index für eine lokale temporäre Tabelle erstellt wird und die Option `auto_commit_on_create_local_temp_index` aktiviert ist. Standardmäßig ist die Option auf Off gesetzt.

Die Erstellung eines Indexes für eine Funktion (eine implizite berechnete Spalte) bewirkt einen Checkpoint.

Spaltenstatistiken werden aktualisiert (oder erstellt, falls sie nicht existieren).

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf die **Indizes** und klicken Sie auf **Neu » Index**.
3. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen von Indizes**.

Ergebnisse

Der neue Index erscheint auf der Registerkarte **Index** für die Tabelle und in **Indizes**. Der neue Index ist zur Verwendung durch Abfragen verfügbar.

Siehe auch

- „CREATE INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Tools für Performanceüberwachung und Diagnose“ auf Seite 153
- „auto_commit_on_create_local_temp_index-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Validieren eines Indexes

Sie können einen Index validieren, um sicherzustellen, dass jede Zeile, die in dem Index referenziert wird, in der Tabelle tatsächlich vorhanden ist. Bei Fremdschlüsselindizes wird durch eine Validierung auch gewährleistet, dass die entsprechende Zeile in der Primärtabelle vorhanden ist.

Voraussetzungen

Sie müssen Eigentümer des Indexes sein oder das **VALIDATE ANY OBJECT**-Systemprivileg haben.

Führen Sie Validierung nur dann aus, wenn keine Verbindungen Änderungen in der Datenbank vornehmen.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Indizes**.
3. Rechtsklicken Sie auf einen Index und klicken Sie auf **Validieren**.
4. Klicken Sie auf **OK**.

Ergebnisse

Eine Prüfung wird ausgeführt, um sicherzustellen, dass jede in dem Index referenzierte Zeile in der Tabelle tatsächlich vorhanden ist. Bei Fremdschlüsselindizes wird außerdem sichergestellt, dass die entsprechende Zeile in der Primärtabelle vorhanden ist.

Siehe auch

- „VALIDATE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Validierungs-Dienstprogramm (dbvalid)“ [*SQL Anywhere Server - Datenbankadministration*]

Neuerstellen eines Indexes

Erstellen Sie einen Index neu, der aufgrund zahlreicher Einfügungen oder Löschungen in der Tabelle oder der materialisierten Ansicht stark fragmentiert ist.

Voraussetzungen

Wenn Sie einen Index für eine Tabelle neu aufbauen 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 Index für eine materialisierte Ansicht neu aufbauen möchten, müssen Sie Eigentümer der materialisierten Ansicht sein oder eines der folgenden Privilegien haben:

- ALTER ANY INDEX-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Kontext und Bemerkungen

Bei der Neuerstellung eines Indexes bauen Sie den physischen Index neu auf. Der Neuaufbau wirkt sich direkt auf alle logischen Indizes aus, die den physischen Index verwenden. Es ist nicht notwendig, logische Indizes neu aufzubauen.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Indizes**.
3. Rechtsklicken Sie auf den Index und klicken Sie auf **Neu erstellen**.
4. Klicken Sie auf **OK**.

Ergebnisse

Der Index wird neu erstellt und dabei wird die Fragmentierung entfernt.

Siehe auch

- „Fortgeschrittene Aufgaben: Logische und physische Indizes“ auf Seite 37
- „REORGANIZE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Indexfragmentierung und -schiefe (Skew) reduzieren“ auf Seite 254
- „Verwenden des Assistenten für die Anwendungsprofilerstellung“ auf Seite 157
- „sa_index_density-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Index entfernen

Löschen Sie einen Index, wenn er nicht mehr benötigt wird oder wenn Sie die Definition einer Spalte ändern müssen, die Teil eines Primär- oder Fremdschlüssels ist.

Voraussetzungen

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 einen Fremdschlüssel, einen Primärschlüssel oder eine Eindeutigkeits-Integritätsregel löschen möchten, müssen Sie Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Tabelle
- ALTER ANY TABLE-Systemprivileg
- ALTER 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

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Indizes**.
3. Rechtsklicken Sie auf den Index und klicken Sie auf **Löschen**.
4. Klicken Sie auf **Ja**.

Ergebnisse

Der Index wird aus der Datenbank gelöscht.

Nächste Schritte

Wenn Sie einen Index gelöscht haben, um die Definition einer Spalte zu löschen oder zu ändern, die Teil eines Primär- oder Fremdschlüssels ist, müssen Sie einen neuen Index hinzufügen.

Siehe auch

- „DROP INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Indizes erstellen“ auf Seite 33

Fortgeschrittene Aufgaben: Indexinformationen im Katalog

Die Systemtabelle ISYSIDX enthält eine Liste aller Indizes in der Datenbank, einschließlich der Primär- und Fremdschlüsselindizes. Zusätzliche Informationen über die Indizes befinden sich in den Systemtabellen ISYSPHYSIDX, ISYSIDXCOL und ISYSFKEY. Sie können Sybase Central oder Interactive SQL benutzen, um die Systemansichten für diese Tabellen zu durchsuchen und die enthaltenen Daten anzuzeigen.

Nachfolgend finden Sie einen Überblick darüber, wie Indexdaten in den Systemtabellen gespeichert werden:

- **ISYSIDX-Systemtabelle** Dies ist die zentrale Tabelle für das Verfolgen von Indizes. Jede Zeile in der Systemtabelle ISYSIDX definiert einen logischen Index in der Datenbank (PKEY, FKEY, UNIQUE-Integritätsregel, Sekundärindex).
- **ISYSPHYSIDX-Systemtabelle** Jede Zeile in der Systemtabelle ISYSPHYSIDX definiert einen physischen Index in der Datenbank.
- **ISYSIDXCOL-Systemtabelle** Während jede Zeile in der Systemansicht SYSIDX einen Index in der Datenbank beschreibt, so beschreibt jede Zeile in der Systemansicht SYSIDXCOL eine Spalte eines Indexes, der in der Systemansicht SYSIDX beschrieben wird.
- **ISYSFKEY-Systemtabelle** Jeder Fremdschlüssel in der Datenbank wird durch eine Zeile in der Systemtabelle ISYSFKEY und eine Zeile in der Systemtabelle ISYSIDX definiert.

Siehe auch

- „SYSIDX-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSPHYSIDX-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSIDXCOL-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSFKEY-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Fortgeschrittene Aufgaben: Logische und physische Indizes“ auf Seite 37

Fortgeschrittene Aufgaben: Logische und physische Indizes

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 Primärschlüssel, Sekundärschlüssel, Fremdschlüssel oder eine Eindeutigkeits-Integritätsregel erstellen, gewährleistet der Datenbankserver referenzielle Integrität, indem er einen logischen Index für die Integritätsregel erstellt. Dann prüft der Datenbankserver, ob bereits ein physischer Index vorhanden ist, der die Integritätsregel erfüllt. Falls bereits ein qualifizierter physischer Index vorhanden ist, verweist der Datenbankserver den logischen Index auf diesen physischen Index. Falls kein solcher Index vorhanden ist, erstellt der Datenbankserver einen neuen physischen Index und verweist den logischen Index auf den physischen Index.

Damit ein physischer Index die Anforderungen eines logischen Indexes erfüllt, müssen die Spalten, die Spaltenreihenfolge und die Sortierfolge (aufsteigend, absteigend) der Daten bei jeder Spalte identisch sein.

In den Systemtabellen ISYSIDX und ISYSPHYSIDX werden Informationen über alle logischen und physischen Indizes der Datenbank aufgezeichnet. Wenn Sie einen logischen Index erstellen, wird ein Eintrag in die Systemtabelle ISYSIDX eingefügt, der die Indexdefinition enthält. In der Spalte ISYSIDX.phys_id wird ein Verweis auf den physischen Index aufgezeichnet, der die Anforderungen des logischen Indexes erfüllt. Der physische Index wird in der Systemtabelle ISYSPHYSIDX definiert.

Durch die Verwendung logischer Indizes braucht der Datenbankserver keine doppelten physischen Indizes zu erstellen und zu verwalten, weil mehrere logische Indizes auf einen einzelnen physischen Index verweisen können.

Wenn Sie einen logischen Index löschen, wird seine Definition aus der Systemtabelle ISYSIDX entfernt. Falls es sich dabei um den einzigen logischen Index handelt, der auf einen bestimmten physischen Index verweist, werden auch der physische Index und seine zugehörigen Einträge in der Systemtabelle ISYSPHYSIDX gelöscht.

Für entfernte Tabellen werden keine physischen Indizes erstellt. Für temporäre Tabellen werden physische Indizes erstellt, jedoch werden sie nicht in ISYSPHYSIDX aufgezeichnet, und nach der Verwendung werden sie gelöscht. Weiterhin werden physische Indizes für temporäre Tabellen nicht gemeinsam genutzt.

Siehe auch

- „SYSIDX-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSPHYSIDX-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Ermitteln, welche logischen Indizes einen physischen Index gemeinsam nutzen

Wenn Sie einen Index löschen, wird der logische Index gelöscht. Der physische Index, auf den er sich bezieht, wird jedoch nicht in allen Fällen gelöscht. Wenn sich ein anderer logischer Index auf denselben physischen Index bezieht, wird der physische Index nicht gelöscht. Es ist wichtig, dies zu wissen, besonders dann, wenn Sie den Index löschen, um Platz auf der Platte freizugeben oder um den Index physisch neu aufzubauen.

Sie können feststellen, ob ein Index für eine Tabelle einen physischen Index gemeinsam mit anderen Indizes nutzt, indem Sie in Sybase Central die Tabelle auswählen und auf die Registerkarte **Indizes** klicken. Achten Sie darauf, ob der Wert für "Phys.-ID" auch bei anderen Indizes in der Liste erscheint.

Übereinstimmende Werte für "Phys.-ID" bedeuten, dass diese Indizes denselben physischen Index nutzen. Um einen physischen Index neu zu erstellen, können Sie die Anweisung ALTER INDEX...REBUILD verwenden. Alternativ können Sie auch alle Indizes löschen und dann neu erstellen.

Ermitteln von Tabellen, in denen physische Indizes gemeinsam genutzt werden

Sie können jederzeit eine Liste der Tabellen abrufen, in denen physische Indizes gemeinsam genutzt werden, indem Sie eine Abfrage wie die folgende ausführen:

```
SELECT tab.table_name, idx.table_id, phys.phys_index_id, COUNT(*)
FROM SYSIDX idx JOIN SYSTAB tab ON (idx.table_id = tab.table_id)
JOIN SYSPHYSIDX phys ON ( idx.phys_index_id = phys.phys_index_id
AND idx.table_id = phys.table_id )
GROUP BY tab.table_name, idx.table_id, phys.phys_index_id
HAVING COUNT(*) > 1
ORDER BY tab.table_name;
```

Es folgt ein Beispielergebnis für die Abfrage:

table_name	table_id	phys_index_id	COUNT()
ISYSHECK	57	0	2
ISYSCOLSTAT	50	0	2
ISYSFKEY	6	0	2
ISYSSOURCE	58	0	2
MAINLIST	94	0	3
MAINLIST	94	1	2

Die Anzahl der Zeilen einer Tabelle gibt an, wie viele gemeinsam genutzte physische Indizes es für diese Tabelle gibt. In diesem Beispiel verfügen alle Tabellen über einen gemeinsam genutzten physischen Index, außer der fiktiven Tabelle MAINLIST, die über zwei Indizes verfügt. Der Wert in "phys_index_id" zeigt an, welcher physische Index gemeinsam genutzt wird. Der Wert in der Spalte COUNT gibt an, wie viele logische Indizes den physischen Index gemeinsam nutzen.

Sie können mit Sybase Central auch festzustellen, welche Indizes einer bestimmten Tabelle einen physischen Index gemeinsam nutzen. Wählen Sie dazu im linken Fensterausschnitt die Tabelle aus und klicken Sie im rechten Ausschnitt auf die Registerkarte **Indizes**. Suchen Sie dann nach mehreren Zeilen mit dem gleichen Wert in der Spalte "Phys.-ID". Indizes mit dem gleichen Wert in "Phys.-ID" nutzen denselben physischen Index.

Siehe auch

- „[Neuerstellen eines Indexes](#)“ auf Seite 35
- „[ALTER INDEX-Anweisung](#)“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „[SYSIDX-Systemansicht](#)“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Fortgeschrittene Aufgaben: Indexselektivität und Auffächerung

Indexselektivität bezieht sich auf die Fähigkeit eines Indexes, einen gewünschten Indexeintrag zu finden, ohne zusätzliche Daten lesen zu müssen.

Bei niedriger Selektivität müssen zusätzliche Informationen von der Tabellenseite, die der Index referenziert, abgerufen werden. Diese Abrufe werden **Vollvergleiche** (full compares) genannt und beeinträchtigen die Index-Performance.

Die Eigenschaft FullCompare protokolliert die Anzahl der aufgetretenen vollen Vergleiche. Sie können diese Statistik auch mit dem Systemmonitor von Sybase Central oder dem Windows-Systemmonitor überwachen.

Hinweis

Der Windows-Systemmonitor ist unter Windows Mobile möglicherweise nicht verfügbar.

Außerdem wird die Anzahl der vollen Vergleiche im grafischen Plan mit Statistik angezeigt.

Indizes sind auf mehreren Ebenen organisiert, vergleichbar mit einer Baumstruktur. Die erste Seite eines Indexes, auch Stammseite genannt, unterteilt sich in eine oder mehrere Seiten auf der nächsten Ebene und jede dieser Seiten unterteilt sich erneut, bis die unterste Ebene des Indexes erreicht ist. Diese Indexseiten auf der untersten Ebene werden Blattseiten genannt. Um eine bestimmte Zeile zu finden, benötigt ein Index mit n Ebenen n Lesevorgänge für Indexseiten und einen Lesevorgang für die Datenseite mit der gesuchten Zeile. Im Allgemeinen sind weniger als n Lesevorgänge von der Festplatte notwendig, da häufig verwendete Indexseiten meistens im Cache gespeichert sind.

Die **Indexauffächerung** ist die auf einer Seite gespeicherte Anzahl von Indexeinträgen. Ein Index mit einer höheren Auffächerung hat eventuell weniger Ebenen als ein Index mit einer niedrigeren Auffächerung. Daher bedeutet eine höhere Indexauffächerung im Allgemeinen eine bessere Index-Performance. Durch die Wahl der richtigen Seitengröße für Ihre Datenbank können Sie die Indexauffächerung verbessern.

Die Anzahl der Ebenen in einem Index können Sie mit der Systemprozedur `sa_index_levels` anzeigen.

Siehe auch

- [Häufig im Plan verwendete Statistiken auf Seite 370](#)
- [„Tipp: Angemessene Seitengröße verwenden“ auf Seite 260](#)
- [„sa_index_levels-Systemprozedur“ \[*SQL Anywhere Server - SQL-Referenzhandbuch*\]](#)

Fortgeschrittene Aufgaben: Andere Verwendungen für Indizes in SQL Anywhere

SQL Anywhere verwendet Indizes auch, um weitere Performancesteigerungen zu erzielen. Mit einem Index kann SQL Anywhere Spalteneindeutigkeit erzwingen, um die Anzahl der Zeilen und Seiten, die gesperrt werden müssen, zu reduzieren und um die Selektivität eines Prädikats besser schätzen zu können.

- **Spalteneindeutigkeit erzwingen** Ohne Index muss SQL Anywhere jedesmal die gesamte Tabelle durchsuchen, wenn ein Wert eingefügt wird, um sicherzustellen, dass er eindeutig ist. Aus diesem Grunde erstellt SQL Anywhere automatisch einen Index für jede Spalte mit einer Eindeutigkeits-Integritätsregel.
- **Sperren reduzieren** Indizes reduzieren die Anzahl von Sperren und Seiten, die während Einfügungen, Aktualisierungen und Löschvorgängen gesperrt werden müssen. Diese Reduzierung ist das Ergebnis der Sortierfolge, die Indizes einer Tabelle vorgeben.
- **Selektivität schätzen** Da ein Index sortiert ist, kann der Optimierer den Prozentsatz der Werte schätzen, die eine gegebene Abfrage erfüllen, indem er die oberen Ebenen des Indexes durchsucht. Dieser Vorgang wird partieller Index-Scan bezeichnet.

Siehe auch

- „Funktionsweise von Sperren“ auf Seite 906

Ansichten

Eine Ansicht ist eine berechnete Tabelle, die durch die Ergebnismenge ihrer Ansichtsdefinition definiert wird, die wiederum als SQL-Abfrage ausgedrückt wird. Sie können Ansichten verwenden, um den Datenbankbenutzern genau die Informationen zu zeigen, die Sie darstellen wollen, und zwar in einem von Ihnen gewählten Format. SQL Anywhere unterstützt zwei Arten von Ansichten: **Reguläre Ansichten** und **materialisierte Ansichten**.

Die Definition jeder Ansicht in der Datenbank ist in der Systemtabelle ISYSVIEW gespeichert.

Siehe auch

- „SYSVIEW-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Dokumentationskonventionen für Ansichten

Der Begriff **reguläre Ansicht** wird verwendet, um eine Ansicht zu beschreiben, die jedes Mal neu berechnet wird, wenn Sie die Ansicht referenzieren, und deren Ergebnismenge nicht auf dem Plattenspeicher gespeichert wird. Dies ist der am häufigsten verwendete Ansichtstyp. Der Großteil der Dokumentation bezieht sich auf reguläre Ansichten.

Der Ausdruck **materialisierte Ansicht** wird verwendet, um eine Ansicht zu beschreiben, deren Ergebnismenge vorausberechnet und auf dem Plattenspeicher materialisiert wird, ähnlich wie der Inhalt einer Basistabelle.

Die Bedeutung des Ausdrucks **Ansicht** (allein stehend) in der Dokumentation ist kontextabhängig. Wenn er in einem Abschnitt verwendet wird, der die gemeinsamen Aspekte von regulären und materialisierten Ansichten behandelt, bezieht er sich sowohl auf reguläre als auch auf materialisierte Ansichten. Wenn der Ausdruck in der Dokumentation für materialisierte Ansichten verwendet wird, bezieht er sich auf materialisierte Ansichten. Das Gleiche gilt entsprechend für reguläre Ansichten.

Fähigkeiten von regulären Ansichten, materialisierten Ansichten und Tabellen

Die folgende Tabelle vergleicht die Fähigkeiten von regulären Ansichten, materialisierten Ansichten und Tabellen:

Funktion	Reguläre Ansichten	Materialisierte Ansichten	Tabellen
Zugriffsprivilegien	Ja	Ja	Ja
SELECT	Ja	Ja	Ja
UPDATE	Manche	Nein	Ja
INSERT	Manche	Nein	Ja
DELETE	Manche	Nein	Ja
Abhängige Ansichten	Ja	Ja	Ja
Indizes	Nein	Ja	Ja
Integritätsregeln	Nein	Nein	Ja
Schlüssel	Nein	Nein	Ja

Vorteile der Verwendung von Ansichten

Mit Ansichten wird der Zugriff auf Daten in der Datenbank an den jeweiligen Benutzer angepasst. Diese Anpassung verfolgt mehrere Zwecke:

- **Effizienter Einsatz von Ressourcen** Reguläre Ansichten benötigen keinen zusätzlichen Speicherplatz für Daten. Sie werden bei jedem Aufruf neu berechnet. Materialisierte Ansichten benötigen Speicherplatz auf der Festplatte, aber sie müssen nicht jedes Mal neu berechnet werden, wenn sie aufgerufen werden. Materialisierte Ansichten können die Reaktionszeit in Umgebungen verbessern, in denen die Datenbank groß ist und der Datenbankserver häufige und wiederholte Anforderungen ausführt, um Joins für die gleichen Tabellen durchzuführen.
- **Verbesserte Sicherheit** Es wird nur der Zugriff auf relevante Informationen erlaubt.
- **Verbesserte Benutzbarkeit** Die Benutzer und Anwendungsentwickler erhalten die Daten in einer leichter verständlichen Form präsentiert als in den Basistabellen.
- **Verbesserte Konsistenz** Sie zentralisiert die Definition von häufigen Abfragen in der Datenbank.

Ansichtenabhängigkeiten

Eine Ansichtsdefinition kann auf andere Objekte wie etwa Spalten, Tabellen oder andere Ansichten verweisen. Wenn eine Ansicht auf ein anderes Objekt verweist, wird diese Ansicht als **referenzierendes Objekt** bezeichnet, und das Objekt, auf das sie verweist, wird als **referenziertes Objekt** bezeichnet. Außerdem ist ein referenzierendes Objekt **abhängig** von den Objekten, die es referenziert.

Die Gruppe der referenzierten Objekte für eine bestimmte Ansicht umfasst alle Objekte, die sie direkt oder indirekt referenziert. Eine Ansicht kann beispielsweise eine Tabelle indirekt referenzieren, indem sie eine andere Ansicht referenziert, die diese Tabelle referenziert.

Betrachten Sie die folgende Gruppe von Tabellen und Ansichten:

```
CREATE TABLE t1 ( c1 INT, c2 INT );
CREATE TABLE t2( c3 INT, c4 INT );
CREATE VIEW v1 AS SELECT * FROM t1;
CREATE VIEW v2 AS SELECT c3 FROM t2;
CREATE VIEW v3 AS SELECT c1, c3 FROM v1, v2;
```

Aus den obigen Definitionen sind folgende Ansichtenabhängigkeiten erkennbar:

- Ansicht "v1" ist von jeder einzelnen Spalte in "t1" sowie von "t1" selbst abhängig.
- Ansicht "v2" ist von "t2.c3" und von "t2" selbst abhängig.
- Ansicht "v3" ist von den Spalten "t1.c1" und "t2.c3", den Tabellen "t1" und "t2" sowie den Ansichten "v1" und "v2" abhängig.

Der Datenbankserver verfolgt, welche Spalten, Tabellen und Ansichten von einer Ansicht referenziert werden. Er verwendet diese Informationen über die Abhängigkeiten, um sicherzustellen, dass Schemaänderungen für referenzierte Objekte nicht zur Unbrauchbarkeit einer referenzierenden Ansicht führen.

Abhängigkeiten und Vorgänge, die ein Schema ändern

Für den Versuch, das für eine Tabelle oder eine Ansicht definierte Schema zu ändern, ist es erforderlich, dass der Datenbankserver prüft, ob es abhängige Ansichten gibt, die von der Änderung berührt werden. Nachfolgend finden Sie Beispiele von Vorgängen, die ein Schema ändern:

- Löschen einer Tabelle, Ansicht, materialisierten Ansicht oder Spalte
- Umbenennen einer Tabelle, Ansicht, materialisierten Ansicht oder Spalte
- Spalten hinzufügen, löschen oder ändern
- Ändern des Datentyps, der Größe oder der Nullwertfähigkeit einer Spalte
- Deaktivieren von Ansichten oder Abhängigkeiten von Tabellenansichten

Ereignisse während Vorgängen, die ein Schema ändern

1. Der Datenbankserver erstellt eine Liste von Ansichten, die direkt oder indirekt von der zu ändernden Tabelle oder Ansicht abhängen. Ansichten mit dem Status DEAKTIVIERT werden ignoriert.

Falls es unter den abhängigen Ansichten materialisierte Ansichten gibt, schlägt die Anforderung fehl, ein Fehler wird gemeldet und die noch ausstehenden Ereignisse geschehen nicht. Sie müssen abhängige materialisierte Ansichten explizit deaktivieren, bevor Sie mit dem Schema-Änderungsvorgang fortfahren können.
2. Der Datenbankserver setzt exklusive Schemasperrungen für das zu ändernde Objekt und für alle abhängigen regulären Ansichten.
3. Der Datenbankserver setzt den Status aller abhängigen regulären Ansichten auf UNGÜLTIG.
4. Der Datenbankserver führt den Vorgang durch, der das Schema ändert. Falls der Vorgang fehlschlägt, werden die Sperren freigegeben, der Status der abhängigen regulären Ansichten wird auf GÜLTIG zurückgesetzt, ein Fehler wird gemeldet und der folgende Schritt wird nicht durchgeführt.
5. Der Datenbankserver kompiliert die abhängigen regulären Ansichten neu und setzt den Status jeder Ansicht auf GÜLTIG, wenn die Kompilation erfolgreich war. Falls die Kompilation für eine reguläre Ansicht fehlschlägt, bleibt der Status für diese Ansicht auf UNGÜLTIG gesetzt. Nachfolgende Anforderungen für eine reguläre Ansicht mit dem Status UNGÜLTIG führen dazu, dass der Datenbankserver eine erneute Kompilation der Ansicht versucht. Falls weitere Versuche fehlschlagen, ist wahrscheinlich eine Änderung der ungültigen Ansicht oder eines Objektes erforderlich, von dem die Ansicht abhängig ist.

Reguläre Ansichten: Abhängigkeiten und Schemaänderungen

- Eine reguläre Ansicht kann Tabellen oder Ansichten referenzieren, einschließlich materialisierter Ansichten.
- Wenn Sie das Schema einer Tabelle oder Ansicht ändern, kompiliert die Datenbank automatisch alle referenzierenden regulären Ansichten neu.
- Wenn Sie eine Ansicht oder Tabelle deaktivieren oder löschen, werden alle abhängigen regulären Ansichten automatisch deaktiviert.
- Sie können die DISABLE VIEW DEPENDENCIES-Klausel der ALTER TABLE-Anweisung verwenden, um abhängige reguläre Ansichten zu deaktivieren.

Materialisierte Ansichten: Abhängigkeiten und Schemaänderungen

- Eine materialisierte Ansicht kann nur Basistabellen referenzieren.
- Schemaänderungen an einer Basistabelle sind nicht zulässig, wenn sie von aktivierten materialisierten Ansichten referenziert wird. Sie können der Tabelle Fremdschlüssel hinzufügen (z.B. ALTER TABLE ADD FOREIGN KEY).
- Bevor Sie eine Tabelle löschen oder ändern, müssen Sie alle abhängigen materialisierten Ansichten deaktivieren oder löschen.

- Die DISABLE VIEW DEPENDENCIES-Klausel der ALTER TABLE-Anweisung wirkt sich nicht auf materialisierte Ansichten aus. Um eine materialisierte Ansicht zu deaktivieren, müssen Sie die Anweisung ALTER MATERIALIZED VIEW ... DISABLE verwenden.
- Wenn Sie eine materialisierte Ansicht deaktiviert haben, müssen Sie sie explizit reaktivieren, z.B. mit der Anweisung ALTER MATERIALIZED VIEW ... ENABLE.

Siehe auch

- „Aktivieren oder Deaktivieren einer materialisierten Ansicht“ auf Seite 68

Abhängigkeitsinformationen abrufen (SQL)

Für jede Tabelle oder Ansicht in der Datenbank können Sie eine Liste der Objekte abrufen, die von diesem Objekt abhängig sind. Dies ist nützlich, wenn Sie eine Tabelle oder Ansicht ändern möchten und wissen müssen, welche anderen Objekte von der Änderung betroffen sein werden.

Voraussetzungen

Die Ausführung der Aufgabe erfordert keine Privilegien und erfolgt unter der Annahme, dass PUBLIC Zugriff auf den Katalog hat.

Kontext und Bemerkungen

In der Systemansicht SYSDEPENDENCY werden Informationen über Abhängigkeiten gespeichert. Jede Zeile in der Systemansicht SYSDEPENDENCY beschreibt eine Abhängigkeit zwischen zwei Datenbankobjekten. Eine direkte Abhängigkeit besteht, wenn ein Objekt in seiner Definition ein anderes Objekt direkt referenziert. Der Datenbankserver benutzt die Informationen über direkte Abhängigkeiten, um auch indirekte Abhängigkeiten zu ermitteln. Beispiel: Ansicht A referenziert Ansicht B, die wiederum Tabelle C referenziert. In diesem Fall ist Ansicht A direkt abhängig von Ansicht B und indirekt abhängig von Tabelle C.

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Führen Sie eine Anweisung aus, die die sa_dependent_views-Systemprozedur aufruft.

Ergebnisse

Eine Liste von IDs für die abhängigen Ansichten wird zurückgegeben.

Beispiel

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;
```

Siehe auch

- „SYSDEPENDENCY-Systemansicht“ [SQL Anywhere Server - SQL-Referenzhandbuch]
- „sa_dependent_views-Systemprozedur“ [SQL Anywhere Server - SQL-Referenzhandbuch]

Reguläre Ansichten

Wenn Sie Daten durchsuchen, arbeitet eine Abfrage mit einem oder mehreren Datenbankobjekten und erzeugt eine Ergebnismenge. Genau wie eine Basistabelle hat eine Ergebnismenge aus einer Abfrage Spalten und Zeilen. Eine Ansicht gibt einer Abfrage einen bestimmten Namen und hält die Definition in den Systemtabellen der Datenbank fest.

Wenn Sie eine reguläre Ansicht erstellen, speichert der Datenbankserver die Ansichtsdefinition in der Datenbank. Für die Ansicht werden keine Daten gespeichert. Stattdessen wird die Ansichtsdefinition nur ausgeführt, wenn sie referenziert wird, und zwar nur für die Zeit, in der die Ansicht in Gebrauch ist. Das Erstellen einer Ansicht erfordert kein Speichern von redundanten Daten in der Datenbank.

Nehmen wir an, Sie müssen häufig die Anzahl der Mitarbeiter in jeder Abteilung auflisten. Sie können diese Liste mit der folgenden Anweisung zusammenstellen:

```
SELECT DepartmentID, COUNT(*)  
FROM Employees  
GROUP BY DepartmentID;
```

Beschränkungen für SELECT-Anweisungen bei regulären Ansichten

Es bestehen einige Einschränkungen bezüglich der SELECT-Anweisungen, die Sie als reguläre Ansichten verwenden können. Insbesondere können Sie keine ORDER BY-Klausel in einer SELECT-Abfrage verwenden. Es gehört zum Wesen von relationalen Datenbanktabellen, dass die Reihenfolge der Zeilen und Spalten keine Bedeutung hat. Eine ORDER BY-Klausel würde die Zeilen der Ansicht in eine Reihenfolge zwingen. Sie können die GROUP BY-Klausel, Unterabfragen und Joins in Ansichtsdefinitionen verwenden.

Um eine Ansicht zu entwickeln, sollten Sie die SELECT-Abfrage selbst so abstimmen, bis sie genau die Ergebnisse und das Format liefert, die Sie benötigen. Wenn die SELECT-Anweisung richtig abgestimmt ist, können Sie vor der Abfrage die folgende Klausel hinzufügen, um die Ansicht zu erstellen:

```
CREATE VIEW view-name AS query;
```

Anweisungen, durch die reguläre Ansichten aktualisiert werden

Aktualisierungen können in einer Ansicht mit den UPDATE-, INSERT- oder DELETE-Anweisungen durchgeführt werden, wenn die Abfragenspezifikation, die die Ansicht festlegt, aktualisierbar ist. Ansichten sind von Natur aus *nicht-aktualisierbar*, wenn ihre Definition eine der folgenden Elemente in ihrer Abfragenspezifikation enthält:

- UNION, EXCEPT oder INTERSECT.
- DISTINCT-Klausel.
- GROUP BY-Klausel.

- WINDOW-Klausel.
- FIRST-, TOP- oder LIMIT-Klausel.
- Aggregatfunktionen.
- Mehr als eine Tabelle in der FROM-Klausel, wenn die Option `ansi_update_constraints` auf 'Strict' oder 'Cursor' gesetzt ist.
- ORDER BY-Klausel, wenn die Option `ansi_update_constraints` auf 'Strict' oder 'Cursor' gesetzt ist.
- Alle SELECT-Listenelemente, die keine Basistabellenspalten sind.

WITH CHECK OPTION-Klausel

Die WITH CHECK OPTION-Klausel ist beim Erstellen einer Ansicht nützlich, um zu steuern, welche Daten geändert werden, wenn über eine Ansicht Daten in eine Basistabelle eingefügt oder aktualisiert werden. Mit dem folgenden Beispiel wird dies veranschaulicht.

Führen Sie die folgende Anweisung aus, um die Ansicht "SalesEmployees" mit einer WITH CHECK OPTION-Klausel zu erstellen.

```
CREATE VIEW SalesEmployees AS
  SELECT EmployeeID, GivenName, Surname, DepartmentID
  FROM Employees
  WHERE DepartmentID = 200
  WITH CHECK OPTION;
```

Zeigen Sie die Inhalte dieser Ansicht wie folgt an:

```
SELECT * FROM SalesEmployees;
```

EmployeeID	GivenName	Surname	DepartmentID
129	Philip	Chin	200
195	Marc	Dill	200
299	Rollin	Overbey	200
467	James	Klobucher	200
...

Versuchen Sie nun, die "DepartmentID" für Philip Chin auf 400 zu ändern:

```
UPDATE SalesEmployees
SET DepartmentID = 400
WHERE EmployeeID = 129;
```

Da WITH CHECK OPTION festgelegt wurde, prüft der Datenbankserver, ob die Aktualisierung die neue Ansichtsdefinition verletzt (in diesem Fall den Ausdruck in der WHERE-Klausel). Die Anweisung schlägt fehl (DepartmentID muss 200 sein) und der Datenbankserver meldet den Fehler "Verletzung von WITH CHECK OPTION für INSERT/UPDATE in Basistabelle 'Employees'".

Hätten Sie WITH CHECK OPTION nicht in der Ansichtsdefinition angegeben, wäre die Aktualisierung fortgesetzt worden. Die Tabelle "Employees" wäre auf den neuen Wert geändert worden, und Philip Chin wäre aus der Ansicht verschwunden.

Wenn eine Ansicht (z.B. "View2") erstellt wird, welche die Ansicht "SalesEmployee" referenziert, werden alle Aktualisierungen oder Einfügungen in "View2" zurückgewiesen, die dem WITH CHECK OPTION-Kriterium für "SalesEmployee" widersprechen, auch wenn "View2" ohne WITH CHECK OPTION definiert ist.

Siehe auch

- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Abfrageergebnisse zusammenfassen, gruppieren und sortieren“ auf Seite 470
- „Materialisierte Ansichten“ auf Seite 56
- „ansi_update_constraints-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Status für reguläre Ansichten

Reguläre Ansichten verfügen über einen Status. Der Status zeigt die Verfügbarkeit der Ansicht für die Verwendung durch den Datenbankserver an. Sie können den Status aller Ansichten anzeigen, indem Sie im linken Fensterausschnitt von Sybase Central auf **Ansichten** klicken und im rechten Fensterausschnitt die Werte in der Spalte **Status** prüfen. Um den Status einer einzelnen Ansicht zu prüfen, rechtsklicken Sie in Sybase Central auf die Ansicht und klicken auf **Eigenschaften**, um den Wert für **Status** zu prüfen.

Es folgen Beschreibungen des möglichen Status für reguläre Ansichten:

- **GÜLTIG** Die Ansicht ist gültig und garantiert mit ihrer Definition übereinstimmend. Der Datenbankserver kann diese Ansicht ohne zusätzliche Arbeitsgänge verwenden. Eine aktivierte Ansicht hat den Status GÜLTIG.

In der Systemansicht SYSOBJECT steht der Wert 1 für den Status GÜLTIG.

- **UNGÜLTIG** Der Status UNGÜLTIG tritt auf, wenn ein Schema für ein referenziertes Objekt geändert wurde, wobei die Änderung zu einem erfolglosen Versuch zum Aktivieren der Ansicht führt. Beispiel: Ansicht "v1" referenziert Spalte "c1" in Tabelle "t". Falls Sie "t" ändern, um "c1" zu löschen, wird der Status von "v1" auf UNGÜLTIG gesetzt. Dies geschieht, während der Datenbankserver versucht, die Ansicht neu zu kompilieren, als Teil der ALTER-Operation zum Löschen der Spalte. In diesem Fall kann "v1" nur neu kompiliert werden, nachdem "c1" wieder zu "t" hinzugefügt wurde, oder nachdem "v1" so geändert wurde, dass sie nicht mehr auf "c1" verweist. Ansichten können den Status UNGÜLTIG auch dann erhalten, wenn eine Tabelle oder eine Ansicht, die sie referenzieren, gelöscht wird.

Eine Ansicht mit dem Status UNGÜLTIG unterscheidet sich von einer Ansicht mit dem Status DEAKTIVIERT dadurch, dass der Datenbankserver jedes Mal eine Neukompilierung versucht, wenn eine Ansicht mit dem Status UNGÜLTIG referenziert wird, etwa durch eine Abfrage. Falls die Kompilation erfolgreich ist, wird die Abfrage durchgeführt. Der Status der Ansicht bleibt UNGÜLTIG, bis sie ausdrücklich aktiviert wird. Wenn die Kompilation fehlschlägt, wird ein Fehler gemeldet.

Wenn der Datenbankserver intern eine Ansicht mit dem Status UNGÜLTIG aktiviert, gibt er eine Performance-Warnung aus.

In der Systemansicht SYSOBJECT steht der Wert 2 für den Status UNGÜLTIG.

- **DEAKTIVIERT** Ansichten mit dem Status DEAKTIVIERT stehen für die Verwendung durch den Datenbankserver zum Beantworten von Abfragen nicht zur Verfügung. Alle Abfragen, die versuchen, eine Ansicht mit dem Status DEAKTIVIERT zu verwenden, liefern eine Fehlermeldung.

Eine reguläre Ansicht hat diesen Status, wenn Folgendes zutrifft:

- Sie haben die Ansicht ausdrücklich deaktiviert, z.B. durch Ausführen der Anweisung `ALTER VIEW ... DISABLE`.
- Sie haben eine materialisierte oder nicht materialisierte Ansicht deaktiviert, von der die erste Ansicht abhängig ist.
- Sie haben Ansichtenabhängigkeiten für eine Tabelle deaktiviert, z.B. durch Ausführen der Anweisung `ALTER TABLE ... DISABLE VIEW DEPENDENCIES`.

In der Systemansicht SYSOBJECT steht der Wert 4 für den Status DEAKTIVIERT.

Siehe auch

- „SYSOBJECT-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Deaktivieren oder Aktivieren einer regulären Ansicht (SQL)“ auf Seite 54

Reguläre Ansichten erstellen

Erstellen Sie eine Ansicht mit Daten aus einer oder mehreren Quellen. Ansichten können die Performance verbessern und ermöglichen es Ihnen, zu steuern, welche Daten Benutzer abfragen können.

Voraussetzungen

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.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16-Plug-Ins** her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf **Ansichten** und klicken Sie auf **Neu » Ansicht**.
3. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen einer Ansicht**.
4. Klicken Sie im rechten Fensterausschnitt auf die Registerkarte **SQL** und bearbeiten Sie die Definition der Ansicht. Um Ihre Änderungen zu speichern, klicken Sie auf **Datei » Speichern**.

Ergebnisse

Die Definition für die erstellte Ansicht wird der Datenbank hinzugefügt. Jedes Mal, wenn eine Abfrage die Ansicht referenziert, wird die Definition verwendet, um die Ansicht mit Daten zu füllen und Ergebnisse zurückzugeben.

Nächste Schritte

Führen Sie eine Abfrage der Ansicht aus, um die Ergebnisse zu prüfen und sicherzustellen, dass die richtigen Daten zurückgegeben werden.

Siehe auch

- „CREATE VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „auto_commit_on_create_local_temp_index-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Reguläre Ansichten ändern

Sie bearbeiten eine reguläre Ansicht, indem Sie ihre Definition in der Datenbank bearbeiten. Beispiel: Wenn die Ansicht Daten aus einer zusätzlichen Tabelle enthalten soll, aktualisieren Sie die Ansichtsdefinition so, dass die Tabellendaten mit den vorhandenen Datenquellen in der Ansichtsdefinition verbunden werden.

Voraussetzungen

Sie müssen Eigentümer der Ansicht sein oder eines der folgenden Privilegien haben:

- ALTER ANY VIEW-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Kontext und Bemerkungen

Sie müssen gegebenenfalls eine Ansicht ändern, wenn die Ansichtsdefinition nicht mehr aktuell ist (kann aufgrund einer Schemaänderung in den zugrunde liegenden Daten nicht mehr kompiliert werden), Spalten hinzugefügt oder entfernt werden müssen oder Änderungen in Bezug auf Einstellungen erforderlich sind.

Sie können eine vorhandene Ansicht nicht umbenennen. Stattdessen müssen Sie eine neue Ansicht mit einem neuen Namen erstellen, die vorherige Definition für die neue Ansicht kopieren und dann die alte Ansicht löschen.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Ansichten**.
3. Wählen Sie die Ansicht aus.

4. Klicken Sie im rechten Fensterausschnitt auf die Registerkarte **SQL** und bearbeiten Sie die Definition der Ansicht.

Tipp

Zum Bearbeiten mehrerer Ansichten können Sie für jede Ansicht ein separates Fenster öffnen, statt jede Ansicht auf der Registerkarte **SQL** im rechten Fensterausschnitt zu bearbeiten. Sie können ein separates Fenster öffnen, indem Sie eine Ansicht auswählen und anschließend auf **Datei » In neuem Fenster bearbeiten** klicken.

5. Klicken Sie auf **Datei » Speichern**.

Ergebnisse

Die Definition der Ansicht wird in der Datenbank aktualisiert.

Nächste Schritte

Führen Sie eine Abfrage der Ansicht aus, um die Ergebnisse zu prüfen und sicherzustellen, dass die richtigen Daten zurückgegeben werden.

Wenn Sie eine reguläre Ansicht ändern und andere Ansichten von der Ansicht abhängig sind, müssen Sie möglicherweise zusätzliche Schritte durchführen, nachdem die Ansichtsänderung abgeschlossen ist. Beispiel: Wenn Sie eine Ansicht geändert haben, wird sie vom Datenbankserver automatisch neu kompiliert und für die Verwendung durch den Datenbankserver aktiviert. Wenn es abhängige reguläre Ansichten gibt, werden auch diese vom Datenbankserver deaktiviert und dann neu aktiviert. Falls sie nicht aktiviert werden können, erhalten sie den Status UNGÜLTIG. Sie müssen dann die Definition der regulären Ansicht an die Definitionen der abhängigen regulären Ansichten anpassen oder umgekehrt. Um zu ermitteln, ob eine reguläre Ansicht abhängige Ansichten hat, verwenden Sie die Systemprozedur `sa_dependent_views`.

Siehe auch

- „Ansichtenabhängigkeiten“ auf Seite 43
- „`sa_dependent_views`-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`ALTER VIEW`-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Reguläre Ansichten löschen“ auf Seite 51

Reguläre Ansichten löschen

Löschen Sie eine Ansicht, wenn sie nicht mehr benötigt wird. Außerdem müssen Sie eine Ansicht löschen (und neu erstellen), wenn Sie den Namen einer Ansicht ändern möchten.

Voraussetzungen

Sie müssen entweder Eigentümer sein oder das `DROP ANY VIEW`-Systemprivileg oder das `DROP ANY OBJECT`-Systemprivileg haben.

Bevor die Ansicht gelöscht werden kann, müssen Sie alle `INSTEAD OF`-Trigger löschen, die die Ansicht referenzieren.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Ansichten**.
3. Rechtsklicken Sie auf die Ansicht und wählen Sie **Löschen**.
4. Klicken Sie auf **Ja**.

Ergebnisse

Die Definition der regulären Ansicht wird aus der Datenbank gelöscht.

Nächste Schritte

Wenn Sie eine reguläre Ansicht löschen, zu der es abhängige Ansichten gibt, erhalten die abhängigen Ansichten als Folge der Löschung den Status UNGÜLTIG. Die abhängigen Ansichten können solange nicht benutzt werden, bis sie geändert werden oder die gelöschte Originalansicht wiederhergestellt wird.

Um zu ermitteln, ob eine reguläre Ansicht abhängige Ansichten hat, verwenden Sie die Systemprozedur `sa_dependent_views`.

Siehe auch

- „`sa_dependent_views`-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Reguläre Ansichten ändern“ auf Seite 50
- „DROP VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Ansichtenabhängigkeiten“ auf Seite 43
- „Trigger löschen“ auf Seite 106
- „DROP TRIGGER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „INSTEAD OF-Trigger“ auf Seite 110

Deaktivieren oder Aktivieren einer regulären Ansicht (Sybase Central)

Sie können festlegen, ob eine reguläre Ansicht für die Verwendung durch den Datenbankserver verfügbar ist, indem Sie sie aktivieren oder deaktivieren.

Voraussetzungen

Sie müssen Eigentümer sein oder eines der folgenden Privilegien haben:

- ALTER ANY VIEW-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Zum **Aktivieren** einer regulären Ansicht benötigen Sie *außerdem* das SELECT-Privileg für die Basistabelle(n) oder das SELECT ANY TABLE-Systemprivileg.

Bevor Sie eine reguläre Ansicht aktivieren, müssen Sie deaktivierte Ansichten, die sie referenziert, wieder aktivieren.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Ansichten**.
3. Um eine reguläre Ansicht zu deaktivieren, rechtsklicken Sie auf die Ansicht und klicken Sie auf **Deaktivieren**.
4. Um eine reguläre Ansicht zu aktivieren, rechtsklicken Sie auf die Ansicht und klicken Sie auf **Neukompilieren und aktivieren**.

Ergebnisse

Wenn Sie eine reguläre Ansicht deaktivieren, bewahrt der Datenbankserver die Definition der Ansicht in der Datenbank auf. Die Ansicht steht dann jedoch nicht mehr für die Verwendung mit einer Abfrage zur Verfügung.

Wenn eine Abfrage ausdrücklich eine deaktivierte Ansicht referenziert, schlägt die Abfrage fehl und es wird ein Fehler gemeldet.

Nächste Schritte

Daher müssen Sie, wenn Sie eine Ansicht wieder aktivieren, auch alle anderen Ansichten wieder aktivieren, die von der ersten Ansicht abhängig waren, bevor diese deaktiviert wurde. Mit der `sa_dependent_views`-Systemprozedur können Sie eine Liste der abhängigen Ansichten ermitteln, bevor Sie eine Ansicht deaktivieren.

Wenn Sie eine reguläre Ansicht aktivieren, wird sie vom Datenbankserver anhand der Definition, die für die Ansicht in der Datenbank gespeichert ist, neu kompiliert. Wenn die Kompilierung erfolgreich ist, wechselt der Ansichtsstatus auf GÜLTIG. Eine nicht erfolgreiche Neukompilation könnte darauf hinweisen, dass das Schema in einem oder mehreren der referenzierten Objekte geändert wurde. Falls dies so ist, müssen Sie die Ansichtsdefinition oder die referenzierten Objekte ändern, bis sie miteinander übereinstimmen, und dann die Ansicht aktivieren.

Wenn eine Ansicht deaktiviert wurde, muss sie ausdrücklich erneut aktiviert werden, damit der Datenbankserver sie wieder verwenden kann.

Siehe auch

- „`sa_dependent_views`-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`ALTER VIEW`-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`SYSDEPENDENCY`-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Deaktivieren oder Aktivieren einer regulären Ansicht (SQL)

Sie können festlegen, ob eine reguläre Ansicht für die Verwendung durch den Datenbankserver verfügbar ist, indem Sie sie aktivieren oder deaktivieren.

Voraussetzungen

Sie müssen Eigentümer sein oder eines der folgenden Privilegien haben:

- ALTER ANY VIEW-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Zum **Aktivieren** einer regulären Ansicht benötigen Sie *außerdem* das SELECT-Privileg für die Basistabelle(n) oder das SELECT ANY TABLE-Systemprivileg.

Bevor Sie eine reguläre Ansicht aktivieren, müssen Sie deaktivierte Ansichten, die sie referenziert, wieder aktivieren.

Kontext und Bemerkungen

Wenn Sie eine Ansicht deaktivieren, werden automatisch andere Ansichten deaktiviert, die sie direkt oder indirekt referenzieren. Daher müssen Sie, wenn Sie eine Ansicht wieder aktivieren, auch alle anderen Ansichten wieder aktivieren, die von der ersten Ansicht abhängig waren, als diese deaktiviert wurde. Mit der sa_dependent_views-Systemprozedur können Sie eine Liste der abhängigen Ansichten ermitteln, bevor Sie eine Ansicht deaktivieren.

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Um eine reguläre Ansicht zu deaktivieren, führen Sie die Anweisung ALTER VIEW ... DISABLE aus.
3. Um eine reguläre Ansicht zu aktivieren, führen Sie die Anweisung ALTER VIEW ... ENABLE aus.

Ergebnisse

Wenn Sie eine reguläre Ansicht deaktivieren, bewahrt der Datenbankserver die Definition der Ansicht in der Datenbank auf. Die Ansicht steht dann jedoch nicht mehr für die Verwendung mit einer Abfrage zur Verfügung.

Wenn eine Abfrage ausdrücklich eine deaktivierte Ansicht referenziert, schlägt die Abfrage fehl und es wird ein Fehler gemeldet.

Nächste Schritte

Daher müssen Sie, wenn Sie eine Ansicht wieder aktivieren, auch alle anderen Ansichten wieder aktivieren, die von der ersten Ansicht abhängig waren, bevor diese deaktiviert wurde. Mit der sa_dependent_views-Systemprozedur können Sie eine Liste der abhängigen Ansichten ermitteln, bevor Sie eine Ansicht deaktivieren.

Wenn Sie eine reguläre Ansicht aktivieren, wird sie vom Datenbankserver anhand der Definition, die für die Ansicht in der Datenbank gespeichert ist, neu kompiliert. Wenn die Kompilierung erfolgreich ist, wechselt der Ansichtsstatus auf GÜLTIG. Eine nicht erfolgreiche Neukompilation könnte darauf hinweisen, dass das Schema in einem oder mehreren der referenzierten Objekte geändert wurde. Falls dies so ist, müssen Sie die Ansichtsdefinition oder die referenzierten Objekte ändern, bis sie miteinander übereinstimmen, und dann die Ansicht aktivieren.

Wenn eine Ansicht deaktiviert wurde, muss sie ausdrücklich erneut aktiviert werden, damit der Datenbankserver sie wieder verwenden kann.

Beispiel

Das folgende Beispiel deaktiviert eine reguläre Ansicht namens ViewSalesOrders mit dem Eigentümer GROUPO.

```
ALTER VIEW GROUPO.ViewSalesOrders DISABLE;
```

Das folgende Beispiel reaktiviert eine normale Ansicht namens ViewSalesOrders mit dem Eigentümer GROUPO.

```
ALTER VIEW GROUPO.ViewSalesOrders ENABLE;
```

Siehe auch

- „sa_dependent_views-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ALTER VIEW-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSDEPENDENCY-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Daten in einer regulären Ansicht durchsuchen

Sie können Daten in einer regulären Ansicht durchsuchen. Reguläre Ansichten werden in der Datenbank als Definitionen für die Ansicht gespeichert. Die Ansicht wird mit Daten gefüllt, wenn sie abgefragt wird, sodass die Daten in der Ansicht aktuell sind.

Voraussetzungen

Die reguläre Ansicht muss bereits festgelegt sein und es muss sich um eine gültige Ansicht handeln, die aktiviert ist.

Sie müssen Eigentümer sein oder eines der folgenden Privilegien haben:

- SELECT-Privileg für die Ansicht
- SELECT ANY TABLE-Systemprivileg

Kontext und Bemerkungen

Diese Aufgabe wird in Sybase Central gestartet, wo Sie die anzuzeigende reguläre Ansicht anfordern, und in Interactive SQL abgeschlossen, wo die Daten für die reguläre Ansicht angezeigt werden.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie im linken Fensterausschnitt auf **Ansichten**.
3. Wählen Sie eine Ansicht aus und klicken Sie dann auf **Datei » Daten in Interactive SQL anzeigen**.

Ergebnisse

Interactive SQL wird geöffnet und der Inhalt der Ansicht wird im Fensterausschnitt **Ergebnisse** auf der Registerkarte **Ergebnisse** angezeigt.

Siehe auch

- „Reguläre Ansichten“ auf Seite 46
- „Deaktivieren oder Aktivieren einer regulären Ansicht (Sybase Central)“ auf Seite 52
- „Abfragen“ auf Seite 291
- „Interactive SQL“ [*SQL Anywhere Server - Datenbankadministration*]
- „SELECT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Materialisierte Ansichten

Eine **materialisierte Ansicht** ist eine Ansicht, deren Ergebnismenge vorab aus den Basistabellen berechnet wurde, auf die sie sich bezieht, und auf der Festplatte gespeichert ist, ähnlich wie bei einer Basistabelle. Vom Konzept her ist eine materialisierte Ansicht sowohl eine Ansicht (sie verfügt über eine im Katalog gespeicherte Abfragespezifikation) als auch eine Tabelle (sie verfügt über beständige materialisierte Zeilen). Sie können daher viele Vorgänge, die für Tabellen gedacht sind, auch für materialisierte Ansichten durchführen. Beispielsweise können Sie Indizes für materialisierte Ansichten erstellen.

Wenn Sie eine materialisierte Ansicht erstellen, validiert der Datenbankserver die Definition, um sicherzustellen, dass die Kompilation einwandfrei durchgeführt wird. Alle Spalten- und Tabellenreferenzen werden vom Datenbankserver vollständig qualifiziert, um sicherzustellen, dass alle Benutzer mit Zugriff auf die Ansicht eine identische Definition sehen. Nachdem Sie eine materialisierte Ansicht erstellt haben, füllen Sie sie mit Daten. Dieser Vorgang wird auch als **Initialisieren** der Ansicht bezeichnet.

Materialisierte Ansichten werden im Ordner **Ansichten** in Sybase Central angezeigt.

Siehe auch

- „Tipp: Verbessern der Abfrageperformance durch materialisierte Ansichten“ auf Seite 234

Performanceverbesserungen durch materialisierte Ansichten

Bei Verwendung unter den richtigen Bedingungen können materialisierte Ansichten die Performance erheblich verbessern, indem kostenträchtige Vorgänge wie etwa Joins vorab berechnet und die Ergebnisse in Form einer Ansicht auf der Festplatte gespeichert werden. Der Optimierer berücksichtigt materialisierte Ansichten bei der Entscheidung über die effizienteste Methode, um eine Abfrage zu erfüllen, auch wenn die materialisierte Ansicht nicht in der Abfrage referenziert wird.

Beim Design Ihrer Anwendung sollten Sie in Erwägung ziehen, materialisierte Ansichten für häufig ausgeführte kostenträchtige Abfragen oder kostenträchtige Teile Ihrer Abfragen festzulegen, etwa wenn intensive Aggregations- oder Join-Vorgänge erforderlich sind. Materialisierte Ansichten dienen der Performance-Verbesserung in Umgebungen, in denen Folgendes zutrifft:

- Die Datenbank ist groß.
- Häufige Abfragen führen zu wiederholter Aggregation und zu Join-Vorgängen für große Datenmengen.
- Änderungen der zugrunde liegenden Daten kommen nur selten vor.
- Der Zugriff auf aktuelle Daten ist nicht von entscheidender Bedeutung.

Berücksichtigen Sie die folgenden Anforderungen, Einstellungen und Einschränkungen, bevor Sie eine materialisierte Ansicht verwenden:

- **Anforderungen an den Plattenspeicher** Da materialisierte Ansichten ein Duplikat von Daten aus Basistabellen enthalten, müssen Sie möglicherweise zusätzlichen Speicherplatz für die Datenbank bereitstellen, um die erstellten materialisierten Ansichten zu speichern. Die Anforderungen an zusätzlichen Speicherplatz müssen sorgfältig geprüft werden, damit der erhaltene Vorteil den Kosten für die Verwendung materialisierter Ansichten gegenübergestellt werden kann.
- **Wartungskosten und Anforderungen an die Datenaktualität** Die Daten in materialisierten Ansichten müssen aktualisiert werden, wenn sich die Daten in den Basistabellen ändern. Beim Festlegen der Intervalle für die Aktualisierung materialisierter Ansichten müssen Faktoren berücksichtigt werden, die möglicherweise im Konflikt miteinander stehen, wie z.B.:
 - **Häufigkeit der Änderungen zugrunde liegender Daten** Häufige oder umfangreiche Datenänderungen führen dazu, dass manuelle Ansichten veralten. Wenn die Aktualität der Daten wichtig ist, sollten Sie sich die Verwendung einer Sofortansicht überlegen.
 - **Kosten der Aktualisierung** Abhängig von der Komplexität der Abfrage für eine materialisierte Ansicht und der betroffenen Datenmenge kann die Berechnung, die für eine Aktualisierung erforderlich ist, sehr kostenträchtig sein, und eine häufige Aktualisierung von materialisierten Ansichten führt möglicherweise zu einer nicht akzeptablen Systemlast auf dem Datenbankserver. Zusätzlich kann auf materialisierte Ansichten während des Aktualisierungsvorgangs nicht zugegriffen werden.
 - **Anwendungsanforderungen an die Datenaktualität** Wenn der Datenbankserver eine veraltete materialisierte Ansicht verwendet, liefert er den Anwendungen veraltete Daten. Veraltete

Daten entsprechen nicht mehr dem aktuellen Zustand der Basistabellen. Der Grad der Veraltung wird durch die Häufigkeit bestimmt, mit der die materialisierte Ansicht aktualisiert wird. Eine Anwendung muss in der Lage sein, den Grad der Veraltung festzulegen, den sie zum Erreichen einer verbesserten Performance tolerieren kann.

- **Anforderungen an die Datenkonsistenz** Beim Aktualisieren von materialisierten Ansichten müssen Sie die Konsistenz festlegen, mit der die materialisierten Ansichten aktualisiert werden sollen.
 - **Verwendung bei der Optimierung** Sie sollten sicherstellen, dass der Optimierer die materialisierte Ansicht berücksichtigt, wenn er eine Abfrage ausführt. Eine Liste der materialisierten Ansichten, die für eine bestimmte Abfrage verwendet werden, finden Sie im grafischen Plan der Abfrage im Fenster **Erweiterte Details** in Interactive SQL.
- Sie können außerdem den Anwendungsprofilerstellungsmodus in Sybase Central benutzen, um festzustellen, ob eine materialisierte Ansicht während der Enumerationsphase einer Abfrage in Betracht gezogen wurde, indem Sie die Zugriffspläne prüfen, die durch den Optimierer aufgelistet wurden. Die Protokollierung muss aktiviert und so konfiguriert werden, dass der Protokollierungstyp `OPTIMIZATION_LOGGING` einbezogen wird, um die Zugriffspläne anzuzeigen, die vom Optimierer aufgelistet werden.
- **Datenändernde Vorgänge** Materialisierte Ansichten sind schreibgeschützt. Datenändernde Vorgänge wie `INSERT`, `LOAD`, `DELETE` oder `UPDATE` können mit ihnen nicht ausgeführt werden.
 - **Schlüssel, Integritätsregeln, Trigger und Artikel** Während Sie Indizes für materialisierte Ansichten erstellen können, ist es nicht möglich, Schlüssel, Integritätsregeln, Trigger oder Artikel für diese zu erstellen.

Siehe auch

- „Fortgeschrittene Aufgaben: Einstellungen zum Steuern der Datenveraltung in materialisierten Ansichten“ auf Seite 79
- „Tipp: Verbessern der Abfrageperformance durch materialisierte Ansichten“ auf Seite 234
- „Aktivieren und Deaktivieren der Verwendung einer materialisierten Ansicht durch den Optimierer“ auf Seite 72
- „REFRESH MATERIALIZED VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ auf Seite 341
- „Anwendungsprofilerstellung“ auf Seite 156
- „Einstellen des Aktualisierungstyps auf "Manuell" oder "Sofort"“ auf Seite 59

Materialisierte Ansichten und Ansichtsabhängigkeiten

Sie können kontrollieren, ob eine materialisierte Ansicht für die Verwendung durch den Datenbankserver verfügbar ist, indem Sie sie aktivieren oder deaktivieren. Eine deaktivierte materialisierte Ansicht wird vom Optimierer während der Optimierung nicht berücksichtigt. Wenn eine Abfrage ausdrücklich eine deaktivierte materialisierte Ansicht referenziert, schlägt die Abfrage fehl und es wird ein Fehler gemeldet. Wenn Sie eine materialisierte Ansicht deaktivieren, löscht der Datenbankserver die Daten der Ansicht, behält die Definition jedoch in der Datenbank. Wenn Sie eine materialisierte Ansicht wieder aktivieren,

befindet sie sich im nicht initialisierten Zustand, und Sie müssen sie aktualisieren, um sie mit Daten zu füllen.

Reguläre Ansichten, die von einer materialisierten Ansicht abhängig sind, werden automatisch vom Datenbankserver deaktiviert, wenn die materialisierte Ansicht deaktiviert wird. Daher müssen Sie, wenn Sie eine materialisierte Ansicht wieder aktivieren, auch alle abhängigen Ansichten wieder aktivieren. Aus diesem Grund sollten Sie die Liste der Ansichten abrufen, die von der materialisierten Ansicht abhängig sind, bevor Sie diese deaktivieren. Sie können dazu die `sa_dependent_views`-Systemprozedur verwenden. Diese Prozedur prüft die `ISYSDEPENDENCY`-Systemtabelle und liefert eine Liste der abhängigen Ansichten, falls vorhanden.

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.

Siehe auch

- „Aktivieren oder Deaktivieren einer materialisierten Ansicht“ auf Seite 68
- „Abhängigkeiten und Vorgänge, die ein Schema ändern“ auf Seite 43
- „Abhängigkeitsinformationen abrufen (SQL)“ auf Seite 45
- „`sa_dependent_views`-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Einstellen des Aktualisierungstyps auf "Manuell" oder "Sofort"

Es gibt zwei **Aktualisierungstypen** bei materialisierten Ansichten: manuelle und sofortige.

- **Manuelle Ansichten** Eine manuelle materialisierte Ansicht, oder **manuelle Ansicht**, ist eine materialisierte Ansicht mit einem als `MANUAL REFRESH` definierten Aktualisierungstyp. Daten in manuellen Ansichten können veralten, weil manuelle Ansichten nicht aktualisiert werden, bis eine Aktualisierung explizit angefordert wird, z.B. unter Verwendung der Anweisung `REFRESH MATERIALIZED VIEW` oder der Systemprozedur `sa_refresh_materialized_views`. Wenn Sie eine materialisierte Ansicht erstellen, ist sie standardmäßig eine manuelle Ansicht.

Eine manuelle Ansicht wird als veraltet angesehen, wenn sich eine der Basistabellen ändert, selbst wenn sich die Änderung nicht auf die Daten in der materialisierten Ansicht auswirkt. Sie können ermitteln, ob eine manuelle Ansicht als veraltet angesehen wird, indem Sie den Wert "DataStatus" prüfen, der von der Systemprozedur `sa_materialized_view_info` zurückgegeben wird. Wenn "S" (für "stale") zurückgegeben wird, ist die manuelle Ansicht veraltet.

- **Sofortansichten** Eine materialisierte Sofortansicht oder **Sofortansicht**, ist eine materialisierte Ansicht mit einem als `IMMEDIATE REFRESH` definierten Aktualisierungstyp. Daten in einer Sofortansicht werden automatisch aktualisiert, wenn sich Änderungen in den Basistabellen auf die Daten in der Ansicht auswirken. Wenn sich Änderungen in den Basistabellen nicht auf die Daten in der Ansicht auswirken, wird die Ansicht nicht aktualisiert.

Außerdem müssen beim Aktualisieren einer Sofortansicht nur veraltete Zeilen geändert werden. Dies ist der Unterschied zu einer manuellen Ansicht, bei der beim Aktualisieren alle Daten gelöscht und neu erstellt werden.

Sie können eine manuelle Ansicht in eine Sofortansicht ändern, und umgekehrt. Allerdings besteht der Vorgang, der eine manuelle Ansicht in eine Sofortansicht ändert, aus mehr Schritten.

Ein Ändern des Aktualisierungstyps bei einer materialisierten Ansicht kann sich auf den Status und die Eigenschaften der Ansicht auswirken, besonders wenn Sie eine manuelle Ansicht in eine Sofortansicht ändern.

Siehe auch

- „Veraltung und manuelle materialisierte Ansichten“ auf Seite 60
- „Fortgeschrittene Aufgaben: Aktualisierungstyp für eine materialisierte Ansicht ändern“ auf Seite 74
- „Fortgeschrittene Aufgaben: Status und Eigenschaften von materialisierten Ansichten“ auf Seite 76
- „sa_materialized_view_info-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Veraltung und manuelle materialisierte Ansichten

Materialisierte Ansichten, die manuell aktualisiert werden, veralten, wenn Änderungen an den zugrunde liegenden Basistabellen auftreten. Der Optimierer berücksichtigt eine materialisierte Ansicht nicht als Möglichkeit zum Erfüllen einer Abfrage, wenn die Daten den für die Ansicht konfigurierten Veraltungsschwellenwert überschritten haben. Ein Aktualisieren einer manuellen Ansicht bedeutet, dass der Datenbankserver die Abfragedefinition für die Ansicht wieder ausführt und die Ansichtsdaten durch die neue Ergebnismenge der Abfrage ersetzt. Das Aktualisieren macht die Ansichtsdaten mit den Basisdaten konsistent. Sie sollten eine akzeptable Grenze in Bezug auf die Veralterung der Daten für die manuelle Ansicht festlegen und eine Aktualisierungsstrategie entwerfen. Ihre Strategie sollte die Zeit berücksichtigen, die für die Durchführung einer Aktualisierung benötigt wird, weil die Ansicht während des Aktualisierungsvorgangs für Abfragen nicht verfügbar ist.

Sie können auch eine Strategie einsetzen, bei der die Ansicht unter Verwendung von Ereignissen aktualisiert wird. Sie können beispielsweise ein Ereignis erstellen, das die Aktualisierung in regulären Intervallen durchführt.

Materialisierte Sofortansichten müssen nicht aktualisiert werden, es sei denn, sie sind nicht initialisiert (enthalten keine Daten), z.B. nach einer Kürzung.

Mit der Datenbankoption `materialized_view_optimization` können Sie einen Veraltungsschwellenwert konfigurieren, nach dessen Überschreitung der Optimierer die materialisierte Ansicht beim Verarbeiten von Abfragen nicht mehr benutzen darf.

Upgrade von Datenbanken mit materialisierten Ansichten

Es empfiehlt sich, materialisierte Ansichten zu aktualisieren, nachdem Sie ein Upgrade des Datenbankservers durchgeführt oder die Datenbank neu aufgebaut bzw. umgestellt haben.

Siehe auch

- „Fortgeschrittene Aufgaben: Einstellungen zum Steuern der Datenveraltung in materialisierten Ansichten“ auf Seite 79
- „Materialisierten Ansichten manuell aktualisieren“ auf Seite 67
- „materialized_view_optimization-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Einschränkungen für materialisierte Ansichten

Einschränkungen bei Erstellung, Initialisierung, Aktualisierung und Ansichtenübereinstimmung von materialisierten Ansichten

- Wenn Sie eine materialisierte Ansicht erstellen, muss die Definition der materialisierten Ansicht die Spaltennamen ausdrücklich definieren. Sie dürfen kein `SELECT *`-Konstrukt als Teil der Spaltendefinition verwenden.
- Beziehen Sie keine Spalten, die als `TIMESTAMP WITH TIME ZONE` definiert sind, in die materialisierte Ansicht ein. Der Wert der `time_zone_adjustment`-Option variiert zwischen Verbindungen, abhängig von ihrem Standort und der Jahreszeit. Dies führt zu falschen Ergebnissen und unerwartetem Verhalten.
- Wenn Sie eine materialisierte Ansicht erstellen, darf die Definition der materialisierten Ansicht Folgendes nicht enthalten:
 - Referenzen zu anderen Ansichten (materialisierte oder nicht materialisierte)
 - Referenzen zu entfernten oder temporären Tabellen
 - Variablen wie `CURRENT USER`; alle Ausdrücke müssen deterministisch sein
 - Aufrufe von gespeicherten Prozeduren, benutzerdefinierten Funktionen oder externen Funktionen
 - `Transact-SQL`-Outer-Joins
 - `FOR XML`-Klauseln

Der `grouped-select-project-join`-Abfrageblock muss `COUNT(*)` in der `SELECT`-Liste enthalten und als Aggregatfunktionen sind nur `SUM` und `COUNT` erlaubt.

- Die folgenden Datenbankoptionen müssen die angegebenen Einstellungen haben, wenn eine materialisierte Ansicht erstellt wird. Andernfalls wird ein Fehler gemeldet. Diese Einstellungen der Datenbankoptionen sind außerdem erforderlich, damit die Ansicht vom Optimierer benutzt werden kann:
 - `ansinull=On`
 - `conversion_error=On`
 - `divide_by_zero_error=On`
 - `sort_collation=Internal`
 - `string_truncation=On`
- Die folgenden Einstellungen der Datenbankoptionen werden für jede materialisierte Ansicht gespeichert, wenn sie erstellt wird. Die aktuellen Optionswerte für die Verbindung müssen mit den für

eine materialisierte Ansicht gespeicherten Werten übereinstimmen, damit die Ansicht bei der Optimierung verwendet werden kann:

- date_format
 - date_order
 - default_timestamp_increment
 - first_day_of_week
 - nearest_century
 - precision
 - scale
 - time_format
 - timestamp_format
 - timestamp_with_time_zone_format
 - default_timestamp_increment
 - uuid_has_hyphens
- Wenn eine Ansicht aktualisiert wird, werden alle oben in der Liste angeführten Verbindungseinstellungen ignoriert. Stattdessen werden die Einstellungen der Datenbankoptionen (die mit den gespeicherten Einstellungen für die Ansicht übereinstimmen müssen) verwendet.

ORDER BY-Klausel in der Definition einer materialisierten Ansicht hat keine Auswirkung

Materialisierte Ansichten ähneln Basistabellen insofern, als die Zeilen nicht in einer bestimmten Reihenfolge gespeichert werden. Der Datenbankserver ordnet die Zeilen so effizient wie möglich an, wenn er die Daten berechnet. Die Verwendung einer ORDER BY-Klausel in der Definition einer materialisierten Ansicht hat daher keine Auswirkung auf die Reihenfolge der Zeilen, wenn die Ansicht materialisiert wird. Außerdem wird die ORDER BY-Klausel in der Ansichtsdefinition vom Optimierer ignoriert, wenn die Ansichtenübereinstimmung durchgeführt wird.

Einschränkungen beim Ändern des Typs einer materialisierten Ansicht von "Manuell" auf "Sofort"

Es gelten die folgenden Einschränkungen, wenn eine manuelle Ansicht in eine Sofortansicht geändert wird. Es wird ein Fehler gemeldet, wenn die Ansicht eine dieser Einschränkungen verletzt:

Hinweis

Sie können die Systemprozedur `sa_materialized_view_can_be_immediate` verwenden um herauszufinden, ob eine manuelle Ansicht in Frage kommt, in eine Sofortansicht geändert zu werden.

- Die Ansicht muss nicht-initialisiert sein.
- Wenn die Ansicht keine Outer-Joins enthält, muss die Ansicht einen eindeutigen Index auf nicht-nullwertfähigen Spalten haben. Falls die Ansicht Outer-Joins enthält, muss die Ansicht einen eindeutigen Index auf nicht-nullwertfähigen Spalten haben bzw. ein eindeutiger Index muss als `WITH NULLS NOT DISTINCT` für nullwertfähige Spalten deklariert sein.
- Wenn die Ansichtsdefinition eine gruppierte Abfrage ist, müssen die Spalten des eindeutigen Indexes den SELECT-Listeneinträgen entsprechen, die keine Aggregatfunktionen sind.

- Die Ansichtsdefinition darf nicht enthalten:
 - GROUPING SETS-Klauseln
 - CUBE-Klauseln
 - ROLLUP-Klauseln
 - DISTINCT-Klauseln
 - Zeilenlimit-Klauseln
 - Nicht-deterministische Ausdrücke
 - Self- und Recursive-Joins
 - LATERAL-, CROSS APPLY- oder APPLY-Klausel
- Die Ansichtsdefinition muss ein einzelner "select-project-join"- oder ein "grouped-select-project-join"-Abfrageblock sein, und der "grouped-select-project-join"-Abfrageblock darf keine HAVING-Klausel enthalten.
- Der "grouped-select-project-join"-Abfrageblock muss COUNT (*) in der SELECT-Liste enthalten und als Aggregatfunktionen sind nur SUM und COUNT erlaubt.
- Eine Aggregatfunktion in der SELECT-Liste kann nicht in einem komplexen Ausdruck referenziert werden. Beispielsweise ist SUM(expression) + 1 in der SELECT-Liste nicht zulässig.
- Wenn die SELECT-Liste die Aggregatfunktion SUM(expression) enthält und expression ein nullwertfähiger Ausdruck ist, muss die SELECT-Liste eine Aggregatfunktion COUNT(expression) enthalten.
- Wenn die Ansichtsdefinition Outer-Joins enthält (LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN), muss die Ansichtsdefinition die folgenden zusätzlichen Bedingungen erfüllen:
 1. Wenn eine Tabelle "T" in einer ON-Bedingung eines OUTER-Joins als bewahrte Seite referenziert wird, muss "T" einen Primärschlüssel haben und die Primärschlüsselspalten müssen in der SELECT-Liste der Ansicht enthalten sein. Die materialisierte Sofortansicht "V", die als `SELECT T1.pk, R1.X FROM T1, T2 LEFT OUTER JOIN (R1 KEY JOIN R2) ON T1.Y = R.Y` festgelegt ist, hat die bewahrte Tabelle "T1", die in der ON-Klausel referenziert wird, und ihre Primärschlüsselspalte "T1.pk" ist in der SELECT-Liste der materialisierten Sofortansicht "V" enthalten.
 2. Für jede nullwertliefernde Seite eines Outer-Joins muss mindestens eine Basistabelle vorhanden sein, sodass eine der nicht nullwertfähigen Spalten in der SELECT-Liste der materialisierten Sofortansicht enthalten ist. Beispiel: Für die materialisierte Sofortansicht "V", die als `SELECT T1.pk, R1.X FROM T1, T2 LEFT OUTER JOIN (R1 KEY JOIN R2) ON T1.Y = R1.Y` definiert ist, ist die nullwertliefernde Seite des Left-Outer-Joins der Tabellenausdruck `(R1 KEY JOIN R2)`. Die Spalte "R1.X" befindet sich in der SELECT-Liste von "V" und "R1.X" ist eine nicht nullwertfähige Spalte der Tabelle "R1".
 3. Wenn es sich bei der Ansicht um eine gruppierte Ansicht handelt und die frühere Bedingung nicht mehr zutrifft, muss für jede nullwertliefernde Seite eines Outer-Joins mindestens eine Basistabelle "T" vorhanden sein, sodass eine ihrer nicht nullwertfähigen Spalten, "T.C", in der Aggregatfunktion COUNT(T.C) in der SELECT-Liste der materialisierten Sofortansicht verwendet wird. Für die materialisierte Sofortansicht "V", die als `SELECT T1.pk,`

`COUNT(R1.X) FROM T1, T2 LEFT OUTER JOIN (R1 KEY JOIN R2) ON T1.Y = R1.Y GROUP BY T1.pk` definiert ist, ist beispielsweise die nullwertliefernde Seite des Left-Outer-Joins der Tabellenausdruck `(R1 KEY JOIN R2)`. Die Aggregatfunktion `COUNT(R1.X)` in der SELECT-Liste von "V" und "R1.X" ist eine nicht nullwertfähige Spalte der Tabelle "R1".

4. Die folgenden Bedingungen müssen durch die Prädikate der Ansichten mit Outer-Joins erfüllt sein:

- Die Prädikate der ON-Klausel für LEFT, RIGHT und FULL OUTER JOINS müssen sich sowohl auf bewahrende als auch auf nullwertliefernde Tabellenausdrücke beziehen. Beispiel: `T LEFT OUTER JOIN R ON R.X = 1` erfüllt diese Bedingung nicht, da das Prädikat "`R.X=1`" nur die nullwertliefernde Seite "R" referenziert.
- Alle Prädikate müssen nullwertliefernde Zeilen ablehnen, die mit Hilfe eines verschachtelten Outer-Joins erstellt wurden. Wenn ein Prädikat also einen Tabellenausdruck referenziert, der einen Nullwert von einem verschachtelten Outer-Join erhält, dann muss er alle Zeilen ablehnen, für die von diesem Outer-Join Nullwerte generiert wurden.

Die Ansicht "V1" `SELECT T1.pk, R1.X FROM T1, T2 LEFT OUTER JOIN (R1 KEY JOIN R2) ON (T1.Y = R1.Y) WHERE R1.Z = 10` hat z.B. das Prädikat `R1.Z=10`, das die Tabelle "R1" referenziert, die Nullwerte von `T2 LEFT OUTER JOIN (R1 KEY JOIN R2)` erhalten kann. Daher muss sie alle Zeilen, für die Nullwerte generiert wurden, ablehnen. Dies ist der Fall, weil das Prädikat als UNKNOWN ausgewertet wird, wenn die Spalte "R1.Z" NULL ist.

Die Ansicht "V2" `SELECT T1.pk, R1.X FROM T1, T2 LEFT OUTER JOIN (R1 KEY JOIN R2) ON (T1.Y = R1.Y) WHERE R1.Z IS NULL` verfügt jedoch nicht über diese Eigenschaft. Das Prädikat `R1.Z IS NULL` referenziert die nullwertliefernde Seite "R1", aber es wird als TRUE ausgewertet, wenn die Tabelle "R1" einen Nullwert liefert (d.h. die Spalte "R1.Z" gleich NULL ist). Die Methode der Ablehnung von nullwertliefernden Zeilen ist nicht so restriktiv wie eine nullwertintolerante Eigenschaft. Beispiel: Das Prädikat `R.X IS NOT DISTINCT FROM T.X and rowid(T) IS NOT NULL` ist nicht nullwertintolerant gegenüber Tabelle "T", da es als TRUE ausgewertet wird, wenn "T.X" NULL ist. Das Prädikat lehnt jedoch alle Zeilen ab, die Nullwerte aus der Basistabelle "T" erhalten.

Siehe auch

- [„sa_materialized_view_can_be_immediate-Systemprozedur“ \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- [„Fortgeschrittene Aufgaben: Status und Eigenschaften von materialisierten Ansichten“ auf Seite 76](#)
- [„Indizes erstellen“ auf Seite 33](#)
- [„Materialisierte Ansichten erstellen“ auf Seite 65](#)
- [TimeZoneAdjustment-Verbindungseigenschaft \[SQL Anywhere Server - Datenbankadministration\]](#)
- [„CREATE MATERIALIZED VIEW-Anweisung“ \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- [„REFRESH MATERIALIZED VIEW-Anweisung“ \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- [„Outer-Joins“ auf Seite 505](#)

Materialisierte Ansichten erstellen

Erstellen Sie eine materialisierte Ansicht zum Speichern von Daten aus Abfragen, die häufig ausgeführt werden und die zu repetitiven Aggregations- und Join-Vorgängen für große Datenmengen führen.

Materialisierte Ansichten können die Performance verbessern, indem kostenträchtige Vorgänge vorab berechnet und die Ergebnisse in Form einer Ansicht auf der Festplatte gespeichert werden.

Voraussetzungen

Wenn Sie eine materialisierte Ansicht erstellen möchten, deren Eigentümer Sie sind, müssen Sie das CREATE MATERIALIZED VIEW-Systemprivileg sowie das SELECT-Privileg für alle Basistabellen haben.

Wenn Sie eine materialisierte Ansicht erstellen möchten, deren Eigentümer andere Benutzer sind, müssen Sie das CREATE ANY MATERIALIZED VIEW-Systemprivileg oder CREATE ANY OBJECT-Systemprivileg sowie das SELECT-Privileg für alle Basistabellen haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf **Ansichten** und klicken Sie auf **Neu » Materialisierte Ansicht**.
3. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen von materialisierten Ansichten**.

Ergebnisse

Eine nicht initialisierte materialisierte Ansicht wird in der Datenbank erstellt. Sie enthält noch keine Daten.

Nächste Schritte

Sie müssen die materialisierte Ansicht initialisieren, um sie mit Daten zu füllen, bevor Sie sie verwenden.

Siehe auch

- „Materialisierte Ansichten“ auf Seite 56
- „SQL Anywhere-Beispieldatenbank“ [*SQL Anywhere 16 - Einführung*]
- „Einschränkungen für materialisierte Ansichten“ auf Seite 61
- „Materialisierte Ansichten löschen“ auf Seite 70
- „Initialisieren einer materialisierten Ansicht“ auf Seite 66
- „CREATE MATERIALIZED VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REFRESH MATERIALIZED VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Initialisieren einer materialisierten Ansicht

Initialisieren Sie eine materialisierte Ansicht, um sie mit Daten zu füllen und für die Verwendung durch den Datenbankserver verfügbar zu machen. Zum Initialisieren einer materialisierten Ansicht müssen Sie dieselben Schritte befolgen wie zum Aktualisieren einer materialisierten Ansicht.

Voraussetzungen

Sie müssen Eigentümer der materialisierten Ansicht sein, das INSERT-Privileg für die materialisierte Ansicht haben oder das INSERT ANY TABLE-Privileg haben.

Bevor Sie eine materialisierte Ansicht erstellen, initialisieren oder aktualisieren, müssen Sie dafür sorgen, dass alle Einschränkungen befolgt wurden.

Kontext und Bemerkungen

Mit der sa_refresh_materialized_views-Systemprozedur können Sie alle nicht initialisierten materialisierten Ansichten in der Datenbank auf einmal initialisieren.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Ansichten**.
3. Rechtsklicken Sie auf eine materialisierte Ansicht und klicken Sie auf **Daten aktualisieren**.
4. Wählen Sie eine Isolationsstufe und klicken Sie auf **OK**.

Ergebnisse

Die materialisierte Ansicht wird mit Daten gefüllt und wird dadurch für die Verwendung durch den Datenbankserver verfügbar. Sie können nun die materialisierte Ansicht abfragen.

Nächste Schritte

Führen Sie eine Abfrage der materialisierten Ansicht aus, um sicherzustellen, dass sie die erwarteten Daten zurückgibt.

Im Fall eines fehlgeschlagenen Initialisierungs- bzw. Aktualisierungsversuchs wird eine materialisierte Ansicht in den nicht initialisierten Zustand zurückversetzt. Wenn die Initialisierung fehlschlägt, überprüfen Sie die Definition der materialisierten Ansicht, um zu bestätigen, dass die angegebenen Basistabellen und Spalten gültige und verfügbare Objekte in Ihrer Datenbank sind.

Siehe auch

- „Materialisierte Ansichten“ auf Seite 56
- „Materialisierte Ansichten löschen“ auf Seite 70
- „Einschränkungen für materialisierte Ansichten“ auf Seite 61
- „Aktivieren oder Deaktivieren einer materialisierten Ansicht“ auf Seite 68
- „CREATE MATERIALIZED VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REFRESH MATERIALIZED VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_refresh_materialized_views-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Materialisierten Ansichten manuell aktualisieren

Materialisierte Ansichten, die nicht so konfiguriert sind, dass die Aktualisierung automatisch erfolgt, müssen manuell aktualisiert werden, um ihre Daten auf den neuesten Stand zu bringen.

Voraussetzungen

Sie müssen Eigentümer der materialisierten Ansicht sein, das INSERT-Privileg für die materialisierte Ansicht haben oder das INSERT ANY TABLE-Systemprivileg haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Ansichten**.
3. Rechtsklicken Sie auf eine materialisierte Ansicht und klicken Sie auf **Daten aktualisieren**.
4. Wählen Sie eine Isolationsstufe und klicken Sie auf **OK**.

Ergebnisse

Die Daten in der materialisierten Ansicht werden aktualisiert und die jüngsten Daten aus den zugrunde liegenden Objekten angezeigt.

Nächste Schritte

Führen Sie eine Abfrage der materialisierten Ansicht aus, um sicherzustellen, dass sie die erwarteten Daten zurückgibt.

Im Fall eines fehlgeschlagenen Aktualisierungsversuchs wird eine materialisierte Ansicht in den nicht initialisierten Zustand konvertiert. Wenn dies vorkommt, überprüfen Sie die Definition der materialisierten Ansicht, um zu bestätigen, dass die angegebenen Basistabellen und Spalten gültige und verfügbare Objekte in Ihrer Datenbank sind.

Siehe auch

- „Materialisierte Ansichten“ auf Seite 56
- „REFRESH MATERIALIZED VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Materialisierte Ansichten löschen“ auf Seite 70
- „Fortgeschrittene Aufgaben: Aktualisierungstyp für eine materialisierte Ansicht ändern“ auf Seite 74
- „Aufgabenautomatisierung mit Abfolgeplanung und Ereignissen“ [*SQL Anywhere Server - Datenbankadministration*]
- „materialized_view_optimization-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „sa_refresh_materialized_views-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Aktivieren oder Deaktivieren einer materialisierten Ansicht

Sie können steuern, ob eine materialisierte Ansicht für Abfragen verfügbar ist, indem Sie sie aktivieren bzw. deaktivieren.

Voraussetzungen

Sie müssen Eigentümer der materialisierten Ansicht sein oder eines der folgenden Systemprivilegien haben:

- ALTER ANY MATERIALIZED VIEW
- ALTER ANY OBJECT

Zum **Aktivieren** einer materialisierten Ansicht benötigen Sie *außerdem* das SELECT-Privileg für die Basistabelle(n) oder das SELECT ANY TABLE-Systemprivileg.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Ansichten**.

Option	Aktion
Materialisierte Ansicht aktivieren	<ol style="list-style-type: none">a. Rechtsklicken Sie auf die Ansicht und klicken Sie auf Neukompilieren und aktivieren.b. (Optional) Rechtsklicken Sie auf die Ansicht und klicken Sie auf Daten aktualisieren, um die Ansicht mit Daten zu füllen. Dieser Schritt ist optional, weil die erste Abfrage, die in der Ansicht nach dem Aktivieren ausgeführt wird, ebenfalls dazu führt, dass die Ansicht mit Daten gefüllt wird.
Materialisierte Ansicht deaktivieren	Rechtsklicken Sie auf die Ansicht und wählen Sie Deaktivieren .

Ergebnisse

Wenn Sie eine materialisierte Ansicht aktivieren, wird sie für die Verwendung durch den Datenbankserver verfügbar und Sie können sie abfragen.

Wenn Sie eine materialisierte Ansicht deaktivieren, werden die Daten und Indizes gelöscht. Falls es sich um eine Sofortansicht handelt, wird sie in eine manuelle Ansicht geändert. Das Abfragen einer deaktivierten materialisierten Ansicht schlägt fehl und es wird ein Fehler zurückgegeben.

Nächste Schritte

Nachdem Sie eine Ansicht wieder aktiviert haben, müssen Sie die dazugehörigen Indizes neu erstellen und die Ansicht in eine Sofortansicht zurückändern, falls sie zum Zeitpunkt der Deaktivierung eine Sofortansicht war.

Siehe auch

- „Materialisierte Ansichten“ auf Seite 56
- „Fortgeschrittene Aufgaben: Aktualisierungstyp für eine materialisierte Ansicht ändern“ auf Seite 74
- „sa_dependent_views-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Ansichtenabhängigkeiten“ auf Seite 43
- „SYSDEPENDENCY-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER MATERIALIZED VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REFRESH MATERIALIZED VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Definition einer materialisierten Ansicht verbergen

Sie können die Definition einer materialisierten Ansicht vor den Benutzern verbergen. Damit wird die in der Datenbank gespeicherte Ansichtsdefinition verschleiert. Diese Einstellung ist irreversibel.

Voraussetzungen

Sie müssen Eigentümer der materialisierten Ansicht sein oder eines der folgenden Systemprivilegien haben:

- ALTER ANY MATERIALIZED VIEW
- ALTER ANY OBJECT

Kontext und Bemerkungen

Wenn eine materialisierte Ansicht verborgen ist, wird beim Debugging mithilfe des Debugger weder die Ansichtsdefinition angezeigt noch kann die Definition über die Prozedurprofilerstellung ermittelt werden. Die Ansicht kann weiterhin aus der Datenbank entladen und in andere Datenbanken geladen werden.

Das Verbergen einer materialisierten Ansicht kann nicht rückgängig gemacht werden und ist nur mit SQL möglich.

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Führen Sie die Anweisung ALTER MATERIALIZED VIEW...SET HIDDEN aus.

Ergebnisse

Ein automatisches Festschreiben wird ausgeführt.

Die Ansicht ist beim Durchsuchen des Katalogs nicht mehr sichtbar. Die Ansicht kann jedoch immer noch direkt referenziert werden und sie steht für die Verwendung während der Abfrageverarbeitung zur Verfügung.

Beispiel

Die folgenden Anweisungen erstellen die materialisierte Ansicht "EmployeeConfid3", aktualisieren sie und verschleiern ihre Ansichtsdefinition.

```
CREATE MATERIALIZED VIEW EmployeeConfid3 AS
  SELECT EmployeeID, Employees.DepartmentID, SocialSecurityNumber, Salary,
  ManagerID,
  Departments.DepartmentName, Departments.DepartmentHeadID
  FROM Employees, Departments
  WHERE Employees.DepartmentID=Departments.DepartmentID;
REFRESH MATERIALIZED VIEW EmployeeConfid3;
ALTER MATERIALIZED VIEW EmployeeConfid3 SET HIDDEN;
```

Vorsicht

Wenn Sie dieses Beispiel ausgeführt haben, müssen Sie die dabei erstellte materialisierte Ansicht löschen. Andernfalls können Sie keine Schemaänderungen an ihren Basistabellen Employees und Departments durchführen, wenn Sie andere Beispiele ausprobieren.

Siehe auch

- „Materialisierte Ansichten“ auf Seite 56
- „ALTER MATERIALIZED VIEW-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Materialisierte Ansichten löschen“ auf Seite 70
- „DROP STATEMENT-Anweisung [ESQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Materialisierte Ansichten löschen

Sie können eine materialisierte Ansicht aus der Datenbank löschen. Führen Sie diese Aufgabe aus, wenn Sie die materialisierte Ansicht nicht mehr benötigen oder wenn Sie eine Schemaänderung an einem zugrunde liegenden referenzierten Objekt vorgenommen haben, sodass die Definition der materialisierten Ansicht nicht mehr gültig ist.

Voraussetzungen

Sie müssen entweder Eigentümer sein oder das DROP ANY MATERIALIZED VIEW-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Bevor Sie eine materialisierte Ansicht löschen können, müssen Sie alle abhängigen Ansichten deaktivieren oder löschen. Mit der Systemprozedur "sa_dependent_views" können Sie feststellen, ob es abhängige Ansichten zu einer materialisierten Ansicht gibt.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Ansichten**.
3. Rechtsklicken Sie auf die materialisierte Ansicht und klicken Sie auf **Löschen**.
4. Klicken Sie auf **Ja**.

Ergebnisse

Die materialisierte Ansicht wird aus der Datenbank gelöscht.

Nächste Schritte

Falls es reguläre Ansichten gab, die von der materialisierten Ansicht abhängig waren, können Sie diese nicht mehr aktivieren. Sie müssen ihre Definition ändern oder sie löschen.

Siehe auch

- „Materialisierte Ansichten“ auf Seite 56
- „Ansichtenabhängigkeiten“ auf Seite 43
- „DROP MATERIALIZED VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_dependent_views-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Ver- und Entschlüsseln einer materialisierten Ansicht

Materialisierte Ansichten können verschlüsselt werden, um zusätzliche Sicherheit zu erreichen. Falls eine materialisierte Ansicht beispielsweise Daten enthält, die in der zugrunde liegenden Tabelle verschlüsselt waren, könnten Sie auch die materialisierte Ansicht verschlüsseln.

Voraussetzungen

Sie müssen entweder Eigentümer sein oder sowohl das CREATE ANY MATERIALIZED VIEW-Systemprivileg als auch das DROP ANY MATERIALIZED VIEW-Systemprivileg bzw. sowohl das CREATE ANY OBJECT-Systemprivileg als auch das DROP ANY OBJECT-Systemprivileg haben.

Die Tabellenverschlüsselung muss bereits in der Datenbank aktiviert sein, damit eine materialisierte Ansicht verschlüsselt werden kann.

Kontext und Bemerkungen

Zum Verschlüsseln der materialisierten Ansicht werden der Verschlüsselungsalgorithmus und der Schlüssel verwendet, die in der Datenbank festgelegt wurden. Sie können die für Ihre Datenbank gültigen

Verschlüsselungseinstellungen abrufen und feststellen, ob die Tabellenverschlüsselung aktiviert ist, indem Sie die Datenbankverschlüsselungseigenschaft mit die Funktion DB_PROPERTY abfragen:

```
SELECT DB_PROPERTY( 'Encryption' );
```

Ebenso wie die Tabellenverschlüsselung kann sich auch die Verschlüsselung einer materialisierten Ansicht auf die Performance auswirken, da der Datenbankserver die Daten entschlüsseln muss, die er aus der Ansicht abruft.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Ansichten**.
3. Rechtsklicken Sie auf die materialisierte Ansicht und klicken Sie auf **Eigenschaften**.
4. Klicken Sie auf die Registerkarte **Sonstiges**.
5. Aktivieren bzw. deaktivieren Sie das Kontrollkästchen **Ansichtsdaten werden verschlüsselt**.
6. Klicken Sie auf **OK**.

Ergebnisse

Die Daten der materialisierten Ansicht werden verschlüsselt.

Siehe auch

- „Tabellen verschlüsseln“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „ALTER MATERIALIZED VIEW-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DB_PROPERTY-Funktion [System]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Aktivieren und Deaktivieren der Verwendung einer materialisierten Ansicht durch den Optimierer

Auch wenn eine Abfrage keine materialisierte Ansicht referenziert, kann der Optimierer entscheiden, die Ansicht zum Erfüllen einer Abfrage zu verwenden, wenn sich dadurch die Performance verbessert. Sie können die Verwendung einer materialisierten Ansicht durch den Optimierer zum Erfüllen einer Abfrage aktivieren oder deaktivieren.

Voraussetzungen

Sie müssen entweder Eigentümer sein oder das ALTER ANY MATERIALIZED VIEW-Systemprivileg oder das ALTER ANY OBJECT-Systemprivileg haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Ansichten**.
3. Rechtsklicken Sie auf die materialisierte Ansicht und klicken Sie auf **Eigenschaften**.
4. Klicken Sie auf die Registerkarte **Allgemein** und aktivieren bzw. deaktivieren Sie das Kontrollkästchen **In der Optimierung verwendet**.
5. Klicken Sie auf **OK**.

Ergebnisse

Sobald eine materialisierte Ansicht für die Verwendung durch den Optimierer aktiviert ist, berücksichtigt der Optimierer sie bei der Berechnung des besten Plans zum Erfüllen einer Abfrage, auch wenn die Ansicht in der Abfrage nicht explizit referenziert wird. Wenn eine materialisierte Ansicht für die Verwendung durch den Optimierer deaktiviert ist, berücksichtigt der Optimierer sie nicht.

Nächste Schritte

Es kann sinnvoll sein, eine Abfrage der zugrunde liegenden Objekte der Ansicht auszuführen, um anhand des Abfrageausführungsplans zu überprüfen, ob der Optimierer die Ansicht verwendet. Die Verfügbarkeit der Ansicht garantiert jedoch nicht, dass der Optimierer sie verwendet. Die Wahl des Optimierers basiert auf der Performance.

Siehe auch

- „ALTER MATERIALIZED VIEW-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Performanceverbesserungen durch materialisierte Ansichten“ auf Seite 57
- „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ auf Seite 341

Fortgeschrittene Aufgaben: Anzeigen von Informationen aus materialisierten Ansichten im Katalog

Sie können eine Liste aller materialisierten Ansichten und ihren Status anzeigen und außerdem überprüfen, welche Datenbankoptionen bei der Erstellung auf die einzelnen materialisierten Ansichten angewendet wurden.

Voraussetzungen

Die materialisierte Ansicht kann nicht verborgen werden.

Kontext und Bemerkungen

Abhängigkeitsinformationen befinden sich auch in der SYSDEPENDENCY-Systemansicht.

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Um eine Liste aller materialisierten Anweisungen und ihrer Status anzuzeigen, führen Sie die folgende Anweisung aus:

```
SELECT * FROM sa_materialized_view_info();
```

3. Um die Datenbankoptionen zu überprüfen, die bei der Erstellung auf die einzelnen materialisierten Ansichten angewendet wurden, führen Sie die folgende Anweisung aus:

```
SELECT b.object_id, b.table_name, a.option_id, c.option_name,  
a.option_value  
FROM SYSMVOPTION a, SYSTAB b, SYSMVOPTIONNAME c  
WHERE a.view_object_id=b.object_id  
AND b.table_type=2;
```

4. Um eine Liste der regulären Ansichten anzufordern, die von einer angegebenen materialisierten Ansicht abhängig sind, führen Sie die folgende Anweisung aus:

```
CALL sa_dependent_views( 'materialized-view-name' );
```

Ergebnisse

Die angeforderten Informationen zu materialisierten Ansichten werden zurückgegeben.

Siehe auch

- „sa_materialized_view_info-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Fortgeschrittene Aufgaben: Status und Eigenschaften von materialisierten Ansichten“ auf Seite 76
- „sa_dependent_views-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSDEPENDENCY-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSMVOPTION-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSMVOPTIONNAME-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSTAB-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Fortgeschrittene Aufgaben: Aktualisierungstyp für eine materialisierte Ansicht ändern

Wenn Sie eine materialisierte Ansicht erstellen, ist ihr Aktualisierungstyp "Manuell", aber Sie können das auf "Sofort" ändern. Außerdem können Sie eine Sofortansicht wieder auf "Manuell" ändern.

Voraussetzungen

Sie müssen entweder Eigentümer sein oder sowohl das CREATE ANY MATERIALIZED VIEW-Systemprivileg als auch das DROP ANY MATERIALIZED VIEW-Systemprivileg bzw. sowohl das CREATE ANY OBJECT-Systemprivileg als auch das DROP 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.

Um den Typ von "Manuell" auf "Sofort" zu ändern, muss die Ansicht in einem nicht initialisierten Zustand (ohne Daten) sein. Wenn die Ansicht eben erstellt und noch nicht aktualisiert wurde, ist sie nicht initialisiert. Wenn die materialisierte Ansicht Daten enthält, müssen Sie eine TRUNCATE-Anweisung ausführen, um sie in einen nicht initialisierten Zustand zurückzusetzen, bevor Sie den Aktualisierungstyp auf "Sofort" ändern können. Außerdem muss die materialisierte Ansicht einen eindeutigen Index haben und den Einschränkungen entsprechen, die für eine Sofortansicht gelten.

Eine Sofortansicht kann jederzeit auf "Manuell" geändert werden. Dies erfordert über das Ändern des Aktualisierungstyps hinaus keine zusätzlichen Schritte.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Ansichten**.
3. Rechtsklicken Sie auf die materialisierte Ansicht und klicken Sie auf **Eigenschaften**.
4. Wählen Sie im Feld **Aktualisierungstyp** eine der folgenden Optionen:

Option	Aktion
Manuelle Ansicht in Sofortansicht ändern	Sofort
Sofortansicht in manuelle Ansicht ändern	Manuell

5. Klicken Sie auf **OK**.

Ergebnisse

Der Aktualisierungstyp der materialisierten Ansicht wird geändert. Sofortansichten werden aktualisiert, sobald sich die Daten in den zugrunde liegenden Objekten ändern. Manuelle Ansichten werden jedes Mal aktualisiert, wenn Sie sie aktualisieren.

Nächste Schritte

Nachdem Sie eine Ansicht von "Manuell" auf "Sofort" geändert haben, muss die Ansicht initialisiert (aktualisiert) werden, um sie mit Daten zu füllen.

Siehe auch

- „Einstellen des Aktualisierungstyps auf "Manuell" oder "Sofort"“ auf Seite 59
- „Initialisieren einer materialisierten Ansicht“ auf Seite 66
- „sa_materialized_view_can_be_immediate-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- Einschränkungen beim Ändern des Typs einer materialisierten Ansicht von "Manuell" auf "Sofort" auf Seite 62
- „ALTER MATERIALIZED VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „TRUNCATE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Indizes erstellen“ auf Seite 33

Fortgeschrittene Aufgaben: Status und Eigenschaften von materialisierten Ansichten

Materialisierte Ansichten sind durch eine Kombination aus Status und Eigenschaften gekennzeichnet. Der Status einer materialisierten Ansicht zeigt die Verfügbarkeit der Ansicht für die Verwendung durch den Datenbankserver an. Die Eigenschaften einer materialisierten Ansicht zeigen den Status der Daten innerhalb der Ansicht an.

Die beste Möglichkeit, um den Status und die Eigenschaften von bestehenden materialisierten Ansichten zu ermitteln, besteht darin, die Systemprozedur `sa_materialized_view_info` zu verwenden.

Sie können auch Informationen über materialisierte Ansichten einsehen, indem Sie den Ordner **Ansichten** in Sybase Central auswählen und die Details zu den einzelnen Ansichten prüfen oder indem Sie die Systemansichten SYSTAB und SYSVIEW abfragen.

Siehe auch

- „sa_materialized_view_info-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSTAB-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSVIEW-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Status von materialisierten Ansichten

Es gibt zwei mögliche Status bei materialisierten Ansichten:

- **Gültig** Die materialisierte Ansicht wurde erfolgreich kompiliert und ist für die Verwendung durch den Datenbankserver verfügbar. Eine aktivierte materialisierte Ansicht muss nicht unbedingt Daten enthalten. Wenn Sie beispielsweise die Daten von einer aktivierten materialisierten Ansicht kürzen, ändert sich der Status auf "Gültig" und "Nicht initialisiert". Eine materialisierte Ansicht kann initialisiert aber leer sein, wenn es keine Daten in den Basistabellen gibt, die der Definition für die materialisierte Ansicht entsprechen. Das ist nicht das Gleiche wie eine materialisierte Ansicht, die keine Daten enthält, weil sie nicht initialisiert ist.
- **Deaktiviert** Die materialisierte Ansicht wurde ausdrücklich deaktiviert, z.B. durch Ausführen der Anweisung `ALTER MATERIALIZED VIEW ... DISABLE`. Wenn Sie eine materialisierte Ansicht

deaktivieren, werden die Daten und Indizes für die Ansicht gelöscht. Beachten Sie auch: Wenn Sie eine Sofortansicht deaktivieren, ändert sie sich in eine manuelle Ansicht.

Um zu ermitteln, ob eine Ansicht aktiviert oder deaktiviert ist, verwenden Sie die Systemprozedur `sa_materialized_view_info`, um die Status-Eigenschaft für die Ansicht zu erhalten.

Siehe auch

- „[ALTER MATERIALIZED VIEW-Anweisung](#)“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „[sa_materialized_view_info-Systemprozedur](#)“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „[Aktivieren oder Deaktivieren einer materialisierten Ansicht](#)“ auf Seite 68
- „[Eigenschaften von materialisierten Ansichten](#)“ auf Seite 77

Eigenschaften von materialisierten Ansichten

Eigenschaften von materialisierten Ansichten werden vom Optimierer zum Einschätzen verwendet, ob eine Ansicht verwendet werden soll. Die folgende Liste beschreibt die Eigenschaften einer materialisierten Ansicht, wie sie von der Systemprozedur `sa_materialized_view_info` zurückgegeben werden:

- **Status** Gibt an, ob die Ansicht aktiviert oder deaktiviert ist.
- **DataStatus** Zeigt den Zustand der Daten in der Ansicht an. Das sagt Ihnen beispielsweise, ob die Ansicht initialisiert oder veraltet ist. Manuelle Ansichten sind veraltet, wenn sich die Daten in den Basistabellen geändert haben, seitdem die materialisierte Ansicht zuletzt aktualisiert wurde. Sofortansichten sind nie veraltet.
- **ViewLastRefreshed** Zeigt den Zeitpunkt an, zu dem die Ansicht zuletzt aktualisiert wurde.
- **DateLastModified** Zeigt den Zeitpunkt, zu dem die Daten in einer Basistabelle zuletzt geändert wurden, falls die Ansicht veraltet ist.
- **AvailForOptimization** Zeigt an, ob die Ansicht zur Verwendung durch den Optimierer verfügbar ist.
- **RefreshType** Zeigt an, ob es sich um eine manuelle Ansicht oder um eine Sofortansicht handelt.

Eine Liste der möglichen Werte für die einzelnen Eigenschaften können Sie mithilfe der `sa_materialized_view_info`-Systemprozedur anzeigen.

Es gibt keine Eigenschaft, die Ihnen mitteilt, ob eine manuelle Ansicht in eine Sofortansicht geändert werden kann. Um dies zu ermitteln, verwenden Sie die Systemprozedur `sa_materialized_view_can_be_immediate`.

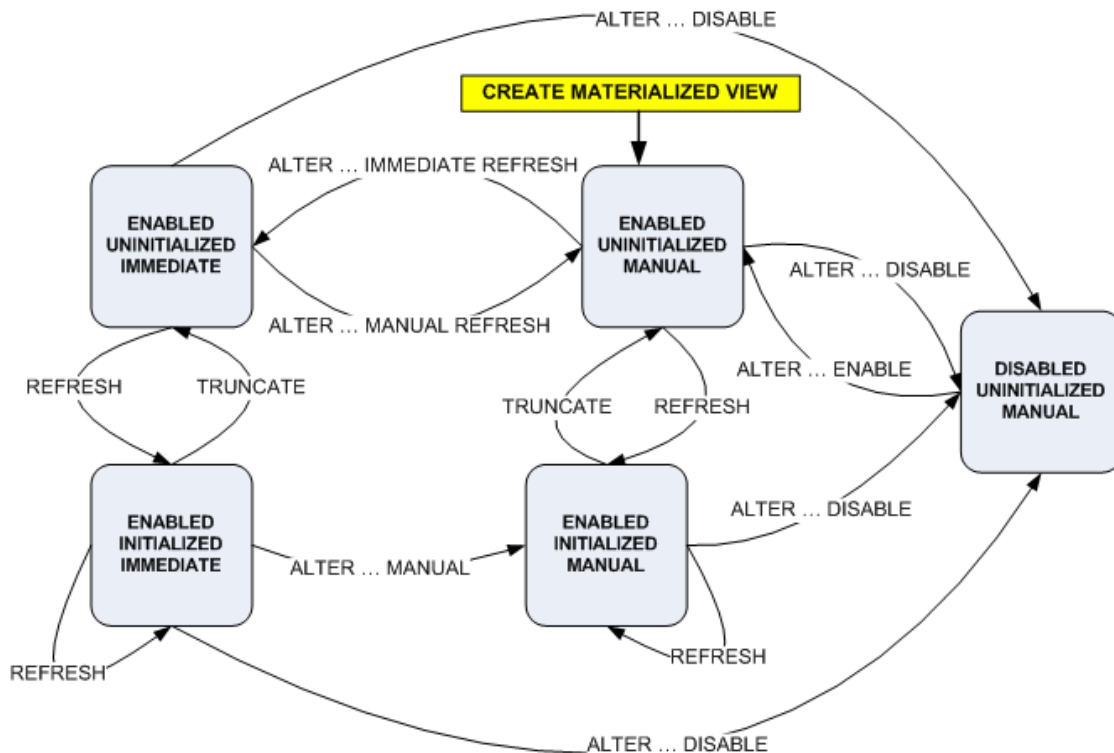
Siehe auch

- „[sa_materialized_view_info-Systemprozedur](#)“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „[sa_materialized_view_can_be_immediate-Systemprozedur](#)“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „[Status von materialisierten Ansichten](#)“ auf Seite 76

Status- und Eigenschaftsänderungen beim Ändern, Aktualisieren und Kürzen einer materialisierten Ansicht

Vorgänge, die Sie mit einer materialisierten Ansicht ausführen (wie Ändern, Aktualisieren und Kürzen), wirken sich auf den Status und die Eigenschaften der Ansicht aus. Das folgende Diagramm zeigt, wie sich diese Aufgaben auf den Status und auf einige Eigenschaften einer materialisierten Ansicht auswirken.

Im Diagramm stellt jedes graue Quadrat eine materialisierte Ansicht dar. Sofortansichten werden durch den Ausdruck "IMMEDIATE" und manuelle Ansichten durch den Ausdruck "MANUAL" gekennzeichnet. Der Ausdruck ALTER in den Verbindungslinien zwischen den grauen Quadraten ist die Abkürzung für ALTER MATERIALIZED VIEW. Auch wenn hier SQL-Anweisungen zum Ändern des Status von materialisierten Ansichten verwendet werden, so können Sie auch Sybase Central verwenden, um diese Vorgänge auszuführen.



- Wenn Sie eine materialisierte Ansicht erstellen, handelt es sich um eine aktivierte manuelle Ansicht, die nicht initialisiert ist (enthält keine Daten).
- Wenn Sie eine nicht initialisierte Ansicht aktualisieren, wird sie initialisiert (mit Daten aufgefüllt).
- Das Ändern einer manuellen Ansicht in eine Sofortansicht erfordert mehrere Schritte, und es gibt zusätzliche Einschränkungen bei Sofortansichten.

- Wenn Sie eine materialisierte Ansicht deaktivieren, geschieht Folgendes:
 - Die Daten werden gelöscht.
 - Die Ansicht kehrt in den nicht initialisierten Zustand zurück.
 - Die Indizes werden gelöscht.
 - Eine Sofortansicht ändert sich zu einer manuellen Ansicht.

Siehe auch

- „ALTER MATERIALIZED VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Einstellen des Aktualisierungstyps auf "Manuell" oder "Sofort"“ auf Seite 59
- „Fortgeschrittene Aufgaben: Aktualisierungstyp für eine materialisierte Ansicht ändern“ auf Seite 74
- Einschränkungen beim Ändern des Typs einer materialisierten Ansicht von "Manuell" auf "Sofort" auf Seite 62
- „Eigenschaften von materialisierten Ansichten“ auf Seite 77
- „Status von materialisierten Ansichten“ auf Seite 76

Fortgeschrittene Aufgaben: Einstellungen zum Steuern der Datenveraltung in materialisierten Ansichten

Daten in materialisierten Ansichten veralten, wenn Daten in den Tabellen geändert werden, die von der materialisierten Ansicht referenziert werden. Wenn Sie feststellen, dass die materialisierte Ansicht vom Optimierer nicht berücksichtigt wird, kann dies an der Veraltung liegen. Sie können den Veraltungsschwellenwert für materialisierte Ansichten mithilfe der Datenbankoption `materialized_view_optimization` anpassen.

Außerdem können Sie das für das Ereignis oder den Trigger zum Aktualisieren der Ansicht festgelegte Intervall anpassen.

Wenn eine Abfrage eine materialisierte Ansicht explizit referenziert, wird die Ansicht zur Verarbeitung der Abfrage herangezogen, unabhängig von der Aktualität der Daten in der Ansicht. Außerdem kann die `OPTION`-Klausel in Anweisungen wie `SELECT`, `UPDATE` oder `INSERT` verwendet werden, um die Einstellung der Datenbankoption `materialized_view_optimization` aufzuheben, wodurch die Verwendung einer materialisierten Ansicht erzwungen wird.

Wenn die Snapshot-Isolation verwendet wird, vermeidet der Optimierer es, eine materialisierte Ansicht zu verwenden, wenn sie nach dem Start des Snapshots für eine Transaktion aktualisiert wurde.

Siehe auch

- „Ermitteln, welche materialisierten Ansichten vom Optimierer in Betracht gezogen wurden“ auf Seite 238
- „Materialisierte Ansichten und Ansichtenübereinstimmung“ auf Seite 235
- „materialized_view_optimization-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen

Prozeduren und Trigger speichern prozedurale SQL-Anweisungen in einer Datenbank, damit sie von allen Anwendungen verwendet werden können. Sie können Steueranweisungen enthalten, die die Wiederholung (LOOP-Anweisung) und die bedingte Ausführung (IF-Anweisung und CASE-Anweisung) von SQL-Anweisungen ermöglichen. Batches sind Gruppen von SQL-Anweisungen, die dem Datenbankserver als Gruppe übergeben werden. Viele Merkmale, die bei Prozeduren und Triggern verfügbar sind, etwa Steueranweisungen, stehen auch für Batches zur Verfügung.

Versionsverwaltung

Verwenden Sie eine Software zur Versionsverwaltung, um Änderungen im Quellcode und Änderungen an Objekten zu verfolgen, die aus der Quelle (auch gespeicherten Prozeduren) erstellt werden, die Sie in der Datenbank bereitstellen.

Prozeduren werden mit einer CALL-Anweisung aufgerufen, verwenden Parameter zur Übernahme von Werten und geben Werte an die aufrufende Umgebung zurück. SELECT-Anweisungen können auch mit Prozedur-Ergebnismengen operieren, indem Sie den Prozedurnamen in die FROM-Klausel aufnehmen.

Prozeduren können auch Ergebnismengen an den Aufrufer zurückgeben, andere Prozeduren aufrufen oder Trigger auslösen. Eine benutzerdefinierte Funktion ist beispielsweise eine Art von gespeicherter Prozedur, die einen einzelnen Wert an die aufrufende Umgebung zurückgibt. Benutzerdefinierte Funktionen erweitern den Bereich der für Abfragen und andere SQL-Anweisungen verfügbaren Funktionen.

Trigger sind bestimmten Datenbanktabellen zugeordnet. Sie werden automatisch ausgelöst, wenn ein Benutzer Aktualisierungen einfügt oder Zeilen der zugehörigen Tabelle löscht. Trigger können Prozeduren aufrufen und andere Trigger auslösen. Sie haben aber keine Parameter und können nicht durch eine CALL-Anweisung aufgerufen werden.

SQL Anywhere-Debugger

Mithilfe des SQL Anywhere-Debuggers können Sie eine Fehlersuche in gespeicherten Prozeduren und Triggern durchführen.

Sie können ein Profil von gespeicherten Prozeduren erstellen, um Performance-Merkmale in Sybase Central zu analysieren.

Siehe auch

- „Prozedurprofilerstellung mit Systemprozeduren“ auf Seite 197
- „SQL Anywhere-Debugger“ auf Seite 971

Vorteile von Prozeduren, Triggern und benutzerdefinierten Funktionen

Prozeduren und Trigger verbessern die Sicherheit, den Wirkungsgrad und die Standardisierung der Datenbanken.

Definitionen für Prozeduren und Trigger erscheinen in der Datenbank jeweils getrennt von der Datenbankanwendung. Diese Trennung bietet mehrere Vorteile.

Standardisierung

Prozeduren und Trigger standardisieren Aktionen, die von mehr als einem Anwendungsprogramm vorgenommen werden. Indem die Aktion einmal in einen Programmcode geschrieben und in der Datenbank gespeichert wird, brauchen die Anwendungen die Prozeduren nur aufzurufen oder den Trigger auszulösen, um wiederholt das gewünschte Ergebnis zu erzielen. Da die Änderungen nur an einer Stelle vorgenommen werden, übernehmen alle Anwendungen, die diese Aktion verwenden, automatisch die neuen Funktionen, sobald die Implementierung der Aktion geändert wird.

Wirkungsgrad

Prozeduren und Trigger, die in einem Netzwerk-Datenbankserver verwendet werden, können auf die Daten in der Datenbank zugreifen, ohne dass Datenverkehr über das Netzwerk erforderlich ist. Das bedeutet, dass sie schneller und mit weniger Auswirkungen auf die Netzlast ausgeführt werden können, als wenn sie in einer Anwendung auf einem der Clientcomputer eingerichtet worden wären.

Prozeduren oder Trigger werden bei der Erstellung automatisch auf syntaktische Richtigkeit geprüft und in den Systemtabellen gespeichert. Wenn die Anwendung das erste Mal eine Prozedur aufruft oder einen Trigger auslöst, werden sie aus den Systemtabellen in den virtuellen Speicher übertragen und von dort ausgeführt. Da eine Kopie der Prozedur oder des Triggers nach dem ersten Ausführen im Arbeitsspeicher bleibt, können wiederholte Ausführungen derselben Prozedur oder desselben Triggers unverzüglich erfolgen. Außerdem können mehrere Anwendungen eine Prozedur oder einen Trigger gleichzeitig benutzen, und eine Anwendung kann sie wiederholt aufrufen.

Sicherheit

Siehe „Sicherheit: Prozeduren und Trigger“ [[SQL Anywhere Server - Datenbankadministration](#)]

Siehe auch

- „Sicherheit: Mit Ansichten und Prozeduren die Daten einschränken, auf die Benutzer Zugriff haben“ [[SQL Anywhere Server - Datenbankadministration](#)]

Prozeduren

Prozeduren und Funktionen zum Ausführen mit Eigentümer- oder Aufruferprivilegien einrichten

Beim Erstellen einer Prozedur oder Funktion können Sie angeben, ob die Prozedur oder Funktion mit den Privilegien ihres **Eigentümers** ausgeführt werden soll oder mit den Privilegien der Person oder der

Prozedur, die sie aufruft (**Aufrufer**). Die Identifizierung des Aufrufers ist nicht immer offensichtlich. Ein Benutzer kann zwar eine Prozedur aufrufen, aber diese Prozedur kann auch andere Prozeduren aufrufen. In diesen Fällen kann ein Unterschied bestehen zwischen dem *angemeldeten Benutzer* (dem Benutzer, der den ersten Aufruf der Prozedur der obersten Ebene durchgeführt hat) und dem *effektiven Benutzer*, der der Eigentümer einer Prozedur sein kann, die durch die erste Prozedur aufgerufen wird. Wenn eine Prozedur mit den Privilegien des Aufrufers ausgeführt wird, werden die Privilegien des effektiven Benutzers erzwungen.

Beim Erstellen einer Prozedur oder Funktion legt die SQL SECURITY-Klausel der CREATE PROCEDURE-Anweisung oder der CREATE FUNCTION-Anweisung fest, welche Privilegien gelten, wenn die Prozedur oder Funktion ausgeführt wird, und wer Eigentümer nicht qualifizierter Objekte ist. Die Wahl für diese Klausel ist INVOKER oder DEFINER. Ein Benutzer kann allerdings eine Prozedur oder Funktion erstellen, die einem anderen Benutzer gehört. In diesem Fall sind die Privilegien des Eigentümers gültig und nicht diejenigen des Definierers.

Wenn Sie Prozeduren oder Funktionen erstellen, sollten Sie daher mit Umsicht vorgehen und alle Objektnamen (Tabellen, Prozeduren, etc.) mit ihrem richtigen Eigentümer qualifizieren. Wenn die Objekte in der Prozedur keinen qualifizierten Eigentümer haben, hängt das Eigentum davon ab, ob die Prozedur unter dem Eigentümer oder dem Aufrufer läuft. Nehmen Sie beispielsweise an, Benutzer1 erstellt die folgende Prozedur:

```
CREATE PROCEDURE user1.myProcedure()
  RESULT( columnA INT )
  SQL SECURITY INVOKER
  BEGIN
    SELECT columnA FROM table1;
  END;
```

Wenn ein anderer Benutzer, user2, versucht, diese Prozedur auszuführen, und eine Tabelle user2.table1 nicht existiert, wird ein Fehler zurückgegeben. Wenn die Tabelle user2.table1 existiert, wird diese Tabelle verwendet, nicht user1.table1.

Wenn Prozeduren oder Funktionen mit den Privilegien des Aufrufers ausgeführt werden, benötigt der Aufrufer das EXECUTE-Privileg für die Prozedur sowie die erforderlichen Privilegien für die Datenbankobjekte, mit denen die Prozedur, Funktion oder Systemprozedur arbeitet.

Wenn Sie sich nicht sicher sind, ob Prozeduren oder Funktionen als Aufrufer oder Definierer ausgeführt werden, können Sie die SQL SECURITY-Klausel in ihren SQL-Definitionen zu Rate ziehen.

Mithilfe der sp_proc_priv-Systemprozedur können Sie bestimmen, welche Privilegien zum Ausführen einer Prozedur oder Funktion erforderlich sein sollen, die mit Privilegien verbundene Vorgänge in der Datenbank ausführt. Siehe „sp_proc_priv-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)].

Siehe auch

- „sp_proc_priv-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE DATABASE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Dienstprogramm Initialisierung (dbinit)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Dienstprogramm zum Upgrade (dbupgrad)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „ALTER DATABASE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE FUNCTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE PROCEDURE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Dienstprogramm zum Upgrade (dbupgrad)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Systemprozeduren vor Version 16.0 als Aufrufer oder Definierer ausführen

Einige Systemprozeduren in der Software vor Version 16.0, die mit Privilegien verbundene Aufgaben in der Datenbank ausführen, z.B. Ändern von Tabellen, können mit den Privilegien des Aufrufers oder des Definierers (Eigentümer) ausgeführt werden. Beim Erstellen oder Initialisieren einer Datenbank können Sie angeben, ob diese speziellen Systemprozeduren mit den Privilegien ihrer Eigentümer (Definierer) oder den Privilegien des Aufrufers ausgeführt werden sollen.

Wenn die Datenbank so konfiguriert ist, dass diese Systemprozeduren im Definierermodus ausgeführt werden, benötigt der Aufrufer keine zusätzlichen Privilegien, weil die Prozedur mit den Privilegien des Definierers (meist die dbo- oder SYS-Rolle) ausgeführt wird, der bereits die erforderlichen Privilegien hat.

Wenn die Datenbank so konfiguriert ist, dass sie im Aufrufermodus ausgeführt wird, muss der Aufrufer die Privilegien haben, die in der Dokumentation für die Prozedur verlangt werden. Der Aufrufer benötigt außerdem das EXECUTE-Privileg für die Prozedur, das er jedoch als Mitglied von PUBLIC erbt.

Hinweis

Das Standardverhalten für *benutzerdefinierte* Prozeduren wird durch den Aufrufer-/Definierermodus nicht beeinflusst. Das bedeutet: Wenn die Definition der benutzerdefinierten Prozedur keinen Aufrufer oder Definierer angibt, wird die Prozedur mit den Privilegien des Definierers ausgeführt.

Wie diese Systemprozeduren bei der Datenbankerstellung oder beim Upgrade ausgeführt werden, steuern Sie mit einer der folgenden Methoden:

- **CREATE DATABASE...SYSTEM PROCEDURE AS DEFINER-Anweisung** Wenn Sie CREATE DATABASE...SYSTEM PROCEDURE AS DEFINER OFF angeben, erzwingt der Datenbankserver die Privilegien des Aufrufers. Dies ist die Standardeinstellung für neue Datenbanken.

Wenn Sie CREATE DATABASE...SYSTEM PROCEDURE AS DEFINER ON angeben, erzwingt der Datenbankserver die Privilegien des Definierers (Eigentümers). Dies war das Standardverhalten bei Datenbanken vor Version 16.0.

- **ALTER DATABASE UPGRADE...SYSTEM PROCEDURE AS DEFINER-Anweisung** Diese Klausel verhält sich so wie bei der CREATE DATABASE-Anweisung. Wenn die Klausel nicht

angegeben ist, wird das bestehende Verhalten der Datenbank, die einem Upgrade unterzogen wird, beibehalten. Wenn Sie beispielsweise ein Upgrade einer Datenbank vor Version 16.0 durchführen, werden die Vorgänge standardmäßig mit den Privilegien des Definierers ausgeführt.

- **-pd-Option, Initialisierungsdienstprogramm (dbinit)** Wenn Sie beim Erstellen der Datenbank die Option -pd angeben, erzwingt der Datenbankserver beim Ausführen dieser Systemprozeduren die Privilegien des Definierers. Wenn Sie die Option -pd nicht angeben, ist das Standardverhalten, dass die Privilegien des Aufrufers erzwungen werden.
- **-pd-Option, Dienstprogramm zum Upgrade (dbupgrad)** Wenn Sie beim Upgrade der Datenbank die Option -pd Y angeben, erzwingt der Datenbankserver beim Ausführen dieser Systemprozeduren die Privilegien des Definierers.

Wenn Sie die Option -pd N angeben, erzwingt der Datenbankserver beim Ausführen dieser Systemprozeduren die Privilegien des Aufrufers.

Wenn diese Option nicht angegeben ist, wird das bestehende Verhalten der Datenbank, die einem Upgrade unterzogen wird, beibehalten.

Hinweis

Der PUBLIC-Systemrolle wird das EXECUTE-Privileg für alle Systemprozeduren erteilt. Neu erstellten Benutzern wird standardmäßig die PUBLIC-Rolle erteilt, sodass die Benutzer bereits das EXECUTE-Privileg für Systemprozeduren haben.

Der Standardwert für benutzerdefinierte Funktionen und Prozeduren bleibt durch die Entscheidung über Aufrufer/Definierer unbeeinflusst. Daher gilt: Auch wenn Sie diese Systemprozeduren als Aufrufer ausführen, bleibt die Standardeinstellung für benutzerdefinierte Prozeduren der Definierer.

Liste der Prozeduren, die durch die Aufrufer/Definierer-Einstellung beeinflusst werden

Im Folgenden finden Sie eine Liste der Systemprozeduren, die durch die Aufrufer/Definierer-Einstellung beeinflusst werden. Dies sind die Systemprozeduren in Versionen vor SQL Anywhere 16.0, die in der Datenbank mit Privilegien verbundene Vorgänge ausgeführt haben. Wenn die Datenbank so konfiguriert wurde, dass diese als Definierer ausgeführt werden, benötigt der Benutzer nur das EXECUTE-Privileg für jede Prozedur, die er ausführen muss. Wenn die Datenbank für die Ausführung mit INVOKER

konfiguriert ist, benötigt der Benutzer nicht das EXECUTE-Privileg für jede Prozedur, sondern die einzelnen Privilegien, die für jede Prozedur erforderlich sind, damit sie erfolgreich ausgeführt wird.

- sa_audit_string
- sa_clean_database
- sa_column_stats
- sa_conn_activity
- sa_conn_compression_info
- sa_conn_info
- sa_conn_list
- sa_conn_options
- sa_conn_properties
- sa_db_info
- sa_db_list
- sa_db_properties
- sa_disable_auditing_type
- sa_disk_free_space
- sa_enable_auditing_type
- sa_external_library_unload
- sa_flush_cache
- sa_flush_statistics
- sa_get_histogram
- sa_get_request_profile
- sa_get_request_times
- sa_get_table_definition
- sa_index_density
- sa_index_levels
- sa_install_feature
- sa_java_loaded_classes
- sa_load_cost_model
- sa_make_object
- sa_materialized_view_can_be_immediate
- sa_procedure_profile
- sa_procedure_profile_summary
- sa_recompile_views
- sa_refresh_materialized_views
- sa_refresh_text_indexes
- sa_remove_tracing_data
- sa_reset_identity
- sa_save_trace_data
- sa_send_udp
- sa_server_option
- sa_set_tracing_level
- sa_table_fragmentation
- sa_table_page_usage
- sa_table_stats
- sa_text_index_vocab_nchar
- sa_unload_cost_model

- sa_user_defined_counter_add
- sa_user_defined_counter_set
- sa_validate
- sa_verify_password
- sp_copy_directory
- sp_copy_file
- sp_create_directory
- sp_delete_directory
- sp_delete_file
- sp_forward_to_remote_server
- sp_get_last_synchronize_result
- sp_list_directory
- sp_move_directory
- sp_move_file
- sp_remote_columns
- sp_remote_exported_keys
- sp_remote_imported_keys
- sp_remote_primary_keys
- sp_remote_procedures
- sp_remote_tables
- st_geometry_predefined_srs
- st_geometry_predefined_uom
- xp_cmdshell
- xp_getenv
- xp_read_file
- xp_sendmail
- xp_startmail
- xp_startsmtp
- xp_stopmail
- xp_stopsmt
- xp_write_file

Liste der Prozeduren, die unabhängig von der Aufrufer/Definierer-Einstellung mit den Privilegien des Aufrufers ausgeführt werden

Eine kleine Teilmenge der Systemprozeduren vor Version 16.0, die mit Privilegien verbundene Vorgänge ausführen, erfordern, dass der Aufrufer zusätzliche Privilegien hat, um die Aufgaben auszuführen,

unabhängig von der Aufrufer/Definierer-Einstellung. Die Liste zusätzlicher erforderlicher Privilegien für diese Prozeduren finden Sie in der Dokumentation der einzelnen Prozeduren:

- sa_locks
- sa_report_deadlocks
- sa_snapshots
- sa_transactions
- sa_performance_statistics
- sa_performance_diagnostics
- sa_describe_shapefile
- sa_text_index_stats
- sa_get_user

Siehe auch

- „sp_proc_priv-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE DATABASE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Dienstprogramm Initialisierung (dbinit)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Dienstprogramm zum Upgrade (dbupgrad)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „ALTER DATABASE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE FUNCTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE PROCEDURE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Dienstprogramm zum Upgrade (dbupgrad)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Festlegen des von einer Datenbank benutzten Sicherheitsmodells (SQL)

Rufen Sie die Einstellung für das Sicherheitsmodell (Aufrufer gegen Definierer) auf, die bei der Datenbankerstellung oder beim Upgrade festgelegt wurde, indem Sie die Database-Eigenschaft Capabilities abfragen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Standardmäßig werden in einer neuen Datenbank mit Privilegien verbundene Systemprozeduren nur mit dem INVOKER-Modell ausgeführt. Das bedeutet, dass Systemprozeduren vor Version 16.0, die mit Privilegien verbundene Vorgänge ausführen, mit den Privilegien des Benutzers ausgeführt werden, der die Prozedur aufruft. Diese Einstellung kann bei der Datenbankerstellung und beim Upgrade geändert werden. Sie können die festgelegte Einstellung des Sicherheitsmodells (Aufrufer gegen Definierer) mit dieser Methode ermitteln.

Aufgabe

- Melden Sie sich in Interactive SQL bei der Datenbank an und führen Sie folgende SQL-Anweisung aus:

```
SELECT IF ((HEXTOINT(SUBSTRING(DB_PROPERTY('Capabilities'),  
1,LENGTH(DB_PROPERTY('Capabilities'))-20)) & 8) = 8)
```

```

THEN 1
ELSE 0
END IF

```

Ergebnisse

1 bedeutet, dass Systemprozeduren vor Version 16.0, die mit Privilegien verbundene Vorgänge ausführen, mit den Privilegien des Aufrufermodells ausgeführt werden. 0 bedeutet, dass die Prozeduren mit den Privilegien des Definierers (Eigentümers) ausgeführt werden.

Nächste Schritte

Keine.

Siehe auch

- [Liste der Prozeduren, die durch die Aufrufer/Definierer-Einstellung beeinflusst werden auf Seite 85](#)
- [„Systemprozeduren vor Version 16.0 als Aufrufer oder Definierer ausführen“ auf Seite 84](#)

Prozeduren erstellen (Sybase Central)

In Sybase Central bietet der **Assistent zum Erstellen von Prozeduren** die Option, Prozedurvorlagen zu verwenden.

Voraussetzungen

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.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16-Plug-Ins** her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Prozeduren und Funktionen**.
3. Klicken Sie auf **Datei » Neu » Prozedur**.
4. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen von Prozeduren**.
5. Klicken Sie im rechten Fensterausschnitt auf die Registerkarte **SQL** und schreiben Sie den Prozedurcode fertig.

Ergebnisse

Die neue Prozedur erscheint in **Prozeduren und Funktionen**. Sie können diese Prozedur in Ihrer Anwendung nicht verwenden.

Siehe auch

- „CREATE PROCEDURE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER PROCEDURE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Zusammengesetzte Anweisungen“ auf Seite 119
- „Erstellen von entfernten Prozeduren (Sybase Central)“ auf Seite 815
- „Fremdserver“ auf Seite 787
- „CALL-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Benannte Parameter“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Prozeduren ändern (Sybase Central)

Sie können eine bestehende Prozedur ändern.

Voraussetzungen

Sie müssen Eigentümer der Prozedur sein oder eines der folgenden Privilegien haben:

- ALTER ANY PROCEDURE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Kontext und Bemerkungen

In Sybase Central können Sie eine bestehende Prozedur nicht direkt umbenennen. Sie müssen eine neue Prozedur mit demselben Namen erstellen, den Programmcode aus der alten Prozedur hinein kopieren und dann die alte Prozedur löschen.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Prozeduren und Funktionen**.
3. Wählen Sie die Prozedur aus.
4. Verwenden Sie eine der folgenden Methoden, um die Prozedur zu bearbeiten:
 - Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **SQL**.
 - Rechtsklicken Sie auf die Prozedur und klicken Sie auf **In neuem Fenster bearbeiten**.

Tipp

Sie können für jede Prozedur ein neues Fenster öffnen und den Code von einer Prozedur in die andere kopieren.

- Um einen Prozedurkommentar hinzuzufügen oder zu bearbeiten, rechtsklicken Sie auf die Prozedur und klicken Sie auf **Eigenschaften**.

Wenn Sie den **Assistenten zum Erstellen der Datenbankdokumentation** verwenden, um Ihre SQL Anywhere-Datenbank zu dokumentieren, haben Sie die Möglichkeit, diese Kommentare in die Ausgabe einzubeziehen.

Ergebnisse

Der Code der Prozedur wird geändert.

Siehe auch

- „Datenbank dokumentieren“ [*SQL Anywhere Server - Datenbankadministration*]
- „ALTER PROCEDURE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE PROCEDURE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Prozeduren erstellen (Sybase Central)“ auf Seite 89
- „Übersetzen einer gespeicherten Prozedur“ auf Seite 671

Aufrufen einer Prozedur (SQL)

CALL-Anweisungen rufen Prozeduren auf. Prozeduren können durch ein Anwendungsprogramm oder durch andere Prozeduren und Trigger aufgerufen werden.

Voraussetzungen

Sie müssen Eigentümer der Prozedur sein, das EXECUTE-Privileg für die Prozedur haben oder das EXECUTE ANY PROCEDURE-Systemprivileg haben.

Alle Benutzer, denen das EXECUTE-Privileg für die Prozedur erteilt wurde, können die Prozedur aufrufen, auch wenn sie keine Privilegien für die Tabelle haben.

Aufgabe

- Führen Sie die folgende Anweisung aus, eine Prozedur aufzurufen und Werte einzufügen:

```
CALL procedure-name( values );
```

Nach diesem Aufruf kann es sinnvoll sein, sicherzustellen, dass die Werte hinzugefügt wurden.

Hinweis

Sie können eine Prozedur aufrufen, die eine Ergebnismenge zurückgibt, indem Sie eine Abfrage aufrufen. Sie können Abfragen auf Ergebnismengen von Prozeduren ausführen und WHERE-Klauseln oder andere SELECT-Funktionen anwenden, um die Ergebnismenge zu beschränken.

Ergebnisse

Die Prozedur wird aufgerufen und ausgeführt.

Beispiel

Die folgende Anweisung ruft die NewDepartment-Prozedur auf, um die Abteilung "Eastern Sales" einzufügen:

```
CALL NewDepartment( 210, 'Eastern Sales', 902 );
```

Nach diesem Aufruf können Sie in der Tabelle "Departments" prüfen, ob die neue Abteilung hinzugefügt wurde.

Alle Benutzer, denen das EXECUTE-Privileg für die Prozedur erteilt wurde, können die NewDepartment-Prozedur aufrufen, auch wenn sie keine Privilegien für die Tabelle "Departments" haben.

Siehe auch

- „Benannte Parameter“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Benutzersicherheit (Rollen und Privilegien)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „CALL-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „GRANT EXECUTE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Prozeduren kopieren (Sybase Central)

Sie können Prozeduren mithilfe von Sybase Central zwischen Datenbanken oder innerhalb einer Datenbank kopieren. Wenn Sie eine Prozedur in derselben Datenbank kopieren, müssen Sie die Prozedur umbenennen oder einen anderen Eigentümer für die kopierte Prozedur wählen.

Voraussetzungen

Wenn Sie eine Prozedur kopieren und sich selbst als Eigentümer zuordnen möchten, benötigen Sie das CREATE PROCEDURE-Systemprivileg in der Datenbank, in die Sie die Prozedur kopieren. Wenn Sie eine Prozedur kopieren und einen anderen Benutzer als Eigentümer zuordnen möchten, benötigen Sie das CREATE ANY PROCEDURE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg in der Datenbank, in die Sie die Prozedur kopieren.

Aufgabe

1. Stellen Sie in Sybase Central mithilfe des **SQL Anywhere 16**-Plug-Ins eine Verbindung zur Datenbank her, die die Prozedur enthält, die Sie kopieren möchten.
2. Verbinden Sie sich mit der Datenbank, in die Sie die Prozedur kopieren möchten.
3. Wählen Sie die zu kopierende Prozedur im linken Fensterausschnitt der ersten Datenbank aus und ziehen Sie sie in den Ordner **Prozeduren und Funktionen** der zweiten Datenbank.

Ergebnisse

Eine neue Prozedur wird erstellt, und der Programmcode der ursprünglichen Prozedur wird in sie kopiert. Nur der Prozedurcode wird in die neue Prozedur kopiert. Andere Prozedureigenschaften, z.B. Privilegien, werden nicht kopiert.

Prozeduren löschen (Sybase Central)

Sie können Sybase Central verwenden, um eine Prozedur aus Ihrer Datenbank zu löschen, z.B. wenn sie nicht mehr benötigt wird.

Voraussetzungen

Sie müssen Eigentümer der Prozedur sein oder eines der folgenden Systemprivilegien haben:

- DROP ANY PROCEDURE
- DROP ANY OBJECT

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Prozeduren und Funktionen**.
3. Rechtsklicken Sie auf die Prozedur und klicken Sie auf **Löschen**.
4. Klicken Sie auf **Ja**.

Ergebnisse

Die Prozedur wird aus der Datenbank entfernt.

Nächste Schritte

Die Definition der abhängigen Datenbankobjekte wurde geändert, um die Referenz zur gelöschten Prozedur zu entfernen.

Siehe auch

- „DROP PROCEDURE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Benutzerdefinierte Funktionen

Benutzerdefinierte Funktionen sind eine Klasse von Prozeduren, die einen einzelnen Wert an die aufrufende Umgebung zurückgeben.

Hinweis

SQL Anywhere trifft keine Annahmen, was die Thread-Sicherheit von benutzerdefinierten Funktionen betrifft. Dies liegt in der Verantwortung des Anwendungsentwicklers.

Die Syntax von CREATE FUNCTION unterscheidet sich leicht von der Syntax der CREATE PROCEDURE-Anweisung.

- IN-, OUT- oder INOUT-Schlüsselwörter sind nicht erforderlich, da alle Parameter IN-Parameter sind.

- Die Klausel RETURNS ist erforderlich, um den Datentyp für die Rückgabe festzulegen.
- Die RETURN-Anweisung ist erforderlich, um den zurückgegebenen Wert festzulegen.
- Benannte Parameter werden nicht unterstützt.

Benutzerdefinierte Funktionen erstellen

Sie können benutzerdefinierte Funktionen in Sybase Central erstellen. Benutzerdefinierte Funktionen sind eine Klasse von Prozeduren, die einen einzelnen Wert an die aufrufende Umgebung zurückgeben.

Voraussetzungen

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.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf **Prozeduren und Funktionen** und klicken Sie auf **Neu » Funktion**.
3. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen von Funktionen**.
4. Klicken Sie im rechten Fensterausschnitt auf die Registerkarte **SQL** und schreiben Sie den Funktionscode fertig.

Ergebnisse

Die neue Funktion erscheint in **Prozeduren und Funktionen**.

Siehe auch

- „CREATE FUNCTION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE FUNCTION-Anweisung [Webdienst]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE FUNCTION-Anweisung [externer Aufruf]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Aufrufen einer benutzerdefinierten Funktion.

Eine benutzerdefinierte Funktion kann überall eingesetzt werden, wo eine integrierte Nicht-Aggregatfunktion verwendet werden kann.

Voraussetzungen

Sie müssen das EXECUTE-Privileg für die Funktion haben.

Aufgabe

1. Stellen Sie in Interactive SQL eine Verbindung mit der Datenbank her.
2. Führen Sie mithilfe der benutzerdefinierten Funktion eine SELECT-Anweisung aus.

Ergebnisse

Die Funktion wird aufgerufen und ausgeführt.

Beispiel

Beispiele 1: Aufrufen einer benutzerdefinierten Funktion 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;
```

Führen Sie die folgende Anweisung in Interactive SQL aus, um aus zwei Spalten mit einem Vor- und Nachnamen einen vollständigen Namen zurückzugeben:

```
SELECT FullName( GivenName, Surname )
AS "Full Name"
FROM Employees;
```

Full Name
Fran Whitney
Matthew Cobb
Philip Chin
...

Führen Sie die folgende Anweisung in Interactive SQL aus, damit die benutzerdefinierte Funktion FullName aus einem übergebenen Vor- und Nachnamen einen vollständigen Namen zurückgibt:

```
SELECT FullName('Jane', 'Smith')
AS "Full Name";
```

Full Name
Jane Smith

Beispiel 2: Lokale Deklarationen von Variablen Die folgende benutzerdefinierte Funktion zeigt lokale Deklarationen von Variablen.

Hinweis

Die Funktion ist als Veranschaulichung gut geeignet, kann aber eine schlechte Performance aufweisen, wenn sie in einer SELECT-Anweisung mit zahlreichen Zeilen eingesetzt wird. Wenn Sie die Funktion beispielsweise in der SELECT-Liste einer Abfrage für eine Tabelle verwenden, die 100000 Zeilen enthält, von denen 10000 zurückgegeben werden, wird die Funktion 10000 Mal aufgerufen. Wenn Sie sie in der WHERE-Klausel derselben Abfrage verwenden, wird sie 100000 Mal aufgerufen.

Die Customers-Tabelle umfasst Kunden aus Kanada und den USA. Die benutzerdefinierte Funktion "Nationality" bildet einen dreistelligen Ländercode, der auf der Spalte "Country" basiert.

```
CREATE FUNCTION Nationality( CustomerID INT )
RETURNS CHAR( 3 )
BEGIN
    DECLARE nation_string CHAR(3);
    DECLARE nation country_t;
    SELECT DISTINCT Country INTO nation
    FROM Customers
    WHERE ID = CustomerID;
    IF nation = 'Canada' THEN
        SET nation_string = 'CDN';
    ELSE IF nation = 'USA' OR nation = ' ' THEN
        SET nation_string = 'USA';
    ELSE
        SET nation_string = 'OTH';
    END IF;
    END IF;
    RETURN ( nation_string );
END;
```

Dieses Beispiel deklariert eine Variable namens "nation_string", in der die Zeichenfolge für die Staatszugehörigkeit enthalten sein soll, benutzt eine SET-Anweisung, um einen Wert für die Variable zu setzen, und gibt den Wert von "nation_string" an die aufrufende Anweisung zurück.

Die folgende Abfrage listet alle kanadischen Kunden in der Tabelle "Customers" auf:

```
SELECT *
FROM Customers
WHERE Nationality( ID ) = 'CDN';
```

Siehe auch

- „Benutzerdefinierte Funktionen“ auf Seite 93
- „CREATE FUNCTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Löschen einer benutzerdefinierten Funktion (SQL)

Benutzerdefinierte Funktionen bleiben in der Datenbank, bis sie explizit entfernt werden.

Voraussetzungen

Sie müssen Eigentümer der benutzerdefinierten Funktion sein oder eines der folgenden Systemprivilegien haben:

- DROP ANY PROCEDURE
- DROP ANY OBJECT

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Führen Sie eine DROP FUNCTION-Anweisung ähnlich der folgenden aus:

```
DROP FUNCTION function-name;
```

Ergebnisse

Die benutzerdefinierte Funktion wird gelöscht.

Beispiel

Die folgende Anweisung entfernt die Funktion "FullName" aus der Datenbank:

```
DROP FUNCTION FullName;
```

Siehe auch

- „DROP FUNCTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Berechtigung zum Ausführen einer benutzerdefinierten Funktion erteilen (SQL)

Geben Sie Benutzern die Möglichkeit, eine benutzerdefinierte Funktion auszuführen, indem Sie das EXECUTE-Privileg auf Objektebene erteilen.

Voraussetzungen

Sie müssen entweder Eigentümer der benutzerdefinierten Funktion sein oder das EXECUTE-Privileg mit Administrationsrechten für die Funktion haben.

Eigentümer einer benutzerdefinierten Funktion ist der Benutzer, der sie erstellt hat, und dieser Benutzer benötigt keine Privilegien, um sie auszuführen.

Kontext und Bemerkungen

Sie haben eine Funktion erstellt und Sie möchten, dass andere Benutzer sie nutzen können.

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Führen Sie eine GRANT EXECUTE-Anweisung ähnlich der folgenden aus:

```
GRANT EXECUTE ON function-name TO user-id;
```

Ergebnisse

Der Privilegienempfänger kann nun die Prozedur ausführen.

Nächste Schritte

Keine.

Beispiel

Der Ersteller der Nationality-Funktion kann beispielsweise einem anderen Benutzer die Erlaubnis erteilen, die Nationality-Funktion zu verwenden, indem er folgende Anweisung eingibt:

```
GRANT EXECUTE ON Nationality TO BobS;
```

Siehe auch

- „GRANT EXECUTE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Erweiterte Informationen zu benutzerdefinierten Funktionen

SQL Anywhere behandelt alle benutzerdefinierten Funktionen als **idempotent**, außer sie werden als NOT DETERMINISTIC deklariert. Idempotente Funktionen geben für die gleichen Parameter ein konsistentes Ergebnis zurück und weisen keine Nebenwirkungen auf. Zwei aufeinanderfolgende Aufrufe einer idempotenten Funktion mit denselben Parametern geben dasselbe Ergebnis zurück und haben keine unerwünschten Nebenwirkungen auf die Semantik der Abfrage.

Weitere Hinweise zu nicht-deterministischen und deterministischen Funktionen finden Sie unter [Caching von Funktionen auf Seite 339](#).

Trigger

Ein Trigger ist eine spezielle Form der gespeicherten Prozedur, die automatisch ausgeführt wird, sobald eine datenverändernde Anweisung ausgeführt wird. Trigger werden verwendet, wenn die referenzielle Integrität und andere deklarative Integritätsregeln nicht ausreichen.

Es kann sinnvoll sein, eine komplexere Form der referenziellen Integrität zu erzwingen, bei der entweder detailliertere Prüfungen durchgeführt werden oder neue Daten geprüft werden, während gleichzeitig Altdaten die Möglichkeit erhalten, Integritätsregeln zu verletzen. Eine andere Form des Einsatzes von

Triggern ist das Protokollieren der Aktivität in Datenbanktabellen, unabhängig von den Anwendungen, die die Datenbank benutzen.

Hinweis

Es gibt drei spezielle Anweisungen, nach denen Trigger nicht ausgelöst werden: LOAD TABLE, TRUNCATE und WRITETEXT.

Privilegien zum Ausführen von Triggern

Trigger werden mit den Privilegien des Eigentümers der zugeordneten Tabelle oder Ansicht ausgeführt, nicht mit den Berechtigungen des Benutzers, dessen Aktionen das Auslösen des Triggers bewirkt haben. Ein Trigger kann Zeilen in einer Tabelle modifizieren, die ein Benutzer nicht direkt modifizieren konnte.

Sie können verhindern, dass Trigger ausgelöst werden, indem Sie die Serveroption -gf angeben oder die Option fire_triggers setzen. Siehe „[Datenbankserveroption -gf](#)“ [*SQL Anywhere Server - Datenbankadministration*] oder „[fire_triggers-Option](#)“ [*SQL Anywhere Server - Datenbankadministration*].

- „[Datenbankserveroption -gf](#)“ [*SQL Anywhere Server - Datenbankadministration*]
- „[fire_triggers-Option](#)“ [*SQL Anywhere Server - Datenbankadministration*]

Triggertypen

SQL Anywhere unterstützt folgende Triggertypen:

- **BEFORE-Trigger** Ein BEFORE-Trigger wird ausgelöst, bevor eine auslösende Aktion ausgeführt wird. BEFORE-Trigger können für Tabellen definiert werden, nicht aber für Ansichten.
- **AFTER-Trigger** Ein AFTER-Trigger wird ausgelöst, nachdem eine auslösende Aktion ausgeführt wurde. AFTER-Trigger können für Tabellen definiert werden, nicht aber für Ansichten.
- **INSTEAD OF-Trigger** Ein INSTEAD OF-Trigger ist ein bedingter Trigger, der anstelle eines auslösenden Vorgangs ausgelöst wird. INSTEAD OF-Trigger können für Tabellen und Ansichten definiert werden (außer für materialisierte Ansichten).

Trigger-Ereignisse

Trigger können für eine oder mehrere auslösende Ereignisse definiert werden:

Maßnahme	Beschreibung
INSERT	Der Trigger wird aufgerufen, wenn eine neue Zeile in die Tabelle eingefügt wird, die dem Trigger zugeordnet ist.
DELETE	Der Trigger wird aufgerufen, wenn eine Zeile der zugeordneten Tabelle gelöscht wird.
UPDATE	Der Trigger wird aufgerufen, wenn eine Zeile der zugeordneten Tabelle aktualisiert wird.

Maßnahme	Beschreibung
UPDATE OF <i>column-list</i> -Klausel	Der Trigger wird aufgerufen, wenn eine Zeile der zugeordneten Tabelle so aktualisiert wird, dass eine Spalte in der <i>Spaltenliste</i> geändert wird.

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.

Siehe auch

- „Datenintegrität“ auf Seite 849
- „CREATE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „TRUNCATE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „WRITETEXT-Anweisung [T-SQL]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „LOAD TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „INSTEAD OF-Trigger“ auf Seite 110
- „CREATE TRIGGER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Atomare zusammengesetzte Anweisungen“ auf Seite 120

Trigger-Zeitpunkt

Trigger können entweder auf **Zeilenebene** oder auf **Anweisungsebene** aufgerufen werden:

- Ein Trigger auf Zeilenebene wird einmal für jede Zeile ausgelöst, die sich ändert. Trigger auf Zeilenebene werden ausgeführt, bevor (BEFORE) oder nachdem (AFTER) die Zeile geändert wurde. Spaltenwerte für das neue und alte Abbild der betroffenen Zeile werden dem Trigger über Variablen zur Verfügung gestellt.
- Ein Trigger auf Anweisungsebene wird ausgelöst, nachdem die komplette, Trigger-auslösende Anweisung abgeschlossen ist. Die Zeilen, die von der Trigger-Anweisung betroffen sind, werden dem Trigger über die temporären Tabellen zur Verfügung gestellt, welche die neuen und alten Abbilder der Zeilen darstellen. SQL Anywhere unterstützt keine BEFORE-Trigger auf der Anweisungsebene .

Die Flexibilität bei der Ausführungszeit des Triggers ist bei Triggern sinnvoll, die sich auf Aktionen der referenziellen Integrität verlassen, wie kaskadierendes Aktualisieren oder Löschen.

Wenn ein Fehler auftritt, während ein Trigger ausgeführt wird, schlägt der Vorgang fehl, der den Trigger ausgelöst hat. INSERT, UPDATE und DELETE sind atomare Vorgänge. Wenn sie fehlschlagen, werden alle Veränderungen der Anweisung (einschließlich der Wirkungen von Triggern und Prozeduren, die von Triggern aufgerufen wurden) zurückgesetzt, sodass wieder der Zustand vor dem Vorgang hergestellt wird.

Trigger für Tabellen erstellen (Sybase Central)

Erstellen Sie einen Trigger für eine Tabelle mit dem **Assistenten zum Erstellen von Triggern**.

Voraussetzungen

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

Aufgabe

1. Stellen Sie eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf **Trigger** und klicken Sie auf **Neu » Trigger**.
3. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen von Triggern**.
4. Um den Code zu vervollständigen, klicken Sie im rechten Fensterausschnitt auf die Registerkarte **SQL**.

Ergebnisse

Der neue Trigger wird erstellt.

Siehe auch

- „CREATE TRIGGER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Datenbankverbindungen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Zusammengesetzte Anweisungen“ auf Seite 119

Trigger für Tabellen erstellen (SQL)

Erstellen Sie einen Trigger für eine Tabelle mit der CREATE TRIGGER-Anweisung. Der Hauptteil eines Triggers besteht aus einer zusammengesetzten Anweisung: Einer Gruppe von durch Semikola getrennten SQL-Anweisungen, die zwischen eine BEGIN- und END-Anweisung gesetzt werden.

Voraussetzungen

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

Kontext und Bemerkungen

COMMIT und ROLLBACK sowie manche ROLLBACK TO SAVEPOINT-Anweisungen sind in einem Trigger nicht gestattet.

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Führen Sie eine CREATE TRIGGER-Anweisung aus.

Ergebnisse

Der neue Trigger wird erstellt.

Beispiel

Beispiel 1: Ein Insert-Trigger auf Zeilenebene Der folgende Trigger ist ein Beispiel für einen INSERT-Trigger auf Zeilenebene. Er prüft, ob das für einen neuen Mitarbeiter angegebene Geburtsdatum sinnvoll ist:

```
CREATE TRIGGER check_birth_date
  AFTER INSERT ON Employees
  REFERENCING NEW AS new_employee
  FOR EACH ROW
  BEGIN
    DECLARE err_user_error EXCEPTION
    FOR SQLSTATE '99999';
    IF new_employee.BirthDate > 'June 6, 2001' THEN
      SIGNAL err_user_error;
    END IF;
  END;
```

Hinweis

In der SQL Anywhere-Beispieldatenbank ist möglicherweise bereits ein Trigger namens "check_birth_date" vorhanden. Falls dies der Fall ist und Sie versuchen, die obenstehende SQL-Anweisung auszuführen, wird ein Fehler des Inhalts zurückgegeben, dass die Triggerdefinition in Konflikt zu bestehenden Triggern steht.

Dieser Trigger wird ausgelöst, nachdem eine Zeile in die Tabelle "Employees" eingefügt wurde. Er erkennt und verbietet neue Zeilen, die Geburtsdaten nach dem 06.06.01 enthalten.

Die Phrase REFERENCING NEW AS new_employee ermöglicht die Bezugnahme von Anweisungen im Triggercode auf Daten in der neuen Zeile mit dem Alias new_employee.

Wenn ein Fehler gemeldet wird, werden die Trigger-auslösende Anweisung und alle vorherigen Wirkungen des Triggers ungültig.

Für eine INSERT-Anweisung, die der Tabelle "Employees" viele Zeilen hinzufügt, wird der Trigger "check_birth_date" einmal für jede neue Zeile ausgelöst. Wenn der Trigger für eine der Zeilen fehlschlägt, werden alle Wirkungen der INSERT-Anweisung zurückgesetzt.

Sie können festlegen, dass der Trigger auslöst, bevor die Zeile eingefügt wird, und nicht danach, indem Sie die zweite Zeile des Beispiels wie folgt ändern:

```
BEFORE INSERT ON Employees
```

Die Klausel `REFERENCING NEW` bezieht sich auf die eingefügten Werte der Zeile und ist unabhängig von der Zeitfolge (`BEFORE` oder `AFTER`) des Triggers.

Manchmal ist es einfacher, Integritätsregeln mithilfe von deklarativer referenzieller Integrität oder `CHECK`-Integritätsregeln zu erzwingen, anstelle von Triggern. Die Umsetzung des oben genannten Beispiels mit einer Prüf-Integritätsregel auf Spalten bietet beispielsweise bessere Performance und ist klarer.

```
CHECK (@col <= 'June 6, 2001')
```

Beispiel 2: Ein Beispiel für einen DELETE-Trigger auf Zeilenebene Die folgende `CREATE TRIGGER` Anweisung definiert einen `DELETE`-Trigger auf Zeilenebene:

```
CREATE TRIGGER mytrigger
BEFORE DELETE ON Employees
REFERENCING OLD AS oldtable
FOR EACH ROW
BEGIN
    ...
END;
```

Die Klausel `REFERENCING OLD` ist unabhängig von der Zeitplanung des Triggers (`BEFORE` oder `AFTER`) und ermöglicht es dem `DELETE`-Triggercode, sich mit dem Alias "oldtable" auf die Werte in der zu löschenden Zeile zu beziehen.

Beispiel 3: Ein Beispiel für einen UPDATE-Trigger auf Anweisungsebene Die folgende `CREATE TRIGGER`-Anweisung ist für `UPDATE`-Trigger auf Anweisungsebene geeignet:

```
CREATE TRIGGER mytrigger AFTER UPDATE ON Employees
REFERENCING NEW AS table_after_update
                OLD AS table_before_update
FOR EACH STATEMENT
BEGIN
    ...
END;
```

Die Klausel `REFERENCING NEW` und `REFERENCING OLD` ermöglicht dem `UPDATE`-Triggercode die Bezugnahme auf die alten und neuen Werte der zu aktualisierenden Zeilen. Der Tabellenalias `table_after_update` bezieht sich auf Spalten in der neuen Zeile, der Tabellenalias `table_before_update` auf Spalten in der alten Zeile.

Die Klausel `REFERENCING NEW` und `REFERENCING OLD` hat eine leicht unterschiedliche Bedeutung für Trigger auf Anweisungsebene und Zeilenebene. Für Trigger auf Anweisungsebene sind die Aliase `REFERENCING OLD` oder `NEW` Tabellenalias, während sie sich bei Triggern auf Zeilenebene auf die geänderte Zeile beziehen.

Siehe auch

- „CREATE TRIGGER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Datenbankverbindungen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „COMMIT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ROLLBACK TO SAVEPOINT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Zusammengesetzte Anweisungen“ auf Seite 119

Trigger ausführen

Trigger werden automatisch ausgelöst, wenn INSERT, UPDATE oder DELETE in der vom Trigger benannten Tabelle ausgeführt wird. Ein Trigger auf Zeilenebene wird einmal für jede betroffene Zeile ausgelöst, während ein Trigger auf Anweisungsebene einmal für die komplette Anweisung ausgelöst wird.

Wenn INSERT, UPDATE oder DELETE einen Trigger auslösen, sieht die Reihenfolge der Vorgänge abhängig vom Triggertyp (BEFORE oder AFTER) wie folgt aus:

1. BEFORE-Trigger lösen aus.
2. Der Vorgang selbst wird ausgeführt.
3. Alle referenziellen Aktionen werden ausgeführt.
4. AFTER-Trigger lösen aus.

Hinweis

Wenn Sie mit der Anweisung CREATE TRIGGER einen Trigger erstellen und keinen Triggertyp angeben, wird standardmäßig AFTER benutzt.

Wenn bei einem der Schritte ein Fehler auftritt, der nicht innerhalb der Prozedur oder des Triggers abgewickelt werden kann, werden die vorhergehenden Schritte rückgängig gemacht, die nachfolgenden Schritte werden nicht ausgeführt, und der Vorgang, der den Trigger ausgelöst hat, schlägt fehl.

Siehe auch

- „CREATE TRIGGER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Trigger ändern

Ändern Sie einen Trigger mithilfe von Sybase Central, um den Code des Triggers zu bearbeiten oder einen Kommentar hinzuzufügen.

Voraussetzungen

Wenn Sie einen Kommentar hinzufügen oder bearbeiten möchten, müssen Sie eines der folgenden Systemprivilegien haben:

- COMMENT ANY OBJECT
- ALTER ANY TRIGGER
- ALTER ANY OBJECT
- CREATE ANY TRIGGER
- CREATE ANY OBJECT

Wenn Sie den Code bearbeiten möchten, benötigen Sie das ALTER ANY OBJECT-Systemprivileg oder das ALTER ANY TRIGGER-Systemprivileg sowie eines der folgenden Privilegien:

- Sie müssen Eigentümer der Basistabelle sein
- ALTER ANY TABLE-Systemprivileg
- ALTER-Privileg für die Basistabelle

Kontext und Bemerkungen

In Sybase Central können Sie einen bestehenden Trigger nicht direkt umbenennen. Sie müssen einen neuen Trigger mit demselben Namen erstellen, den Programmcode aus dem alten Trigger hinein kopieren und dann den alten Trigger löschen.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Trigger**.
3. Wählen Sie einen Trigger.
4. Verwenden Sie eine der folgenden Methoden, um den Trigger zu ändern:

Option	Aktion
Bearbeiten Sie den Code.	<p>Rechtsklicken Sie auf den Trigger und klicken Sie auf In neuem Fenster bearbeiten oder bearbeiten Sie den Code auf der Registerkarte SQL im rechten Fensterausschnitt.</p> <div> <p>Tipp</p> <p>Sie können für jede Prozedur ein neues Fenster öffnen und den Code von einem Trigger in den anderen kopieren.</p> </div>

Option	Aktion
Fügen Sie einen Kommentar hinzu.	<p>Um einen Triggerkommentar hinzuzufügen oder zu bearbeiten, rechtsklicken Sie auf den Trigger und klicken Sie auf Eigenschaften.</p> <p>Wenn Sie den Assistenten zum Erstellen der Datenbankdokumentation verwenden, um Ihre SQL Anywhere-Datenbank zu dokumentieren, haben Sie die Möglichkeit, diese Kommentare in die Ausgabe einzubeziehen.</p>

Ergebnisse

Der Code des Triggers wird geändert.

Siehe auch

- „Datenbankverbindungen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbank dokumentieren“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Übersetzen einer gespeicherten Prozedur“ auf Seite 671
- „ALTER TRIGGER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Trigger löschen

Verwenden Sie Sybase Central, um einen Trigger aus Ihrer Datenbank zu löschen, wenn er nicht mehr benötigt wird.

Voraussetzungen

Sie müssen Eigentümer des Triggers sein oder eines der folgenden Systemprivilegien haben:

- DROP ANY TRIGGER
- DROP ANY OBJECT

Aufgabe

1. Stellen Sie eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Trigger**.
3. Wählen Sie den Trigger aus und klicken Sie auf **Bearbeiten » Löschen**.
4. Klicken Sie auf **Ja**.

Ergebnisse

Der Trigger wird aus der Datenbank entfernt.

Nächste Schritte

Die Definition des abhängigen Datenbankobjekts muss geändert werden, um die Referenzen zum gelöschten Trigger zu entfernen.

Siehe auch

- „Datenbankverbindungen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „DROP TRIGGER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Beispiel: Trigger-Vorgänge vorübergehend deaktivieren

Sie können Trigger so einstellen, dass ihre Vorgänge deaktiviert werden, wenn Benutzer Aktionen für Spaltendaten ausführen (durch die der Trigger ausgelöst wird). Der Trigger kann weiterhin ausgelöst werden und seine Vorgänge können ausgeführt werden, und zwar mit einer Prozedur, die eine vordefinierte Verbindungsvariable enthält. Benutzer können anschließend INSERT-, ALTER- oder DELETE-Anweisungen auf Spalten anwenden, ohne dass die Trigger-Vorgänge ausgeführt werden, auch wenn der Trigger ausgelöst wird.

Hinweis

Wenn Sie mit einem Trigger auf Zeilenebene arbeiten, verwenden Sie eine WHEN-Klausel, um anzugeben, wann der Trigger ausgelöst werden soll.

Beispiel: Vorgänge eines einzelnen Triggers vorübergehend deaktivieren

In diesem Beispiel werden die Vorgänge eines Triggers in Abhängigkeit davon deaktiviert, ob eine Verbindungsvariable vorhanden ist.

1. Erstellen Sie einen AFTER INSERT-Trigger, der den Status einer Verbindungsvariablen prüft, um zu ermitteln, ob die Trigger-Logik aktiviert ist. Wenn die Variable nicht vorhanden ist, werden die Vorgänge des Triggers aktiviert:

```
CREATE TRIGGER myTrig AFTER INSERT
REFERENCING NEW AS new-name
FOR EACH STATEMENT
BEGIN
    DECLARE @execute_trigger integer;
    IF varexists('enable_trigger_logic') = 1 THEN
        SET @execute_trigger = enable_trigger_logic;
    ELSE
        SET @execute_trigger = 1;
    END IF;
    IF @execute_trigger = 1 THEN
        ... -your-trigger-logic
    END IF;
END;
```

2. Fügen Sie Ihrer Anweisung den folgenden Code hinzu, um den in Schritt 1 erstellten Trigger aufzurufen. Die Anweisung verwendet eine Verbindungsvariable, um zu steuern, wann der Trigger deaktiviert wird, und muss den Code umschließen, den Sie deaktivieren möchten.

```
...
IF varexists('enable_trigger_logic') = 0 THEN
    CREATE VARIABLE enable_trigger_logic INT;
END IF;
SET enable_trigger_logic = 0;
... execute-your-code-that-you-do-not-want-triggers-to-run
SET enable_trigger_logic = 1;
... now-your-trigger-logic-will-do-its-work
```

Beispiel: Vorgänge für mehrere Trigger vorübergehend deaktivieren

In diesem Beispiel wird die auf einer Verbindungsvariablen basierende Methode aus Beispiel 1 verwendet, um die Vorgänge mehrerer Trigger zu steuern. Dabei werden zwei Prozeduren erstellt, die aufgerufen werden können, um mehrere Trigger zu aktivieren bzw. zu deaktivieren. Außerdem wird eine Funktion erstellt, die verwendet werden kann, um zu überprüfen, ob Trigger-Vorgänge aktiviert sind.

1. Erstellen Sie eine Prozedur, die aufgerufen werden kann, um Trigger-Vorgänge zu deaktivieren. Ihr Verhalten basiert auf dem Wert einer Verbindungsvariablen.

```
CREATE PROCEDURE sp_disable_triggers()
BEGIN
    IF VAREXISTS ('enable_trigger_logic') = 0 THEN
        CREATE VARIABLE enable_trigger_logic INT;
    END IF;
    SET enable_trigger_logic = 0;
END;
```

2. Erstellen Sie eine Prozedur, die aufgerufen werden kann, um Trigger-Vorgänge zu aktivieren. Ihr Verhalten basiert auf dem Wert einer Verbindungsvariablen.

```
CREATE PROCEDURE sp_enable_triggers()
BEGIN
    IF VAREXISTS ('enable_trigger_logic') = 0 THEN
        CREATE VARIABLE enable_trigger_logic INT;
    END IF;
    SET enable_trigger_logic = 1;
END;
```

3. Erstellen Sie eine Funktion, die aufgerufen werden kann, um zu ermitteln, ob Ihre Trigger-Vorgänge aktiviert sind:

```
CREATE FUNCTION f_are_triggers_enabled()
RETURNS INT
BEGIN
    IF VAREXISTS ('enable_trigger_logic') = 1 THEN
        RETURN enable_trigger_logic;
    ELSE
        RETURN 1;
    END IF;
END;
```

4. Fügen Sie eine IF-Klausel zu den Triggern hinzu, deren Vorgänge Sie steuern möchten:

```
IF f_are_triggers_enabled() = 1 THEN
    ... your-trigger-logic
END IF;
```

5. Rufen Sie die in Schritt 2 erstellte Prozedur auf, um Trigger-Vorgänge zu aktivieren:

```
CALL sp_enable_triggers();
... execute-code-where-trigger-logic-runs
```

6. Rufen Sie die in Schritt 1 erstellte Prozedur auf, um Trigger-Vorgänge zu deaktivieren:

```
CALL sp_disable_triggers();
... execute-your-code-where-trigger-logic-is-disabled
```

Siehe auch

- „CREATE TRIGGER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Datenbankserveroption -gf“ [[SQL Anywhere Server - Datenbankadministration](#)]
- [fire_triggers](#)-Verbindungseigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]
- „Trigger“ auf Seite 98

Privilegien zum Ausführen von Triggern

Sie können kein Privileg zum Ausführen eines Triggers erteilen, weil Trigger nicht von Benutzern ausgeführt werden können. SQL Anywhere löst sie vielmehr als Reaktion auf Aktionen in der Datenbank aus. Trotzdem sind einem Trigger Privilegien zugeordnet, die sein Recht definieren, bestimmte Aktionen auszuführen.

Trigger werden mit den Privilegien des Eigentümers der Tabelle ausgeführt, für die sie definiert sind, und weder mit den Privilegien des Benutzers, der das Auslösen des Triggers verursacht hat, noch mit den Privilegien des Benutzers, der den Trigger erstellt hat.

Wenn ein Trigger auf eine Tabelle Bezug nimmt, benutzt er die Rollenmitgliedschaften des Tabellenerstellers zur Ermittlung der Position von Tabellen, für die kein expliziter Eigentümername angegeben wurde. Wenn beispielsweise ein Trigger für die Tabelle "Benutzer_1.Tabelle_A" die Tabelle "Tabelle_B" referenziert und den Eigentümer der "Tabelle_B" nicht angibt, muss entweder die "Tabelle_B" von "Benutzer_1" erstellt worden sein oder "Benutzer_1" muss (direkt oder indirekt) Mitglied einer Rolle sein, die Eigentümer von "Tabelle_B" ist. Wenn keine Bedingung zutrifft, gibt der Datenbankserver beim Auslösen des Triggers eine Meldung zurück, wonach die Tabelle nicht gefunden werden kann.

Benutzer_1 benötigt außerdem die erforderlichen Privilegien zum Ausführen der im Trigger angegebenen Vorgänge.

Siehe auch

- „Benutzersicherheit (Rollen und Privilegien)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Erweiterte Informationen zu Triggern

Ein schwierig zu verstehender Aspekt bei Triggern ist die Reihenfolge, in der Trigger ausgelöst werden, wenn ein auslösender Vorgang Auswirkungen auf mehrere Trigger hat. Ob konkurrierende Trigger ausgelöst werden und in welcher Reihenfolge sie ausgelöst werden, hängt von zwei Dingen ab: Triggertyp (BEFORE, INSTEAD OF oder AFTER) und Triggerbereich (Zeilen- oder Anweisungsebene).

Mit UPDATE-Anweisungen können Spaltenwerte in mehr als einer Tabelle geändert werden. Die Sequenz der Trigger-Auslösung ist für jede Tabelle gleich, aber die Reihenfolge, in der die Tabellen aktualisiert werden, ist nicht garantiert.

Für Trigger auf Zeilenebene gilt: BEFORE-Trigger werden vor INSTEAD OF-Triggern ausgelöst, die wiederum vor AFTER-Triggern ausgelöst werden. Alle Trigger auf Zeilenebene für eine bestimmte Zeile werden vor den Triggern für die Folgezeile ausgelöst.

Bei Triggern auf Anweisungsebene werden INSTEAD OF-Trigger vor AFTER-Triggern ausgelöst. BEFORE-Trigger auf Anweisungsebene werden nicht unterstützt.

Wenn es konkurrierende AFTER-Trigger auf Anweisungs- und Zeilenebene gibt, werden die AFTER-Trigger auf Anweisungsebene ausgelöst, nachdem alle Trigger auf Zeilenebene beendet wurden.

Wenn es konkurrierende INSTEAD OF-Trigger auf Anweisungs- und Zeilenebene gibt, werden die Trigger der Zeilenebene nicht ausgelöst.

Die für den AFTER STATEMENT-Trigger erstellten temporären Tabellen haben dasselbe Schema wie die Basistabelle, mit denselben Spaltennamen und Datentypen. Diese Tabellen enthalten jedoch keine Primärschlüssel, Fremdschlüssel oder Indizes. Die Reihenfolge der Zeilen in den temporären Tabellen ist nicht garantiert und stimmt möglicherweise nicht mit der Reihenfolge überein, in der die Zeilen der Basistabelle ursprünglich aktualisiert wurden.

INSTEAD OF-Trigger

INSTEAD OF-Trigger unterscheiden sich von BEFORE- und AFTER-Triggern, da beim Auslösen eines INSTEAD OF-Triggers die auslösende Aktion übersprungen wird und stattdessen die festgelegte Aktion durchgeführt wird.

Die folgende Liste zeigt die Fähigkeiten und Einschränkungen, die speziell für INSTEAD OF-Trigger gelten:

- Für jedes Trigger-Ereignis in einer bestimmten Tabelle kann es nur einen einzigen INSTEAD OF-Trigger geben.
- INSTEAD OF-Trigger können für eine Tabelle oder eine Ansicht definiert werden. INSTEAD OF-Trigger können jedoch nicht für materialisierte Ansichten definiert werden, da Sie für diese keine DML-Vorgänge ausführen können wie etwa die Anweisungen INSERT, DELETE und UPDATE.
- Wenn Sie einen INSTEAD OF-Trigger definieren, können Sie keine ORDER- oder WHEN-Klauseln angeben.
- Für das Trigger-Ereignis UPDATE OF *column-list* können Sie keinen INSTEAD OF-Trigger festlegen.
- Ob ein INSTEAD OF-Trigger Rekursionen ausführt, hängt davon ab, ob das Ziel des Triggers eine Basistabelle oder eine Ansicht ist. Die Rekursion ist für Ansichten möglich, nicht aber für Basistabellen. D.h., wenn ein INSTEAD OF-Trigger DML-Vorgänge in der Basistabelle durchführt, für die der Trigger definiert ist, werden durch diese Vorgänge keine Trigger ausgelöst (auch keine BEFORE- oder AFTER-Trigger). Falls es sich beim Ziel um eine Ansicht handelt, werden alle Trigger für die Vorgänge ausgelöst, die für die Ansicht ausgeführt werden.
- Wenn ein INSTEAD OF-Trigger für eine Tabelle definiert ist, können Sie in dieser Tabelle keine INSERT-Anweisung mit einer ON EXISTING-Klausel ausführen. Falls Sie dies doch versuchen, wird der Fehler `SQLSTATE=42000` gemeldet.
- Sie können keine INSERT-Anweisung für eine Ansicht ausführen, die mit WITH CHECK OPTION festgelegt wurde (oder die in einer anderen, auf diese Art festgelegten Ansicht verschachtelt ist) und für

die ein INSTEAD OF INSERT-Trigger festgelegt ist. Dies gilt auch für die Anweisungen UPDATE und DELETE. Falls Sie dies doch versuchen, wird der Fehler `SQL_CHECK_TRIGGER_CONFLICT` gemeldet.

- Falls ein INSTEAD OF-Trigger als Ergebnis einer positionsbasierten UPDATE-, DELETE- oder PUT-Anweisung oder einer breiten Einfügung ausgelöst wird, wird der Fehler `SQL_INSTEAD_TRIGGER_POSITIONED` gemeldet.

Nicht aktualisierbare Ansichten mit INSTEAD OF-Trigger aktualisieren

INSTEAD OF-Trigger ermöglichen es Ihnen, INSERT-, UPDATE- oder DELETE-Anweisungen auf Ansichten anzuwenden, die normalerweise nicht aktualisierbar sind. Der Hauptteil des Triggers definiert, was die Ausführung der entsprechenden INSERT-, UPDATE- oder DELETE-Anweisung bedeutet. Angenommen, Sie erstellen den folgenden Index:

```
CREATE VIEW V1 ( Surname, GivenName, State )
  AS SELECT DISTINCT Surname, GivenName, State
  FROM Contacts;
```

Sie können keine Zeilen aus V1 löschen, da das Schlüsselwort DISTINCT dazu führt, dass V1 nicht aktualisierbar ist. In anderen Worten, der Datenbankserver kann nicht eindeutig ermitteln, was es bedeutet, eine Zeile aus V1 zu löschen. Sie könnten jedoch einen INSTEAD OF DELETE-Trigger definieren, der einen Löschvorgang in V1 durchführt. Der folgende Trigger löscht beispielsweise alle Zeilen mit einem bestimmten "Surname", "GivenName" und "State" aus "Contacts", wenn diese Zeile aus V1 gelöscht wird:

```
CREATE TRIGGER V1_Delete
  INSTEAD OF DELETE ON V1
  REFERENCING OLD AS old_row
  FOR EACH ROW
  BEGIN
    DELETE FROM Contacts
      WHERE Surname = old_row.Surname
      AND GivenName = old_row.GivenName
      AND State = old_row.State
  END;
```

Wenn der Trigger "V1_Delete" definiert ist, können Sie Zeilen aus V1 löschen. Sie können auch andere INSTEAD OF-Trigger festlegen, um INSERT- und UPDATE-Anweisungen auf V1 zu ermöglichen.

Falls eine Ansicht mit einem INSTEAD OF DELETE-Trigger in einer anderen Ansicht verschachtelt ist, wird sie zum Prüfen der Aktualisierbarkeit eines DELETE-Vorgangs wie eine Basistabelle behandelt. Dies trifft auch für INSERT- und UPDATE-Vorgänge zu. Um das vorherige Beispiel fortzuführen, erstellen Sie nun eine andere Ansicht:

```
CREATE VIEW V2 ( Surname, GivenName ) AS
  SELECT Surname, GivenName from V1;
```

Ohne den Trigger "V1_Delete" können Sie keine Zeilen aus V2 löschen, weil V1 normalerweise nicht aktualisierbar ist, und das Gleiche gilt für V2. Wenn Sie allerdings einen INSTEAD OF DELETE-Trigger auf V1 definieren, können Sie Zeilen aus V2 löschen. Jede aus V2 gelöschte Zeile führt zum Löschen einer Zeile aus V1, wodurch der Trigger "V1_Delete" ausgelöst wird.

Seien Sie bei der Definition eines INSTEAD OF-Triggers auf einer verschachtelten Ansicht vorsichtig, da das Auslösen des Triggers unvorhersehbare Konsequenzen haben kann. Um das gewünschte Verhalten

explizit anzugeben, definieren Sie den INSTEAD OF-Trigger für jede Ansicht, die die verschachtelte Ansicht referenziert.

Der folgende Trigger könnte für V2 definiert werden, um das gewünschte Verhalten für eine DELETE-Anweisung zu bewirken:

```
CREATE TRIGGER V2_Delete
  INSTEAD OF DELETE ON V2
  REFERENCING OLD AS old_row
  FOR EACH ROW
  BEGIN
    DELETE FROM Contacts
      WHERE Surname = old_row.Surname
      AND GivenName = old_row.GivenName
  END;
```

Der Trigger "V2_Delete" sorgt dafür, dass das Verhalten eines Löschvorgangs auf V2 unverändert bleibt, sogar wenn der INSTEAD OF DELETE-Trigger für V1 gelöscht oder geändert wird.

Siehe auch

- „CREATE TRIGGER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Batchanweisungen

Ein Batch ist ein Satz von SQL-Anweisungen, die gemeinsam abgesetzt und als Gruppe, eine nach der anderen, ausgeführt werden. Die in Prozeduren verwendeten Steueranweisungen (CASE, IF, LOOP ect.) können auch in Batches verwendet werden. Wenn der Batch aus einer zusammengesetzten Anweisung besteht, die von BEGIN/END umschlossen ist, kann er auch Hostvariable, lokale Deklarationen für Variable, Cursor, temporäre Tabellen und Ausnahmefehler enthalten. Hostvariablen-Referenzen sind innerhalb von Batches mit folgenden Einschränkungen zulässig:

- Nur eine Anweisung im Batch darf Hostvariable referenzieren.
- Der Anweisung, die Hostvariable verwendet, darf keine Anweisung vorangehen, die eine Ergebnismenge zurückgibt.

Die Verwendung von BEGIN/END wird empfohlen, um eindeutig anzugeben, wann ein Batch verwendet wird.

Anweisungen im Batch können durch Semikolons begrenzt werden. In diesem Fall entspricht der Batch dem Watcom-SQL-Dialekt. Ein Batch mit mehreren Anweisungen, der keine Semikolons zur Begrenzung der Anweisungen verwendet, entspricht dem Transact-SQL-Dialekt. Der Dialekt des Batches legt fest, welche Anweisungen innerhalb des Batches zulässig sind und wie Fehler im Batch behandelt werden.

Batches sind gespeicherten Prozeduren ähnlich, allerdings gibt es einige Unterschiede:

- Batches haben keine Namen.
- Batches akzeptieren keine Parameter.

- Batches werden nicht dauerhaft in der Datenbank gespeichert.
- Batches können nicht von verschiedenen Verbindungen gemeinsam genutzt werden.

Ein einfacher Batch besteht aus einer Gruppe von SQL-Anweisungen ohne Trennzeichen, gefolgt von einer separaten Zeile, auf der nur das Wort "go" steht. Im folgenden Beispiel wird die Abteilung "Eastern Sales" erstellt und alle Vertreter von Massachusetts werden in diese Abteilung versetzt. Dies ist ein Beispiel eines Transact-SQL-Batches.

```
INSERT
INTO Departments ( DepartmentID, DepartmentName )
VALUES ( 220, 'Eastern Sales' )

UPDATE Employees
SET DepartmentID = 220
WHERE DepartmentID = 200
AND State = 'MA'

COMMIT
go
```

Das Wort "go" wird von Interactive SQL erkannt und bewirkt, dass die vorherigen Anweisungen als ein einziger Batch an den Server gesendet werden.

Das folgende Beispiel sieht zwar ähnlich aus, wird aber von Interactive SQL unterschiedlich behandelt. In diesem Beispiel wird der Transact-SQL-Dialekt nicht benutzt. Jede Anweisung wird durch ein Semikolon abgeschlossen. Interactive SQL sendet jede durch Semikolon abgeschlossene Anweisung separat an den Server. Sie wird nicht als Batch behandelt.

```
INSERT
INTO Departments ( DepartmentID, DepartmentName )
VALUES ( 220, 'Eastern Sales' );

UPDATE Employees
SET DepartmentID = 220
WHERE DepartmentID = 200
AND State = 'MA';

COMMIT;
```

Um die Behandlung durch Interactive SQL als Batch zu erreichen, können Anweisungen mit BEGIN ... END. Die folgende Darstellung ist eine geänderte Version des vorherigen Beispiels. Die drei Anweisungen in der zusammengesetzten Anweisung werden als Batch an den Server gesendet.

```
BEGIN
  INSERT
  INTO Departments ( DepartmentID, DepartmentName )
  VALUES ( 220, 'Eastern Sales' );

  UPDATE Employees
  SET DepartmentID = 220
  WHERE DepartmentID = 200
  AND State = 'MA';

  COMMIT;
END
```

In diesem Beispiel hat es keine Auswirkungen auf das Endergebnis, ob ein Batch oder einzelne Anweisungen vom Server ausgeführt werden. In manchen Situationen kann es jedoch Auswirkungen geben. Beachten Sie das folgende Beispiel.

```
DECLARE @CurrentID INTEGER;
SET @CurrentID = 207;
SELECT Surname FROM Employees
WHERE EmployeeID=@CurrentID;
```

Wenn Sie dieses Beispiel unter Verwendung von Interactive SQL ausführen, meldet der Datenbankserver einen Fehler, der angibt, dass die Variable nicht gefunden werden kann. Dieser Fehler wird gemeldet, da Interactive SQL drei separate Anweisungen an den Server sendet. Sie werden nicht als Batch ausgeführt. Wie bereits erläutert wurde, kann das Problem gelöst werden, indem eine zusammengesetzte Anweisung benutzt wird, um Interactive SQL zu zwingen, diese Anweisungen als Batch an den Server zu senden. Mit dem folgenden Beispiel wird dies erreicht.

```
BEGIN
  DECLARE @CurrentID INTEGER;
  SET @CurrentID = 207;
  SELECT Surname FROM Employees
  WHERE EmployeeID=@CurrentID;
END
```

Wenn eine Gruppe von Anweisungen in BEGIN und END gesetzt wird, verarbeitet Interactive SQL sie als Batch.

Die IF-Anweisung ist ein weiteres Beispiel einer zusammengesetzten Anweisung. Interactive SQL sendet die folgenden Anweisungen als einen einzelnen Batch an den Server.

```
IF EXISTS(  SELECT *
            FROM SYSTAB
            WHERE table_name='Employees' )
THEN
  SELECT  Surname AS LastName,
          GivenName AS FirstName
  FROM Employees;
  SELECT Surname, GivenName
  FROM Customers;
  SELECT Surname, GivenName
  FROM Contacts;
ELSE
  MESSAGE 'The Employees table does not exist'
  TO CLIENT;
END IF
```

Diese Situation tritt nicht auf, wenn andere Techniken zum Vorbereiten und Ausführen von SQL-Anweisungen benutzt werden. Eine Anwendung, die ODBC benutzt, kann beispielsweise eine Reihe von durch Semikolon getrennten Anweisungen als Batch vorbereiten und ausführen.

Beim Mischen von Interactive SQL-Anweisungen mit SQL-Anweisungen, die für den Server bestimmt sind, sollte man Vorsicht walten lassen. Das folgende Beispiel zeigt, welche Probleme beim Mischen von Interactive SQL-Anweisungen und SQL-Anweisungen auftreten können. In diesem Beispiel ist die Interactive SQL OUTPUT-Anweisung in die zusammengesetzte Anweisung eingebettet. Daher wird sie mit allen anderen Anweisungen zusammen als Batch an den Server gesendet, was zu einem Syntaxfehler führt.

```
IF EXISTS(    SELECT *
              FROM SYSTAB
              WHERE table_name='Employees' )
THEN
    SELECT    Surname AS LastName,
              GivenName AS FirstName
    FROM Employees;
    SELECT Surname, GivenName
    FROM Customers;
    SELECT Surname, GivenName
    FROM Contacts;
    OUTPUT TO 'c:\\temp\\query.txt';
ELSE
    MESSAGE 'The Employees table does not exist'
    TO CLIENT;
END IF
```

Die richtige Anordnung der OUTPUT-Anweisung wird unten gezeigt.

```
IF EXISTS(    SELECT *
              FROM SYSTAB
              WHERE table_name='Employees' )
THEN
    SELECT    Surname AS LastName,
              GivenName AS FirstName
    FROM Employees;
    SELECT Surname, GivenName
    FROM Customers;
    SELECT Surname, GivenName
    FROM Contacts;
ELSE
    MESSAGE 'The Employees table does not exist'
    TO CLIENT;
END IF;
OUTPUT TO 'c:\\temp\\query.txt';
```

Siehe auch

- „Transact-SQLBatches“ auf Seite 671
- „SQL-Anweisungen ausführen (Interactive SQL)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Aufbau von Prozeduren, Triggern und benutzerdefinierten Funktionen

Der Hauptteil einer Prozedur oder eines Triggers besteht aus einer zusammengesetzten Anweisung. Eine zusammengesetzte Anweisung besteht aus einer BEGIN- und END-Anweisung, zwischen die die anderen SQL-Anweisungen gesetzt werden. Semikola begrenzen die einzelnen Anweisungen.

Siehe auch

- „Zusammengesetzte Anweisungen“ auf Seite 119

Parameterdeklaration für Prozeduren

Prozedurparameter werden als Liste in der Anweisung CREATE PROCEDURE übergeben. Parameternamen müssen den Regeln für andere Datenbankbezeichner, wie z.B. Spaltennamen, entsprechen. Sie müssen einen gültigen Datentyp haben und es kann ihnen eines der folgenden Schlüsselwörter vorangehen: IN, OUT oder INOUT. Standardmäßig sind Parameter INOUT-Parameter. Diese Schlüsselwörter haben folgende Bedeutungen:

- **IN** Das Argument ist ein Ausdruck, der für die Prozedur einen Wert bereitstellt.
- **OUT** Das Argument ist eine Variable, der von der Prozedur ein Wert verliehen werden kann.
- **INOUT** Das Argument ist eine Variable, die der Prozedur einen Wert liefert, und deren Wert von der Prozedur geändert werden kann.

Sie können Prozedurparametern in der Anweisung CREATE PROCEDURE Standardwerte verleihen. Der Standardwert muss eine Konstante, darf aber auch NULL sein. Beispiel: Die folgende Prozedur benutzt NULL als Standardwert für einen IN-Parameter, damit nicht eine Abfrage ausgeführt wird, die keinen Sinn ergibt:

```
CREATE PROCEDURE CustomerProducts(  
    IN customer_ID  
        INTEGER DEFAULT NULL )  
RESULT ( product_ID INTEGER,  
        quantity_ordered INTEGER )  
BEGIN  
    IF customer_ID IS NULL THEN  
        RETURN;  
    ELSE  
        SELECT      Products.ID,  
                    sum( SalesOrderItems.Quantity )  
        FROM        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;
```

Die folgende Anweisung weist den Standardwert NULL zu, und die Prozedur führt eine Rückgabe durch, anstatt die Abfrage auszuführen.

```
CALL CustomerProducts( );
```

Siehe auch

- „SQL-Datentypen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Möglichkeiten zum Übergeben von Parametern an Prozeduren

Sie können die Standardwerte der Parameter von gespeicherten Prozeduren mit einer von zwei Formen der CALL-Anweisung nutzen.

Wenn die optionalen Parameter am Ende der Argumentliste in der Anweisung CREATE PROCEDURE stehen, können sie in der CALL-Anweisung weggelassen werden. Als Beispiel sehen wir uns eine Prozedur mit drei INOUT-Parametern an:

```
CREATE PROCEDURE SampleProcedure(  
    INOUT var1 INT DEFAULT 1,  
        INOUT var2 int DEFAULT 2,  
        INOUT var3 int DEFAULT 3 )  
...
```

In diesem Beispiel wird vorausgesetzt, dass die aufrufende Umgebung drei Variablen eingerichtet hat, die die an die Prozedur übergebenen Werte enthalten:

```
CREATE VARIABLE V1 INT;  
CREATE VARIABLE V2 INT;  
CREATE VARIABLE V3 INT;
```

Die Prozedur "SampleProcedure" kann wie folgt aufgerufen werden, indem nur der erste Parameter übergeben wird, wobei die Standardwerte für *var2* und *var3* verwendet werden.

```
CALL SampleProcedure( V1 );
```

Die Prozedur kann auch wie folgt aufgerufen werden, indem nur der zweite Parameter übergeben wird, wobei der DEFAULT-Wert für den ersten Parameter verwendet wird:

```
CALL SampleProcedure( DEFAULT, V2 );
```

Eine flexiblere Methode für den Aufruf von Prozeduren mit optionalen Argumenten ist die Übergabe der Parameter nach Namen. Die Prozedur "SampleProcedure" kann wie folgt aufgerufen werden:

```
CALL SampleProcedure( var1 = V1, var3 = V3 );
```

Die zweite Möglichkeit:

```
CALL SampleProcedure( var3 = V3, var1 = V1 );
```

Wie Parameter an Funktionen übergeben werden

Benutzerdefinierte Funktionen werden nicht mit der CALL-Anweisung aufgerufen, sondern auf dieselbe Weise verwendet wie integrierte Funktionen. In der folgenden Anweisung wird beispielsweise die Funktion "FullName" verwendet, um die Namen von Mitarbeitern abzurufen:

Hinweise

- Standardparameter können in aufrufenden Funktionen verwendet werden. Allerdings können Parameter nicht nach Namen an Funktionen übergeben werden.
- Parameter werden nach Werten, und nicht nach Referenz übergeben. Auch wenn die Funktion den Wert des Parameters ändert, wird diese Änderung nicht an die aufrufende Umgebung zurückgegeben.
- Ausgabeparameter können in benutzerdefinierten Funktionen nicht verwendet werden.
- Benutzerdefinierte Funktionen können keine Ergebnismengen zurückgeben.

Beispiel: Namensliste aller Mitarbeiter

Führen Sie in Interactive SQL folgende Abfrage aus:

```
SELECT FullName( GivenName, Surname ) AS Name
FROM Employees;
```

Die folgenden Ergebnisse werden angezeigt:

Name
Fran Whitney
Matthew Cobb
Philip Chin
Julie Jordan
...

Siehe auch

- [„Benutzerdefinierte Funktionen erstellen“ auf Seite 94](#)

Steueranweisungen

Es gibt mehrere Steueranweisungen für die logische Abfolge und für Entscheidungen, die im Hauptteil einer Prozedur, eines Triggers oder einer benutzerdefinierten Funktion bzw. in einem Batch verwendet werden können. Folgende Steueranweisungen sind verfügbar:

Steueranweisung	Syntax
Zusammengesetzte Anweisungen	<pre>BEGIN [ATOMIC] Statement-list END</pre>
Bedingte Ausführung: IF	<pre>IF condition THEN Statement-list ELSEIF condition THEN Statement-list ELSE Statement-list END IF</pre>
Bedingte Ausführung: CASE	<pre>CASE expression WHEN value THEN Statement-list WHEN value THEN Statement-list ELSE Statement-list END CASE</pre>

Steueranweisung	Syntax
Wiederholung: WHILE, LOOP	<pre>WHILE condition LOOP Statement-list END LOOP</pre>
Wiederholung: FOR Cursorschleife	<pre>FOR loop-name AS cursor-name CURSOR FOR select-statement DO Statement-list END FOR</pre>
Abbruch: LEAVE	<pre>LEAVE label</pre>
CALL	<pre>CALL procname(arg, ...)</pre>

Siehe auch

- „BEGIN-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „IF-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CASE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „LOOP-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „FOR-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „LEAVE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CALL-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Zusammengesetzte Anweisungen

Eine zusammengesetzte Anweisung beginnt mit dem Schlüsselwort BEGIN und endet mit dem Schlüsselwort END. Der Hauptteil einer Prozedur oder eines Triggers besteht aus einer **zusammengesetzten Anweisung**. Zusammengesetzte Anweisungen können auch in Batches benutzt werden. Zusammengesetzte Anweisungen können verschachtelt und mit anderen Steueranweisungen kombiniert werden, um die Abfolge der Befehle in Prozeduren und Triggern oder in Batches festzulegen.

Eine zusammengesetzte Anweisung ist eine Gruppe von SQL-Anweisungen, die als Einheit behandelt wird. SQL-Anweisungen in einer zusammengesetzten Anweisung müssen durch Semikola getrennt werden.

Deklarationen in zusammengesetzten Anweisungen

Lokale Deklarationen in einer zusammengesetzten Anweisung folgen unmittelbar auf das Schlüsselwort BEGIN. Diese lokalen Deklarationen bestehen nur in der zusammengesetzten Anweisung. Innerhalb einer zusammengesetzten Anweisung können Sie folgende Elemente deklarieren:

- Variable
- Cursor

- Temporäre Tabellen
- Ausnahmebedingungen (Fehler-IDs)

Lokale Deklarationen können von jeder beliebigen Anweisung in dieser zusammengesetzten Anweisung oder in jeder beliebigen darin verschachtelten zusammengesetzten Anweisung referenziert werden. Lokale Deklarationen sind für andere Prozeduren nicht sichtbar, die aus der zusammengesetzten Anweisung aufgerufen werden.

Atomare zusammengesetzte Anweisungen

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 geänderten Zeilen wieder auf den Ausgangsstatus zurückgesetzt. Die UPDATE-Anweisung ist atomar.

Alle nicht zusammengesetzten SQL-Anweisungen sind atomar. Eine zusammengesetzte Anweisung kann atomar gemacht werden, indem nach dem Schlüsselwort BEGIN das Schlüsselwort ATOMIC angehängt wird.

```
BEGIN ATOMIC
  UPDATE Employees
  SET ManagerID = 501
  WHERE EmployeeID = 467;
  UPDATE Employees
  SET BirthDate = 'bad_data';
END
```

In diesem Beispiel sind die beiden Aktualisierungsanweisungen Teil einer atomaren zusammengesetzten Anweisung. Sie können nur als Gesamtheit gelingen oder fehlschlagen. Die erste Aktualisierungsanweisung wird erfolgreich verlaufen. Die zweite verursacht einen Datenkonvertierungsfehler, da der der Spalte "BirthDate" zugewiesene Wert nicht in ein Datum umgewandelt werden kann.

Die atomare zusammengesetzte Anweisung schlägt fehl, und die Wirkung beider UPDATE-Anweisungen wird annulliert. Auch wenn die derzeit gerade ausgeführte Transaktion festgeschrieben wird, werden die Anweisungen in einer atomaren zusammengesetzten Anweisung nicht wirksam.

Wenn eine atomare zusammengesetzte Anweisung erfolgreich ist, werden die innerhalb der zusammengesetzten Anweisung durchgeführten Änderungen nur wirksam, wenn die derzeit ausführende Transaktion festgeschrieben wird. Für den Fall, dass eine atomare zusammengesetzte Anweisung erfolgreich ist, die Transaktion, in der sie vorkommt, jedoch zurückgesetzt wird, so wird die atomare zusammengesetzte Anweisung ebenfalls zurückgesetzt. Am Anfang der atomaren zusammengesetzten Anweisung wird ein Savepoint eingerichtet. Alle Fehler innerhalb der Anweisung führen zum Zurücksetzen zu diesem Savepoint.

Wenn eine atomare zusammengesetzte Anweisung im Autocommit-Modus (unverkettet) ausgeführt wird, ändert sich der Festschreibemodus auf manuell (verkettet), bis die Anweisungsausführung abgeschlossen ist. Im manuellen Modus bewirken DML-Anweisungen, die innerhalb einer atomaren zusammengesetzten Anweisung ausgeführt werden, kein sofortiges COMMIT oder ROLLBACK. Wenn die atomare

zusammengesetzte Anweisung erfolgreich abschließt, wird eine COMMIT-Anweisung ausgeführt, ansonsten wird eine ROLLBACK-Anweisung ausgeführt.

COMMIT und ROLLBACK sowie einige ROLLBACK TO SAVEPOINT-Anweisungen können nicht in einer atomaren zusammengesetzten Anweisung verwendet werden.

Siehe auch

- „Autocommit-Verhalten steuern“ [[SQL Anywhere Server - Programmierung](#)]
- „Autocommit-Modus und manueller Commit-Modus“ [[SQL Anywhere Server - Programmierung](#)]
- „Transaktionen und Savepoints in Prozeduren, Triggern und benutzerdefinierten Funktionen“ auf Seite 146
- Routine bei Ausnahmefehlern und atomare zusammengesetzte Anweisungen auf Seite 138

Ergebnismengen

Prozeduren können Ergebnisse als einzelne Zeilen mit Daten oder als mehrere Zeilen zurückgeben. Die Ergebnisse können als Argumente von der Prozedur zurückgegeben werden. Ergebnisse aus mehreren Datenzeilen werden als Ergebnismengen zurückgegeben. Prozeduren können auch einen einzelnen Wert zurückgeben, der in der RETURN-Anweisung angegeben wird.

Siehe auch

- „Prozeduren“ auf Seite 82

Einen Wert mit der RETURN-Anweisung zurückgeben

Die RETURN-Anweisung gibt einen einzelnen Ganzzahlwert an die aufrufende Umgebung zurück, wodurch ein sofortiger Ausstieg aus der Prozedur bewirkt wird.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. Führen Sie die folgende Anweisung aus:

```
RETURN expression
```

2. Der Wert des gelieferten Ausdrucks wird an die aufrufende Umgebung zurückgegeben. Verwenden Sie eine Erweiterung der CALL-Anweisung zum Speichern des Rückgabewerts in einer Variablen:

```
CREATE VARIABLE returnval INTEGER;  
returnval = CALL variable/procedure-name? myproc();
```

Ergebnisse

Ein Wert wird zurückgegeben und als Variable gespeichert.

Siehe auch

- „RETURN-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Möglichkeiten zum Zurückgeben von Ergebnissen als Prozedurparameter

Prozeduren können Ergebnisse in den Parametern für die Prozedur an die aufrufende Umgebung zurückgeben. In einer Prozedur können mithilfe einer der folgenden Methoden Parametern und Variablen Werte zugeordnet werden:

- SET-Anweisung

In der nachstehenden Prozedur wird ein Wert in einem OUT-Parameter zurückgegeben, wobei der Wert mit einer SET-Anweisung zugeordnet wird. Sie müssen das CREATE PROCEDURE-Systemprivileg haben, um die folgende Anweisung ausführen zu können:

```
CREATE PROCEDURE greater(  
    IN a INT,  
    IN b INT,  
    OUT c INT )  
BEGIN  
    IF a > b THEN  
        SET c = a;  
    ELSE  
        SET c = b;  
    END IF ;  
END;
```

- SELECT-Anweisung mit INTO-Klausel

Eine einzeilige Abfrage fragt höchstens eine Zeile von der Datenbank ab. Dieser Typ einer Abfrage wird durch eine SELECT-Anweisung mit einer INTO-Klausel durchgeführt. Die INTO-Klausel folgt auf die SELECT-Liste und steht vor der FROM-Klausel. Sie enthält eine Liste der Variablen, die die Werte der einzelnen Elemente aus der SELECT-Liste aufnehmen sollen. Es müssen genauso viele Variablen vorhanden sein wie Elemente in der SELECT-Liste.

Wenn eine SELECT-Anweisung ausgeführt wird, ruft der Datenbankserver die Ergebnisse der SELECT-Anweisung ab und platziert sie in den Variablen. Falls die Abfrageergebnisse mehr als eine Zeile enthalten, gibt der Datenbankserver einen Fehler zurück. Für Abfragen, die mehr als eine Zeile zurückgeben, muss ein Cursor verwendet werden.

Wenn durch die Abfrage keine Zeilen ausgewählt werden, werden die Variablen nicht aktualisiert und es wird eine Warnung zurückgegeben.

Sie benötigen die geeigneten SELECT-Privilegien für das Objekt, um eine SELECT-Anweisung ausführen zu können.

Beispiel: 1: Erstellen Sie eine Prozedur und wählen Sie ihre Prozedur mit einer SELECT...INTO-Anweisung.

1. Starten Sie Interactive SQL und stellen Sie eine Verbindung zur SQL Anywhere-Beispieldatenbank her. Sie benötigen das CREATE PROCEDURE-Systemprivileg und entweder das SELECT-Privileg für die Tabelle "Employees" oder das SELECT ANY TABLE-Systemprivileg.

2. Führen Sie im Fensterausschnitt **SQL-Anweisungen** die folgende Anweisung aus, um eine Prozedur (AverageSalary) zu erstellen, die den Durchschnittslohn der Mitarbeiter als OUT-Parameter zurückgibt:

```
CREATE PROCEDURE AverageSalary( OUT average_salary NUMERIC(20,3) )
BEGIN
    SELECT AVG( Salary )
    INTO average_salary
    FROM GROUPO.Employees;
END;
```

3. Erstellen Sie eine Variable, die die Ausgabe der Prozedur aufnehmen soll. In diesem Fall ist die Ausgabevariable numerisch, mit drei Dezimalstellen.

```
CREATE VARIABLE Average NUMERIC(20,3);
```

4. Rufen Sie die Prozedur auf, indem Sie die erstellte Variable benutzen, um das Ergebnis aufzunehmen:

```
CALL AverageSalary( Average );
```

5. Wenn die Prozedur korrekt erstellt und ausgeführt wurde, zeigt die Registerkarte **Meldungen** von Interactive SQL keine Fehler an.

6. Um den Wert der Variablen zu prüfen, führen Sie folgende Anweisung aus:

```
SELECT Average;
```

7. Sehen Sie sich den Wert in der Ausgabevariable "Average" an. Im Fensterausschnitt **Ergebnisse** der Registerkarte **Ergebnisse** wird ein Wert von 49.988,623 für diese Variable angezeigt: Hierbei handelt es sich um den Durchschnittslohn der Mitarbeiter.

Beispiel 2: Rückgabe der Ergebnisse einer einzeiligen SELECT-Anweisung

1. Starten Sie Interactive SQL und stellen Sie eine Verbindung zur SQL Anywhere-Beispieldatenbank her. Sie benötigen das CREATE PROCEDURE-Systemprivileg und entweder das SELECT-Privileg für die Tabelle "Customers" oder das SELECT ANY TABLE-Systemprivileg.
2. Führen Sie die folgende Anweisung aus, um die Anzahl der Bestellungen zurückzugeben, die von einem bestimmten Kunden aufgegeben wurden:

```
CREATE PROCEDURE OrderCount(
    IN customer_ID INT,
    OUT Orders INT )
BEGIN
    SELECT COUNT(SalesOrders.ID)
    INTO Orders
    FROM GROUPO.Customers
    KEY LEFT OUTER JOIN SalesOrders
    WHERE Customers.ID = customer_ID;
END;
```

3. Testen Sie diese Prozedur mit den folgenden Anweisungen, die die Anzahl der Bestellungen zeigen, die vom Kunden mit der ID 102 aufgegeben wurden:

```
CREATE VARIABLE orders INT;  
CALL OrderCount ( 102, orders );  
SELECT orders;
```

Hinweise zu Beispiel 2

- Der customer_ID-Parameter wird als ein IN-Parameter deklariert. Dieser Parameter hält die Kundenkennung, die an die Prozedur weitergegeben wird.
- Der Orders-Parameter wird als ein OUT-Parameter deklariert. Er enthält den Wert der Bestellungsvariablen, die an die aufrufende Umgebung zurückgegeben wurde.
- Für die Orders-Variable ist keine DECLARE-Anweisung erforderlich, weil sie in der Argumentliste der Prozedur deklariert wird.
- Die SELECT-Anweisung gibt eine einzelne Zeile zurück und setzt sie in die Variable Orders.

Siehe auch

- „In Ergebnismengen aus Prozeduren zurückgegebene Informationen“ auf Seite 124
- „Datenbankverbindungen“ [[SQL Anywhere Server - Datenbankadministration](#)]

In Ergebnismengen aus Prozeduren zurückgegebene Informationen

Prozeduren können nicht nur in einzelnen Parametern Ergebnisse zurückgeben, sondern auch in Ergebnismengen. Eine Ergebnismenge ist in der Regel das Ergebnis einer Abfrage.

Die Anzahl der Variablen in der RESULT-Klausel muss der Anzahl der Elemente in der SELECT-Liste entsprechen. Automatische Datentypkonvertierung erfolgt, wenn die Datentypen nicht zusammenpassen und die Konvertierung zulässig ist.

Die RESULT-Klausel ist Teil der CREATE PROCEDURE-Anweisung und hat kein Trennzeichen für Anweisungen.

Die Namen der Elemente in der SELECT-Liste müssen nicht mit denen in der RESULT-Klausel übereinstimmen.

Um die Ergebnismengen von Prozeduren in einer Ansicht ändern zu können, muss der Benutzer die entsprechenden Privilegien für die Basistabelle haben.

Wenn eine gespeicherte Prozedur oder eine benutzerdefinierte Funktion ein Ergebnis zurückgibt, kann sie nicht zusätzlich Ausgabeparameter oder Rückgabewerte unterstützen.

Standardmäßig wird in Interactive SQL nur die erste Ergebnismenge angezeigt. Damit eine Prozedur in Interactive SQL mehr als eine Zeile mit Ergebnissen zurückgeben kann, müssen Sie im Dialogfeld **Optionen** auf der Registerkarte **Ergebnisse** die Option **Mehrere Ergebnismengen zeigen** aktivieren.

Beispiel 1

Die folgende Prozedur gibt eine Liste von Kunden, die Bestellungen aufgegeben haben sowie den Gesamtwert der aufgegebenen Bestellungen zurück.

Führen Sie die folgende Anweisung in Interactive SQL aus:

```
CREATE PROCEDURE ListCustomerValue()
RESULT ( "Company" CHAR(36), "Value" INT )
BEGIN
    SELECT CompanyName,
           CAST( SUM( SalesOrderItems.Quantity *
                     Products.UnitPrice )
                AS INTEGER ) AS value
    FROM Customers
    INNER JOIN SalesOrders
    INNER JOIN SalesOrderItems
    INNER JOIN Products
    GROUP BY CompanyName
    ORDER BY value DESC;
END;
```

Beim Ausführen von `CALL ListCustomerValue () ;` wird die folgende Ergebnismenge zurückgegeben:

Company	Value
The Hat Company	5016
The Igloo	3564
The Ultimate	3348
North Land Trading	3144
Molly's	2808
...	...

Beispiel 2

Die folgende Prozedur gibt eine Ergebnismenge zurück, die den Lohn für jeden Mitarbeiter in einer bestimmten Abteilung enthält. Führen Sie die folgende Anweisung in Interactive SQL aus:

```
CREATE PROCEDURE SalaryList( IN department_id INT )
RESULT ( "Employee ID" INT, Salary NUMERIC(20,3) )
BEGIN
    SELECT EmployeeID, Salary
    FROM Employees
    WHERE Employees.DepartmentID = department_id;
END;
```

Die Namen in der RESULT-Klausel werden an die Ergebnisse der Abfrage angepasst und in den angezeigten Ergebnissen als Spaltentitel verwendet.

Um die Löhne der Mitarbeiter in der F+E-Abteilung (Abteilungs-ID 100) aufzulisten, führen Sie die folgende Anweisung aus:

```
CALL SalaryList( 100 );
```

Die folgende Ergebnismenge erscheint im Fensterausschnitt **Ergebnisse**:

Employee ID	Salary
102	45700.000
105	62000.000
160	57490.000
243	72995.000
...	...

Rückgabe von mehreren Ergebnismengen

Sie können Interactive SQL verwenden, um mehr als eine Ergebnismenge aus einer Prozedur zurückzugeben.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Standardmäßig zeigt Interactive SQL nicht mehrere Ergebnismengen an.

Aufgabe

1. Stellen Sie in Interactive SQL eine Verbindung mit der Datenbank her.
2. Klicken Sie auf **Extras » Optionen**.
3. Klicken Sie auf **SQL Anywhere**.
4. Klicken Sie auf der Registerkarte **Ergebnisse** auf **Alle Ergebnismengen anzeigen**.
5. Klicken Sie auf **OK**.

Ergebnisse

Nachdem Sie diese Option aktiviert haben, zeigt Interactive SQL mehrere Ergebnismengen an. Die Einstellung wird sofort wirksam und bleibt für zukünftige Sitzungen wirksam, bis sie deaktiviert wird.

Nächste Schritte

Wenn in einer Prozedurdefinition eine RESULT-Klausel enthalten ist, müssen die Ergebnismengen kompatibel sein: Sie müssen dieselbe Anzahl von Elementen in den SELECT-Listen aufweisen und alle Datentypen müssen automatisch in die Datentypen aus der RESULT-Klausel konvertierbar sein.

Wenn keine RESULT-Klausel angegeben ist, kann eine Prozedur Ergebnismengen zurückgeben, die sich im Hinblick auf die Anzahl und den Typ der zurückgegebenen Spalten unterscheiden.

Beispiel

Die folgende Prozedur listet die Namen aller Mitarbeiter, Kunden und Kontaktpersonen auf, die in der Datenbank enthalten sind:

```
CREATE PROCEDURE ListPeople()  
RESULT ( Surname CHAR(36), GivenName CHAR(36) )  
BEGIN  
    SELECT Surname, GivenName  
    FROM Employees;  
    SELECT Surname, GivenName  
    FROM Customers;  
    SELECT Surname, GivenName  
    FROM Contacts;  
END;
```

Siehe auch

- „Variable Ergebnismengen für Prozeduren“ auf Seite 127
- „isql_show_multiple_result_sets-Option [Interactive SQL]“ [*SQL Anywhere Server - Datenbankadministration*]

Variable Ergebnismengen für Prozeduren

Die RESULT-Klausel ist bei Prozeduren optional. Wenn Sie die RESULT-Klausel auslassen, können Sie Prozeduren schreiben, die je nach Art ihrer Ausführung unterschiedliche Ergebnismengen mit unterschiedlichen Anzahlen oder Typen von Spalten zurückgeben.

Wenn Sie die Funktion der variablen Ergebnismengen nicht benutzen, sollten Sie aus Performancegründen eine RESULT-Klausel programmieren.

Beispiel: Die folgende Prozedur gibt zwei Spalten zurück, wenn die Eingabevariable Y ist, sonst nur eine:

```
CREATE PROCEDURE Names( IN formal char(1) )  
BEGIN  
    IF formal = 'y' THEN  
        SELECT Surname, GivenName  
        FROM Employees  
    ELSE  
        SELECT GivenName  
        FROM Employees  
    END IF  
END;
```

Die Verwendung von variablen Ergebnismengen in Prozeduren unterliegt je nach der Schnittstelle, die von der Clientanwendung benutzt wird, bestimmten Beschränkungen.

- **Embedded SQL** Um die richtige Form der Ergebnismenge zu erhalten, müssen Sie mit DESCRIBE den Prozeduraufruf beschreiben, nachdem der Cursor für die Ergebnismenge geöffnet wurde, aber bevor Zeilen zurückgegeben werden.

Wenn Sie eine Prozedur ohne RESULT-Klausel erstellen und die Prozedur eine variable Ergebnismenge zurückgibt, kann ein DESCRIBE einer SELECT-Anweisung mit Verweis auf die Prozedur fehlschlagen. Um das Fehlschlagen des DESCRIBE-Vorgangs zu verhindern, ist es empfehlenswert, eine WITH-Klausel in die FROM-Klausel der SELECT-Anweisung einzubeziehen. Alternativ können Sie die WITH VARIABLE RESULT-Klausel in der DESCRIBE-Anweisung verwenden. Mithilfe der WITH VARIABLE RESULT-Klausel kann ermittelt werden, ob der Prozeduraufruf nach jeder OPEN-Anweisung beschrieben werden sollte.

- **ODBC** Prozeduren mit variablen Ergebnismengen können von ODBC-Anwendungen verwendet werden. Der SQL Anywhere ODBC-Treiber führt die richtige Beschreibung der variablen Ergebnismengen aus.
- **Open Client-Anwendungen** Open Client-Anwendungen können Prozeduren mit variablen Ergebnismengen verwenden. SQL Anywhere führt die richtige Beschreibung der variablen Ergebnismengen aus.

Siehe auch

- „DESCRIBE-Anweisung [ESQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Cursor in Prozeduren, Triggern, benutzerdefinierten Funktionen und Batches

Ein Cursor wird verwendet, um jeweils eine Zeile gleichzeitig aus einer Abfrage oder einer gespeicherten Prozedur abzurufen, die in ihrer Ergebnismenge mehrere Zeilen haben. Ein Cursor ist eine Verarbeitungsroutine oder ein Bezeichner für die Abfrage oder Prozedur sowie für eine bestimmte Position innerhalb der Ergebnismenge.

Cursor-Verwaltung

Die Verwaltung eines Cursors ist der Verwaltung einer Datei in einer Programmiersprache ähnlich. Folgende Aufgaben sind bei der Verwaltung von Cursor vorgesehen:

1. Deklarieren Sie einen Cursor für eine bestimmte SELECT-Anweisung oder Prozedur mit der DECLARE-Anweisung.
2. Öffnen Sie den Cursor mit der Anweisung OPEN.
3. Rufen Sie mit der FETCH-Anweisung für jeweils eine Zeile Ergebnisse aus dem Cursor ab.
4. Die Warnung "Zeile nicht gefunden" signalisiert das Ende der Ergebnismenge.
5. Schließen Sie den Cursor mit der Anweisung CLOSE.

Standardmäßig wird ein Cursor am Ende einer Transaktion automatisch geschlossen (bei COMMIT- oder ROLLBACK-Anweisungen). Wenn ein Cursor mit der WITH HOLD-Klausel geöffnet wurde, bleibt er für nachfolgende Transaktionen geöffnet, bis er explizit geschlossen wird.

Siehe auch

- „Cursor positionieren“ [[SQL Anywhere Server - Programmierung](#)]

Cursor in SELECT-Anweisungen

Die folgende Prozedur benutzt einen Cursor für die SELECT-Anweisung. Sie basiert auf derselben Abfrage, die in der ListCustomerValue-Prozedur verwendet wird, und veranschaulicht mehrere Funktionen der Sprache von gespeicherten Prozeduren.

```
CREATE PROCEDURE TopCustomerValue(
    OUT TopCompany CHAR(36),
    OUT TopValue INT )
BEGIN
    -- 1. Declare the "row not found" exception
    DECLARE err_notfound
        EXCEPTION FOR SQLSTATE '02000';
    -- 2. Declare variables to hold
    --     each company name and its value
    DECLARE ThisName CHAR(36);
    DECLARE ThisValue INT;
    -- 3. Declare the cursor ThisCompany
    --     for the query
    DECLARE ThisCompany CURSOR FOR
    SELECT CompanyName,
        CAST( sum( SalesOrderItems.Quantity *
            Products.UnitPrice ) AS INTEGER )
        AS value
    FROM Customers
        INNER JOIN SalesOrders
        INNER JOIN SalesOrderItems
        INNER JOIN Products
    GROUP BY CompanyName;
    -- 4. Initialize the values of TopValue
    SET TopValue = 0;
    -- 5. Open the cursor
    OPEN ThisCompany;
    -- 6. Loop over the rows of the query
    CompanyLoop:
    LOOP
        FETCH NEXT ThisCompany
            INTO ThisName, ThisValue;
        IF SQLSTATE = err_notfound THEN
            LEAVE CompanyLoop;
        END IF;
        IF ThisValue > TopValue THEN
            SET TopCompany = ThisName;
            SET TopValue = ThisValue;
        END IF;
    END LOOP CompanyLoop;
    -- 7. Close the cursor
    CLOSE ThisCompany;
END;
```

Hinweise

Die Prozedur TopCustomerValue hat folgende bemerkenswerte Merkmale:

- Eine Ausnahmebedingung wird deklariert. Diese Ausnahmebedingung signalisiert später in der Prozedur, dass eine Schleife über die Ergebnisse einer Abfrage abgeschlossen wurde.
- Zwei lokale Variable ThisName und ThisValue werden deklariert, um die Ergebnisse jeder Zeile der Abfrage aufzunehmen.
- Der Cursor ThisCompany wird deklariert. Die SELECT-Anweisung erzeugt eine Liste der Firmennamen und den Gesamtwert der Bestellungen, die von diesen Unternehmen aufgegeben wurden.
- Der Wert von TopValue wird auf einen Anfangswert von 0 gesetzt, der später in der Schleife verwendet wird.
- Der Cursor ThisCompany wird geöffnet.
- Die LOOP-Anweisung fragt in einer Schleife jede Zeile der Abfrage ab und setzt jeden Firmennamen in die Variablen ThisName und ThisValue. Wenn ThisValue größer als der aktuelle Topwert ist, werden TopCompany und TopValue auf ThisName und ThisValue zurückgesetzt.
- Der Cursor wird am Ende der Prozedur geschlossen.
- Sie können diese Prozedur auch ohne LOOP schreiben, indem Sie eine "ORDER BY Wert"-DESC-Klausel der SELECT-Anweisung hinzufügen. Dann muss nur die erste Zeile des Cursors abgefragt werden.

Das LOOP-Konstrukt in der Prozedur "TopCompanyValue" ist eine Standardform, die beendet wird, nachdem die letzte Zeile verarbeitet wurde. Sie können diese Prozedur in kompakterer Form neu programmieren, indem Sie eine FOR-Schleife verwenden. Die FOR-Anweisung kombiniert einige Aspekte der oben genannten Prozedur in einer einzigen Anweisung.

```
CREATE PROCEDURE TopCustomerValue2(  
    OUT TopCompany CHAR(36),  
    OUT TopValue INT )  
BEGIN  
    -- 1. Initialize the TopValue variable  
    SET TopValue = 0;  
    -- 2. Do the For Loop  
    FOR CompanyFor AS ThisCompany  
        CURSOR FOR  
            SELECT CompanyName AS ThisName,  
                CAST( sum( SalesOrderItems.Quantity *  
                    Products.UnitPrice ) AS INTEGER )  
                AS ThisValue  
            FROM Customers  
                INNER JOIN SalesOrders  
                INNER JOIN SalesOrderItems  
                INNER JOIN Products  
            GROUP BY ThisName  
    DO  
        IF ThisValue > TopValue THEN  
            SET TopCompany = ThisName;  
            SET TopValue = ThisValue;  
        END IF;
```

```
END FOR;
END;
```

Siehe auch

- „In Ergebnismengen aus Prozeduren zurückgegebene Informationen“ auf Seite 124
- „Behandlung von Fehlern und Warnungen“ auf Seite 131
- „Zeile nicht gefunden“ [*Fehlermeldungen*]

Positionierte Aktualisierungen innerhalb von Prozeduren, Triggern, benutzerdefinierten Funktionen und Batches

Die folgende Prozedur benutzt einen aktualisierbaren Cursor für die SELECT-Anweisung. Sie veranschaulicht, wie eine positionierte Aktualisierung für eine Zeile mit der gespeicherten Prozedur ausgeführt wird.

```
CREATE PROCEDURE UpdateSalary(
    IN employeeIdent INT,
    IN salaryIncrease NUMERIC(10,3) )
BEGIN
    -- Procedure to increase (or decrease) an employee's salary
    DECLARE err_notfound
        EXCEPTION FOR SQLSTATE '02000';
    DECLARE oldSalary NUMERIC(20,3);
    DECLARE employeeCursor
        CURSOR FOR SELECT Salary from Employees
            WHERE EmployeeID = employeeIdent
        FOR UPDATE;
    OPEN employeeCursor;
    FETCH employeeCursor INTO oldSalary FOR UPDATE;
    IF SQLSTATE = err_notfound THEN
        MESSAGE 'No such employee' TO CLIENT;
    ELSE
        UPDATE Employees SET Salary = oldSalary + salaryIncrease
            WHERE CURRENT OF employeeCursor;
    END IF;
    CLOSE employeeCursor;
END;
```

Die folgende Anweisung ruft die obige gespeicherte Prozedur auf:

```
CALL UpdateSalary( 105, 220.00 );
```

Behandlung von Fehlern und Warnungen

Nachdem ein Anwendungsprogramm eine SQL-Anweisung ausgeführt hat, kann es einen **Statuscode** ablesen. Dieser Statuscode (oder Rückgabecode) zeigt, ob die ausgeführte Anweisung erfolgreich abgeschlossen wurde oder fehlschlug, und gibt den Grund für ein eventuelles Fehlschlagen an. Derselbe Mechanismus kann verwendet werden, um den Erfolg oder den Fehlschlag einer CALL-Anweisung für eine Prozedur anzuzeigen.

Die Fehlerberichte benutzen die Statusbeschreibungen SQLCODE oder SQLSTATE.

Bei jeder ausgeführten SQL-Anweisung wird ein Wert in spezielle Prozedurvariablen gesetzt, die `SQLSTATE` und `SQLCODE` genannt werden. Der Spezialwert gibt an, ob während der Anweisungsausführung ungewöhnliche Bedingungen aufgetreten sind. Sie können den Wert von `SQLSTATE` oder `SQLCODE` in einer IF-Anwendung nach einer SQL-Anweisung prüfen und entsprechende Maßnahmen bei Erfolg oder Fehlschlag der Anweisung ergreifen.

Beispiel: Die `SQLSTATE`-Variable kann benutzt werden, um anzuzeigen, ob eine Zeile erfolgreich abgerufen werden konnte. In der `TopCustomerValue`-Prozedur wurde der `SQLSTATE`-Test verwendet, um zu erkennen, dass alle Zeilen einer `SELECT`-Anweisung verarbeitet wurden.

Siehe auch

- [Fehlermeldungen](#)

Standardmäßige Behandlung von Fehlern

In diesem Abschnitt wird beschrieben, wie SQL Anywhere Fehler behandelt, die während des Ausführens einer Prozedur entstehen, wenn Sie keine Anweisungen für die Fehlerbehandlung in Ihrer Prozedur einprogrammiert haben.

Um ein anderes Verhalten zu erzielen, können Sie Ausnahmeroutinen verwenden.

Warnungen werden etwas anders behandelt als Fehler.

Es gibt zwei Arten für die Fehlerbehandlung mit expliziten Fehlerbehandlungsroutinen:

- **Standard-Fehlerbehandlung** Die Prozedur oder der Trigger schlägt fehl und gibt einen Fehlercode an die aufrufende Umgebung zurück.
- **ON EXCEPTION RESUME** Wenn die Klausel `ON EXCEPTION RESUME` in der Anweisung `CREATE PROCEDURE` erscheint, setzt die Prozedur die Arbeit nach einem Fehler fort und arbeitet mit der Anweisung weiter, die nach der fehlerverursachenden Anweisung kommt.

Das genaue Verhalten für Prozeduren, die `ON EXCEPTION RESUME` verwenden, wird durch die Einstellung der Option `"on_tsql_error"` bestimmt.

Standard-Fehlerbehandlung

Im Allgemeinen gilt: Wenn eine SQL-Anweisung in einer Prozedur oder einem Trigger fehlschlägt, beendet die Prozedur oder der Trigger die Ausführung und gibt die Kontrolle mit einer entsprechenden Einstellung der Werte `SQLSTATE` und `SQLCODE` an das Anwendungsprogramm zurück. Dies gilt auch, wenn der Fehler in einer Prozedur oder einem Trigger auftrat, die direkt oder indirekt aufgerufen wurden. Bei Triggern wird der Vorgang, der den Trigger ausgelöst hat, ebenfalls rückgängig gemacht, und der Fehler wird an die Anwendung zurückgegeben.

Die folgende Demo-Prozedur zeigt, was passiert, wenn eine Anwendung die Prozedur `OuterProc` aufruft, `OuterProc` wiederum die Prozedur `InnerProc` aufruft, und diese dann auf einen Fehler trifft.

```
CREATE PROCEDURE OuterProc()  
BEGIN  
    MESSAGE 'Hello from OuterProc.' TO CLIENT;
```

```

CALL InnerProc();
MESSAGE 'SQLSTATE set to ',
        SQLSTATE, ' in OuterProc.' TO CLIENT
END;
CREATE PROCEDURE InnerProc()
BEGIN
    DECLARE column_not_found
        EXCEPTION FOR SQLSTATE '52003';
    MESSAGE 'Hello from InnerProc.' TO CLIENT;
    SIGNAL column_not_found;
    MESSAGE 'SQLSTATE set to ',
        SQLSTATE, ' in InnerProc.' TO CLIENT;
END;

CALL OuterProc();

```

Die Registerkarte "Meldungen" von Interactive SQL zeigt Folgendes an:

```

Hello from OuterProc.
Hello from InnerProc.

```

Die DECLARE-Anweisung in InnerProc deklariert einen symbolischen Namen für einen der vordefinierten SQLSTATE-Werte, die den dem Server bereits bekannten Fehlerbedingungen zugeordnet sind.

Die MESSAGE-Anweisung sendet eine Meldung an die Registerkarte "Meldungen" von Interactive SQL.

Die SIGNAL-Anweisung generiert eine Fehlerbedingung aus der Prozedur InnerProc.

Nach der SIGNAL-Anweisung in InnerProc werden keine anderen Anweisungen ausgeführt: InnerProc gibt die Kontrolle sofort an die aufrufende Umgebung zurück, die in diesem Fall die Prozedur OuterProc ist. Nach der CALL-Anweisung in OuterProc werden keine Anweisungen ausgeführt. Die Fehlerbedingung wird an die aufrufende Umgebung zurückgegeben, um von dort aus weiter behandelt zu werden. Beispiel: Interactive SQL behandelt den Fehler, indem in einem Fenster eine Beschreibung des Fehlers ausgegeben wird.

Die TRACEBACK-Funktion bietet eine Liste der Anweisungen, die ausgeführt wurden, als der Fehler auftrat. Sie können die TRACEBACK-Funktion aus Interactive SQL verwenden, indem Sie folgende Anweisung eingeben:

```
SELECT TRACEBACK( );
```

Siehe auch

- „Ausnahmeroutinen“ auf Seite 136
- „Standardmäßige Behandlung von Warnungen“ auf Seite 135
- „on_tsql_error-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Fehlerbehandlung mit ON EXCEPTION RESUME

Wenn die Klausel ON EXCEPTION RESUME in der Anweisung CREATE PROCEDURE erscheint, prüft die Prozedur die folgende Anweisung, wenn ein Fehler auftritt. Wenn die Anweisung den Fehler verarbeiten kann, wird die Prozedur weiter ausgeführt und beginnt bei der Anweisung, die nach der

fehlerverursachenden Anweisung kommt. Sie gibt die Kontrolle nicht an die aufrufende Umgebung zurück, wenn ein Fehler eintritt.

Das Verhalten für Prozeduren, die ON EXCEPTION RESUME verwenden, wird durch die Einstellung der Option "on_tsq_error" bestimmt.

Anweisungen für die Fehlerbehandlung enthalten folgende Elemente:

- IF
- SELECT @variable =
- CASE
- LOOP
- LEAVE
- CONTINUE
- CALL
- EXECUTE
- SIGNAL
- RESIGNAL
- DECLARE
- SET VARIABLE

Die folgende Demo-Prozedur zeigt, was passiert, wenn eine Anwendung die Prozedur OuterProc aufruft, OuterProc wiederum die Prozedur InnerProc aufruft, und diese dann auf einen Fehler trifft. Diese Demo-Prozeduren basieren auf den in diesem Abschnitt bereits vorher gezeigten:

```
DROP PROCEDURE OuterProc;
DROP PROCEDURE InnerProc;

CREATE PROCEDURE OuterProc()
ON EXCEPTION RESUME
BEGIN
    DECLARE res CHAR(5);
    MESSAGE 'Hello from OuterProc.' TO CLIENT;
    CALL InnerProc();
    SET res=SQLSTATE;
    IF res='52003' THEN
        MESSAGE 'SQLSTATE set to ',
            res, ' in OuterProc.' TO CLIENT;
    END IF
END;

CREATE PROCEDURE InnerProc()
ON EXCEPTION RESUME
BEGIN
    DECLARE column_not_found
    EXCEPTION FOR SQLSTATE '52003';
    MESSAGE 'Hello from InnerProc.' TO CLIENT;
    SIGNAL column_not_found;
    MESSAGE 'SQLSTATE set to ',
        SQLSTATE, ' in InnerProc.' TO CLIENT;
END;

CALL OuterProc();
```

Die Registerkarte Meldungen von Interactive SQL zeigt Folgendes an:


```
Hello from OuterProc.
Hello from InnerProc.
SQLSTATE set to 52003 in OuterProc.
```

Die gewählte Ausführungsfolge lautet:

1. OuterProc wird ausgeführt und ruft InnerProc auf.
2. In InnerProc gibt die Anweisung SIGNAL einen Fehler bekannt.
3. Die MESSAGE-Anweisung kann Fehler nicht verarbeiten, daher wird die Kontrolle an OuterProc zurückgegeben, und die Meldung wird nicht angezeigt.
4. In OuterProc weist die Anweisung nach dem Fehler den SQLSTATE-Wert der Variablen `res` zu. Da es sich dabei um eine Anweisung mit Fehlerverarbeitungskapazitäten handelt, wird die Ausführung fortgesetzt, und die OuterProc Meldung wird angezeigt.

Siehe auch

- „on_tsq_error-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standardmäßige Behandlung von Warnungen

Fehler und Warnungen werden unterschiedlich verarbeitet. Während die Standardvorgehensweise bei Fehlern darin besteht, einen Wert für die Variablen SQLSTATE und SQLCODE zu setzen sowie beim Auftritt eines Fehlers die Kontrolle an die aufrufende Umgebung zurückzugeben, besteht die Standardmaßnahme bei Warnungen darin, die Werte SQLSTATE und SQLCODE zu setzen, die Prozedur aber fortzusetzen.

Mit den folgenden Demo-Prozeduren wird die Standardverarbeitung von Warnungen gezeigt.

In diesem Fall generiert die SIGNAL-Anweisung eine Bedingung, die angibt, dass die Zeile nicht gefunden werden kann. Hier handelt es sich eher um eine Warnung als um einen Fehler.

```
DROP PROCEDURE OuterProc;
DROP PROCEDURE InnerProc;

CREATE PROCEDURE OuterProc()
BEGIN
    MESSAGE 'Hello from OuterProc.' TO CLIENT;
    CALL InnerProc();
    MESSAGE 'SQLSTATE set to ',
        SQLSTATE, ' in OuterProc.' TO CLIENT;
END;

CREATE PROCEDURE InnerProc()
BEGIN
    DECLARE row_not_found
        EXCEPTION FOR SQLSTATE '02000';
    MESSAGE 'Hello from InnerProc.' TO CLIENT;
    SIGNAL row_not_found;
    MESSAGE 'SQLSTATE set to ',
        SQLSTATE, ' in InnerProc.' TO CLIENT;
END;

CALL OuterProc();
```

Die Registerkarte Meldungen von Interactive SQL zeigt Folgendes an:

```
Hello from OuterProc.  
Hello from InnerProc.  
SQLSTATE set to 02000 in InnerProc.  
SQLSTATE set to 00000 in OuterProc.
```

Beide Prozeduren werden nach Ausgabe der Warnung fortgesetzt, wobei SQLSTATE von der Warnbedingung gesetzt wurde (02000).

Das Ausführen der zweiten MESSAGE-Anweisung in InnerProc setzt die Warnung zurück. Die erfolgreiche Ausführung einer SQL-Anweisung setzt SQLSTATE auf 00000 und SQLCODE auf 0 zurück. Wenn eine Prozedur den Fehlerstatus registrieren soll, muss sie die Zuordnung eines Werts sofort nach dem Ausführen der Anweisung vornehmen, die den Fehler oder die Warnung verursacht hat.

Siehe auch

- „Standardmäßige Behandlung von Fehlern“ auf Seite 132
- „Zeile nicht gefunden“ [*Fehlermeldungen*]

Ausnahmeroutinen

Oft will man bestimmte Fehlertypen abfangen und in einer Prozedur oder einem Trigger behandeln, anstatt den Fehler an die aufrufende Umgebung zurückzugeben. Dies erfolgt durch eine **Ausnahmeroutine**.

Sie können eine Ausnahmeroutine im EXCEPTION-Teil einer zusammengesetzten Anweisung festlegen.

Die Ausnahmeroutine wird ausgeführt, wenn ein Fehler in der zusammengesetzten Anweisung auftritt. Im Gegensatz zu Fehlern führen Warnungen nicht dazu, dass der Programmcode der Ausnahmefehler-Verarbeitungsroutine ausgeführt wird. Der Code der Verarbeitungsroutine bei Ausnahmefehlern wird auch ausgeführt, wenn ein Fehler in einer verschachtelten zusammengesetzten Anweisung oder in einer Prozedur bzw. einem Trigger erscheint, die irgendwo in der zusammengesetzten Anweisung aufgerufen wurden.

Eine Ausnahmeroutine für den Unterbrechungsfehler SQL_INTERRUPT, SQLSTATE 57014 sollte ausschließlich nicht unterbrechbare Anweisungen wie ROLLBACK und ROLLBACK TO SAVEPOINT enthalten. Wenn die Ausnahmeroutine unterbrechbare Anweisungen enthält, die aufgerufen werden, wenn die Verbindung unterbrochen wird, stoppt der Datenbankserver die Ausnahmeroutine bei der ersten unterbrechbaren Anweisung und gibt einen Fehler zurück.

Eine Ausnahmeroutine kann mit den SQLSTATE- oder SQLCODE-Spezialwerten ermitteln, warum eine Anweisung fehlgeschlagen ist. Alternativ dazu kann die Funktion ERRORMSG ohne Argument verwendet werden, um die Fehlerbedingung für SQLSTATE zurückzugeben. Nur die erste Anweisung jeder WHEN-Klausel kann diese Informationen festlegen und die Anweisung kann keine zusammengesetzte Anweisung sein.

Die Demonstrationsprozeduren, die die Verarbeitungsroutine bei Ausnahmefehlern illustrieren, basieren auf jenen in „Standardmäßige Behandlung von Fehlern“ auf Seite 132.

In diesem Beispiel verarbeitet zusätzlicher Code den Fehler in der InnerProc-Prozedur bezüglich der Spalte, die nicht gefunden werden kann.

```
DROP PROCEDURE OuterProc;
DROP PROCEDURE InnerProc;

CREATE PROCEDURE OuterProc()
BEGIN
    MESSAGE 'Hello from OuterProc.' TO CLIENT;
    CALL InnerProc();
    MESSAGE 'SQLSTATE set to ',
        SQLSTATE, ' in OuterProc.' TO CLIENT
END;
CREATE PROCEDURE InnerProc()
BEGIN
    DECLARE column_not_found
    EXCEPTION FOR SQLSTATE '52003';
    MESSAGE 'Hello from InnerProc.' TO CLIENT;
    SIGNAL column_not_found;
    MESSAGE 'Line following SIGNAL.' TO CLIENT;
    EXCEPTION
        WHEN column_not_found THEN
            MESSAGE 'Column not found handling.' TO CLIENT;
        WHEN OTHERS THEN
            RESIGNAL ;
END;

CALL OuterProc();
```

Die Registerkarte **Meldungen** von Interactive SQL zeigt Folgendes an:

```
Hello from OuterProc.
Hello from InnerProc.
Column not found handling.
SQLSTATE set to 00000 in OuterProc.
```

Die EXCEPTION-Klausel deklariert die Ausnahmeroutine. Die der EXCEPTION-Klausel folgenden Zeilen werden nur ausgeführt, wenn ein Fehler auftritt. Jede WHEN-Klausel gibt einen Namen für eine Ausnahmebedingung (deklariert mit einer DECLARE-Anweisung) sowie die Anweisung oder die Anweisungen an, die beim Eintreten des Ausnahmefehlers ausgeführt werden sollen. Die Klausel WHEN OTHERS THEN gibt die Anweisungen an, die ausgeführt werden sollen, wenn die aufgetretene Ausnahmebedingung in den vorhergehenden WHEN-Klauseln nicht vorkommt.

In diesem Beispiel übergibt die RESIGNAL-Anweisung die Ausnahmebedingung an eine Ausnahmeroutine einer höheren Ebene. RESIGNAL ist die Standardmaßnahme, wenn WHEN OTHERS THEN in einer Ausnahmeroutine nicht angegeben wurde.

Weitere Hinweise

- Der EXCEPTION-Handler wird ausgeführt, nicht die Zeilen nach der SIGNAL-Anweisung in InnerProc.
- Da bei dem aufgetretenen Fehler eine Spalte nicht gefunden werden kann, wird die für diesen Fehler vorgesehene MESSAGE-Anweisung ausgeführt und der SQLSTATE-Wert wird auf 0 zurückgesetzt (d.h. keine Fehler).

- Nachdem der Code für die Verarbeitungsroutine bei Ausnahmefehlern ausgeführt wurde, wird die Kontrolle zurück an die Prozedur OuterProc übertragen, die so weiter arbeitet, als wäre kein Fehler aufgetreten.
- Sie dürfen ON EXCEPTION RESUME nicht gemeinsam mit expliziten Verarbeitungsroutinen bei Ausnahmefehlern verwenden. Der Code für die Verarbeitungsroutine bei Ausnahmefehlern wird nicht ausgeführt, wenn ON EXCEPTION RESUME in der Prozedur einprogrammiert ist.
- Wenn es sich bei dem Fehlerbehandlungscode um eine RESIGNAL-Anweisung handelt, wird die Kontrolle an die OuterProc-Prozedur zurückgegeben, wobei SQLSTATE immer noch auf den Wert 52003 gesetzt ist. Dies ist genau so, als wäre in InnerProc kein Fehlerbehandlungscode vorhanden. Da sich in OuterProc kein Code für die Fehlerbehandlung befindet, schlägt die Prozedur fehl.

Routine bei Ausnahmefehlern und atomare zusammengesetzte Anweisungen

Wenn ein Fehler in einer atomaren zusammengesetzten Anweisung auftritt und diese Anweisung eine Ausnahmeroutine hat, die den Fehler handhabt, wird die zusammengesetzte Anweisung ohne aktive Ausnahmebedingung abgeschlossen und die Änderungen vor dem Auftreten der Ausnahmebedingung werden nicht zurückgesetzt. Wenn die Ausnahmeroutine den Fehler nicht behandelt oder einen weiteren Fehler bewirkt (einschließlich via RESIGNAL), werden die in der atomaren Anweisung durchgeführten Änderungen rückgängig gemacht.

Siehe auch

- „Zusammengesetzte Anweisungen“ auf Seite 119
- „Spalte '%1' nicht gefunden“ [*Fehlermeldungen*]
- „SQLCODE-Spezialwert“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SQLSTATE-Spezialwert“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ERRORMSG-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „RESIGNAL-Anweisung [SP]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „TRY-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Verschachtelte zusammengesetzte Anweisungen und Ausnahmeroutinen

Der Programmcode nach einer Anweisung, die einen Fehler verursacht, wird nur ausgeführt, wenn in die Prozedurdefinition eine Klausel ON EXCEPTION RESUME einprogrammiert ist.

Sie können verschachtelte zusammengesetzte Anweisungen verwenden, um mehr Kontrolle über Anweisungen zu erhalten, die nach einem Fehler ausgeführt oder nicht ausgeführt werden.

Das folgende Beispiel zeigt, wie verschachtelte zusammengesetzte Anweisungen für die Ablaufsteuerung verwendet werden können.

```
DROP PROCEDURE OuterProc;
DROP PROCEDURE InnerProc;

CREATE PROCEDURE InnerProc()
BEGIN
    DECLARE column_not_found
```

```

EXCEPTION FOR SQLSTATE VALUE '52003';
MESSAGE 'Hello from InnerProc' TO CLIENT;
SIGNAL column_not_found;
MESSAGE 'Line following SIGNAL' TO CLIENT
EXCEPTION
    WHEN column_not_found THEN
        MESSAGE 'Column not found handling' TO
            CLIENT;
    WHEN OTHERS THEN
        RESIGNAL;
END;
MESSAGE 'Outer compound statement' TO CLIENT;
END;

CALL InnerProc();

```

Die Registerkarte **Meldungen** von Interactive SQL zeigt Folgendes an:

```

Hello from InnerProc
Column not found handling
Outer compound statement

```

Nach der Anweisung SIGNAL, die den Fehler verursacht, geht die Kontrolle auf die Ausnahmeroutine für die zusammengesetzte Anweisung über, und die Meldung Verarbeitung der Ausnahmebedingung wird ausgegeben. Die Kontrolle wird dann wieder an die äußere zusammengesetzte Anweisung zurückgegeben, und die Meldung Outer compound statement wird ausgegeben.

Wenn in der inneren zusammengesetzten Anweisung ein anderer Fehler auftritt als "Spalte nicht gefunden", führt die Ausnahmeroutine die RESIGNAL-Anweisung aus. Die Anweisung RESIGNAL übergibt die Kontrolle direkt zurück an die aufrufende Umgebung, und der Rest der äußeren zusammengesetzten Anweisung wird nicht ausgeführt.

Beispiel

Dieses Beispiel zeigt die Ausgabe der sa_error_stack_trace-Systemprozedur mit RESIGNAL:

```

CREATE PROCEDURE DBA.error_reporting_procedure()
BEGIN
    SELECT *
    FROM sa_error_stack_trace();
END;

CREATE PROCEDURE DBA.proc1()
BEGIN TRY
    BEGIN TRY
        DECLARE v INTEGER = 0;
        SET v = 1 / v;
    END TRY
    BEGIN CATCH
        CALL DBA.proc2();
    END CATCH
END TRY
BEGIN CATCH
    CALL DBA.error_reporting_procedure();
END CATCH;

CREATE PROCEDURE DBA.proc2()
BEGIN
    CALL DBA.proc3();

```

```
END;  
  
CREATE PROCEDURE DBA.proc3()  
BEGIN  
    RESIGNAL;  
END;
```

Wenn die oben genannte Prozedur mit CALL proc1() aufgerufen wird, gibt sie folgende Ergebnismenge aus:

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

Dieses Beispiel zeigt die Ausgabe der sa_error_stack_trace-Systemprozedur mit RESIGNAL und der BEGIN-Anweisung:

```
CREATE PROCEDURE DBA.error_reporting_procedure()  
BEGIN  
    SELECT *  
    FROM sa_error_stack_trace();  
END;  
  
CREATE PROCEDURE DBA.proc1()  
BEGIN  
    BEGIN  
        DECLARE v INTEGER = 0;  
        SET v = 1 / v;  
        EXCEPTION WHEN OTHERS THEN  
            CALL DBA.proc2();  
        END  
    END  
    EXCEPTION WHEN OTHERS THEN  
        CALL DBA.error_reporting_procedure();  
END;  
  
CREATE PROCEDURE DBA.proc2()  
BEGIN  
    CALL DBA.proc3();  
END;  
  
CREATE PROCEDURE DBA.proc3()  
BEGIN  
    RESIGNAL;  
END;
```

Wenn die oben genannte Prozedur mit CALL proc1() aufgerufen wird, gibt sie folgende Ergebnismenge aus:

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

Siehe auch

- „TRY-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „BEGIN-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ERROR_LINE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ERROR_MESSAGE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ERROR_PROCEDURE-Funktion [Funktionstyp]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ERROR_SQLCODE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ERROR_SQLSTATE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ERROR_STACK_TRACE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „STACK_TRACE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_error_stack_trace-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_stack_trace-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Verschachtelte zusammengesetzte Anweisungen und Ausnahmeroutinen“ auf Seite 138

Beispiel: Fehlerprotokollierungsprozedur erstellen, sodass sie von einer Ausnahmeroutine aufgerufen werden kann

Sie können eine Fehlerprotokollierungsprozedur definieren, die Sie in Ausnahmeroutinen in mehreren Anwendungen verwenden können, um eine einheitliche Fehlerprotokollierung zu erhalten.

1. Erstellen Sie die folgenden Tabellen, um Fehlerinformationen zu protokollieren, sobald eine Fehlerprotokollierungsprozedur ausgeführt wird.

```
CREATE TABLE IF NOT EXISTS error_info_table (
    idx INTEGER,
    In UNSIGNED INTEGER,
    code INTEGER,
    state CHAR(5),
    err_msg CHAR(256),
    name CHAR(257),
    err_stack LONG VARCHAR,
    traceback LONG VARCHAR
);

CREATE TABLE IF NOT EXISTS error_stack_trace_table (
    idx UNSIGNED SMALLINT NOT NULL,
    stack_level UNSIGNED SMALLINT NOT NULL,
    user_name VARCHAR(128),
    proc_name VARCHAR(128),
    line_number UNSIGNED INTEGER NOT NULL,
```

```
is_resignal BIT NOT NULL, PRIMARY KEY (idx, stack_level)
);
```

2. Erstellen Sie die folgende Prozedur, die die Fehlerinformationen in den Tabellen `error_info_table` und `error_stack_trace_table` speichert und eine Meldung im Meldungsfenster des Datenbankservers ausgibt:

```
CREATE OR REPLACE PROCEDURE error_report_proc ( IN location_indicator
INTEGER )
NO RESULT SET
BEGIN
    INSERT INTO error_info_table VALUES (
        location_indicator,
        ERROR_LINE(),
        ERROR_SQLCODE(),
        ERROR_SQLSTATE(),
        ERROR_MESSAGE(),
        ERROR_PROCEDURE(),
        ERROR_STACK_TRACE(),
        TRACEBACK()
    );
    INSERT INTO error_stack_trace_table
    SELECT location_indicator, *
    FROM sa_error_stack_trace() ;
    MESSAGE 'The error message is ' || ERROR_MESSAGE() || ' and the stack
trace is ' || ERROR_STACK_TRACE()
    TYPE WARNING TO CONSOLE ;
END;
```

3. Erstellen Sie eine Prozedur wie die folgende und rufen Sie die Fehlerprotokollierungsprozedur aus der Ausnahmeroutine auf.

```
CREATE OR REPLACE PROCEDURE MyProc()
BEGIN
    DECLARE column_not_found
    EXCEPTION FOR SQLSTATE '52003';
    MESSAGE 'Hello from MyProc.' TO CLIENT;
    SIGNAL column_not_found;
    MESSAGE 'Line following SIGNAL.' TO CLIENT;
EXCEPTION
WHEN column_not_found THEN
    MESSAGE 'Column not found handling.' TO CLIENT;
    CALL error_report_proc();
END ;
```


Siehe auch

- „Ausnahmeroutinen“ auf Seite 136
- „CREATE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE PROCEDURE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „MESSAGE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „TRY-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ERROR_LINE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ERROR_SQLCODE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ERROR_SQLSTATE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ERROR_MESSAGE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ERROR_PROCEDURE-Funktion [Funktionstyp]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ERROR_STACK_TRACE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

EXECUTE IMMEDIATE in Prozeduren, Triggern, benutzerdefinierten Funktionen und Batches

Die EXECUTE IMMEDIATE-Anweisung ermöglicht den Aufbau von Anweisungen mit einer Kombination aus Literalzeichenfolgen (in Apostrophen) und Variablen. Beispiel: Die folgende Prozedur enthält eine EXECUTE IMMEDIATE-Anweisung, die eine Tabelle erstellt.

```
CREATE PROCEDURE CreateTableProcedure(  
    IN tablename CHAR(128) )  
BEGIN  
    EXECUTE IMMEDIATE 'CREATE TABLE '  
        || tablename  
        || '( column1 INT PRIMARY KEY )'  
END;
```

Die EXECUTE IMMEDIATE-Anweisung kann mit Abfragen verwendet werden, die Ergebnismengen zurückgeben. Verwenden Sie die Klausel WITH RESULT SET ON mit der EXECUTE IMMEDIATE-Anweisung, um anzuzeigen, dass die Anweisung eine Ergebnismenge zurückgibt. Das Standardverhalten ist, dass die Anweisung keine Ergebnismenge zurückgibt. 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.

Betrachten Sie die folgende Prozedur:

```
CREATE OR REPLACE PROCEDURE test_result_clause()  
BEGIN  
    EXECUTE IMMEDIATE WITH RESULT SET OFF 'SELECT 1';  
END;
```

Obwohl die Definition der Prozedur keine RESULT SET-Klausel enthält, versucht der Datenbankserver zu ermitteln, ob die Prozedur eine solche generiert. Hier gibt die EXECUTE IMMEDIATE-Anweisung an, dass keine Ergebnismenge generiert wird. Daher definiert der Datenbankserver die Prozedur ohne Ergebnismengenspalten und in der SYSPROCPARM-Systemansicht für diese Prozedur sind keine Zeilen vorhanden. Ein DESCRIBE für eine CALL-Anweisung für diese Prozedur würde keine Ergebnisspalten zurückgeben. Wenn eine Embedded SQL-Anwendung diese Informationen verwendet, um zu

entscheiden, ob sie einen Cursor öffnet oder die Anweisung ausführt, wird die Anweisung ausgeführt und dann ein Fehler zurückgegeben.

Das zweite Beispiel ist eine modifizierte Version der obigen Prozedur:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET ON 'SELECT 1';
END;
```

Hier bewirkt die WITH RESULT SET ON-Klausel, dass in der SYSPROCPARM-Systemansicht eine Zeile für diese Prozedur vorhanden ist. Der Datenbankserver weiß nicht, wie die Ergebnismenge aussieht, weil die Prozedur EXECUTE IMMEDIATE verwendet, aber er weiß, dass eine erwartet wird. Deshalb definiert der Datenbankserver in SYSPROCPARM eine Platzhalter-Ergebnismengenspalte mit dem Namen "expression" und dem SMALLINT-Datentyp, um dies anzuzeigen. Es wird nur *eine* Platzhalter-Ergebnismengenspalte erstellt. Der Server hat keine Möglichkeit, die Anzahl und den Typ der einzelnen Ergebnismengenspalten zu ermitteln, wenn eine EXECUTE IMMEDIATE-Anweisung verwendet wird. Betrachten Sie daher dieses leicht geänderte Beispiel:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET ON 'SELECT 1, 2, 3';
END;
```

Hier gibt zwar die SELECT-Anweisung eine Ergebnismenge aus drei Spalten zurück, aber der Server platziert trotzdem nur eine Zeile in der SYSPROCPARM-Systemansicht. Daher schlägt die folgende Abfrage mit SQLCODE -866 fehl:

```
SELECT * FROM test_result_clause();
```

Dies liegt daran, dass die Merkmale der Ergebnismenge zum Zeitpunkt der Ausführung nicht mit dem Platzhalter-Ergebnis in SYSPROCPARM übereinstimmen.

Um die oben genannte Abfrage auszuführen, können Sie explizit die Namen und Typen der Spalten in der Ergebnismenge angeben, und zwar folgendermaßen:

```
SELECT * FROM test_result_clause() WITH (x INTEGER, y INTEGER, z INTEGER);
```

Wenn WITH RESULT SET ON angegeben ist, verarbeitet der Datenbankserver während der Ausführung eine EXECUTE IMMEDIATE-Anweisung, die eine Ergebnismenge zurückgibt. Wenn jedoch WITH RESULT SET OFF angegeben ist oder die Klausel nicht vorhanden ist, prüft der Datenbankserver *dennoch* den Typ der ersten Anweisung im syntaktisch analysierten Zeichenfolgenargument. Wenn diese Anweisung eine SELECT-Anweisung ist, gibt er eine Ergebnismenge zurück. Daher gilt im zweiten Beispiel oben:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET OFF 'SELECT 1';
END;
```

Diese Prozedur kann über Interactive SQL aufgerufen werden. Sie können allerdings die Prozedur so ändern, dass sie einen Batch enthält und keine einzelne SELECT-Anweisung:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
```

```
EXECUTE IMMEDIATE WITH RESULT SET OFF
'begin declare v int; set v=1; select v; end';
END;
```

Dann wird bei einer CALL-Anweisung für die test_result_clause-Systemprozedur ein Fehler zurückgegeben (SQLCODE-946, SQLSTATE 09W03).

Dieses letzte Beispiel zeigt, wie es möglich ist, eine SELECT-Anweisung als Argument einer EXECUTE IMMEDIATE-Anweisung in einer Prozedur zu erstellen und die Prozedur eine Ergebnismenge zurückgeben zu lassen.

```
CREATE PROCEDURE DynamicResult(
  IN Columns LONG VARCHAR,
  IN TableName CHAR(128),
  IN Restriction LONG VARCHAR DEFAULT NULL )
BEGIN
  DECLARE Command LONG VARCHAR;
  SET Command = 'SELECT ' || Columns || ' FROM ' || TableName;
  IF ISNULL( Restriction, '' ) <> '' THEN
    SET Command = Command || ' WHERE ' || Restriction;
  END IF;
  EXECUTE IMMEDIATE WITH RESULT SET ON Command;
END;
```

Außerdem kann die Prozedur oben folgendermaßen aufgerufen werden:

```
CALL DynamicResult(
  'table_id,table_name',
  'SYSTAB',
  'table_id <= 10');
```

Dies liefert folgendes Ergebnis:

table_id	table_name
1	ISYSTAB
2	ISYSTABCOL
3	ISYSIDX
...	...

Die oben angegebene CALL-Anweisung gibt korrekt eine Ergebnismenge zurück, obwohl in der Prozedur EXECUTE IMMEDIATE verwendet wird. Einige Server-APIs, wie ODBC, verwenden die kombinierte Anforderung PREPARE-DESCRIBE-EXECUTE-OR-OPEN, die die Anweisung entweder ausführt oder öffnet, je nachdem, ob eine Ergebnismenge zurückgegeben wurde. Sollte die Anweisung geöffnet werden, kann die API oder Anwendung anschließend eine DESCRIBE CURSOR-Anweisung ausgeben, um zu ermitteln, wie die tatsächliche Ergebnismenge aussehen wird, statt sich auf den Inhalt der SYSPROCPARM-Systemansicht zu verlassen, der beim Erstellen der Prozedur festgelegt wurde. Sowohl DBISQL als auch DBISQLC verwenden diese Methode. In diesen Fällen wird eine CALL-Anweisung für die oben genannte Prozedur ohne Fehler ausgeführt. Anwendungsschnittstellen, die sich auf die DESCRIBE-Ergebnisse der Anweisung verlassen, können jedoch keine beliebige Anweisung verarbeiten.

In atomaren zusammengesetzten Anweisungen können Sie EXECUTE IMMEDIATE-Anweisungen nicht verwenden, die ein COMMIT verursachen, da COMMIT in diesem Kontext nicht zulässig ist.

Siehe auch

- „EXECUTE IMMEDIATE-Anweisung [SP]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „%1 gab eine Ergebnismenge mit einem anderen als dem erwarteten Schema zurück“ [*Fehlermeldungen*]
- „Ergebnismenge in '%1' nicht zulässig“ [*Fehlermeldungen*]

Transaktionen und Savepoints in Prozeduren, Triggern und benutzerdefinierten Funktionen

SQL-Anweisungen in einer Prozedur oder einem Trigger sind Teil der aktuellen Transaktion. Sie können mehrere Prozeduren in einer Transaktion aufrufen oder mehrere Transaktionen in eine Prozedur einbauen.

COMMIT und ROLLBACK sind in atomaren Anweisungen nicht zulässig. Trigger werden nach INSERT, UPDATE oder DELETE ausgelöst, bei denen es sich um atomare Anweisungen handelt. COMMIT und ROLLBACK sind in einem Trigger oder einer von einem Trigger aufgerufenen Prozedur nicht zulässig.

Savepoints können in einer Prozedur oder einem Trigger verwendet werden, aber eine ROLLBACK TO SAVEPOINT-Anweisung darf keinen Savepoint referenzieren, bevor der atomare Vorgang gestartet wurde. Außerdem werden alle Savepoints in einem atomaren Vorgang freigegeben, wenn der atomare Vorgang abgeschlossen wird.

Siehe auch

- „Transaktionen und Isolationsstufen“ auf Seite 881
- „Atomare zusammengesetzte Anweisungen“ auf Seite 120
- „Savepoints innerhalb von Transaktionen“ auf Seite 884

Tipps zum Schreiben von Prozeduren, Triggern, benutzerdefinierten Funktionen und Batches

Dieser Abschnitt enthält einige Hinweise zum Schreiben von Prozeduren, Triggern, benutzerdefinierten Funktionen und Batches.

Überprüfen Sie, ob Sie das Trennzeichen für SQL-Anweisungen ändern müssen.

Sie brauchen beim Schreiben von Prozeduren das Trennzeichen für Anweisungen nicht zu ändern. Wenn Sie hingegen Prozeduren und Trigger aus einer anderen Anwendung ändern und testen, muss das Trennzeichen für Anweisungen eventuell vom Semikolon in ein anderes Zeichen geändert werden.

Jede Anweisung in der Prozedur endet mit einem Semikolon. Damit andere lesende Anwendungen die CREATE PROCEDURE-Anweisung syntaktisch analysieren können, muss als Trennzeichen für Anweisungen ein anderes Zeichen als das Semikolon verwendet werden.

Wenn Sie eine Anwendung verwenden, bei der das Trennzeichen für Anweisungen geändert werden muss, können Sie beispielsweise zwei Semikola (;;) benutzen oder ein Fragezeichen (?), falls das System keine aus mehreren Zeichen bestehenden Trennzeichen zulässt.

Anweisungen müssen in der Prozedur immer getrennt werden

Jede Anweisung in der Prozedur sollte mit dem Befehlstrennzeichen beendet werden. Sie können zwar das Semikolon für die letzte Anweisung in einer Liste weglassen, aber es gehört zu einer sauberen Programmierungsweise, nach jeder Anweisung ein Semikolon zu setzen.

Die Anweisung CREATE PROCEDURE selbst enthält sowohl die RESULT-Angabe, als auch die zusammengesetzte Anweisung, die ihren Hauptteil ausmacht. Nach den Schlüsselwörtern BEGIN oder END oder nach der Klausel RESULT ist kein Semikolon erforderlich.

Vollqualifizierte Namen für Tabellen in Prozeduren benutzen

Wenn in einer Prozedur auf Tabellen Bezug genommen wird, setzen Sie vor den Tabellennamen immer den Namen des Eigentümers (Ersteller) der Tabelle.

Wenn eine Prozedur auf eine Tabelle Bezug nimmt, benutzt sie die Rollenmitgliedschaften des Prozedurerstellers zur Ermittlung der Position von Tabellen, für die kein expliziter Eigentümername angegeben wurde. Wenn beispielsweise eine Prozedur, die von Benutzer_1 erstellt wurde, die Tabelle_B referenziert und den Eigentümer der Tabelle_B nicht angibt, muss entweder die Tabelle_B vom Benutzer_1 erstellt worden sein oder der Benutzer_1 muss (direkt oder indirekt) Mitglied einer Rolle sein, die Eigentümer der Tabelle_B ist. Wenn beide Bedingungen nicht zutreffen, wird beim Aufrufen der Prozedur die Meldung `Tabelle nicht gefunden` zurückgegeben.

Sie können das Problem langer, vollqualifizierter Tabellennamen umgehen, indem Sie einen Korrelationsnamen verwenden. Weitere Hinweise zu Korrelationsnamen finden Sie unter „FROM-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)].

Datums- und Zeitangaben in Prozeduren

Wenn aus Prozeduren Datums- und Zeitangaben an die Datenbank gesendet werden, erfolgt dies in Form von Zeichenfolgen. Der Datumsteil der Zeichenfolge wird gemäß der aktuellen Einstellung der Datenbankoption "date_order" interpretiert. Da unterschiedliche Verbindungen diese Option auf unterschiedliche Werte setzen können, werden manche Zeichenfolgen vielleicht falsch in Datumswerte konvertiert, oder die Datenbank ist nicht in der Lage, die Zeichenfolge richtig als Datum zu lesen.

Sie sollten das eindeutige Datumsformat `yyyy-mm-dd` oder `yyyy/mm/dd` verwenden, wenn Sie in einer Prozedur Datumsangaben benutzen. Der Server interpretiert diese Zeichenfolgen eindeutig als Datumswerte, ganz gleich, wie die Datenbankoption "date_order" eingestellt wurde.

Eingabeargumente der Prozedur auf richtige Übergabe prüfen

Eine Möglichkeit, die Eingabeargumente zu überprüfen, ist den Wert des Parameters auf der Registerkarte **Meldungen** von Interactive SQL mithilfe der MESSAGE-Anweisung anzuzeigen. Die folgende Prozedur zeigt beispielsweise einfach den Wert des Eingabeparameters `var` :

```
CREATE PROCEDURE message_test( IN var char(40) )  
BEGIN
```

```
MESSAGE var TO CLIENT;  
END;
```

Sie können auch den Debugger benutzen, um zu prüfen, ob die Eingabeargumente der Prozedur richtig übergeben wurden.

Siehe auch

- „command_delimiter-Option [Interactive SQL]“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datentypen für Datum und Uhrzeit“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Lektion 2: Fehlerdiagnose“ auf Seite 975

Zulässige Anweisungen in Prozeduren, Triggern, Ereignissen und Batches

Die meisten SQL-Anweisungen sind in Batches zulässig, mit folgenden Ausnahmen:

- ALTER DATABASE (Syntax 3 und 4)
- CONNECT
- CREATE DATABASE
- CREATE DECRYPTED FILE
- CREATE ENCRYPTED FILE
- DISCONNECT
- DROP CONNECTION
- DROP DATABASE
- FORWARD TO
- Interactive SQL-Anweisungen wie INPUT oder OUTPUT
- PREPARE TO COMMIT
- STOP SERVER

Sie können COMMIT-, ROLLBACK- und SAVEPOINT-Anweisungen in Prozeduren, Triggern, Ereignissen und Batches unter bestimmten Einschränkungen verwenden.

Siehe auch

- „SQL-Anweisungen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Transaktionen und Savepoints in Prozeduren, Triggern und benutzerdefinierten Funktionen“ auf Seite 146

SELECT-Anweisungen in Batches

Sie können SELECT-Anweisungen in einen Batch einbeziehen. Zum Beispiel:

```
IF EXISTS( SELECT *  
           FROM SYSTAB  
           WHERE table_name='Employees' )  
THEN  
  SELECT Surname AS LastName,  
         GivenName AS FirstName  
  FROM Employees;
```

```
SELECT Surname, GivenName  
FROM Customers;  
SELECT Surname, GivenName  
FROM Contacts;  
END IF;
```

Der Alias für die Ergebnismenge ist nur in der ersten SELECT-Anweisung erforderlich, da der Server die erste SELECT-Anweisung in dem Batch verwendet, um die Ergebnismenge zu beschreiben.

Eine RESUME-Anweisung ist nach jeder Abfrage erforderlich, um die nächste Ergebnismenge abzurufen.

Verbergen des Inhalts von Prozeduren, Funktionen, Triggern, Ereignissen oder Ansichten

Um eine Anwendung und eine Datenbank zu verteilen ohne die in Prozeduren, Triggern, Ereignissen und Ansichten enthaltene Logik weiterzugeben, können Sie den Inhalt dieser Objekte mit der SET HIDDEN-Klausel der Anweisungen ALTER PROCEDURE, ALTER FUNCTION, ALTER TRIGGER und ALTER VIEW verbergen.

Voraussetzungen

Sie müssen Eigentümer des Objekts sein, das ALTER ANY OBJECT-Systemprivileg haben oder eines der folgenden Privilegien haben:

Prozeduren und Funktionen ALTER ANY PROCEDURE-Systemprivileg

Ansichten ALTER ANY VIEW-Systemprivileg

Ereignisse MANAGE ANY EVENT-Systemprivileg

Trigger

- ALTER ANY TRIGGER-Systemprivileg
- ALTER-Privileg für die Basistabelle und das CREATE ANY OBJECT-Systemprivileg
- Bei Triggern für Ansichten benötigen Sie die Systemprivilegien ALTER ANY TRIGGER und ALTER ANY VIEW

Kontext und Bemerkungen

Die SET HIDDEN-Klausel verschleiert den Inhalt der zugeordneten Objekte und macht sie unlesbar, wobei die Nutzbarkeit der Objekte nach wie vor möglich ist. Des weiteren können Sie die Objekte in eine andere Datenbank entladen und neu laden.

Die Änderung kann nicht rückgängig gemacht werden und löscht den Originaltext des Objekts. Deshalb ist eine Sicherung des Originaltextes dieses Objekts außerhalb der Datenbank notwendig.

Beim Debugging mithilfe des Debuggers wird weder die Definition der Prozedur angezeigt noch zeigen die Prozedurprofilinformationen den Quelltext an.

Hinweis

Wenn die preserve_source_format-Datenbankoption auf "On" gesetzt wurde, speichert der Datenbankserver die formatierte Quelle aus CREATE- und ALTER-Anweisungen für Prozeduren, Ansichten, Trigger und Ereignisse und platziert sie in der source-Spalte der entsprechenden Systemansicht. In diesem Fall wird sowohl die Objektdefinition als auch die Quellendefinition ausgeblendet.

Die Einstellung "On" für die preserve_source_format-Datenbankoption verhindert jedoch *nicht*, dass die ursprüngliche Quellendefinition des Objekts durch die SET HIDDEN-Klausel gelöscht wird.

Aufgabe

- Verwenden Sie die geeignete ALTER-Anweisung mit der SET HIDDEN-Klausel.

Option	Aktion
Ausblenden eines einzelnen Objekts	Führen Sie die geeignete ALTER-Anweisung mit der SET HIDDEN-Klausel aus, um eine Prozedur, einen Trigger, ein Ereignis oder eine Ansicht zu verbergen.
Ausblenden aller Objekte eines bestimmten Typs	Führen Sie die geeignete ALTER-Anweisung mit der SET HIDDEN-Klausel in einer Schleife aus, um alle Prozeduren, Trigger, Ereignisse oder Ansichten zu verbergen.

Ergebnisse

Ein automatisches Festschreiben wird ausgeführt. Die Objekt Definition ist nicht mehr sichtbar. Das Objekt kann jedoch immer noch direkt referenziert werden und steht für die Verwendung während der Abfrageverarbeitung zur Verfügung.

Beispiel

Führen Sie die folgenden Schleife aus, um alle Prozeduren zu verbergen:

```
BEGIN
  FOR hide_lp as hide_cr cursor FOR
    SELECT proc_name, user_name
    FROM SYS.SYSPROCEDURE p, SYS.SYSUSER u
    WHERE p.creator = u.user_id
    AND p.creator NOT IN (0,1,3)
  DO
    MESSAGE 'altering ' || proc_name;
    EXECUTE IMMEDIATE 'ALTER PROCEDURE "' ||
      user_name || '."' || proc_name
      || ' " SET HIDDEN'
  END FOR
END;
```


Siehe auch

- „ALTER FUNCTION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER PROCEDURE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TRIGGER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER EVENT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „preserve_source_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Performanceverbesserungen, Diagnosen und Monitoring

In diesem Abschnitt wird beschrieben, wie Sie die Performance verbessern, Performance-Probleme diagnostizieren und die Performance überwachen können.

Tools für Performanceüberwachung und Diagnose

Dieser Abschnitt bietet Informationen über die Verwendung von SQL Anywhere-Analysertools, um die Datenbankperformance zu analysieren und zu korrigieren.

SQL Anywhere stellt mehrere Diagnosetools bereit, um Performanceprobleme bei Produktionsdatenbanken zu ermitteln. Einige dieser Tools beruhen auf der Infrastruktur für die **Diagnoseprotokollierung**. Dabei handelt es sich um ein System von Tabellen, Dateien und anderen Komponenten, die Diagnosedaten erfassen und speichern. Sie können die Diagnoseprotokollierungsdaten dann verwenden, um Diagnose- und Überwachungsaufgaben wie etwa die **Anwendungsprofilerstellung** durchzuführen.

Es gibt mehrere Methoden zur Analyse der SQL Anywhere-Performance-Daten, und zwar:

Tool	Details
Diagnoseprotokollierung	<ul style="list-style-type: none"> ● Assistent für die Datenbankprotokollierung Mit diesem Assistenten kann der Typ der erfassten Performance-Daten beliebig angepasst werden. Mit diesem Assistenten können Sie die Performance von bestimmten Benutzern oder Aktivitäten überwachen. ● Indexberater Diese Funktion analysiert die Indizes in der Datenbank und gibt Empfehlungen für Verbesserungsmaßnahmen. Sie können auf dieses Tool im Modus für die Anwendungsprofilerstellung zugreifen, aber es steht auch als Standalone-Tool für einzelne Abfragen in Interactive SQL zur Verfügung. ● Anwendungsprofilerstellung Die Anwendungsprofilerstellung generiert Daten, die Sie verwenden können, um zu ermitteln, wie Anwendungen mit der Datenbank interagieren, und Performance-Probleme zu identifizieren und zu beseitigen. Zwei Methoden stehen für die Generierung von Profildaten zur Verfügung: <ul style="list-style-type: none"> ○ Assistent für die Anwendungsprofilerstellung Dieser Assistent, der im Sybase Central-Modus "Anwendungsprofil" zur Verfügung gestellt wird, bietet eine vollautomatische Methode zur Performance-Prüfung. Abschließend gibt der Assistent Empfehlungen für Verbesserungsmaßnahmen. Dieses Tool ist für Entwicklungsumgebungen empfohlen und für die Protokollierung der Abfrageperformance optimiert. ○ Prozedurprofilerstellung Mithilfe dieser Funktion können Sie ermitteln, wie lange die Ausführung von Prozeduren, benutzerdefinierten Funktionen, Ereignissen, Systemtriggern und Triggern dauert. Die Prozedurprofilerstellung steht in Sybase Central als Anwendungsprofilerstellung-Modus zur Verfügung. <p>Sie können auch Systemprozeduren verwenden, um die Prozedurprofilerstellung zu implementieren.</p>

Tool	Details
Andere Tools	<ul style="list-style-type: none"> • Anforderungsprotokollierung Die Anforderungsprotokollierung zeichnet die einzelnen von einer Anwendung empfangenen Anforderungen bzw. die an eine Anwendung gesendeten Antworten in einer Textdatei auf. Das ist besonders hilfreich, um herauszufinden, welche Aktion die Anwendung vom Datenbankserver verlangt. Die Anforderungsprotokollierung ist auch ein guter Ausgangspunkt für die Performanceanalyse einer bestimmten Anwendung, wenn es nicht offensichtlich ist, ob der Datenbankserver oder der Client der Verursacher des Fehlers ist. Sie können eine Anforderungsprotokollierung durchführen, um die spezielle Anforderung an den Datenbankserver zu ermitteln, die für Probleme verantwortlich sein könnte. Das Anforderungslog enthält eine Teilmenge der Daten, die von der Diagnoseprotokollierung und der Ereignisprotokollierung bereitgestellt werden. • Ereignisprotokollierung Die Ereignisprotokollierung wird für Produktionsumgebungen empfohlen und bietet detaillierte Steuerungsmöglichkeiten hinsichtlich der protokollierten Informationen. Sie können benutzer- und systemdefinierte Trace-Ereignisse sowohl für den Datenbankserver als auch für Ihre Anwendung protokollieren und die Trace-Ereignisse so anpassen, dass sich Performanceprobleme ermitteln lassen. • Ausführungspläne Mit dieser Funktion können Sie einen Ausführungsplan prüfen, um auf Informationen in der Datenbank zuzugreifen, die mit einer Anweisung zusammenhängen. Sie können den Ausführungsplan in Interactive SQL bzw. mit den SQL-Funktionen anzeigen. Sie können einen Ausführungsplan in mehreren unterschiedlichen Formaten abrufen und der Plan kann gespeichert werden.

Hinweis

In dieser Dokumentation werden die Begriffe **Anwendungsprofilerstellung** und **Diagnoseprotokollierung** synonym verwendet. Die Diagnoseprotokollierung ist eine Erweiterung der Anwendungsprofilerstellung.

Hinweis

Sie können Sybase Central benutzen, um eine Verbindung zu einem Datenbankserver der Version 9 herzustellen. Das Fensterlayout von Sybase Central wird allerdings dem Layout der Version 9 angepasst, sodass der Modus für die Anwendungsprofilerstellung nicht zur Verfügung steht. Hinweise, wie Sie den Indexberater in Sybase Central finden und ihn verwenden, finden Sie in Ihrer Version 9-Dokumentation.

Siehe auch

- „Anwendungsprofilerstellung“ auf Seite 156
- „Diagnoseprotokollierung“ auf Seite 170
- „Ereignisprotokollierung“ [*SQL Anywhere Server - Datenbankadministration*]
- „Anforderungs-Trace-Analysen ausführen“ auf Seite 192
- „Indexberater“ auf Seite 164
- „Prozedurprofilerstellung im Modus für die Anwendungsprofilerstellung“ auf Seite 158
- „Prozedurprofilerstellung mit Systemprozeduren“ auf Seite 197
- „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ auf Seite 341

Anwendungsprofilerstellung

Die Anwendungsprofilerstellung generiert Daten, die Sie verwenden können, um zu ermitteln, wie Anwendungen mit der Datenbank interagieren, und Performance-Probleme zu identifizieren und zu beseitigen. Es stehen zwei Methoden für die Generierung von Profilerstellungsinformationen zur Verfügung: Eine automatisierte Methode unter Verwendung des **Assistenten für die Anwendungsprofilerstellung**, oder indem Sie die Tools und Funktionen verwenden, die im Anwendungsprofil-Modus von Sybase Central bereit gestellt werden.

Der **Assistent für die Anwendungsprofilerstellung** wird unter Windows Mobile nicht unterstützt, der **Assistent für die Datenbankprotokollierung** hingegen schon. Sie können weder automatisch eine Protokollierungsdatenbank von einem Windows Mobile-Gerät aus erstellen, noch können Sie ein Protokoll für die lokale Datenbank auf einem Windows Mobile-Gerät erstellen. Sie müssen die Protokollierung vom Windows Mobile-Gerät in einer Kopie der Windows Mobile-Datenbank durchführen, die auf einem Datenbankserver auf einem PC läuft.

- **Automatisierte Anwendungsprofilerstellung** Verwenden Sie den **Assistenten für die Anwendungsprofilerstellung** in Sybase Central, um übliche Performance-Probleme zu identifizieren. Der Assistent ermöglicht es Ihnen, die Typen von Aktivitäten festzulegen, deren Profil erstellt werden soll, und liefert nach seiner Ausführung Empfehlungen zur Verbesserung der Datenbank-Performance. Der Indexberater wurde ebenfalls in den Assistenten für die **Anwendungsprofilerstellung** integriert und verwendet die Daten, um Index-Verbesserungen zu empfehlen.

Die automatisierte Methode ist besonders für Umgebungen mit wenigen Datenbankverbindungen geeignet, oder bei denen eine hoch entwickelte Profilerstellung nicht erforderlich ist.

- **Erweiterte Anwendungsprofilerstellung mithilfe der Diagnoseprotokollierung** Verwenden Sie den **Assistenten für die Datenbankprotokollierung**, um die Daten, die während einer Protokollierungssitzung zurückgegeben werden, anzupassen und ihren Speicherort festzulegen. Sie können auch die Befehlszeile verwenden, um angepasste Protokollierungsdaten zurückzugeben und zu speichern. Sie können die Aktivitäten, deren Profil erstellt werden soll, festlegen und gezielt spezifische Probleme einbeziehen. Sie können beispielsweise folgende Aktivitäten prüfen: bestimmte Anweisungen, die durch den Datenbankserver ausgeführt wurden, verwendete Abfragepläne, Deadlocks, Verbindungen, die einander blockieren, und Performance-Statistiken.

Eine erweiterte Methode wird für Umgebungen empfohlen, in der die Datenbank eine hohe Arbeitslast hat, oder wo eine hoch entwickelte Profilerstellung erforderlich ist, um eine Problemdiagnose zu erstellen. Durch Anpassen der Protokollierungssitzung können Sie den Umfang der Protokollierung auf bestimmte Aktivitäten reduzieren, und Sie können die Protokollierungsdaten in eine entfernte Datenbank umleiten. Beide Maßnahmen reduzieren die Arbeitslast der Datenbank, deren Profil erstellt wird.

Siehe auch

- [„Diagnoseprotokollierung“ auf Seite 170](#)

Verwenden des Assistenten für die Anwendungsprofilerstellung

Sie können in Sybase Central den **Assistenten für die Anwendungsprofilerstellung** verwenden, um eine diagnostische Protokollierungssitzung für die Erstellung von Anwendungsprofilen durchzuführen. Der Assistent sammelt Daten darüber, wie Ihre Anwendungen mit der Datenbank interagieren, ermöglicht Ihnen den Zugriff auf die Daten und gibt ggf. Indizierungsempfehlungen.

Voraussetzungen

Sie benötigen die DIAGNOSTICS-Systemrolle sowie die Systemprivilegien **MANAGE PROFILING**, **SERVER OPERATOR** und **SELECT ANY TABLE**.

Kontext und Bemerkungen

Der **Assistent für die Anwendungsprofilerstellung** kann nicht benutzt werden, um eine Protokollierungssitzung für eine Datenbank zu erstellen, die unter Windows CE läuft. Sie müssen den **Assistenten für die Datenbankprotokollierung** benutzen.

Wenn Sie den **Assistenten für die Anwendungsprofilerstellung** in Sybase Central benutzen, erstellt der Assistent automatisch eine Protokollierungsdatenbank mit dem gleichen Namen, den Sie im Assistenten für die Analysedatei festgelegt haben.

Weitere Informationen über die vom **Assistenten für die Anwendungsprofilerstellung** gelieferten Indexierungsempfehlungen finden Sie unter „[Empfehlungen des Indexberaters](#)“ auf Seite 166.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16-Plug-Ins** her.
2. Klicken Sie auf **Modus » Anwendungsprofil**.

Wenn der **Assistent für die Anwendungsprofilerstellung** nicht eingeblendet wird, klicken Sie auf **Anwendungsprofil » Anwendungsprofil-Assistenten öffnen**.

3. Befolgen Sie die Anweisungen des **Assistenten für die Anwendungsprofilerstellung**.

Der Assistent:

- Erstellt eine lokale Datenbank, um die Diagnoseprotokollierungsinformationen aufzunehmen
- Startet den Netzwerkserver
- Startet eine Protokollierungssitzung
- Fordert Sie auf, die Anwendung auszuführen, deren Profil Sie erstellen wollen

Ergebnisse

Der Assistent gibt seine Ergebnisse zurück und Sie können die während der Protokollierungssitzung erfassten Daten prüfen.

Siehe auch

- [„Verwenden des Assistenten für die Anwendungsprofilerstellung“ auf Seite 157](#)
- [„Daten der Protokollierungssitzung“ auf Seite 171](#)
- [So lesen Sie die Ergebnisse der Prozedurprofilerstellung auf Seite 162](#)
- [„Erstellen einer Diagnoseprotokollierungssitzung \(Sybase Central\)“ auf Seite 186](#)

Prozedurprofilerstellung im Modus für die Anwendungsprofilerstellung

Dieser Abschnitt beschreibt, wie Sie den Modus "Anwendungsprofil" in Sybase Central verwenden, um Prozedurprofilerstellungen durchzuführen. Es ist die empfohlene Methode, um auf die Ergebnisse einer Prozedurprofilerstellung zuzugreifen. Sie können allerdings auch SQL-Befehle verwenden, um die Prozedurprofilerstellung durchzuführen.

Prozedurprofile zeigen Ihnen, wie lange es dauert, um Prozeduren, benutzerdefinierte Funktionen, Ereignisse, Systemtrigger und Trigger auszuführen. Sie können die Ausführungszeiten für diese Objekte auch Zeile für Zeile prüfen, sobald sie während der Profilerstellung ausgeführt wurden. Anhand der Daten, die in den Ergebnissen der Prozedurprofilerstellung bereitgestellt werden, können Sie feststellen, welche Objekte detailliert abgestimmt werden sollten, um die Performance der Datenbank zu verbessern.

Prozedurprofile unterstützen Sie auch bei der Analyse bestimmter Datenbankprozeduren (wie gespeicherte Prozeduren, Funktionen, Ereignisse und Trigger), die aufgrund der Anforderungsprotokollierung als kostenträchtig eingeschätzt werden. Damit können Sie kostenträchtige verborgene Prozeduren aufspüren wie Trigger, Ereignisse und verschachtelte Aufrufe von gespeicherten Prozeduren. Überdies können Sie damit potentielle Probleme innerhalb der Struktur einer Prozedur herausfinden.

Ergebnisse von Prozedurprofilerstellungen werden durch den Datenbankserver im Speicher abgelegt. Profilerstellungsinformationen sind kumulativ und auf 1 ms genau.

Siehe auch

- [„Prozedurprofilerstellung mit Systemprozeduren“ auf Seite 197](#)

Prozedurprofilerstellung aktivieren

Wenn die Prozedurprofilerstellung aktiviert ist, kann der Datenbankserver Profilinformationen sammeln, bis die Profilerstellung wieder deaktiviert oder der Datenbankserver heruntergefahren wird.

Voraussetzungen

Sie müssen den Netzwerkserver (dbsrv) ausführen, nicht den Personal Server (dbeng).

Sie benötigen die DIAGNOSTICS-Systemrolle sowie die Systemprivilegien MANAGE PROFILING, SERVER OPERATOR und SELECT ANY TABLE.

Kontext und Bemerkungen

Hinweis

Alle Profilerstellungsinformationen werden gelöscht, wenn der Datenbankserver heruntergefahren wird. Um Profilerstellungsinformationen zu exportieren, verwenden Sie die Systemprozedur `sa_procedure_profile`.

Sie können keine SQL-Anweisungen verwenden, um vom Datenbankserver gespeicherte Profilerstellungsinformationen abzurufen. Profilerstellungsinformationen werden in speicherresidenten Datenbankserver-Datenstrukturen aufbewahrt.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Wählen Sie im linken Fensterausschnitt die Datenbank aus.
3. Klicken Sie auf **Modus » Anwendungsprofil**.

Wenn der **Assistent für die Anwendungsprofilerstellung** nicht eingeblendet wird, klicken Sie auf **Anwendungsprofil » Anwendungsprofil-Assistenten öffnen**.

4. Befolgen Sie die Anweisungen des **Assistenten für die Anwendungsprofilerstellung**.

Klicken Sie auf der Seite **Profilerstellungsoptionen** auf **Ausführungszeit von gespeicherten Prozeduren, Funktionen, Triggern oder Ereignissen**.

Wenn Sie in einen anderen Modus wechseln, wird eine Eingabeaufforderung eingeblendet, in der Sie angeben können, ob Sie das Erfassen von Prozedurprofilerstellungsdaten stoppen wollen. Klicken Sie auf **Nein**, um in einem anderen Modus weiterzuarbeiten, während die Profilerstellung fortgesetzt wird.

Ergebnisse

Die Prozedurprofilerstellung wird aktiviert und der Datenbankserver sammelt Profilerstellungsinformationen.

Siehe auch

- „Prozedurprofilerstellung zurücksetzen“ auf Seite 159
- „Prozedurprofilerstellung deaktivieren“ auf Seite 161
- „sa_procedure_profile-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Ergebnisse der Prozedurprofilerstellung“ auf Seite 162

Prozedurprofilerstellung zurücksetzen

Setzen Sie die Prozedurprofilerstellung zurück, wenn Sie vorhandene Profilinformationen über Prozeduren, Funktionen, Ereignisse und Trigger löschen wollen. Sie müssen die Prozedurprofilerstellung nicht deaktivieren, um sie zurücksetzen zu können.

Voraussetzungen

Sie müssen den Netzwerkserver (dbsrv) ausführen, nicht den Personal Server (dbeng).

Sie benötigen die DIAGNOSTICS-Systemrolle sowie die Systemprivilegien MANAGE PROFILING, SERVER OPERATOR und SELECT ANY TABLE.

Kontext und Bemerkungen

Hinweis
Wenn die Prozedurprofilerstellung aktiviert ist, wird sie durch das Zurücksetzen nicht gestoppt, und wenn die Prozedurprofilerstellung deaktiviert ist, wird sie durch das Zurücksetzen nicht gestartet.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Wählen Sie im linken Fensterausschnitt die Datenbank aus.
3. Klicken Sie auf **Modus » Anwendungsprofil**.

Wenn der **Assistent für die Anwendungsprofilerstellung** eingeblendet wird, klicken Sie auf **Abbrechen**.

4. Wählen Sie einen der folgenden Abschnitte:

Option	Aktion
Wenn die Prozedurprofilerstellung aktiviert ist	Klicken Sie im Fensterausschnitt Details zur Anwendungsprofilerstellung auf die Datenbank, und dann auf Profilerstellungseinstellungen für ausgewählte Datenbanken anzeigen .
Wenn die Prozedurprofilerstellung nicht aktiviert ist	Rechtsklicken Sie im linken Fensterausschnitt auf die Datenbank und klicken Sie auf Eigenschaften .

5. Klicken Sie auf die Registerkarte **Profilerstellungseinstellungen**.
6. Klicken Sie auf **Jetzt zurücksetzen**.
7. Klicken Sie auf **OK**.

Ergebnisse

Die Prozedurprofilerstellung wird zurückgesetzt und vorhandene Profilinformationen werden gelöscht.

Siehe auch

- „Prozedurprofilerstellung aktivieren“ auf Seite 158
- „Prozedurprofilerstellung deaktivieren“ auf Seite 161
- „Ergebnisse der Prozedurprofilerstellung“ auf Seite 162

Prozedurprofilerstellung deaktivieren

Deaktivieren Sie die Prozedurprofilerstellung, wenn die Erfassung von Profilerstellungsinformationen für Prozeduren, Trigger und Funktionen abgeschlossen ist. Sie können die bereits erfassten Profilerstellungsinformationen löschen, wenn Sie Ihre Analyse bereits abgeschlossen haben.

Voraussetzungen

Sie benötigen die DIAGNOSTICS-Systemrolle sowie die Systemprivilegien MANAGE PROFILING, SERVER OPERATOR und SELECT ANY TABLE.

Kontext und Bemerkungen

Wenn Sie die Profilerstellungsinformationen nicht löschen, stehen sie weiterhin im Modus für die **Anwendungsprofilerstellung** in Sybase Central zur Verfügung, auch wenn die Prozedurprofilerstellung deaktiviert wurde.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16-Plug-Ins** her.
2. Wählen Sie im linken Fensterausschnitt die Datenbank aus.
3. Klicken Sie auf **Modus » Anwendungsprofil**.

Wenn der **Assistent für die Anwendungsprofilerstellung** eingeblendet wird, klicken Sie auf **Abbrechen**.

4. (Optional) Löschen Sie die bereits erfassten Profilerstellungsinformationen.
 - a. Wählen Sie im Fensterausschnitt **Details zur Anwendungsprofilerstellung** die gewünschte Datenbank und klicken Sie auf die Option **Profilerstellungseinstellungen für ausgewählte Datenbanken anzeigen**.
 - b. Klicken Sie auf die Registerkarte **Profilerstellungseinstellungen**.
 - c. Klicken Sie auf **Jetzt löschen**.
 - d. Klicken Sie auf **OK**.
5. Im Fensterausschnitt **Details zur Anwendungsprofilerstellung** klicken Sie auf **Sammeln von Profilinformatoren in ausgewählten Datenbanken stoppen**.

Ergebnisse

Die Prozedurprofilerstellung wird deaktiviert und die Profilerstellungsinformationen werden gelöscht, sofern angegeben.

Siehe auch

- „Prozedurprofilerstellung aktivieren“ auf Seite 158
- „Prozedurprofilerstellung zurücksetzen“ auf Seite 159
- „Ergebnisse der Prozedurprofilerstellung“ auf Seite 162

Ergebnisse der Prozedurprofilerstellung

So lesen Sie die Ergebnisse der Prozedurprofilerstellung

Die Registerkarte **Ergebnisse der Profilerstellung** liefert eine nach Typen gruppierte Zusammenfassung für alle Objekte, die innerhalb der Datenbank ausgeführt wurden, seitdem Sie die Prozedurprofilerstellung gestartet haben. Dazu gehören folgende Details:

Spalte	Beschreibung
Name	Der Name des Objekts
Eigentümer	Der Eigentümer des Objekts
Tabelle oder Tabellenname	Tabelle, zu der ein Trigger gehört (die Spalte wird nur auf der Profil-Registerkarte der Datenbank angezeigt)
Ereignis	Enthält den Objekttyp, z.B. eine Prozedur
Typ	Der Triggertyp für Systemtrigger. Dieser Wert kann "Aktualisieren" oder "Löschen" lauten.
# Ausführen.	Zeigt an, wie oft jedes Objekt aufgerufen wurde
ms (Millisekunden)	Die Gesamtausführungszeit für jedes Objekt

Diese Spalten und ihre Inhalte können je nach Objekttyp unterschiedlich sein.

Wenn Sie auf ein bestimmtes Objekt, wie z.B. eine Prozedur, doppelklicken, werden spezifische Details zu diesem Objekt auf der Registerkarte **Ergebnisse der Profilerstellung** eingeblendet. Dazu gehören folgende Details:

Spalte	Beschreibung
Ausführen	Wie oft die Zeile des Codes im Objekt ausgeführt wurde
Millisekunden	Die gesamte Zeit für die Ausführung einer Zeile

Spalte	Beschreibung
%	Der Anteil an der gesamten Zeit für die Ausführung einer Zeile
Zeile	Die Nummer der Zeile innerhalb des Objekts
Quelle	Der ausgeführte Code

Zeilen, deren Ausführungszeiten im Verhältnis zu anderen Zeilen im Code länger sind, sollten analysiert werden, um festzustellen, ob es effizientere Möglichkeiten gibt, die gleiche Funktionalität zu erreichen. Wenn Sie auf Informationen der Anwendungsprofilerstellung zugreifen möchten, müssen Sie mit der Datenbank verbunden sein, die Profilerstellung aktiviert haben und das **MANAGE PROFILING**-Systemprivileg haben.

Siehe auch

- „Prozedurprofilerstellung aktivieren“ auf Seite 158

Ergebnisse der Prozedurprofilerstellung analysieren

Sie können Ergebnisse der Profilerstellung für gespeicherte Prozeduren, benutzerdefinierte Funktionen, Trigger, Systemtrigger und Ereignisse in Ihrer Datenbank anzeigen.

Voraussetzungen

Sie benötigen die **DIAGNOSTICS**-Systemrolle sowie die Systemprivilegien **MANAGE PROFILING**, **SERVER OPERATOR** und **SELECT ANY TABLE**.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Prozedurprofilerstellung aktivieren:
 - a. Klicken Sie auf **Modus » Anwendungsprofil**.
Wenn der **Assistent für die Anwendungsprofilerstellung** nicht eingeblendet wird, klicken Sie auf **Anwendungsprofil » Anwendungsprofil-Assistenten öffnen**.
 - b. Befolgen Sie die Anweisungen des **Assistenten für die Anwendungsprofilerstellung**.
Klicken Sie auf der Seite **Profilerstellungsoptionen** auf **Ausführungszeit von gespeicherten Prozeduren, Funktionen, Triggern oder Ereignissen**.
Wenn Sie in einen anderen Modus wechseln, wird eine Eingabeaufforderung eingeblendet, in der Sie angeben können, ob Sie das Erfassen von Prozedurprofilerstellungsdaten stoppen wollen. Klicken Sie auf **Nein**, um in einem anderen Modus weiterzuarbeiten, während die Profilerstellung fortgesetzt wird.
3. Im linken Fensterausschnitt doppelklicken Sie auf eines der folgenden Objekte: **Trigger**, **Systemtrigger**, **Prozeduren und Funktionen** oder **Ereignisse**.

4. Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **Ergebnisse der Profilerstellung**.

In einer Liste werden alle Objekte des gewählten Typs aufgeführt, die ausgeführt wurden, seitdem Sie die Prozedurprofilerstellung aktiviert haben.

Ein erwartetes Objekt fehlt möglicherweise, weil es nicht ausgeführt wurde. Oder es wurde ausgeführt, aber die Ergebnisse wurden noch nicht aktualisiert. Drücken Sie F5, um die Liste zu aktualisieren.

Falls die Liste mehr Objekte enthält als erwartet, liegt es daran, dass ein Objekt andere Objekte aufrufen kann und dass daher mehr Objekte aufgelistet werden als ausdrücklich aufgerufen wurden.

5. Um detaillierte Profilerstellungsergebnisse für ein bestimmtes Objekt anzuzeigen, doppelklicken Sie auf das Objekt auf der Registerkarte **Ergebnisse der Profilerstellung**.

Ergebnisse

Die Details im rechten Fensterausschnitt werden durch die detaillierten Profilinformatoren für das Objekt ersetzt.

Indexberater

Die Auswahl der passenden Gruppe von Indizes kann die Datenbank-Performance verbessern. Der SQL Anywhere-Indexberater unterstützt Sie bei der Auswahl der Indizes, indem er Empfehlungen für die Gruppe von Indizes gibt, die am besten zu Ihrer Datenbank passt.

Sie können den Indexberater mit Interactive SQL für eine einzelne Abfrage einsetzen, aber auch mithilfe des Modus für die Anwendungsprofilerstellung in Sybase Central für eine Datenbank. Bei der Analyse einer Datenbank verwendet der Indexberater eine Protokollierungssitzung, um Daten zu erfassen und Empfehlungen abzugeben. Um festzustellen, welche Indizes zu verbesserten Ausführungsplänen führen, schätzt der Indexberater die Kosten der Abfrageausführung unter Verwendung dieser Indizes. Der Indexberater bewertet Mehrfach-Spaltenindizes und Einzel-Spaltenindizes und überprüft die Auswirkungen von Clustered- bzw. Nonclustered-Indizes.

Der Indexberater analysiert eine Datenbank oder eine einzelne Abfrage, indem er mögliche Indizes erzeugt und ihre Auswirkung auf die Performance ermittelt. Um die Auswirkung der möglichen Indizes zu untersuchen, optimiert der Indexberater die Abfragen immer wieder mit unterschiedlichen Indexgruppen. Er führt die Abfragen nicht aus.

Sie können alle berichteten Fehler analysieren, die bei Ausführen des Indexberaters aufgetreten sind, indem Sie die Spalten `sql_code` und `log_message` in der Tabelle `"DBO"."ix_consultant_log"` konsultieren.

Siehe auch

- „Indizes“ auf Seite 27
- „Indizes“ auf Seite 27
- „Anwendungsprofilerstellung“ auf Seite 156
- „Anwendungsprofilerstellung“ auf Seite 156
- „Empfehlungen des Indexberaters“ auf Seite 166

Abrufen von Empfehlungen des Indexberaters für eine Abfrage

Sie können auf die Empfehlungen des Indexberaters für eine Abfrage in Interactive SQL zugreifen.

Voraussetzungen

Sie müssen die DIAGNOSTICS-Systemrolle oder alle folgenden Systemprivilegien haben:

- SELECT ANY TABLE
- INSERT ANY TABLE
- DELETE ANY TABLE
- UPDATE ANY TABLE

Aufgabe

1. Stellen Sie in Interactive SQL eine Verbindung mit der Datenbank her.
2. Geben Sie im Fensterausschnitt **SQL-Anweisungen** eine SELECT-, UPDATE- oder DELETE-Anweisung ein:
3. Klicken Sie auf **Extras » Indexberater** und befolgen Sie die Anweisungen.

Ergebnisse

Die Empfehlungen des Indexberaters werden im Indexberater-Assistenten im Fensterausschnitt **Übersicht** angezeigt.

Abrufen von Empfehlungen des Indexberaters für eine Datenbank

Um Empfehlungen des Indexberaters für eine komplette Datenbank zu erhalten, benutzen Sie den Anwendungsprofilmodus in Sybase Central.

Voraussetzungen

Sie müssen die DIAGNOSTICS-Systemrolle oder alle folgenden Systemprivilegien haben:

- SELECT ANY TABLE
- INSERT ANY TABLE
- DELETE ANY TABLE
- UPDATE ANY TABLE

Außerdem müssen Sie die Systemprivilegien MANAGE PROFILING, SERVER OPERATOR und SELECT ANY TABLE haben.

Kontext und Bemerkungen

Der Indexberater benötigt Profilerstellungsdaten, um Empfehlungen abgeben zu können. Das folgende Verfahren stellt eine rasche Methode zur Sammlung von Daten dar, die vom **Assistent für die Anwendungsprofilerstellung** gesammelt werden, um eine Empfehlung abzugeben. Wenn Sie allerdings bereits über Anwendungsprofilierungsdaten verfügen (z.B. wenn Sie für Ihre Datenbank mit dem

Assistenten für die Datenbankprotokollierung) bereits Profildaten gesammelt haben, können Sie den Indexberater auch mit der Protokollierungsdatenbank ausführen, die Sie erstellt haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie auf **Modus » Anwendungsprofil**.

Falls der **Assistent für die Anwendungsprofilerstellung** nicht erscheint, klicken Sie auf **Anwendungsprofil » Anwendungsprofil-Assistenten öffnen** und befolgen die Anweisungen des Assistenten bis zum Abschluss.

3. Befolgen Sie die Anweisungen des **Assistenten für die Anwendungsprofilerstellung**.
4. Klicken Sie in Sybase Central auf **Anwendungsprofil » Indexberater auf Protokollierungsdatenbank ausführen**.
5. Befolgen Sie die Anweisungen des **Indexberater-Assistenten**.

Ergebnisse

Die Empfehlungen des Indexberaters werden angezeigt.

Empfehlungen des Indexberaters

Bevor Sie eine Protokollierungssitzung analysieren, fragt der Indexberater, welche Arten von Empfehlungen er liefern soll:

- **Clustered-Indizes empfehlen** Wenn diese Option ausgewählt ist, analysiert der Indexberater die Auswirkungen von Clustered- und Unclustered-Indizes.

Verglichen mit Unclustered-Indizes können richtig ausgewählte Clustered-Indizes die Performance signifikant steigern, aber Sie müssen die Tabelle reorganisieren (mit der REORGANIZE TABLE-Anweisung), damit sie wirksam werden. Außerdem dauert die Analyse länger, wenn die Auswirkungen von Clustered-Indizes einbezogen werden.

- **Bestehende Sekundärindizes beibehalten** Der Indexberater kann seine Analyse durchführen, indem er entweder die bestehenden Sekundärindizes in der Datenbank beibehält oder die bestehenden Sekundärindizes ignoriert. Ein Sekundärindex ist ein Index, der keine Eindeutigkeits-Integritätsregel bzw. kein Primär- oder Fremdschlüssel ist. Indizes, die zur Erzwingung von Integritätsregeln zur Erhaltung der referenziellen Integrität dienen, werden immer berücksichtigt, wenn Zugriffspläne ausgewählt werden.

Die Analyse umfasst die folgenden Schritte:

- **Mögliche Indizes generieren** Für jede Protokollierungssitzung generiert der Indexberater eine Gruppe möglicher Indizes. Das Erstellen eines realen Indizes für eine große Tabelle kann viel Zeit

beanspruchen. Daher erstellt der Indexberater seine möglichen Indizes als virtuelle Indizes. Ein virtueller Index kann nicht verwendet werden, um tatsächlich Abfragen auszuführen, aber der Optimierer kann virtuelle Indizes verwenden, um die Kosten von Ausführungsplänen zu schätzen, falls so ein Index verfügbar wäre. Mit virtuellen Indizes kann der Indexberater "Was-wäre-wenn"-Analysen ohne die Kosten der Erstellung und Verwaltung realer Indizes durchführen. Virtuelle Indizes sind auf vier Spalten begrenzt.

- **Nutzen und Kosten von möglichen Indizes testen** Der Indexberater fordert den Optimierer auf, die Kosten für das Ausführen der Abfragen in der Protokollierungsdatenbank mit bzw. ohne verschiedene Kombinationen von möglichen Indizes zu schätzen.
- **Empfehlungen generieren** Der Indexberater stellt die Ergebnisse der Abfragekosten zusammen und sortiert die Indizes anhand des Gesamtvorteils, den sie bieten. Er liefert ein SQL-Skript, das Sie ausführen können, um die Empfehlungen anzuwenden, oder das Sie für Ihre eigene Überprüfung und Analyse speichern können.

Siehe auch

- [„Clustered-Indizes“ auf Seite 31](#)

Ergebnisse des Indexberaters

Der Indexberater liefert eine Reihe von Registern mit den Ergebnissen einer gegebenen Analyse. Sie können die Ergebnisse einer Analyse für eine spätere Überprüfung speichern.

Registerkarte "Zusammenfassung"

Die Registerkarte **Zusammenfassung** enthält eine Übersicht über die Analyse, einschließlich der Anzahl der Abfragen, der Anzahl der empfohlenen Indizes, der Anzahl der für die empfohlenen Indizes erforderlichen Seiten und der Vorteile, die die empfohlenen Indizes voraussichtlich bewirken werden. Die Vorteils-Zahl wird in internen Kosteneinheiten gemessen.

Registerkarte "Empfohlene Indizes"

Die Registerkarte **Empfohlene Indizes** enthält Daten über jeden der empfohlenen Indizes. Dazu gehören folgende Details:

- **Clustered** Jede Tabelle kann einen einzigen Clustered-Index haben. In manchen Fällen kann ein Clustered-Index signifikant mehr Vorteile als ein Unclustered-Index bieten.
- **Seiten** Die geschätzte Anzahl von Datenbankseiten, die zur Aufnahme des Indizes erforderlich sind, falls Sie sich entscheiden, ihn zu erstellen.
- **Relativer Vorteil** Eine Zahl von eins bis zehn, die den geschätzten Gesamtvorteil bei Erstellung des angegebenen Indizes darstellt. Je höher die Zahl, desto größer ist der Vorteil.

Der relative Vorteil wird unter Verwendung eines internen Algorithmus berechnet, und zwar unabhängig von der Spalte "Kostenvorteile insgesamt". Bei der Schätzung des relativen Vorteils gibt es mehrere Faktoren, die in den Kostenvorteilen insgesamt nicht berücksichtigt werden. So kann es zum Beispiel passieren, dass sich das Vorhandensein des einen Indexes dramatisch auf die Vorteile

auswirkt, die einem zweiten Index zugeordnet sind. In diesem Fall versucht die Funktion "Relativer Vorteil", die jeweilige Auswirkung der einzelnen Indizes abzuschätzen.

- **Gesamtvorteil** Der Gesamtvorteil ist die dem Index zugeordnete Kostenminderung als Summe aller Vorgänge in der Protokollierungssitzung, in internen Kosteneinheiten gemessen (das Kostenmodell).
- **Aktualisierungskosten** Das Hinzufügen eines Indizes bewirkt einen Aufwand sowohl in Bezug auf den zusätzlichen Speicherplatzbedarf als auch die erhöhte Arbeitsleistung, die anfällt, wenn Daten geändert werden. Die Spalte "Aktualisierungskosten" besteht aus der Schätzung der zusätzlichen Kosten zur Aufrechterhaltung eines Indizes. Sie werden in internen Kosteneinheiten gemessen.
- **Kostenvorteile insgesamt** Der Gesamtvorteil minus der Aktualisierungskosten, die dem Index zugeordnet sind.

Registerkarte "Anforderungen"

Die Registerkarte **Anforderungen** enthält eine Aufschlüsselung der Auswirkungen der Empfehlungen auf die einzelnen Anforderungen innerhalb der Protokollierungssitzung. Die Informationen umfassen die geschätzten Kosten vor und nach der Anwendung der empfohlenen Indizes sowie auch die virtuellen Indizes, die von der Abfrage verwendet werden. Eine Schaltfläche ermöglicht es Ihnen, den besten Ausführungsplan anzuzeigen, der für die Anforderung gefunden wurde.

Registerkarte "Aktualisierungen"

Die Registerkarte **Aktualisierungen** enthält eine Aufschlüsselung der Auswirkungen der Empfehlungen.

Registerkarte "Nicht benutzte Indizes"

Die Registerkarte **Nicht benutzte Indizes** listet in der Datenbank vorhandene Indizes auf, die aber bei der Ausführung einer Anforderung in der Protokollierungssitzung nicht verwendet wurden. Es werden nur Sekundärindizes aufgelistet: Das heißt, dass weder Indizes auf Primärschlüssel oder Fremdschlüssel noch Eindeutigkeits-Integritätsregeln aufgelistet werden.

Registerkarte "Log"

Die Registerkarte **Log** listet Aktivitäten auf, die für diese Analyse durchgeführt wurden.

Siehe auch

- „Tipp: Angemessene Seitengröße verwenden“ auf Seite 260
- „Implementierung von Indexberater-Ergebnissen“ auf Seite 168
- „So funktioniert der Optimierer“ auf Seite 333
- „Clustered-Indizes“ auf Seite 31
- „Indizes“ auf Seite 27
- „Indizes“ auf Seite 27
- „Anwendungsprofilerstellung“ auf Seite 156

Implementierung von Indexberater-Ergebnissen

Auch wenn der Indexberater ein SQL-Skript bereitstellt, das Sie zum Implementieren der Ergebnisse ausführen können, kann es sinnvoll sein, die Ergebnisse zuvor zu bewerten. So kann es beispielsweise

sinnvoll sein, die vorgeschlagenen Indexnamen umzubenennen, die während der Analyse generiert wurden.

Während Sie die Ergebnisse bewerten, beachten Sie Folgendes:

- **Entsprechen die vorgeschlagenen Indizes Ihren Erwartungen?** Wenn Sie mit den Daten in Ihrer Datenbank und mit den in der Datenbank ausgeführten Abfragen vertraut sind, sollten Sie den Nutzen der vorgeschlagenen Indizes anhand Ihrer Erfahrung abwägen. Möglicherweise wirkt sich ein vorgeschlagener Index nur auf eine einzige, selten ausgeführte Abfrage aus, oder er gilt für eine kleine Tabelle und hätte nur relativ geringe Gesamtauswirkungen. Auch ist es möglich, dass ein Index, den der Indexberater zu löschen empfiehlt, für eine andere Aufgabe benötigt wird, die nicht in Ihrer Protokollierungssitzung enthalten war.
- **Gibt es starke Korrelationen zwischen den Auswirkungen der vorgeschlagenen Indizes?** In den Indexempfehlungen sollte der relative Vorteil der einzelnen Indizes separat bewertet werden. Zwei Indizes haben allerdings nur Sinn, wenn beide vorhanden sind (eine Abfrage kann beide verwenden, sofern sie vorhanden sind, oder keinen, wenn einer fehlt). Es kann sinnvoll sein, die Registerkarte **Anforderungen** und die Abfragepläne zu untersuchen, um herauszufinden, wie die vorgeschlagenen Indizes zusammenwirken.
- **Sind Sie in der Lage, eine Tabelle neu zu organisieren, wenn Sie einen Clustered-Index erstellen?** Um den vollen Nutzen aus einem Clustered-Index zu ziehen, sollten Sie die Tabelle, auf deren Grundlage er erstellt wird, mit der REORGANIZE TABLE-Anweisung neu organisieren. Wenn der Indexberater viele Clustered-Indizes empfiehlt, werden Sie möglicherweise Ihre Datenbank entladen und wieder laden müssen, um die Vorteile ausnützen zu können. Das Entladen und Neuladen von Tabellen kann ein zeitraubender Vorgang sein, der möglicherweise große Plattenspeicher-Ressourcen erfordert. Daher sollten Sie überprüfen, ob Ihnen die Zeit und die Ressourcen zur Verfügung stehen, um die Empfehlungen zu implementieren.
- **Stellen Server- und Verbindungsstatus während der Analyse ein realistisches Abbild des Status unter Produktionsbedingungen dar?** Die Ergebnisse der Analyse hängen vom Status des Datenbankservers, einschließlich der Daten im Cache, ab. Sie hängen auch vom Status der Verbindung und bestimmten Datenbankoptionseinstellungen ab. Da die Analyse nur virtuelle Indizes erstellt und keine Anforderungen ausführt, ist der Status des Datenbankservers während der Analyse im Grunde ein statischer (abgesehen von Änderungen, die von anderen Verbindungen durchgeführt werden). Wenn der Status nicht dem normalen Zustand Ihrer Datenbank entspricht, sollten Sie möglicherweise die Analyse unter anderen Bedingungen erneut durchführen.

Siehe auch

- „Empfehlungen des Indexberaters“ auf Seite 166
- „SQL-Skriptdateien“ auf Seite 780
- „Indizes“ auf Seite 27
- „Indizes“ auf Seite 27
- „REORGANIZE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Anwendungsprofilerstellung“ auf Seite 156

Diagnoseprotokollierung

Die Diagnoseprotokollierung ist eine erweiterte Methode der Anwendungsprofilerstellung. Die vom Datenbankserver gelieferten Daten der Diagnoseprotokollierung können Zeitstempel und Verbindungs-IDs der Anweisungen umfassen, die vom Datenbankserver bearbeitet werden. Bei Abfragen umfassen die Diagnoseprotokollierungsdaten die Isolationsstufe, die Anzahl der abgerufenen Zeilen, den Cursortyp und den Ausführungsplan. Bei INSERT-, UPDATE- und DELETE-Anweisungen wird auch die Anzahl der betroffenen Zeilen einbezogen. Sie können die Diagnoseprotokollierung auch verwenden, um Informationen über Sperren und Deadlocks aufzuzeichnen und Performance-Statistiken zu erfassen.

Die während der Diagnoseprotokollierung erfassten Daten können Sie für detailliertere Anwendungsprofilerstellungsaktivitäten wie etwa Identifizierung und Problembehebung verwenden:

- Spezielle Performance-Probleme
- Anweisungen, die ungewöhnlich langsam ausgeführt werden
- Falsche Optionseinstellungen
- Umstände, die den Optimierer veranlassen, einen nicht optimalen Plan zu wählen
- Ressourcenkonflikte (CPUs, Speicher, Festplatten-I/O)
- Probleme mit der Anwendungslogik

Die Protokollierung von Daten wird auch von Tools wie etwa dem Indexberater eingesetzt, um spezielle Empfehlungen dazu abzugeben, wie die Datenbank oder die Anwendung geändert werden sollte, um die Performance zu verbessern.

Die Protokollierungsarchitektur ist robust und skalierbar. Sie kann alle Informationen aufzeichnen, die die Anforderungsprotokollierung erfasst, sowie Details zur Unterstützung einer benutzerdefinierten Analyse.

Hinweis

Die Anwendungsprofilerstellung erfasst nicht ein Ereignis, sondern lang laufende Anweisungen in dem Ereignis. Um ein Ereignis zu erfassen, betten Sie den Ereigniscode in eine Prozedur ein und rufen Sie die Prozedur aus dem Ereignis auf. Bevor Sie die Prozedur ausführen, aktivieren Sie die Protokollierung mit einer benutzerdefinierten Ebene, die geringe Details ohne Bedingungen über die Anweisungserfassung festlegt. Wenn die Protokollierungssitzung beendet ist, deaktivieren Sie die Anwendungsprofilerstellung.

Siehe auch

- „Anwendungsprofilerstellung“ auf Seite 156
- „Anforderungs-Trace-Analysen ausführen“ auf Seite 192

Daten der Protokollierungssitzung

Daten für die Diagnoseprotokollierung werden während einer **Protokollierungssitzung** erfasst. Es sind drei Methoden verfügbar, um Daten einer Protokollierungssitzung aufzuzeichnen:

- Verwendung des **Assistenten für die Datenbankprotokollierung** in Sybase Central
- Transparente Verwendung als Teil der automatisierten Aktivitäten des **Assistenten für die Anwendungsprofilerstellung**
- Verwendung der Anweisungen ATTACH TRACING und DETACH TRACING

Wenn eine Protokollierungssitzung ausgeführt wird, generiert SQL Anywhere Diagnosedaten für die angegebene Datenbank. Die Menge der generierten Protokollierungsdaten hängt von den Protokollierungseinstellungen ab.

Die Datenbank, für die ein Profil erstellt wird, wird als **Produktionsdatenbank**, als Quelldatenbank oder einfach als die zu profilierende Datenbank bezeichnet. Die Datenbank, in der die Protokollierungsdaten gespeichert werden, wird als **Protokollierungsdatenbank** bezeichnet. Bei der Produktions- und der Protokollierungsdatenbank kann es sich um die gleiche Datenbank handeln. Um eine Vergrößerung der Produktionsdatenbank zu vermeiden, wird empfohlen, dass Sie die Protokollierungsdaten in einer separaten Datenbank speichern. Die Größe von Datenbankdateien kann nicht vermindert werden, nachdem sie angewachsen sind. Außerdem hat die Produktionsdatenbank eine bessere Performance, wenn der Overhead für das Speichern und Verwalten der Protokollierungsdaten in einer anderen Datenbank durchgeführt wird, besonders wenn die Produktionsdatenbank groß ist und intensiv genutzt wird.

Die Tabellen in der Protokollierungsdatenbank, die die Protokollierungsdaten enthalten, werden als **Diagnoseprotokollierungstabellen** bezeichnet. Diese Tabellen gehören dem dbo.

Hinweis

Der **Assistent für die Anwendungsprofilerstellung** wird unter Windows Mobile nicht unterstützt, der **Assistent für die Datenbankprotokollierung** hingegen schon. Weiterhin müssen Sie die Protokollierung vom Windows Mobile-Gerät in einer Kopie der Windows Mobile-Datenbank durchführen, die auf einem Datenbankserver auf einem PC läuft. Sie können weder automatisch eine Protokollierungsdatenbank von einem Windows Mobile-Gerät aus erstellen, noch können Sie ein Protokoll für die lokale Datenbank auf einem Windows Mobile-Gerät erstellen.

Dateien, die während einer Protokollierungssitzung erstellt werden

Welche Dateien für eine Protokollierungssitzung erstellt und verwendet werden, hängt davon ab, ob Sie den **Assistenten für die Anwendungsprofilerstellung** oder den **Assistenten für die Datenbankprotokollierung** einsetzen.

Wenn Sie den **Assistenten für die Anwendungsprofilerstellung** ausführen, zeichnet dieser im Hintergrund eine Protokollierungssitzung auf und erstellt eine Protokollierungsdatenbank zum Speichern der Diagnosetabellen. Beim Erstellen dieser externen Datenbank werden der Name und der Speicherort benutzt, die Sie im Assistenten angeben, und die Erweiterung lautet *.adb*. Außerdem erstellt der Assistent eine Analyselogdatei in dem Verzeichnis, in dem die Protokollierungsdatenbank gespeichert wird, und verwendet dabei den gleichen Namen, jedoch mit der Erweiterung *.alg*. Diese Analyselogdatei enthält die Ergebnisse der vom Assistenten durchgeführten Analyse und kann jederzeit mit einem Texteditor geöffnet werden.

Wenn Sie die Daten, die der **Assistent für die Anwendungsprofilerstellung** generiert hat, nicht mehr benötigen, können Sie die Protokollierungsdatenbank und die Analyselogdatei dieser Sitzung löschen.

Wenn Sie eine Protokollierungsdatenbank mit dem **Assistenten für die Datenbankprotokollierung** erstellen, werden Sie gefragt, ob die Protokollierungsdaten intern, in der Produktionsdatenbank, oder extern, in einer separaten Datenbank (z.B. *tracingData.db*) gespeichert werden sollen. Das Erstellen einer externen Protokollierungsdatenbank wird empfohlen.

Hinweis

Protokollierungsdaten werden *nicht* als Teil eines Datenbank-Entladungs- oder -Aktualisierungsvorgangs entladen. Wenn Sie Protokollierungsdaten von einer Datenbank auf eine andere übertragen, müssen Sie den Inhalt der sa_diagnostic_*-Tabellen manuell kopieren. Dies wird allerdings nicht empfohlen.

Siehe auch

- „Erstellen einer externen Protokollierungsdatenbank (Sybase Central)“ auf Seite 193
- „Diagnoseprotokollierungskonfiguration“ auf Seite 172
- „Diagnoseprotokollierungstabellen“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Diagnoseprotokollierungskonfiguration

Sie können die vorkonfigurierten Protokollierungseinstellungen des **Assistenten für die Anwendungsprofilerstellung** in Sybase Central nicht ändern. Sie können allerdings den **Assistenten für die Datenbankprotokollierung** verwenden, um fast alle Aspekte Ihrer Protokollierungsaktivitäten zu konfigurieren. Verwenden Sie eine der folgenden Methoden, um die Einstellungen der Diagnoseprotokollierung zu konfigurieren:

- Verwendung des **Assistenten für die Datenbankprotokollierung** in Sybase Central. Diese Methode wird empfohlen, da sie Ihnen ermöglicht, alle aktivierten Protokollierungseinstellungen zu sehen.
- Verwenden von Systemprozeduren, um die Einstellungen in den Diagnoseprotokollierungstabellen zu ändern.

Die Protokollierungseinstellungen werden in der Systemtabelle "sa_diagnostic_tracing_level" gespeichert.

Die Datenbankeigenschaften "SendingTracingTo" und "ReceivingTracingFrom" identifizieren die Protokollierungs- und Produktionsdatenbanken.

Siehe auch

- „Diagnoseprotokollierungstypen“ auf Seite 176
- „sa_diagnostic_tracing_level-Tabelle“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_set_tracing_level-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_save_trace_data-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Liste der Datenbankeigenschaften“ [*SQL Anywhere Server - Datenbankadministration*]
- „Einstellungen für die Diagnoseprotokollierung“ auf Seite 184

Diagnoseprotokollierungsstufen

Untenstehend finden Sie eine Liste der Diagnoseprotokollierungsstufen, die im **Assistenten für die Datenbankprotokollierung** enthalten sind.

Die geschätzten Auswirkungen auf die Performance beruhen auf der Annahme, dass die Protokollierungsdaten an eine Protokollierungsdatenbank auf einem anderen Datenbankserver gesendet werden (empfohlen).

- **Stufe 0** Diese Stufe hält die Protokollierungssitzung aktiv, sendet aber keine Protokollierungsdaten an die Diagnoseprotokollierungstabellen.
- **Stufe 1** Performance-Zähler und Stichproben der ausgeführten Anweisungen werden alle fünf Sekunden aufgezeichnet. Für diese Stufe gibt es folgende Diagnoseprotokollierungstypen:

- "volatile_statistics", die jede Sekunde erfasst werden
- "non_volatile_statistics", die alle 60 Sekunden erfasst werden

Die Auswirkungen dieser Stufe auf die Performance sind nur gering.

- **Stufe 2** In dieser Stufe werden Performance-Zähler und Stichproben der ausgeführten Pläne alle fünf Sekunden aufgezeichnet und alle ausgeführten Anweisungen protokolliert. Für diese Stufe gibt es folgende Diagnoseprotokollierungstypen:

- "volatile_statistics", die jede Sekunde erfasst werden
- "non_volatile_statistics", die alle 60 Sekunden erfasst werden
- statements
- Pläne, die alle 5 Sekunden erfasst werden

Diese Stufe hat mittlere Auswirkungen auf die Performance: bis zu 20 % Overhead, aber nicht mehr.

- **Stufe 3** In dieser Stufe werden die gleichen Details wie in Stufe 2 aufgezeichnet. Stichproben von Plänen werden jedoch häufiger aufgezeichnet (alle 2 Sekunden) und es werden detaillierte Daten über Blockierungen und Deadlocks erfasst. Für diese Stufe gibt es folgende Diagnoseprotokollierungstypen:

- "volatile_statistics", die jede Sekunde erfasst werden
- "non_volatile_statistics", die alle 60 Sekunden erfasst werden
- statements
- blocking
- deadlock
- "statements_with_variables"

- Pläne, die alle 2 Sekunden erfasst werden

Die Stufe hat die größten Auswirkungen auf die Performance: der Overhead beträgt über 20 %.

Bereiche der Diagnoseprotokollierung

Die folgende Liste enthält die möglichen Einstellungen für die **Bereiche** der Diagnoseprotokollierung. Diese Einstellungen können benutzt werden, um die Protokollierung auf das zu beschränken, was die Aktivität in der Datenbank verursacht. Sie können beispielsweise den Bereich so festlegen, dass Anforderungen protokolliert werden, die von einer bestimmten Verbindung kommen. Bereichswerte werden in der Bereichsspalte der Diagnosetabelle "dbo.sa_diagnostic_tracing_level" gespeichert und können zugehörige Argumente haben. Normalerweise handelt es sich dabei um Bezeichner wie Objektnamen oder Benutzernamen, die in der Bezeichnerspalte gespeichert werden. Die Werte in der Bereichsspalte reflektieren die Einstellungen, die im **Assistenten für die Datenbankprotokollierung** festgelegt wurden.

Werte in der Bereichsspalte	Beschreibung
DATABASE	<p>Zeichnet Protokollierungsdaten für jedes Ereignis in der Datenbank auf, sofern das Ereignis der angegebenen Stufe und Bedingung entspricht. Wird für die langfristige Hintergrundüberwachung der Datenbank benutzt bzw. für kurzfristige Diagnosen, wenn die Ursachen kostspieliger Abfragen ermittelt werden müssen.</p> <p>Für DATABASE braucht kein Bezeichner angegeben zu werden.</p>
ORIGIN	<p>Zeichnet die Protokollierungsdaten für die Abfragen auf, die ihren Ursprung außerhalb oder innerhalb der Datenbank haben.</p> <p>Wenn Sie den Bereich ORIGIN festlegen, können Sie zwei Bezeichner angeben. "Internal" oder "External". Mit "External" wird festgelegt, dass der Anweisungstext und die zugehörigen Details für Abfragen zu protokollieren sind, die ihren Ursprung außerhalb des Datenbankservers haben und die der angegebenen Stufe und Bedingung entsprechen. Mit "Internal" wird festgelegt, dass für Abfragen die Daten zu protokollieren sind, die ihren Ursprung innerhalb des Datenbankservers haben und die der angegebenen Stufe und Bedingung entsprechen.</p>
USER	<p>Zeichnet Protokollierungsdaten nur für die Abfragen auf, die vom angegebenen Benutzer kommen, und für Verbindungen, die vom angegebenen Benutzer hergestellt wurden. Dieser Bereich wird verwendet, um problematische Abfragen zu prüfen, die von einem bestimmten Benutzer kommen.</p> <p>Die ID für diesen Bereich ist die Benutzer-ID des Benutzers, für den die Protokollierung durchgeführt werden soll.</p>

Werte in der Bereichsspalte	Beschreibung
CONNECTI- ON_NAME oder CONNECTI- ON_NUMBER	<p>Protokolliert Daten nur für die Anweisungen, die von der aktuellen Verbindung ausgeführt werden. Diese Einstellungen werden benutzt, wenn ein Benutzer über mehrere Verbindungen verfügt, von denen eine kostspielige Anweisungen verursacht.</p> <p>Der Bezeichner für diese Einstellungen ist der Name oder die Nummer der Verbindung.</p>
FUNCTION, PROCEDURE, EVENT, TRIG- GER oder TABLE	<p>Zeichnet Protokollierungsdaten für die Anweisungen auf, die das angegebene Objekt benutzen. Falls das Objekt andere Objekte referenziert, werden auch alle Daten für die anderen Objekte aufgezeichnet. Wenn die Protokollierung beispielsweise für eine Prozedur durchgeführt wird, die eine Funktion benutzt, welche wiederum ein Ereignis auslöst, werden die Anweisungen für alle drei Objekte protokolliert, sofern sie der angegebenen Stufe und der Bedingung entsprechen, die für die Protokollierung festgelegt wurden. Diese Einstellung wird benutzt, wenn ein bestimmtes Objekt kostspielig ist, oder wenn die Anweisungen, die das Objekt referenzieren, eine ungewöhnlich lange Ausführungszeit haben.</p> <p>Der Bereich TABLE wird für Tabellen, materialisierte Ansichten und nicht materialisierte Ansichten benutzt.</p> <p>Der Bezeichner für diese Einstellung ist der voll qualifizierte Name des Objekts.</p>

Siehe auch

- „Diagnoseprotokollierungstypen“ auf Seite 176
- „Bedingungen für die Diagnoseprotokollierung“ auf Seite 181

Benutzerdefinierte Diagnoseprotokollierungsstufen

Einstellungen der Diagnoseprotokollierung sind auf verschiedenen Stufen zusammengefasst. Innerhalb dieser Stufen können Sie die Einstellungen weiter anpassen. Die Typen von Informationen, die auf den unterschiedlichen Stufen erfasst werden, werden als **Diagnoseprotokollierungstypen** bezeichnet. Es folgt eine Beschreibung der Stufen, die Sie angeben können, und der Diagnoseprotokollierungstypen, die in den Stufen enthalten sind.

Durch die Anpassung der Einstellungen für die Diagnoseprotokollierung können Sie den Umfang unerwünschter Protokollierungsdaten in der Protokollierungssitzung reduzieren. Angenommen, Benutzerin AliceB hat sich beschwert, dass ihre Anwendung zu langsam läuft, während die restlichen Benutzer dieses Problem nicht haben. Sie möchten nun herausfinden, wodurch die Probleme bei AliceBs Abfragen verursacht werden. Sie sollten daher eine Liste aller Abfragen und sonstigen Anweisungen erstellen, die AliceB als Teil ihrer Anwendung ausführt, und aller Abfragepläne für länger laufende Abfragen. Um dies durchzuführen, könnten Sie einfach die Diagnoseprotokollierungsstufe auf 3 setzen und Protokollierungsdaten für einen oder zwei Tage erstellen. Da die Protokollierung jedoch beträchtliche Performance-Auswirkungen für die anderen Benutzer hat, sollten Sie die Protokollierung auf AliceBs

Aktivitäten beschränken. Setzen Sie dazu die Diagnoseprotokollierungsstufe auf 3, stellen Sie dann den Umfang der Protokollierung auf USER und geben Sie "AliceB" als Benutzernamen an. Lassen Sie die Diagnoseprotokollierungssitzung für einige Stunden laufen und prüfen Sie die Ergebnisse.

Die empfohlene Methode zum Anpassen von Einstellungen für die Diagnoseprotokollierung ist die Verwendung des **Assistenten für die Datenbankprotokollierung**.

Sie können auch die Systemprozedur "sa_set_tracing_level" benutzen, jedoch sind mit dieser Methode nicht so viele Anpassungen wie mit dem Assistenten möglich.

Es ist empfehlenswert, die Einstellungen für die Diagnoseprotokollierung während einer Protokollierungssitzung nicht zu ändern, da die Interpretation der Daten sonst erschwert wird.

Siehe auch

- „Diagnoseprotokollierungstypen“ auf Seite 176
- „Einstellungen für die Diagnoseprotokollierung“ auf Seite 184
- „sa_set_tracing_level-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Einstellungen für die Diagnoseprotokollierung während einer Protokollierungssitzung ändern“ auf Seite 185

Diagnoseprotokollierungstypen

Die folgende Tabelle listet die **Typen** der Protokollierung auf, die Sie für die Diagnoseprotokollierung auswählen können. Jeder Diagnoseprotokollierungstyp erfordert eine entsprechende Bedingung, wie unten angeführt, und wird in der Spalte "trace_type" der Diagnosetabelle "dbo.sa_diagnostic_tracing_level" gespeichert. Er kann entsprechende Diagnoseprotokollierungsbedingungen haben, die in der Spalte "trace_condition" gespeichert werden.

Die Werte in der Spalte "trace_type" reflektieren die Einstellungen, die im **Assistenten für die Datenbankprotokollierung** festgelegt wurden.

Wert in der Spalte "trace_type"	Beschreibung
VOLATILE_STATISTICS	<p>Erfasst Stichproben von sich häufig ändernden Datenbank- und Serverstatistiken.</p> <p>Bereiche und Bedingungen: Dieser Diagnoseprotokollierungstyp erfordert den Bereich DATA-BASE und verwendet die Bedingung SAMPLE_EVERY als Intervall, in dem Daten gesammelt werden.</p>

Wert in der Spalte "trace_type"	Beschreibung
NONVOLATILE_STATISTICS	<p>Erfasst Stichproben der Datenbank- und Servers- statistiken, die sich nicht häufig ändern. Nicht volatile Statistiken können nicht häufiger erfasst werden als volatile Statistiken. Volatile Statisti- ken müssen erfasst werden, damit nicht volatile Statistiken erfasst werden können, und der Zeit- unterschied zwischen den Stichprobenerfassun- gen für nicht volatile Statistiken sollte ein Mehr- faches des Zeitunterschiedes betragen, der für volatile Statistiken festgelegt wurde.</p> <p>Bereiche und Bedingungen: Dieser Diagnosepro- tokollierungstyp erfordert den Bereich DATA- BASE und verwendet die Bedingung SAM- PLE_EVERY als Intervall, in dem Daten gesam- melt werden.</p>
CONNECTION_STATISTICS	<p>Erfasst Stichproben von Verbindungsstatistiken. Wenn der Bereich DATABASE ist, werden Sta- tistiken für alle Verbindungen zur Datenbank auf- gezeichnet. Wenn der Bereich USER ist, werden Statistiken für alle Verbindungen für den ange- gebenen Benutzer aufgezeichnet. Wenn der Bereich CONNECTION_NAME oder CONNECTI- ON_NUMBER lautet, werden nur die Statistiken für die angegebene Verbindung erfasst. Volatile Statistiken müssen erfasst werden, damit CON- NECTION_STATISTICS erfasst werden können, und das Intervall zwischen den Stichprobenerfas- sungen sollte ein Mehrfaches des Intervalls betra- gen, das für VOLATILE_STATISTICS festge- legt wurde.</p> <p>Bereiche und Bedingungen: Dieser Diagnosepro- tokollierungstyp kann mit den Bereichen DATA- BASE, USER, CONNECTION_NUMBER und CONNECTION_NAME benutzt werden, und er verwendet die Bedingung SAMPLE_EVERY als Intervall, in dem Daten gesammelt werden.</p>

Wert in der Spalte "trace_type"	Beschreibung
BLOCKING	<p>Erfasst Daten zu Blockierungen entsprechend des festgelegten Bereichs und der Bedingung. Lautet der Bereich CONNECTION_NAME oder CONNECTION_NUMBER, kann die Sperre aufgezeichnet werden, wenn die Verbindung eine andere Verbindung blockiert oder durch eine andere Verbindung blockiert wird.</p> <p>Bereiche und Bedingungen: Dieser Diagnoseprotokollierungstyp kann mit allen Bereichen verwendet werden und verwendet eine der folgenden Bedingungen für das Sammeln: NONE, NULL, SAMPLE_EVERY.</p>
PLANS	<p>Erfasst Ausführungspläne für Abfragen, abhängig von der Bedingung und dem Bereich.</p> <p>Bereiche und Bedingungen: Dieser Diagnoseprotokollierungstyp kann mit allen Bereichen verwendet werden und verwendet eine der folgenden Bedingungen für das Sammeln: NONE, NULL, SAMPLE_EVERY und ABSOLUTE_COST.</p>
PLANS_WITH_STATISTICS	<p>Erfasst Pläne mit Ausführungsstatistiken. Pläne werden zum Zeitpunkt des Cursorschließens erfasst. Wenn die Bedingung RELATIVE_COST_DIFFERENCE festgelegt wurde, handelt es sich bei Teilen der Statistik in der Ausgabe möglicherweise um bestmöglich geschätzte Statistiken.</p> <p>Bereiche und Bedingungen: Dieser Diagnoseprotokollierungstyp kann mit allen Bereichen verwendet werden und jede der Bedingungen für das Sammeln verwenden.</p>

Wert in der Spalte "trace_type"	Beschreibung
STATEMENTS	<p>Erfasst SQL-Anweisungen für den angegebenen Bereich und die Bedingung. Interne Variablen werden bei der ersten Ausführung der einzelnen Prozeduren erfasst. Dieser Diagnoseprotokollierungstyp ist automatisch enthalten, wenn die Diagnoseprotokollierungstypen STATEMENTS_WITH_VARIABLES, PLANS, PLANS_WITH_STATISTICS, OPTIMIZATION_LOGGING oder OPTIMIZATION_LOGGING_WITH_PLANS festgelegt sind.</p> <p>Bereiche und Bedingungen: Dieser Diagnoseprotokollierungstyp kann mit allen Bereichen verwendet werden und jede der Bedingungen für das Sammeln verwenden.</p>
STATEMENTS_WITH_VARIABLES	<p>Erfasst SQL-Anweisungen und die Variablen, die mit den Anweisungen verbunden sind. Für jede Variable, egal ob interne Variablen oder Hostvariablen, werden auch alle zugewiesenen Werte erfasst.</p> <p>Bereiche und Bedingungen: Dieser Diagnoseprotokollierungstyp kann mit allen Bereichen verwendet werden und jede der Bedingungen für das Sammeln verwenden.</p>

Wert in der Spalte "trace_type"	Beschreibung
OPTIMIZATION_LOGGING	<p>Erfasst Daten über Join-Strategien, die vom Optimierer für die Ausführung der einzelnen Abfragen in Betracht gezogen werden. Außerdem werden Informationen über die Ausführungskosten für jede Strategie und die Basisinformationen erfasst, die für die Rekonstruktion der Baumstruktur erforderlich sind. Auch Daten über die durchgeführten Neuschreibungen für die Abfrage werden erfasst. Wenn ein anderer Bereich als DATABASE, CONNECTION_NAME, CONNECTION_NUMBER, ORIGIN oder USER benutzt wird, unterscheidet sich die erste aufgezeichnete Anweisung möglicherweise vom ursprünglichen Text der Abfrage, da Neuschreibungen durchgeführt werden könnten, bevor festgestellt wird, dass die Optimierungsprotokollierung auf die aktuelle Anweisung angewendet werden soll. Dieser Diagnoseprotokollierungstyp wird automatisch hinzugefügt, wenn der Protokollierungstyp OPTIMIZATION_LOGGING_WITH_PLANS festgelegt wird.</p> <p>Dieser Diagnoseprotokollierungstyp kann mit allen Bereichen verwendet werden und übernimmt keine Bedingung.</p>

Wert in der Spalte "trace_type"	Beschreibung
OPTIMIZATION_LOGGING_WITH_PLANS	<p>Erfasst Daten über Join-Strategien, die vom Optimierer in Betracht gezogen werden. Außerdem werden Informationen über die Ausführungskosten jeder Strategie und der komplette XML-Plan zur Beschreibung der Baumstruktur der Join-Strategie erfasst. Auch Daten über die durchgeführten Neuschreibungen für die Abfrage werden erfasst. Wenn ein anderer Bereich als DATABASE, CONNECTION_NAME, CONNECTION_NUMBER, ORIGIN oder USER benutzt wird, unterscheidet sich die erste aufgezeichnete Anweisung möglicherweise vom ursprünglichen Text der Abfrage, da Neuschreibungen durchgeführt werden könnten, bevor festgestellt wird, dass die Optimierungsprotokollierung auf die aktuelle Anweisung angewendet werden soll. Der Protokollierungstyp OPTIMIZATION_LOGGING wird automatisch hinzugefügt, wenn der Protokollierungstyp OPTIMIZATION_LOGGING_WITH_PLANS festgelegt wird.</p> <p>Dieser Diagnoseprotokollierungstyp kann mit allen Bereichen verwendet werden und übernimmt keine Bedingung.</p>

Siehe auch

- „Bereiche der Diagnoseprotokollierung“ auf Seite 174
- „Bedingungen für die Diagnoseprotokollierung“ auf Seite 181

Bedingungen für die Diagnoseprotokollierung

Die folgende Tabelle listet die **Bedingungen** der Diagnoseprotokollierung auf, die Sie festlegen können. Bedingungen steuern die Kriterien, die erfüllt sein müssen, damit ein Protokollierungseintrag für einen bestimmten Diagnoseprotokollierungstyp erfolgt. Die meisten Bedingungen benötigen einen Wert wie unten beschrieben. Bedingungen werden in der Spalte "trace_condition" der Diagnosetabelle "dbo.sa_diagnostic_tracing_level" gespeichert und können über einen zugehörigen Wert verfügen, wie etwa eine Zeitangabe in Millisekunden, gespeichert in der Wertespalte. Die Werte in der Bedingungsspalte reflektieren die Einstellungen, die im **Assistenten für die Datenbankprotokollierung** festgelegt wurden.

Wert in der Spalte "trace_condition"	Beschreibung
NONE oder NULL	Erfasst alle Protokollierungsdaten, die den Anforderungen der Stufe und des Bereichs entsprechen. Es wird nicht empfohlen, kostenträchtige Diagnoseprotokollierungsstufen (z.B. Pläne) für längere Zeit mit dieser Bedingung zu verwenden.
SAMPLE EVERY	Erfasst Protokollierungsdaten, welche die Stufen- und Bereichsanforderungen erfüllen, wenn seit der letzten Aufzeichnung des Ereignisses mehr als der angegebene Zeitraum verstrichen ist. Werte: Der Wert für diese Bedingung ist eine positive Ganzzahl, welche die Zeit in Millisekunden darstellt.
ABSOLUTE_COST	Erfasst die Anweisungen mit Ausführungskosten, die größer oder gleich dem angegebenen Wert sind. Werte: Der Wert für diese Bedingung ist eine Kostenangabe in Millisekunden.
RELATIVE_COST_DIFFERENCE	Zeichnet die Anweisungen auf, bei denen der Unterschied zwischen der erwarteten Ausführungszeit und der tatsächlichen Ausführungszeit größer oder gleich dem angegebenen Wert ist. Werte: Der Wert für diese Bedingung ist eine Kostenangabe als Prozentanteil. Um beispielsweise Anweisungen aufzuzeichnen, die mindestens doppelt so langsam wie erwartet sind, müssten Sie einen Wert von 200 angeben.

Siehe auch

- „Bereiche der Diagnoseprotokollierung“ auf Seite 174
- „Diagnoseprotokollierungstypen“ auf Seite 176

Ermitteln der aktuellen Einstellungen für die Diagnoseprotokollierung (Sybase Central)

Verwenden Sie den **Assistenten für die Datenbankprotokollierung** in Sybase Central, um die aktuellen Einstellungen für die Diagnoseprotokollierung anzuzeigen.

Voraussetzungen

Um eine Protokollierungssitzung zu starten, muss TCP/IP auf dem/den Datenbankserver(n) laufen, auf dem die Protokollierungsdatenbank und die Produktionsdatenbank ausgeführt werden.

Sie benötigen die DIAGNOSTICS-Systemrolle sowie die Systemprivilegien MANAGE PROFILING, SERVER OPERATOR und SELECT ANY TABLE.

Kontext und Bemerkungen

Sie können die Einstellungen für die Diagnoseprotokollierung unabhängig davon abrufen, ob eine Protokollierungssitzung läuft oder nicht.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie auf **Modus » Anwendungsprofil**.

Wenn der **Assistent für die Anwendungsprofilerstellung** eingeblendet wird, klicken Sie auf **Abbrechen**.

3. Rechtsklicken Sie im linken Fensterausschnitt auf die Datenbank, klicken Sie auf **Protokollierung** und befolgen Sie die Anweisungen im **Assistenten für die Datenbankprotokollierung**.

Ergebnisse

Die Einstellungen, die derzeit für die Diagnoseprotokollierung festgelegt sind, werden in der Liste **Protokollierungsstufen bearbeiten** angezeigt.

Siehe auch

- „sa_diagnostic_tracing_level-Tabelle“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Ermitteln der aktuellen Einstellungen für die Diagnoseprotokollierung (SQL)

Sie können die Einstellungen für die Diagnoseprotokollierung einfach abrufen, indem Sie die sa_diagnostic_tracing_level-Tabelle abfragen.

Voraussetzungen

Sie müssen die DIAGNOSTICS-Systemrolle und das MANAGE PROFILING-Systemprivileg haben.

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Fragen Sie die Tabelle "sa_diagnostic_tracing_level" auf Zeilen ab, in denen die Spalte "enabled" eine 1 enthält.

Ergebnisse

Der Datenbankserver liefert die Einstellungen für die Diagnoseprotokollierung, die zurzeit in Verwendung sind. Eine 1 in der Spalte "enabled" zeigt an, dass die Einstellung aktiv ist.

Beispiel

In der folgenden Anweisung wird gezeigt, wie Sie die Diagnosetabelle "sa_diagnostic_tracing_level" abfragen können, um die aktuellen Einstellungen für die Diagnoseprotokollierung abzurufen:

```
SELECT * FROM sa_diagnostic_tracing_level WHERE enabled = 1;
```

Die nachstehende Tabelle enthält ein Beispiel für eine Ergebnismenge, die aus dieser Abfrage resultiert:

id	scope	identifier	trace_type	trace_condition	value	enabled
1	database	(NULL)	volatile_statistics	sample_every	1,000	1
2	database	(NULL)	nonvolatile_statistics	sample_every	60.000	1
3	database	(NULL)	connection_statistics	(NULL)	60,000	1
4	database	(NULL)	blocking	(NULL)	(NULL)	1
5	database	(NULL)	deadlock	(NULL)	(NULL)	1
6	database	(NULL)	plans_with_statistics	sample_every	2,000	1

Siehe auch

- „sa_diagnostic_tracing_level-Tabelle“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Einstellungen für die Diagnoseprotokollierung

Einstellungen für die Diagnoseprotokollierung gelten jeweils für eine Produktionsdatenbank. Mit dem **Assistenten für die Datenbankprotokollierung** in Sybase Central können Sie die Einstellungen für die Diagnoseprotokollierung ändern, wenn Sie eine Protokollierungssitzung erstellen.

Die Protokollierungseinstellungen, die im **Assistenten für die Datenbankprotokollierung** konfiguriert werden, wirken sich nicht auf die Einstellungen oder das Verhalten des **Assistenten für die Anwendungsprofilerstellung** aus. Die Einstellungen des **Assistenten für die Anwendungsprofilerstellung** sind vorkonfiguriert und können nicht geändert werden.

Sie können die Diagnoseprotokollierungsstufe auch mit der Systemprozedur "sa_set_tracing_level" ändern. Mit dieser Prozedur wird keine Protokollierungssitzung gestartet, und wenn bereits eine Protokollierungssitzung läuft, schlägt die Änderung fehl. Mit der Systemprozedur haben Sie auch nicht so viel Kontrolle über andere Einstellungen, wie z.B. Bereiche, Bedingungen, Werte usw., wie mit Sybase Central.

Beispiel

In der folgenden Anweisung wird die Systemprozedur "sa_set_tracing_level" verwendet, um die Diagnoseprotokollierungsstufe auf 1 zu setzen:

```
CALL sa_set_tracing_level( 1 );
```

Vorhandene Einstellungen werden mit den Standardeinstellungen für die Diagnoseprotokollierungsstufe 1 überschrieben.

Siehe auch

- „sa_set_tracing_level-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Diagnoseprotokollierungsstufen“ auf Seite 173
- „Erstellen einer Diagnoseprotokollierungssitzung (Sybase Central)“ auf Seite 186

Einstellungen für die Diagnoseprotokollierung während einer Protokollierungssitzung ändern

In Sybase Central können Sie neue Protokollierungsstufen hinzufügen oder vorhandene Protokollierungsstufen löschen, während eine Protokollierungssitzung läuft.

Voraussetzungen

Um eine Protokollierungssitzung zu starten, muss TCP/IP auf dem/den Datenbankserver(n) laufen, auf dem die Protokollierungsdatenbank und die Produktionsdatenbank ausgeführt werden.

Sie benötigen die DIAGNOSTICS-Systemrolle sowie die Systemprivilegien MANAGE PROFILING, SERVER OPERATOR und SELECT ANY TABLE.

Kontext und Bemerkungen

Hinweis

Es ist empfehlenswert, die Einstellungen für die Diagnoseprotokollierung während einer Protokollierungssitzung nicht zu ändern, da die Interpretation der Daten sonst erschwert wird.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf die Datenbank und klicken Sie auf **Protokollierung » Protokollierungsstufen ändern**.
3. Fügen Sie neue Protokollierungsstufen hinzu oder löschen Sie bestehende.
4. Klicken Sie auf **OK**.

Ergebnisse

Die Protokollierungseinstellungen werden geändert.

Siehe auch

- „Benutzerdefinierte Diagnoseprotokollierungsstufen“ auf Seite 175

Erstellen einer Diagnoseprotokollierungssitzung (Sybase Central)

In Sybase Central können Sie eine Diagnoseprotokollierungssitzung zum Sammeln von Protokollierungsdaten aus Ihrer Datenbank erstellen.

Voraussetzungen

Um eine Protokollierungssitzung zu starten, muss TCP/IP auf dem/den Datenbankserver(n) laufen, auf dem die Protokollierungsdatenbank und die Produktionsdatenbank ausgeführt werden.

Sie benötigen die DIAGNOSTICS-Systemrolle sowie die Systemprivilegien MANAGE PROFILING, SERVER OPERATOR und SELECT ANY TABLE.

Kontext und Bemerkungen

Wenn Sie eine Diagnoseprotokollierungssitzung starten, konfigurieren Sie auch den auszuführenden Protokollierungstyp und geben an, wo die Protokollierungsdaten gespeichert werden sollen. Die Protokollierungssitzung wird fortgeführt, bis Sie sie ausdrücklich stoppen.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie auf die Datenbank und klicken Sie auf **Protokollierung**.
3. Klicken Sie auf **Weiter**.
4. Auf der Seite **Protokollierungsdetailtiefe** wählen Sie die Protokollierungsstufe aus.
5. Auf der Seite **Protokollierungsstufen bearbeiten** passen Sie die Einstellungen für die Diagnoseprotokollierung an.
6. Auf der Seite **Externe Datenbank erstellen** führen Sie folgende Schritte durch:
 - Klicken Sie auf **Neue Protokollierungsdatenbank erstellen**.
 - Wählen Sie einen Speicherort für die Datenbank.
 - Füllen Sie die Felder **Benutzername** und **Kennwort** aus.
 - Klicken Sie auf **Datenbank auf aktuellem Server starten**.
 - Klicken Sie auf **Datenbank erstellen**.
7. Führen Sie auf der Seite **Protokollierung starten** folgende Schritte durch:
 - Klicken Sie auf **Protokollierungsdaten in einer externen Datenbank speichern**.
 - Füllen Sie die Felder **Benutzername** und **Kennwort** aus. Geben Sie den Benutzernamen und das Kennwort an, die für die Verbindung mit der Produktionsdatenbank benutzt wurden.
 - Geben Sie im Feld **Weitere Verbindungsparameter** den Datenbankserver und den Datenbanknamen in Form einer partiellen Verbindungszeichenfolge an. Zum Beispiel:
`Server=Server47;DBN=TracingDB`

Hinweis

Nur DBN, DBF, Server, DBKEY, HOST und LINKS (CommLinks) werden in der Verbindungszeichenfolge für eine externe Datenbank unterstützt.

- Wählen Sie in der Liste **Wollen Sie die Menge der zu speichernden Protokollierungsdaten beschränken?** eine Option aus.
- 8. Klicken Sie auf **Fertig stellen**.
- 9. Wenn Sie die benötigten Diagnoseprotokollierungsdaten gesammelt haben, rechtsklicken Sie auf die Datenbank und klicken auf **Protokollierung » Protokollierung mit Speichern stoppen**.

Ergebnisse

Die diagnostische Protokollierungssitzung wurde gestartet und abgeschlossen und die Daten werden gespeichert.

Siehe auch

- „Anwendungsprofilerstellung“ auf Seite 156
- „TCP/IP-Protokoll“ [*SQL Anywhere Server - Datenbankadministration*]
- „ATTACH TRACING-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „DETACH TRACING-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_set_tracing_level-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Erstellen einer Diagnoseprotokollierungssitzung (SQL)

Sie können eine Protokollierungssitzung starten, indem Sie die ATTACH TRACING-Anweisung in Interactive SQL ausführen.

Voraussetzungen

Sie müssen die DIAGNOSTICS-Systemrolle und das MANAGE PROFILING-Systemprivileg haben.

Kontext und Bemerkungen

Das Starten einer Protokollierungssitzung wird auch als Anhängen einer Protokollierung bezeichnet. Dementsprechend wird das Stoppen einer Protokollierungssitzung als Abtrennen einer Protokollierung bezeichnet. Die SQL-Anweisungen für das Starten und Stoppen von Protokollierungen lauten ATTACH TRACING und DETACH TRACING.

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Verwenden Sie die sa_set_tracing_level-Systemprozedur, um die Protokollierungsstufen festzulegen.
Beispiel:

```
CALL sa_set_tracing_level( 1 );
```

3. Starten Sie die Protokollierung, indem Sie die Anweisung ATTACH TRACING ausführen.
4. Beenden Sie die Protokollierung, indem Sie die Anweisung DETACH TRACING ausführen.

Ergebnisse

Die Protokollierungssitzung wird erstellt und abgeschlossen.

Nächste Schritte

Die Diagnoseprotokollierungsdaten können in Sybase Central im Modus "Anwendungsprofil" angezeigt werden.

Beispiel

In diesem Beispiel wird gezeigt, wie die Diagnoseprotokollierung für die aktuelle Datenbank gestartet wird, wie die Protokollierungsdaten in einer separaten Datenbank gespeichert werden und wie ein zweistündiges Limit für die zu speichernden Daten gesetzt wird. In diesem Beispiel wird davon ausgegangen, dass eine Benutzer-ID "DBA" mit dem Kennwort "sql" und den richtigen Privilegien vorhanden ist:

```
ATTACH TRACING TO 'UID=DBA;PWD=sql;Server=server47;DBN=tracing;Host=myhost'  
LIMIT HISTORY 2 HOURS;
```

In diesem Beispiel wird gezeigt, wie die Diagnoseprotokollierung für die aktuelle Datenbank gestartet wird, wie die Protokollierungsdaten in der lokalen Datenbank gespeichert werden und wie ein Zwei-Megabyte-Limit für die zu speichernden Daten gesetzt wird:

```
ATTACH TRACING TO LOCAL DATABASE LIMIT SIZE 2 MB;
```

In diesem Beispiel wird gezeigt, wie die Diagnoseprotokollierung beendet wird und wie die Diagnosedaten gespeichert werden, die während der Protokollierungssitzung erfasst wurden:

```
DETACH TRACING WITH SAVE;
```

In diesem Beispiel wird gezeigt, wie die Diagnoseprotokollierung beendet wird, die Diagnosedaten aber nicht gespeichert werden.

```
DETACH TRACING WITHOUT SAVE;
```

Siehe auch

- „sa_save_trace_data-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Anwendungsprofilerstellung“ auf Seite 156
- „TCP/IP-Protokoll“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „ATTACH TRACING-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DETACH TRACING-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „sa_set_tracing_level-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Analyse von Diagnoseprotokollierungsdaten

Die Diagnoseprotokollierungsdaten bieten eine Aufzeichnung aller Aktivitäten, die auf dem Datenbankserver stattgefunden haben und die den Diagnoseprotokollierungsstufen und den Einstellungen

für die Protokollierungssitzung entsprechen. Sie müssen beim Auswerten der Daten die Einstellungen berücksichtigen, die bei der Protokollierung galten. Die Abwesenheit einer Anweisung, die Sie in einer Protokollierungssitzung zu sehen erwartet hatten, könnte beispielsweise darauf hinweisen, dass die Anweisung nicht ausgeführt wurde. Es ist aber auch möglich, dass die Anweisung nicht kostspielig genug war, um eine Bedingung zu erfüllen, nach der nur kostspielige Anweisungen protokolliert werden.

Es gibt viele Gründe, warum Sie detailliert prüfen sollten, welche Aktivitäten der Datenbankserver ausführt. Dazu gehören das Beheben von Performance-Problemen, das Schätzen der Ressourcennutzung für die Planung zukünftiger Auslastungen und das Debugging von Anwendungslogik.

Hinweis

Die Anwendungsprofilerstellung erfasst nicht ein Ereignis, sondern lang laufende Anweisungen in dem Ereignis. Um ein Ereignis zu erfassen, betten Sie den Ereigniscode in eine Prozedur ein und rufen Sie die Prozedur aus dem Ereignis auf. Bevor Sie die Prozedur ausführen, aktivieren Sie die Protokollierung mit einer benutzerdefinierten Ebene, die geringe Details ohne Bedingungen über die Anweisungserfassung festlegt. Wenn die Protokollierungssitzung beendet ist, deaktivieren Sie die Anwendungsprofilerstellung.

Siehe auch

- [„Praktische Einführungen in die Anwendungsprofilerstellung“ auf Seite 264](#)

Performance-Probleme beheben

Verwenden Sie die Anwendungsprofilerstellung um zu ermitteln, ob Performance-Probleme aus folgenden Ursachen entstehen:

- Lange Verarbeitungszeiten der Anwendung
- Ungeeignete Abfragepläne
- Konflikte bei gemeinsam genutzten Hardware-Ressourcen wie etwa CPU oder Platten-I/O
- Konflikte bei Datenbankobjekten
- Nicht optimales Datenbankdesign

Bei der Fehlerbehandlung einer schlechten Datenbank-Performance besteht die erste Aufgabe darin zu ermitteln, ob die Anwendung oder der Datenbankserver der Hauptverursacher ist. Um zu bestimmen, wie viel Verarbeitungszeit eine Clientanwendung benötigt, verwenden Sie die Registerkarte **Details** im Anwendungsprofilerstellungs-Tool und filtern Sie die Ergebnisse anhand einer einzelnen Verbindung. Wenn es Zeitunterschiede zwischen verschiedenen Anforderungen von dieser Verbindung gibt, liegt die Hauptverzögerung beim Anwendungsclient.

Wenn der Datenbankserver die Performance beeinträchtigt, müssen Sie die spezifische Ursache identifizieren.

Hinweis

Die Anwendungsprofilerstellung erfasst nicht ein Ereignis, sondern lang laufende Anweisungen in dem Ereignis. Um ein Ereignis zu erfassen, betten Sie den Ereigniscode in eine Prozedur ein und rufen Sie die Prozedur aus dem Ereignis auf. Bevor Sie die Prozedur ausführen, aktivieren Sie die Protokollierung mit einer benutzerdefinierten Ebene, die geringe Details ohne Bedingungen über die Anweisungserfassung festlegt. Wenn die Protokollierungssitzung beendet ist, deaktivieren Sie die Anwendungsprofilerstellung.

Siehe auch

- „Praktische Einführungen in die Anwendungsprofilerstellung“ auf Seite 264

Wenn Hardware-Ressourcen ein einschränkender Faktor sind

Wenn eine Datenbank mit mehr Arbeitslast fertig werden muss, wird die Performance üblicherweise durch CPU-Zyklen, den Arbeitsspeicher oder die I/O-Bandbreite der Festplatte eingeschränkt. Eine ineffiziente Anwendung oder ein ineffizienter Datenbankserver könnte die Ursache sein. Wenn Sie keine Effizienzmängel feststellen können, müssen Sie möglicherweise zusätzliche Hardwareressourcen zur Verfügung stellen.

Durch Hinzufügen von Ressourcen werden möglicherweise nicht alle Probleme im Zusammenhang mit der Skalierbarkeit gelöst oder die Computer-Performance gesteigert. Wenn ein Datenbankserver beispielsweise seine ihm zugewiesenen CPUs voll ausnützt, könnte dies darauf hinweisen, dass Sie weitere CPU-Ressourcen zuweisen sollten. Eine Verdoppelung der dem Datenbankserver verfügbaren CPUs verdoppelt allerdings nicht unbedingt die Arbeitsleistung, die der Datenbankserver ausführen kann.

Verwenden Sie die Registerkarte **Statistiken** im Bereich **Details zur Anwendungsprofilerstellung** um zu ermitteln, ob Hardware-Ressourcen ein einschränkender Faktor für die Performance sind.

- **Ermitteln, ob die CPU ein einschränkender Faktor ist** Um zu ermitteln, ob die CPU ein einschränkender Faktor ist, prüfen Sie die Statistik "ProcessCPU". Falls diese Statistik im Diagramm nicht vorhanden ist, klicken Sie auf die Schaltfläche **Statistik hinzufügen** und anschließend auf **ProcessCPU**. Wenn auf dem Diagramm erkennbar ist, dass sich der Zählerwert um ca. 1 Punkt pro Sekunde für jede CPU erhöht, die dem Datenbankserver zugeordnet ist, dann handelt es sich bei der CPU um einen einschränkenden Faktor. (Beispiel: Ein Datenbankserver läuft auf zwei CPUs. Wenn der CPU-Prozesszähler in 10 Sekunden von 2220 auf 2237 steigt, war die CPU-Nutzung in dieser Zeit $(2237-2220)/10s * 100 \% = 170 \%$. Somit arbeitet jede CPU mit $170 \%/2 = 85 \%$ ihrer Kapazität.
- **Ermitteln, ob der Arbeitsspeicher ein einschränkender Faktor ist** Um zu ermitteln, ob der Arbeitsspeicher (Größe des Pufferpools) ein einschränkender Faktor ist, prüfen Sie die Datenbankstatistiken "CacheHits" und "CacheReads". Falls diese Statistiken im Diagramm nicht vorhanden sind, klicken Sie auf die Schaltfläche **Statistik hinzufügen** und anschließend auf **CacheHits** und **CacheReads**. Falls "CacheHits" weniger als 10 % von "CacheReads" beträgt, bedeutet dies, dass der Pufferpool zu klein ist. Falls das Verhältnis im Rahmen von 10 - 70 % liegt, kann dies eine Hinweis auf einen zu kleinen Pufferpool sein — Sie sollten versuchen, die Cachegröße für den Datenbankserver zu erhöhen. Falls das Verhältnis über 70 % liegt, ist die Cachegröße wahrscheinlich ausreichend. Diese Strategie gilt nur, während der Datenbankserver gleichmäßig ausgelastet ist – d.h., wenn die normalen Arbeiten ausgeführt werden, nicht aber, wenn die Datenbank gerade gestartet wurde.

- **Erkennen, ob die I/O-Bandbreite ein einschränkender Faktor ist** Um zu ermitteln, ob die I/O-Bandbreite ein einschränkender Faktor ist, prüfen Sie die Datenbankstatistik "CurrIO". Falls diese Statistik im Diagramm nicht vorhanden ist, klicken Sie auf die Schaltfläche **Statistik hinzufügen** und anschließend auf **CurrIO**. Suchen Sie nach dem höchsten anhaltenden Wert in der Statistik. Wenn es beispielsweise ein hohes Plateau in der Grafik gibt, ist es um so wichtiger, je breiter es ist. Wenn das Diagramm anhaltende Werte von 3 oder mehr plus die Anzahl der vom Datenbankserver verwendeten physischen Platten aufweist, kann dies ein Hinweis darauf sein, dass das Plattensystem dem Ausmaß der Aktivitäten des Datenbankservers nicht gewachsen ist.

Siehe auch

- [„Systemmonitor-Statistiken“ auf Seite 207](#)
- [„Praktische Einführungen in die Anwendungsprofilerstellung“ auf Seite 264](#)
- [„Performance-Probleme beheben“ auf Seite 189](#)

Tools für die Fehlersuche in der Anwendungslogik

Wenn es Fehler im Anwendungscode oder in gespeicherten Prozeduren, Triggern, Funktionen oder Ereignissen gibt, kann es hilfreich sein, alle Anweisungen zu prüfen, die vom Datenbankserver ausgeführt werden und die mit dem fehlerhaften Code zusammenhängen. Bei Anwendungen, die SQL dynamisch generieren, können Sie den Text prüfen, den der Datenbankserver tatsächlich sieht, um nach Fehlern zu suchen, die von der Anwendung in den SQL-Text eingebaut wurden. Solche Fehler können dazu führen, dass die Ausführung von Abfragen fehlschlägt oder dass andere Ergebnisse geliefert werden als beabsichtigt. Die Anwendung könnte beispielsweise in der Entwicklungsphase gelegentlich einen SQL-Syntaxfehler melden, andererseits ist sie aber möglicherweise nicht in der Lage, den SQL-Text der misslungenen Abfrage zu liefern. Wenn Sie ein Protokoll zum Zeitpunkt der Anwendung ausgeführt haben, können Sie nach Anweisungen suchen, die Syntaxfehler oder andere Fehler verursacht haben, und Sie können den genauen Text sehen, der von der Anwendung generiert wurde.

Für interne Datenbankobjekte wie Prozeduren und Trigger können Sie den Debugger in Sybase Central verwenden. Manchmal kann es jedoch effektiver sein, wenn der Datenbankserver alle Anweisungen protokolliert, die von einer bestimmten Prozedur ausgeführt werden, und wenn Sie dann diese Anweisungen mit dem Tool zur Anwendungsprofilerstellung prüfen. Eine gespeicherte Prozedur könnte beispielsweise unter 1000 Aufrufen einmal ein falsches Ergebnis liefern, aber es ist nicht ersichtlich, unter welchen Bedingungen der Fehler auftritt. Statt die Prozedur 1000 mal mit dem Debugger zu prüfen, könnten Sie die Diagnoseprotokollierung für die Prozedur aktivieren und die Anwendung ausführen. Anschließend könnten Sie die Gruppe von Anweisungen prüfen, die vom Datenbankserver ausgeführt wurden, die Gruppe von Anweisungen ermitteln, die zur fehlerhaften Ausführung der Prozedur gehören, und dann entweder ermitteln, warum die Prozedur fehlgeschlagen ist, oder unter welchen Bedingungen sie sich anders als erwartet verhält. Wenn Sie wissen, unter welchen Bedingungen sich die Prozedur anders als erwartet verhält, können Sie einen Breakpoint in die Prozedur einfügen und weitere Nachforschungen mit dem Debugger anstellen.

Siehe auch

- [„SQL Anywhere-Debugger“ auf Seite 971](#)

Anforderungs-Trace-Analysen ausführen

Wenn eine bestimmte Anwendung oder Anforderung Probleme bereitet, können Sie eine Anforderungs-Trace-Analyse ausführen, um das Problem zu ergründen. Bei der Anforderungs-Trace-Analyse wird der **Assistent für die Datenbankprotokollierung** so konfiguriert, dass das Sammeln der Diagnosedaten auf lediglich den Benutzer, die Verbindung oder die Anforderung eingeschränkt wird, bei denen das Problem auftritt.

Voraussetzungen

Um eine Protokollierungssitzung zu starten, muss TCP/IP auf dem/den Datenbankserver(n) laufen, auf dem die Protokollierungsdatenbank und die Produktionsdatenbank ausgeführt werden.

Sie benötigen die DIAGNOSTICS-Systemrolle sowie die Systemprivilegien MANAGE PROFILING, SERVER OPERATOR und SELECT ANY TABLE.

Kontext und Bemerkungen

Sie können potenzielle Konflikte oder Engpässe mithilfe der verschiedenen Datenanzeigetools im Modus "Anwendungsprofil" ermitteln.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16-Plug-Ins** her.
2. Klicken Sie auf **Modus » Anwendungsprofil**.

Wenn der **Assistent für die Anwendungsprofilerstellung** eingeblendet wird, klicken Sie auf **Abbrechen**.

3. Rechtsklicken Sie auf die Datenbank und klicken Sie auf **Protokollierung** oder klicken Sie auf **Protokollierung » Protokollierung konfigurieren und starten**.
4. Befolgen Sie die Anweisungen im **Assistenten für die Datenbankprotokollierung**.
5. Wenn Sie die benötigten Protokollierungsdaten gesammelt haben, rechtsklicken Sie auf die Datenbank und klicken Sie auf **Protokollierung » Protokollierung mit Speichern stoppen**.
6. Klicken Sie im Fensterausschnitt **Details zur Anwendungsprofilerstellung** auf **Eine Analysedatei öffnen oder mit einer Protokollierungsdatenbank verbinden**.
7. Klicken Sie auf **In einer Protokollierungsdatenbank** und anschließend auf **Öffnen**.
8. Füllen Sie die Felder **Benutzername** und **Kennwort** aus und klicken Sie auf **OK**.
9. Klicken Sie im Fensterausschnitt **Details zur Anwendungsprofilerstellung** auf den letzten Eintrag in der Liste **Protokollierungssitzungs-ID**.
10. Klicken Sie unten im Fensterausschnitt **Details zur Anwendungsprofilerstellung** auf die Registerkarte **Datenbank-Protokollierungsdaten**.

Ergebnisse

Es stehen Ihnen verschiedene Registerkarten zur Verfügung, die Ihnen unterschiedliche Ansichten der für die Analyse erfassten Daten bieten. Auf der Registerkarte **Zusammenfassung** können Sie beispielsweise alle Anforderungen sehen, die während der Protokollierungssitzung an die Datenbank gestellt wurden. Dazu gehören die Anzahl der Ausführungen jeder Anforderung, die Dauer der Ausführungen, der Benutzer, der die Anforderung ausgeführt hat, usw. Wenn die Liste lang ist und Sie nach einer bestimmten Anforderung suchen, klicken Sie auf die Titelleiste **Filtern** auf der Registerkarte **Zusammenfassung** und geben eine Zeichenfolge in das Feld **SQL-Anweisungen mit** ein.

Nächste Schritte

Sie können weitere Details zu einer bestimmten Anforderung anzeigen, indem Sie auf die Anforderung rechtsklicken und auf **Detaillierte SQL-Anweisungen für die ausgewählte Zusammenfassungs-SQL-Anweisung anzeigen** klicken. Die Registerkarte **Details** wird geöffnet. Klicken Sie mit der rechten Maustaste auf die Zeile, in der die Anforderung steht. Damit werden weitere Auswahlen für Informationen geboten, etwa die Anzeige von SQL-Anweisung, Verbindung und Details zu Blockierungen.

Erstellen einer externen Protokollierungsdatenbank (Sybase Central)

Wenn Sie eine Protokollierungssitzung erstellen, können Sie die Protokollierungsdaten innerhalb der zu profilierenden Datenbank speichern. Dies ist bei Entwicklungsumgebungen angebracht, wenn Sie Anwendungen testen oder wenn es nur wenige Verbindungen zur Datenbank gibt.

Voraussetzungen

Um eine Protokollierungssitzung zu starten, muss TCP/IP auf dem/den Datenbankserver(n) laufen, auf dem die Protokollierungsdatenbank und die Produktionsdatenbank ausgeführt werden.

Sie benötigen die DIAGNOSTICS-Systemrolle sowie die Systemprivilegien MANAGE PROFILING, SERVER OPERATOR und SELECT ANY TABLE.

Kontext und Bemerkungen

Hinweis

Wenn Ihre Datenbank normalerweise zehn oder mehr Verbindungen gleichzeitig verarbeitet, wird empfohlen, Protokollierungsdaten in einer externen Protokollierungsdatenbank zu speichern, um die Auswirkungen auf die Performance zu vermindern.

Sie können die Protokollierungsdatenbank verwenden, um Daten für nachfolgende Protokollierungssitzungen zu speichern.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.

2. Klicken Sie auf **Modus » Anwendungsprofil**.

Wenn der **Assistent für die Anwendungsprofilerstellung** eingeblendet wird, klicken Sie auf **Abbrechen**.

3. Rechtsklicken Sie im linken Fensterausschnitt auf die Datenbank und klicken Sie auf **Protokollierung**.
4. Klicken Sie im **Assistenten für die Datenbankprotokollierung** auf der Seite **Externe Datenbank erstellen** auf **Neue Protokollierungsdatenbank erstellen** und befolgen Sie die Anweisungen.

Ergebnisse

Eine externe Datenbank zum Speichern der Analysedaten wird erstellt und eine Protokollierungsverbindung wird hergestellt.

Siehe auch

- „Erstellen einer Diagnoseprotokollierungssitzung (Sybase Central)“ auf Seite 186
- „Dienstprogramm zum Entladen (dbunload)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Erstellen einer externen Protokollierungsdatenbank (Befehlszeile)

Verwenden Sie das Entladedienstprogramm (dbunload), um manuell eine Protokollierungsdatenbank ohne eine Protokollierungssitzung zu erstellen.

Voraussetzungen

Sie müssen die DIAGNOSTICS-Systemrolle und das MANAGE PROFILING-Systemprivileg haben.

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Führen Sie den dbunload-Befehl aus, um das Schema aus der Produktionsdatenbank in die neue Protokollierungsdatenbank zu entladen:

Beispiel:

```
dbunload -c "UID=DBA;PWD=sql;Server=demo;DBN=demo" -an tracing.db -n -k -kd
```

In diesem Beispiel wird eine neue Datenbank mit dem Namen erstellt, der durch die Option -an angegeben wird (*tracing.db*). Mit der Option -n wird das Schema aus der profilierten Datenbank (in diesem Fall die SQL Anywhere-Beispieldatenbank *demo.db*) in die neue Protokollierungsdatenbank entladen. Mit der Option -k wird die Protokollierungsdatenbank mit den Daten gefüllt, die das Tool für die Anwendungsprofilerstellung zum Analysieren der Protokollierungsdaten verwendet. Die Option -kd schreibt alle DBSpaces in eine einzige DBSpace-Datei.

3. Um die Protokollierungsdatenbank auf einem anderen Computer zu speichern, können Sie sie an den neuen Speicherort kopieren.

Ergebnisse

Eine externe Datenbank zum Speichern der Analysedaten wird erstellt und es wird keine Protokollierungssitzung erstellt.

Siehe auch

- „Erstellen einer Diagnoseprotokollierungssitzung (Sybase Central)“ auf Seite 186
- „Dienstprogramm zum Entladen (dbunload)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Sonstige Diagnose-Tools und -Techniken

Neben den Funktionen zur Anwendungsprofilerstellung und Diagnoseprotokollierung steht Ihnen eine Vielzahl weiterer Diagnose-Tools und -Techniken zur Verfügung, die Sie beim Analysieren und Überwachen der aktuellen Performance der SQL Anywhere-Datenbank unterstützen.

Anforderungsprotokollierung

Die Anforderungsprotokollierung zeichnet die einzelnen von einer Anwendung empfangenen Anforderungen bzw. die an eine Anwendung gesendeten Antworten auf. Das ist besonders hilfreich, um herauszufinden, welche Aktion die Anwendung vom Datenbankserver verlangt.

Die Anforderungsprotokollierung ist auch ein guter Ausgangspunkt für die Performanceanalyse einer bestimmten Anwendung, wenn es nicht offensichtlich ist, ob der Datenbankserver oder der Client der Verursacher des Fehlers ist. Sie können eine Anforderungsprotokollierung durchführen, um die spezielle Anforderung an den Datenbankserver zu ermitteln, die für Probleme verantwortlich sein könnte.

Hinweis

Die gesamte Funktionalität sowie die Daten, die von der Anforderungsprotokollierungsfunktion geliefert werden, stehen auch bei der Diagnoseprotokollierung zur Verfügung. Die Diagnoseprotokollierung bietet außerdem zusätzliche Funktionen und Daten.

Zu den protokollierten Daten gehören beispielsweise Zeitstempel, Verbindungs-IDs und Anforderungstyp. Bei Abfragen umfassen die Daten außerdem die Isolationsstufe, die Anzahl der abgerufenen Zeilen und den Cursortyp. Für die Anweisungen INSERT, UPDATE und DELETE werden auch die Anzahl der betroffenen Zeilen und die Anzahl der ausgelösten Trigger erfasst.

Hinweis

Das Anforderungslog enthält alle Anweisungen mit verschleierte vertraulichen Daten. Der einzige Fall, bei dem sensitive Daten nicht verschleiert werden, ist eine Anweisung mit einem syntaktischen Analysefehler.

Mit der Serveroption -zr können Sie die Anforderungsprotokollierung einschalten, wenn Sie den Datenbankserver starten. Sie können die Ausgabe zur späteren Analyse in eine Anforderungs-Logdatei umleiten, indem Sie die Serveroption -zo verwenden. Mit den Optionen -zn und -zs legen Sie fest, wie viele Anforderungs-Logdateien gespeichert werden und welche maximale Größe die Anforderungs-Logdateien haben können.

Weitere Hinweise zu diesen Optionen finden Sie unter:

- „Datenbankserveroption -zr“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption -zo“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption -zn“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption -zs“ [[SQL Anywhere Server - Datenbankadministration](#)]

Hinweis

Diese Serveroptionen wirken sich nicht auf die Diagnoseprotokollierung in Sybase Central aus. Die Anforderungsprotokollierung auf Dateibasis ist vollkommen getrennt von der Diagnoseprotokollierungsfunktion in Sybase Central, welche dbo-eigene Diagnosetabellen in der Datenbank verwendet, um Daten des Anforderungslogs zu speichern.

Die `sa_get_request_times`-Systemprozedur liest ein Anforderungslog und füllt die globale temporäre Tabelle (`satmp_request_time`) mit Anweisungen aus dem Log und deren Ausführungszeiten. Die aufgezeichneten Zeitangaben für die Anweisungen INSERT/UPDATE/DELETE beziehen sich auf die Zeit, zu denen die Anweisungen ausgeführt wurden. Bei Abfragen bestehen die aufgezeichneten Zeitdaten aus der Zeitspanne von PREPARE bis DROP (describe/open/fetch/close). Das bedeutet, dass Sie auf geöffnete Cursor achten müssen.

Prüfen Sie `satmp_request_time` auf Anweisungen, die Kandidaten für Verbesserungen sein könnten. Anweisungen, die kostengünstig sind, aber häufig ausgeführt werden, können Performance-Probleme darstellen.

Sie können `sa_get_request_profile` verwenden, um `sa_get_request_times` aufzurufen und `satmp_request_time` in einer anderen globalen temporären Tabelle namens `satmp_request_profile` zusammenzufassen. Diese Prozedur gruppiert überdies die Anweisungen und liefert die Anzahl der Aufrufe, die Ausführungszeiten usw.

Vorsicht

Wenn das Log mit dem Perl-Skript *tracetime.pl* analysiert wird, sollte die `max_client_statements_cached`-Option auf 0 gesetzt werden, damit das clientseitige Caching von Anweisungen deaktiviert wird, während das Anforderungslog erfasst wird.

Siehe auch

- „Diagnoseprotokollierung“ auf Seite 170
- „`sa_get_request_times`-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`sa_get_request_profile`-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`max_client_statements_cached`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „`sa_server_option`-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Beispiele

Sie können die Ausgabe in ein Anforderungslog filtern, um nur Anforderungen von einer bestimmten Verbindung oder einer bestimmten Datenbank aufzuzeichnen, indem Sie die `sa_server_option`-Systemprozedur verwenden. Dies kann den Umfang des Protokolls verringern, wenn Sie einen Datenbankserver mit vielen aktiven Verbindungen oder einer Vielzahl von Datenbanken überwachen.

- **Filtern anhand einer Verbindung** Verwenden Sie die folgende Syntax:

```
CALL sa_server_option( 'RequestFilterConn' , connection-id );
```

Sie können die *connection-id* abrufen, indem Sie `CALL sa_conn_info()` ausführen.

- **Filtern anhand einer Datenbank** Verwenden Sie die folgende Syntax:

```
CALL sa_server_option( 'RequestFilterDB' , database-id );
```

Die *database-id* kann durch die Ausführung von `SELECT CONNECTION_PROPERTY('DBNumber')` bezogen werden, wenn Sie mit der Datenbank verbunden sind. Das Filtern bleibt aktiviert, bis es explizit zurückgesetzt bzw. der Datenbankserver heruntergefahren wird.

- **Zurücksetzen des Filterns** Verwenden Sie eine der beiden folgenden Anweisungen, um das Filtern anhand der Verbindung bzw. anhand der Datenbank zurückzusetzen:

```
CALL sa_server_option( 'RequestFilterConn' , -1 );
```

```
CALL sa_server_option( 'RequestFilterDB' , -1 );
```

- **Hostvariablen in Anforderungslogs ausgeben** Hostvariablenwerte in das Anforderungslog einbeziehen:

- Verwenden Sie die Option `-zr` mit einem Wert von **hostvars**
- Führen Sie Folgendes aus:

```
CALL sa_server_option( 'RequestLogging' , 'hostvars' );
```

Die `sa_get_request_times`-Prozedur für die Anforderungsloganalyse erkennt Hostvariablen im Log und fügt sie der globalen temporären Tabelle "satmp_request_hostvar" hinzu.

Prozedurprofilerstellung mit Systemprozeduren

Die Prozedurprofilerstellung liefert wertvolle Informationen über die Verwendung von gespeicherten Prozeduren, benutzerdefinierten Funktionen, Ereignissen, Systemtriggern und Triggern durch alle Verbindungen. Sie können die Prozedurprofilerstellung unter Verwendung von Systemprozeduraufrufen durchführen. Es wird empfohlen, dass Sie die Prozedurprofilerstellungsfunktionen im Modus für die Anwendungsprofilerstellung von Sybase Central verwenden.

Siehe auch

- „Prozedurprofilerstellung im Modus für die Anwendungsprofilerstellung“ auf Seite 158

Aktivieren der Prozedurprofilerstellung (SQL)

Sie können die Prozedurprofilerstellung mithilfe der `sa_server_option`-Systemprozedur aktivieren.

Voraussetzungen

Sie müssen die DIAGNOSTICS-Systemrolle und das MANAGE PROFILING-Systemprivileg haben.

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Rufen Sie die Systemprozedur "sa_server_option" auf und setzen Sie die Option "ProcedureProfiling" auf ON.

Führen Sie beispielsweise die folgende CALL-Anweisung aus:

```
CALL sa_server_option( 'ProcedureProfiling' , 'ON' );
```

Ergebnisse

Die Prozedurprofilerstellung wird aktiviert.

Siehe auch

- „sa_server_option-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Filtern der Prozedurprofilerstellung nach Benutzer (SQL)

Nötigenfalls können Sie feststellen, welche Prozeduren ein bestimmter Benutzer verwendet, ohne andere Verbindungen daran zu hindern, die Datenbank zu verwenden. Das ist nützlich, wenn die Verbindung bereits besteht oder wenn mehrere Benutzer sich mit derselben Benutzer-ID anmelden.

Voraussetzungen

Sie müssen die DIAGNOSTICS-Systemrolle und das MANAGE PROFILING-Systemprivileg haben.

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Rufen Sie die Systemprozedur "sa_server_option" folgendermaßen auf:

```
CALL sa_server_option( 'ProfileFilterUser' , 'userid' );
```

Ergebnisse

Der Wert von *userid* ist die ID des Benutzers, der überwacht wird.

Siehe auch

- „sa_server_option-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Zurücksetzen der Prozedurprofilerstellung (SQL)

Wenn Sie die Profilinformatoren zurücksetzen, löscht die Datenbank die alten Informationen und beginnt umgehend mit der Erfassung neuer Informationen zu Prozeduren, Funktionen, Ereignissen und

Triggern. Sie können die Prozedurprofilerstellung mithilfe der sa_server_option-Systemprozedur aus Interactive SQL zurücksetzen.

Voraussetzungen

Sie müssen die DIAGNOSTICS-Systemrolle und das MANAGE PROFILING-Systemprivileg haben.

Die Prozedurprofilerstellung muss aktiviert sein.

Aufgabe

- Rufen Sie die sa_server_option-Systemprozedur auf und setzen Sie die ProcedureProfiling-Option auf RESET.

Führen Sie beispielsweise die folgende CALL-Anweisung aus:

```
CALL sa_server_option( 'ProcedureProfiling' , 'RESET' );
```

Ergebnisse

Die Prozedurprofilerstellung wird zurückgesetzt.

Siehe auch

- „sa_server_option-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Deaktivieren der Prozedurprofilerstellung (SQL)

Sie können die Prozedurprofilerstellung mithilfe der sa_server_option-Systemprozedur aus Interactive SQL deaktivieren und dabei vorhandene Daten löschen.

Voraussetzungen

Sie müssen die DIAGNOSTICS-Systemrolle und das MANAGE PROFILING-Systemprivileg haben.

Kontext und Bemerkungen

Wenn Sie die Profilinformatoren ausgewertet haben, können Sie entweder die Profilerstellung deaktivieren oder die Profilerstellung löschen. Wenn Sie die Profilerstellung deaktivieren, erfasst die Datenbank keine Profilinformatoren mehr, und die bis zu diesem Zeitpunkt erfassten Informationen verbleiben auf der Registerkarte **Profil** in Sybase Central. Wenn Sie die Profilerstellung löschen, deaktiviert die Datenbank die Profilerstellung und löscht sämtliche Profildaten aus der Registerkarte **Profil** in Sybase Central.

Aufgabe

- Rufen Sie die sa_server_option-Systemprozedur auf und setzen Sie die ProcedureProfiling-Option auf OFF oder CLEAR.

Führen Sie beispielsweise die folgende CALL-Anweisung aus, um die Profilerstellung zu deaktivieren:

```
CALL sa_server_option( 'ProcedureProfiling' , 'OFF' );
```

Oder führen Sie die folgende CALL-Anweisung aus, um die Profilerstellung zu bereinigen:

```
CALL sa_server_option( 'ProcedureProfiling' , 'CLEAR' );
```

Ergebnisse

Die Prozedurprofilerstellung wird deaktiviert, und wenn die CLEAR-Option verwendet wurde, werden die Prozedurprofilerstellungsdaten gelöscht.

Siehe auch

- „sa_server_option-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Systemprozeduren können Prozedurprofilerstellungsinformationen abrufen

Sie können Systemprozeduren verwenden, um Daten zur Prozedurprofilerstellung für folgende Objekte anzuzeigen: gespeicherte Prozeduren, Funktionen, Ereignisse, Systemtrigger und Trigger. Außerdem muss die Prozedurprofilerstellung bereits aktiviert sein.

Die Systemprozedur "sa_procedure_profile" zeigt detaillierte Profilinformatoren einschließlich der Ausführungszeiten für die Zeiten innerhalb der einzelnen Objekte. Jede Zeile in der Ergebnismenge stellt eine ausführbare Codezeile im Objekt dar.

Die Systemprozedur "sa_procedure_profile_summary" zeigt die gesamte Ausführungszeit für die einzelnen Objekte und liefert eine Zusammenfassung aller ausgeführten Objekte. Jede Zeile in der Ergebnismenge stellt die Ausführungsdetails eines Objekts dar.

Wenn Sie die Ergebnisse dieser Systemprozeduren prüfen, werden möglicherweise mehr Objekte aufgeführt als ursprünglich angefordert wurden. Dies liegt daran, dass ein Objekt ein weiteres Objekt aufrufen kann. Ein Trigger kann beispielsweise eine gespeicherte Prozedur aufrufen, die wiederum eine weitere gespeicherte Prozedur aufruft.

Siehe auch

- „Aktivieren der Prozedurprofilerstellung (SQL)“ auf Seite 197
- „sa_procedure_profile_summary-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „sa_procedure_profile-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „sa_server_option-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Zeitfolgen-Dienstprogramme

Einige Dienstprogramme zum Testen der Performance, wie "fetchst", "instest" und "trantest" sind im Verzeichnis %SQLANYSAMPI6%\SQLAnywhere verfügbar.

Das Dienstprogramm "fetchst" misst die Abrufzeiten bei einer beliebigen Abfrage. Das Dienstprogramm "intest" ermittelt die Zeit, die benötigt wird, um Zeilen in eine Tabelle einzufügen. Das Dienstprogramm "trantest" misst die Last, welche von einer gegebenen Serverkonfiguration mit einem gegebenen Datenbankdesign und einer gegebenen Reihe von Transaktionen bewältigt werden kann.

Diese Tools liefern Ihnen genauere Zeitangaben als der grafische Plan mit Statistiken, und sie können Ihnen Hinweise auf die beste erreichbare Performance (beispielsweise Durchsatz) für einen bestimmten Server und eine Datenbankkonfiguration geben.

Eine ausführliche Beschreibung der Tools finden Sie in der *readme.txt*-Datei im selben Ordner wie das Dienstprogramm.

Datenbankperformance überwachen

SQL Anywhere liefert Statistiken, mit denen Sie die Datenbank-Performance überwachen können. Es gibt viele Möglichkeiten, auf diese Statistiken zuzugreifen:

- **SQL-Funktionen** Mit diesen Funktionen kann eine Anwendung direkt auf SQL Anywhere-Datenbankstatistiken zugreifen.
- **Sybase Central-Systemmonitor** Dieses grafische Tool fragt die Datenbank ab und stellt nur diejenigen Statistikdaten grafisch dar, die Sie im Systemmonitor ausgewählt haben.
- **Optionen des Windows-Systemmonitors** Der Systemmonitor ist ein Tool zur Überwachung, das vom Betriebssystem Windows zur Verfügung gestellt wird.
- **Dienstprogramm für Performancestatistik (dbstats)** Dieses Dienstprogramm bietet die Überwachung des Datenbankservers, der Datenbank und der Verbindungsstatistiken für Datenbankserver unter Unix.
- **SQL Anywhere-Konsolendienstprogramm (dbconsole)** Dieses Dienstprogramm stellt Verwaltungs- und Monitoring-Funktionen für Datenbankserver-Verbindungen zur Verfügung.

Diese Methoden sind für die Überwachung in Echtzeit nützlich. Sie können jedoch auch Statistikdaten im Rahmen einer Diagnoseprotokollierung erfassen und für eine spätere Analyse speichern.

Siehe auch

- „SQL-Funktionen zum Überwachen von Statistiken“ auf Seite 201
- „Funktionen des Sybase Central-Systemmonitors zum Überwachen von Statistiken“ auf Seite 203
- „Optionen des Windows-Systemmonitors“ auf Seite 205
- „Dienstprogramm für Performancestatistik (dbstats) (Unix)“ [*SQL Anywhere Server - Datenbankadministration*]
- „SQL Anywhere-Konsolendienstprogramm (dbconsole)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Diagnoseprotokollierung“ auf Seite 170
- „Systemmonitor-Statistiken“ auf Seite 207
- „Datenbank-Monitoring“ [*SQL Anywhere Server - Datenbankadministration*]

SQL-Funktionen zum Überwachen von Statistiken

SQL Anywhere bietet eine Reihe von Systemfunktionen, mit denen Sie Zugang zu Informationen auf Verbindungs-, Datenbank- oder Serverbasis haben. Die verfügbaren Informationen reichen von statischen

Daten wie Datenbankservernamen bis zu Performance-bezogenen Statistiken, die den Plattenspeicher und die Speicherbelegung betreffen.

Funktionen, die Informationen abrufen

Die folgenden Funktionen rufen Systeminformationen ab:

- **PROPERTY-Funktion** Diese Funktion liefert den Wert einer bestimmten Eigenschaft auf Serverbasis.
- **Funktionen DB_PROPERTY und DB_EXTENDED_PROPERTY** Diese Funktionen liefern den Wert einer gegebenen Eigenschaft für eine bestimmte Datenbank oder standardmäßig für die aktuelle Datenbank.
- **Funktionen CONNECTION_PROPERTY und CONNECTION_EXTENDED_PROPERTY** Diese Funktion liefert den Wert einer gegebenen Eigenschaft für eine bestimmte Verbindung oder standardmäßig für die aktuelle Verbindung.

Geben Sie als Argument nur den Namen der Eigenschaft an, die Sie abrufen möchten. Die Funktionen geben den Wert für den aktuellen Server, die Verbindung oder Datenbank zurück.

Wirkungsgrad der Abfragen steigern

Um eine bessere Performance zu erreichen, sollte eine Clientanwendung, welche die Datenbankaktivität überwacht, die PROPERTY_NUMBER-Funktion einsetzen, um eine benannte Eigenschaft zu identifizieren, und dann die Nummer verwenden, um die Statistik wiederholt abzurufen.

Auf diese Weise abgerufene Eigenschaftsnamen stehen für viele verschiedene Statistiken zur Verfügung, von der Anzahl der Seiten-Schreibvorgänge im Transaktionslog und der Anzahl der gesetzten Checkpoints bis zu der Anzahl der Lesevorgänge von Indexblattseiten aus dem Cachespeicher.

Die folgende Reihe von Anweisungen veranschaulicht den Prozess von Interactive SQL aus:

```
CREATE VARIABLE propnum INT;  
CREATE VARIABLE propval INT;  
SET propnum = PROPERTY_NUMBER( 'CacheRead' );  
SET propval = DB_PROPERTY( propnum );
```

Beispiele

Die folgende Anweisung setzt eine Variable namens server_name auf den Namen des aktuellen Servers:

```
SET server_name = PROPERTY( 'name' );
```

Die folgende Abfrage gibt die Benutzer-ID für die aktuelle Verbindung zurück:

```
SELECT CONNECTION_PROPERTY( 'UserID' );
```

Die folgende Abfrage gibt den Dateinamen für die Stammdatei der aktuellen Datenbank zurück:

```
SELECT DB_PROPERTY( 'file' );
```

Siehe auch

- „PROPERTY_NUMBER-Funktion [System]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „PROPERTY-Funktion [System]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DB_PROPERTY-Funktion [System]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DB_EXTENDED_PROPERTY-Funktion [System]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CONNECTION_PROPERTY-Funktion [System]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CONNECTION_EXTENDED_PROPERTY-Funktion [Zeichenfolge]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Systemfunktionen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Funktionen des Sybase Central-Systemmonitors zum Überwachen von Statistiken“ auf Seite 203

Funktionen des Sybase Central-Systemmonitors zum Überwachen von Statistiken

Der Sybase Central-Systemmonitor ist nützlich für das Protokollieren von detaillierten Informationen über Aktionen des Datenbankservers, wie etwa Festplatten- und Speicherzugriff. Der Sybase Central-Systemmonitor kann Statistiken für alle SQL Anywhere-Datenbankserver grafisch darstellen, zu denen Sie eine Verbindung herstellen können.

Der Sybase Central-Systemmonitor bietet folgende Funktionen:

- Aktualisierungen in Echtzeit (in anpassbaren Intervallen)
- Farbkodierte und in der Größe veränderbare Legende
- Konfigurierbare Eigenschaften für das Erscheinungsbild

Der Sybase Central-Systemmonitor fragt die Datenbank ab, um die Statistikzahlen zu erfassen. Dies kann sich auf einige Statistiken wie etwa Cachelesevorgänge pro Sekunde auswirken. Wenn Sie dies verhindern möchten, können Sie stattdessen den Windows-Systemmonitor verwenden.

Wenn Sie mehrere Versionen von SQL Anywhere gleichzeitig ausführen, können Sie auch mehrere Versionen des Systemmonitors gleichzeitig betreiben.

Siehe auch

- „Optionen des Windows-Systemmonitors“ auf Seite 205
- „Systemmonitor-Statistiken“ auf Seite 207

Sybase Central-Systemmonitor öffnen

Der Sybase Central-Systemmonitor wird im rechten Fensterausschnitt von Sybase Central angezeigt, wenn die Registerkarte **Systemmonitor** aktiviert ist. Im Diagramm werden nur die Statistiken angezeigt, die Sie entsprechend konfiguriert haben.

Voraussetzungen

Es gibt keine Voraussetzungen für das Ausführen dieser Aufgabe.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Wählen Sie im linken Fensterausschnitt den Server aus.
3. Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **Systemmonitor**.

Ergebnisse

Der **Systemmonitor** wird in Sybase Central geöffnet.

Siehe auch

- „Statistiken hinzufügen und entfernen“ auf Seite 204
- „Optionen des Windows-Systemmonitors“ auf Seite 205
- „Statistiken hinzufügen und entfernen“ auf Seite 204

Statistiken hinzufügen und entfernen

Sie können Sybase Central verwenden, um im Systemmonitor überwachte Statistiken hinzuzufügen oder zu entfernen.

Voraussetzungen

Es gibt keine Voraussetzungen für das Ausführen dieser Aufgabe.

Kontext und Bemerkungen

Tipp

Sie können auch im Statistik-Eigenschaftsfenster des Systemmonitors von Sybase Central eine Statistik hinzufügen bzw. entfernen.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Wählen Sie im linken Fensterausschnitt den Server aus.
3. Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **Statistiken**.
4. Rechtsklicken Sie auf eine Statistik, die derzeit nicht überwacht wird, und klicken Sie auf **Zum Systemmonitor hinzufügen** oder **Aus dem Systemmonitor entfernen**.

Ergebnisse

Die angegebenen Statistiken werden im Sybase Central-Systemmonitor hinzugefügt bzw. entfernt.

Siehe auch

- „Systemmonitor-Statistiken“ auf Seite 207
- „Sybase Central-Systemmonitor öffnen“ auf Seite 203
- „Optionen des Windows-Systemmonitors“ auf Seite 205

Optionen des Windows-Systemmonitors

Eine Alternative zum Sybase Central-Systemmonitor ist der Windows-Systemmonitor.

Der Windows-Systemmonitor bietet mehr Statistikwerte zur Performance als der Sybase Central-Systemmonitor, besonders bei den Statistikwerten zur Netzwerkkommunikation. Er verwendet ein Schema mit gemeinsam genutztem Speicher, anstatt Abfragen im Datenbankserver auszuführen, und beeinträchtigt auf diese Weise also nicht die statistischen Ergebnisse.

Wenn Sie gleichzeitig mehrere Versionen von SQL Anywhere verwenden, können Sie auch mehrere Versionen des Systemmonitors zeitgleich ausführen.

Beim Starten des Datenbankservers, der den vom Windows-Systemmonitor verwendeten Speicher steuert, können Sie die Datenbankserver-Optionen sowie die maximale Anzahl von Verbindungen oder Datenbanken angeben, die der Systemmonitor überwacht.

- „Datenbankserveroption -ks“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -ksc“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -ksd“ [*SQL Anywhere Server - Datenbankadministration*]

Siehe auch

- „Systemmonitor-Statistiken“ auf Seite 207

Verwenden des Windows-Systemmonitors

Verwenden Sie den Windows-Systemmonitor, wenn Sie Datenquellen anzeigen möchten, die mit Ihren SQL Anywhere-Datenbanken, Ihrem Server oder Ihrer Verbindung verknüpft sind.

Voraussetzungen

Ein SQL Anywhere-Datenbankserver muss laufen.

Kontext und Bemerkungen

Bei anderen Windows-Versionen informieren Sie sich in der Dokumentation des Windows-Betriebssystems, wie Sie den Windows-Systemmonitor starten.

Aufgabe

1. Starten Sie den Systemmonitor:
 - a. Klicken Sie in der Windows-Systemsteuerung auf **Verwaltung**.

- b. Klicken Sie auf **Leistung**.
2. Klicken Sie in der Symbolleiste auf das Pluszeichen (+).
3. Wählen Sie aus der Liste **Datenobjekt** eine der folgenden Optionen:
 - **SQL Anywhere 16-Verbindung** Damit wird die Performance einer einzelnen Verbindung überwacht. Es muss eine Verbindung bestehen, damit diese Auswahl angezeigt wird.
 - **SQL Anywhere 16-Datenbank** Damit wird die Performance einer einzelnen Datenbank überwacht.
 - **SQL Anywhere 16-Server** Damit wird die Performance auf dem gesamten Server überwacht.

Im Feld **Leistungsindikatoren auswählen** erscheint eine Liste der Statistiken, die angezeigt werden können.

Falls Sie auf **SQL Anywhere 16-Verbindung** oder **SQL Anywhere 16-Datenbank** geklickt haben, wird im Feld **Instanzen** eine Liste der Verbindungen oder Datenbanken angezeigt, für die Sie Statistiken anzeigen können.

4. Wählen Sie in der Liste **Leistungsindikatoren auswählen** eine anzuzeigende Statistik.
5. Falls Sie auf **SQL Anywhere 16-Verbindung** oder **SQL Anywhere 16-Datenbank** geklickt haben, wählen Sie im Feld **Instanzen** eine zu überwachende Datenbankverbindung oder Datenbank aus.
6. Klicken Sie auf **Erklärung**, um Informationen zur ausgewählten Datenquelle anzuzeigen.
7. Um die Datenquelle anzuzeigen, klicken Sie auf **Hinzufügen**.
8. Wenn Sie alle gewünschten Datenquellen ausgewählt haben, klicken Sie auf **Schließen**.

Ergebnisse

Die angegebenen Statistiken werden im Windows-Systemmonitor angezeigt.

Systemmonitor-Statistiken

SQL Anywhere stellt folgende Statistiken bereit:

- „Cachestatistiken“
- „Checkpoint- und Wiederherstellungs-Statistiken“
- „Kommunikationsstatistiken“
- „Statistiken über I/O-Vorgänge“
- „Statistiken über Festplatten-Lesezugriffe“
- „Statistiken über Festplatten-Schreibzugriffe“
- „Indexstatistiken“
- „Speicherseiten-Statistiken“
- „Anforderungsstatistiken“
- „Benutzerdefinierte Statistiken“
- „Verschiedene Statistiken“

Die Raten werden in 1-Sekunden-Intervallen protokolliert.

Cachestatistiken

Diese Statistiken beschreiben die Verwendung des Caches.

Statistik	Bereich	Beschreibung
Cachetreffer/Sek.	Verbindung und Datenbank	Zeigt die Rate, mit der die Suche nach Datenbankseiten durch eine im Cache gefundene Seite Erfolg hat
Cachelesevorgänge: Index intern	Verbindung und Datenbank	Zeigt die Rate, mit der interne Knotenseiten des Index vom Cache gelesen werden
Cachelesevorgänge: Index-Blatt	Verbindung und Datenbank	Zeigt die Rate, mit der Indexblattseiten vom Cache gelesen werden
Cachelesevorgänge: Tabelle/Sek.	Verbindung und Datenbank	Zeigt die Rate, mit der Tabellenseiten vom Cache gelesen werden
Cachelesevorgänge: Lesezugr. Gesamts./Sek	Verbindung und Datenbank	Zeigt die Rate, mit der Datenbankseiten im Cache gesucht werden
Cachelesevorgänge: Arbeitstabelle	Verbindung und Datenbank	Zeigt die Rate, mit der Arbeitstabellenseiten vom Cache gelesen werden
Cache-Ersetzungen: Lesezugr. Gesamts./Sek	Server	Die Rate, mit der Datenbankseiten aus dem Cache ausgelesen werden, um Platz für eine andere Seite zu machen
Cachegröße: Aktuell	Server	Zeigt die aktuelle Cachegröße des Datenbankservers in kB

Statistik	Bereich	Beschreibung
Cachegröße: Maximum	Server	Zeigt die maximal zulässige Cachegröße des Datenbankservers in kB
Cachegröße: Minimum	Server	Zeigt die minimal zulässige Cachegröße des Datenbankservers in kB
Cachegröße: Spitze	Server	Zeigt den Spitzenwert der Cachegröße des Datenbankservers in kB

Checkpoint- und Wiederherstellungs-Statistiken

Diese Statistiken grenzen die Checkpoint- und Wiederherstellungsaktionen ein, die ausgeführt werden, wenn die Datenbank in einem inaktiven Zustand ist.

Statistik	Bereich	Beschreibung
Checkpoint-Auslesevorgänge/Sek.	Datenbank	Die Rate, mit der Bereiche benachbarter Seiten während eines Checkpoints weggeschrieben werden
Checkpoint-Dringlichkeit	Datenbank	Die Checkpoint-Dringlichkeit in Prozent
Checkpoints/Sek.	Datenbank	Die Rate, mit der Checkpoints ausgeführt werden
ChkptLog: Bitmap-Größe	Datenbank	Die Größe des Checkpoint-Log-Bitmaps
ChkptLog: Festschreiben auf Platte/Sek	Datenbank	Die Rate, mit der die Checkpoint-Log-Vorgänge "commit_to_disk" durchgeführt werden
ChkptLog: Log-Größe	Datenbank	Die Größe des Checkpoint-Logs Seiten
ChkptLog: Seitenabbilder gesp./Sek	Datenbank	Die Rate, mit der Seiten vor Änderungen im Checkpoint-Log gespeichert werden
ChkptLog: Seiten in Gebrauch	Datenbank	Die Anzahl der Seiten im Checkpoint-Log, die derzeit verwendet werden
ChkptLog: Verschieben Seiten/Sek.	Datenbank	Die Rate, mit der Seiten im Checkpoint-Log verschoben werden
ChkptLog: Vorabbild/Sek. speichern	Datenbank	Die Rate, mit der Vorabbilder neuer Datenbankseiten dem Checkpoint-Log hinzugefügt werden

Statistik	Bereich	Beschreibung
ChkptLog: Schr. Seiten/ Sek.	Datenbank	Die Rate, mit der Seiten ins Checkpoint-Log geschrieben werden
ChkptLog: Schr. Seiten/ Sek.	Datenbank	Die Rate, mit der Festplatten-Schreibvorgänge im Checkpoint-Log durchgeführt werden. Ein Schreibvorgang kann mehrere Seiten umfassen.
ChkptLog: Schreiben in Bitmap/Sek	Datenbank	Die Rate, mit der Platten-Schreibvorgänge im Checkpoint-Log für Bitmap-Seiten durchgeführt werden
Leerlauf aktiv/Sek.	Datenbank	Die Rate, mit der der Leerlauf-Thread des Datenbankservers aktiv wird, um Leerlauf-Schreibvorgänge, Leerlauf-Checkpoints etc. auszuführen
Checkp.-Inakt.-Zeit	Datenbank	Die Gesamtdauer der inaktiven Checkpoints in Sekunden
Leerl.-Checkpoints/Sek.	Datenbank	Die Rate, mit der Checkpoints vom Leerlauf-Thread des Datenbankservers gesetzt werden. Ein Leerlauf-Checkpoint kommt vor, wenn der Leerlauf-Thread die letzte noch nicht abgeschlossene Seite im Cache ausliest.
Leerlauf-Schreibvorgänge/ Sek.	Datenbank	Die Rate, mit der Schreibvorgänge vom Leerlauf-Thread des Servers auf der Festplatte ausgegeben werden
Wiedherst.I/O-Schätzung	Datenbank	Die geschätzte Anzahl von I/O-Vorgängen, die für die Wiederherstellung einer Datenbank erforderlich sind
Wiederherstellungs- Dringlichkeit	Datenbank	Die Wiederherstellungs-Dringlichkeit in Prozent

Siehe auch

- „So entscheidet die Datenbank, wann ein Checkpoint auszuführen ist“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „checkpoint_time-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption -gc“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „recovery_time-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Checkpoint-Logs“ [[SQL Anywhere Server - Datenbankadministration](#)]

Kommunikationsstatistiken

Diese Statistiken beschreiben die Aktivität der Client/Server-Kommunikation.

Statistik	Bereich	Beschreibung
Comm: Empfangene Byte/Sek.	Verbindung, Datenbank und Server	Die Rate, mit der Netzwerkdaten (in Byte) empfangen werden
Comm: Empfangene Byte unkomprimiert/Sek.	Verbindung, Datenbank und Server	Die Rate, mit der Byte bei deaktivierter Komprimierung empfangen worden wären
Comm: Gesendete Byte/Sek.	Verbindung, Datenbank und Server	Die Rate, mit der Byte über das Netzwerk übertragen werden
Comm: Gesendete Byte unkomprimiert/Sek.	Verbindung, Datenbank und Server	Die Rate, mit der Byte bei deaktivierter Komprimierung gesendet worden wären
Comm: Freie Puffer	Server	Die Anzahl der freien Netzwerkpuffer
Comm: Empfangene Mehrfach-Pakete/Sek.	Server	Die Rate, mit der Mehrfach-Paketzustellungen empfangen werden
Comm: Gesendete Mehrfach-Pakete/Sek.	Server	Die Rate, mit der Mehrfach-Paketzustellungen übertragen werden
Comm: Empfangene Pakete/Sek.	Verbindung, Datenbank und Server	Die Rate, mit der Netzwerkpakete empfangen werden
Comm: Empfangene Pakete unkomprimiert/Sek	Verbindung, Datenbank und Server	Die Rate, mit der Netzwerkpakete bei deaktivierter Komprimierung empfangen worden wären
Comm: Gesendete Pakete/Sek.	Verbindung, Datenbank und Server	Die Rate, mit der Netzwerkpakete übertragen werden
Comm: Gesendete Pakete unkomprimiert/Sek.	Verbindung, Datenbank und Server	Die Rate, mit der Netzwerkpakete bei deaktivierter Komprimierung übertragen worden wären
Comm: Remoteput-Warten/Sek.	Server	Die Rate, bei der die Kommunikationsverbindung warten muss, weil sie keine Puffer zum Senden von Daten zur Verfügung hat. Dieser Statistikwert wird nur für TCP/IP erfasst.

Statistik	Bereich	Beschreibung
Comm: Empfangene Anforderungen	Verbindung, Datenbank und Server	Die Anzahl der Client-/Server-Kommunikationsanforderungen oder Roundtrips. Dies unterscheidet sich von "Comm: Empfangene Pakete" dadurch, dass Mehrfach-Paketanforderungen als eine Anforderung zählen und Verfügbarkeitspakete nicht einbezogen werden.
Comm: Sendefehler/Sek.	Server	Die Rate, bei der die Basisprotokolle einen Sendefehler verzeichnet haben
Comm: Gesamtpuffer	Server	Die Gesamtzahl der Netzwerkpuffer
Comm: Eindeutige Clientadressen	Server	Die Anzahl der eindeutigen Client-Netzwerkadressen, die mit dem Datenbankserver verbunden sind. Dies ist normalerweise die Anzahl der verbundenen Client-Computer, und die Zahl kann kleiner sein als die Gesamtzahl der Verbindungen.

Statistiken über I/O-Vorgänge

Diese Statistiken verbinden Festplatten-Lesezugriffe und Festplatten-Schreibzugriffe, um zusammenfassende Informationen über den Umfang der Aktivitäten durch I/O-Vorgänge zu liefern.

Statistik	Bereich	Beschreibung
Platte: Akt. I/O	Datenbank	Die aktuelle Anzahl von Datei-I/O-Vorgängen des Datenbankservers, die noch nicht abgeschlossen sind
Platte: Max. Akt. I/O	Datenbank	Zeigt den maximalen Wert, den "Platte: Aktiv I/O" erreicht hat.

Statistiken über Festplatten-Lesezugriffe

Diese Statistiken beschreiben Umfang und Typ der Aktivität beim Schreiben von Informationen auf Festplatten.

Statistik	Bereich	Beschreibung
Festplatten-Lesezugriffe: Gesamtseiten/Sek.	Verbindung und Datenbank	Die Rate, mit der Seiten aus einer Datei gelesen werden
Pl.-Lesezugr.: Akt.	Datenbank	Die aktuelle Anzahl von Datei-Lesevorgängen des Datenbankservers, die noch nicht abgeschlossen sind

Statistik	Bereich	Beschreibung
Festplatten-Lesezugriffe: Index-intern/Sek.	Verbindung und Datenbank	Die Rate, mit der interne Knotenseiten des Indexes von der Festplatte gelesen werden
Festplatten-Lesezugriffe: Index-Blatt/Sek.	Verbindung und Datenbank	Die Rate, mit der Indexblattseiten von der Festplatte gelesen werden
Festplatten-Lesezugriffe: Tabelle/Sek	Verbindung und Datenbank	Die Rate, mit der Tabellenseiten von der Festplatte gelesen werden
Pl.-Lesezugr.: Max. Akt.	Datenbank	Zeigt den maximalen Wert, den "Festplatten-Lesezugriffe: Aktiv" erreicht hat.
Platten-Lesezugriffe: Arbeitstabelle	Verbindung und Datenbank	Die Rate, mit der Arbeitstabellenseiten von der Festplatte gelesen werden

Statistiken über Festplatten-Schreibzugriffe

Diese Statistiken beschreiben Umfang und Typ der Aktivität beim Schreiben von Informationen auf Festplatten.

Statistik	Bereich	Beschreibung
Pl.-Schr.zugr.: Akt.	Datenbank	Die aktuelle Anzahl von Datei-Schreibvorgängen des Datenbankservers, die noch nicht abgeschlossen sind
Pl.-Schr.zugr.: Max. Akt.	Datenbank	Zeigt den maximalen Wert, den "Festplatten-Schreibzugriffe: Aktiv" erreicht hat.
Festplatten-Schreibzugriffe: Dateifestschreibung/Sek.	Datenbank	Die Rate, mit der der Datenbankserver ein Wegschreiben des Festplattencaches erzwingt. Windows-Plattformen verwenden ungepufferte (direkte) I/O-Vorgänge, sodass der Plattencache nicht geleert werden muss.
Festplatten-Schreibzugriffe: Datenbank-Erweiterung/Sek.	Datenbank	Die Rate (in Seiten/Sekunde), mit der die Datenbankdatei erweitert wird
Festplatten-Schreibzugriffe: Temp-Erweiterung/Sek.	Datenbank	Die Rate (in Seiten/Sekunde), mit der temporäre Dateien erweitert werden
Festplatten-Schreibzugriffe: Seiten/Sek.	Verbindung und Datenbank	Die Rate, mit der geänderte Seiten auf die Festplatte geschrieben werden

Statistik	Bereich	Beschreibung
Festplatten-Schreibzugriffe: Transaktionslog/Sek.	Verbindung und Datenbank	Die Rate, mit der Seiten ins Transaktionslog geschrieben werden
Tr.Log-Gr.-Festschr./Sek.	Verbindung und Datenbank	Die Rate, mit der eine Festschreibung des Transaktionslogs angefordert wurde, wobei das Log aber bereits geschrieben wurde (sodass die Festschreibung umsonst war)

Indexstatistiken

Diese Statistiken beschreiben die Verwendung des Index.

Statistik	Bereich	Beschreibung
Index: Hinzufügungen/Sek.	Verbindung und Datenbank	Die Rate, mit der einem Index Einträge hinzugefügt werden
Index:Suchen/Sek.	Verbindung und Datenbank	Die Rate, mit der Einträge in Indizes gesucht werden
Index: Volle Vergleiche/Sek.	Verbindung und Datenbank	Die Rate, mit der Vergleiche zusätzlich zum Hash-Wert in einem Index durchgeführt werden müssen

Speicherdiagnose-Statistiken

Diese Statistikwerte geben Aufschluss darüber, wie der Datenbankserver den Arbeitsspeicher nutzt.

Statistik	Bereich	Beschreibung
Cache: Zuweisung von Mehrfachseiten	Server	Die Anzahl der Mehrfachseiten-Zuweisungen
Cache: Instabilität	Server	Die Anzahl der Fehlversuche des Cachemanagers, eine Seite für eine Zuweisung zu finden
Cache: Aufräumen bei Besuch	Server	Die Anzahl der Seiten, die besucht wurden, während für eine zuzuweisende Seite aufgeräumt wurde.
Cache: Aufräumvorgänge	Server	Die Anzahl der Aufräumvorgänge des Cachemanagers für eine zuzuweisende Seite

Statistik	Bereich	Beschreibung
Cacheseiten: Zugewiesene Strukturen	Server	Die Anzahl der Cacheseiten, die für Datenstrukturen des Datenbankservers zugewiesen wurden
Cacheseiten: Datei	Server	Die Anzahl der Cacheseiten, die für Daten aus Datenbankdateien benutzt werden
Cacheseiten: Datei verändert	Server	Die Anzahl der Cacheseiten, die verändert sind (geschrieben werden müssen).
Cacheseiten: Frei	Server	Die Anzahl der Cacheseiten, die nicht benutzt werden
Cacheseiten: verbraucht	Server	Die Anzahl der Seiten, die zurzeit nicht für die Wiederverwendung verfügbar sind
Cache-Ersetzungen: Lesezugr. Gesamts./Sek	Server	Die Rate, mit der Datenbankseiten aus dem Cache ausgelesen werden, um Platz für eine andere Seite zu machen
Heaps: Carver	Verbindung, Datenbank und Server	Die Anzahl der Heaps, die für kurzfristige Zwecke wie Abfrageoptimierung benutzt werden
Heaps: Abfrageverarbeitung	Verbindung, Datenbank und Server	Die Anzahl der Heaps, die für die Abfrageverarbeitung benutzt werden (Hash- und Sort-Vorgänge).
Heaps: Neu positionierbar	Verbindung, Datenbank und Server	Die Anzahl der neu positionierbaren Heaps
Heaps: Neu positionierbar gesperrt	Verbindung, Datenbank und Server	Die Anzahl der neu positionierbaren Heaps, die zurzeit im Cache gesperrt sind
Belegung physischen Speichers/Sek.	Server	Die Rate, mit der Fenster für den Adressraum auf Datenbankseiten dem physischen Speicher im Cache mithilfe von Address Windowing Extensions zugeordnet werden
Mem Pages: Carver	Verbindung, Datenbank und Server	Die Anzahl der Heap-Seiten, die für kurzfristige Zwecke wie die Abfrageoptimierung benutzt werden
Mem Pages: Verbrauchter Cursor	Verbindung, Datenbank und Server	Die Anzahl der Seiten, die benutzt werden, um Cursor-heaps im Speicher zu bewahren

Statistik	Bereich	Beschreibung
Mem Pages: Abfrageverarbeitung	Verbindung, Datenbank und Server	Die Anzahl der Cacheseiten, die für die Abfrageverarbeitung benutzt werden (Hash- und Sort-Vorgänge)
Abfragespeicher: Derzeit aktiv	Verbindung, Datenbank und Server	Zeigt die aktuelle Anzahl der Anforderungen an, die den Abfragespeicher aktiv nutzen.
Abfragespeicher: Geschätzt aktiv	Server	Zeigt die Schätzung des Datenbankservers zum Steady-state-Mittelwert der Anzahl der Aufgaben an, die aktiv den Abfragespeicher nutzen.
Abfragespeicher: Zusätzl. verfügbar	Server	Zeigt die Speichermenge an, die zusätzlich zur grundlegenden speicherintensiven Bereitstellung verfügbar ist.
Abfragespeicher: Anzahl Zugriffsfehlschläge	Verbindung, Datenbank und Server	Zeigt an, wie viele Male eine Anforderung insgesamt auf Zuweisung von Abfragespeicher gewartet und nicht erhalten hat.
Abfragespeicher: Anzahl Abfrageanforderungen	Verbindung, Datenbank und Server	Zeigt an, wie viele Male eine Anforderung versucht hat, Abfragespeicher zu belegen.
Abfragespeicher: Anzahl Abfrage-Warten	Verbindung, Datenbank und Server	Zeigt an, wie viele Male eine Anforderung insgesamt auf Speicherzuweisung gewartet hat.
Abfragespeicher: Bereitgestellte Seiten	Verbindung, Datenbank und Server	Zeigt die Anzahl der Seiten an, die derzeit für Anforderungen bereitgestellt sind.
Abfragespeicher: Wartende Abfragen	Verbindung, Datenbank und Server	Zeigt die aktuelle Anzahl der Anforderungen, die auf Zuweisung von Abfragespeicher warten.

Speicherseiten-Statistiken

Diese Statistiken beschreiben Umfang und Zweck des Speichers, der vom Datenbankserver verwendet wird.

Statistik	Bereich	Beschreibung
Mem Pages: Sperrentabelle	Datenbank	Die Anzahl von Seiten, die zum Speichern von Sperrendaten verwendet werden

Statistik	Bereich	Beschreibung
Mem Pages: Gesperrter Heap	Server	Die Anzahl der im Cache gesperrten Heap-Seiten
Mem Pages: Haupt-Heap	Server	Die Anzahl der Seiten, die für globale Datenstrukturen des Datenbankservers verwendet wurden
Mem Pages: Zuordnung von Seiten	Datenbank	Die Anzahl der Zuordnungen von Seiten, die zum Zugriff auf Sperrtabellen, Häufigkeitstabellen und Tabellenlayouts verwendet wurden
Mem Pages: Prozedurdefinitionen	Datenbank	Die Anzahl der verlegbaren Heap-Seiten, die für Prozeduren verwendet wurden
Mem Pages: Verlegbarkeit	Datenbank	Die Anzahl der Seiten, die für verlegbare Heaps (Cursors, Anweisungen, Prozeduren, Trigger, Ansichten etc.) verwendet werden
Mem Pages: Verlegungen/Sek.	Datenbank	Die Rate, mit der verlegbare Heap-Seiten von der temporären Datei gelesen werden
Mem Pages: Rollback-Log	Verbindung und Datenbank	Die Anzahl der Seiten im Rollback-Log
Mem Pages: Triggerdefinitionen	Datenbank	Die Anzahl der verlegbaren Heap-Seiten, die für Trigger verwendet wurden
Mem Pages: Ansichtsdefinitionen	Datenbank	Die Anzahl der verlegbaren Heap-Seiten, die für Ansichten verwendet wurden

Anforderungsstatistiken

Diese Statistiken beschreiben die Aktivität des Datenbankservers beim Beantworten der Anforderungen von Clientanwendungen.

Statistik	Bereich	Beschreibung
Cursor	Verbindung, Datenbank und Server	Die Anzahl der deklarierten Cursor, die derzeit vom Datenbankserver verwaltet werden
Cursor offen	Verbindung, Datenbank und Server	Die Anzahl der offenen Cursors, die derzeit vom Datenbankserver aufrechterhalten werden
Sperrenanzahl	Verbindung und Datenbank	Die Anzahl der Sperren

Statistik	Bereich	Beschreibung
Anforderungen/ Sek.	Server	Die Rate, mit der der Datenbankserver reagiert, um eine neue Anforderung zu verarbeiten oder die Verarbeitung einer laufenden Anforderung fortgesetzt wird
Anforderungen: Aktiv	Server	Die Anzahl der Datenbankserver-Threads, die derzeit eine Anforderung bearbeitet
Aufgaben: Exchange	Server	Die Anzahl der Datenbankserver-Threads, die derzeit für die parallele Ausführung einer Abfrage benutzt werden
Anforderungen: Ungeplant	Server	Die Anzahl der Anforderungen, die derzeit in einer Warteschlange auf einen verfügbaren Datenbankserver-Thread warten
Snapshot-Anzahl	Verbindung und Datenbank	Die Anzahl der aktiven Snapshots
Anweisungscache-Treffer	Verbindung, Datenbank und Server	Die Rate, mit der Anweisungsvorbereitungen, die vom Client in den Cache geschrieben wurden, vom Datenbankserver wieder verwendet werden
Anweisungscache-Fehlschläge	Verbindung, Datenbank und Server	Die Rate, mit der Anweisungsvorbereitungen, die vom Client in den Cache geschrieben wurden, vom Datenbankserver erneut vorbereitet werden müssen
Anw.Vorbereitungen	Verbindung und Datenbank	Die Rate, mit der Anweisungsvorbereitungen vom Datenbankserver verarbeitet werden
Anweisungen	Verbindung, Datenbank und Server	Die Anzahl der vorbereiteten Anweisungen, die derzeit vom Datenbankserver aufrechterhalten werden
Tr.-Festschreibungen	Verbindung, Datenbank und Server	Die Rate, mit der Festschreibungsanforderungen ausgeführt werden
Transaktions-Rollbacks	Verbindung, Datenbank und Server	Die Rate, mit der Rücksetzungsanforderungen ausgeführt werden

Benutzerdefinierte Statistiken

Diese Statistiken beschreiben die Aktivität in Bezug auf Werte, die von Ihrer Anwendung protokolliert werden.

Statistik	Bereich	Beschreibung
Benutzerdefinierte Rate: Counter1	Verbindung, Datenbank und Server	Die Rate des Werts für den benutzerdefinierten Zähler im Laufe der Zeit.
Benutzerdefinierte Rate: Counter2	Verbindung, Datenbank und Server	Die Rate des Werts für den benutzerdefinierten Zähler im Laufe der Zeit.
Benutzerdefinierte Rate: Counter3	Verbindung, Datenbank und Server	Die Rate des Werts für den benutzerdefinierten Zähler im Laufe der Zeit.
Benutzerdefinierte Rate: Counter4	Verbindung, Datenbank und Server	Die Rate des Werts für den benutzerdefinierten Zähler im Laufe der Zeit.
Benutzerdefinierte Rate: Counter5	Verbindung, Datenbank und Server	Die Rate des Werts für den benutzerdefinierten Zähler im Laufe der Zeit.
Benutzerdefinierte Rohdaten: Counter1	Verbindung, Datenbank und Server	Zeigt den aktuellen Wert für den benutzerdefinierten Zähler.
Benutzerdefinierte Rohdaten: Counter2	Verbindung, Datenbank und Server	Zeigt den aktuellen Wert für den benutzerdefinierten Zähler.
Benutzerdefinierte Rohdaten: Counter3	Verbindung, Datenbank und Server	Zeigt den aktuellen Wert für den benutzerdefinierten Zähler.
Benutzerdefinierte Rohdaten: Counter4	Verbindung, Datenbank und Server	Zeigt den aktuellen Wert für den benutzerdefinierten Zähler.
Benutzerdefinierte Rohdaten: Counter5	Verbindung, Datenbank und Server	Zeigt den aktuellen Wert für den benutzerdefinierten Zähler.

Siehe auch

- „Benutzerdefinierte Eigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]

Verschiedene Statistiken

Statistik	Bereich	Beschreibung
Verfügbare I/O	Server	Die aktuelle Anzahl von verfügbaren I/O-Kontrollblöcken
Verbindungsanzahl	Datenbank	Die Anzahl von Verbindungen zur Datenbank
Haupt-Heap-Byte	Server	Die Anzahl von Byte, die für globale Datenbankserver-Datenstrukturen verwendet werden

Statistik	Bereich	Beschreibung
Abfrage: Plan-Cache-Seiten	Verbindung und Datenbank	Die Anzahl der Seiten, die für die Speicherung der Ausführungspläne benutzt werden
Abfrage: Strategien für Speichermangel	Verbindung und Datenbank	Die Anzahl, mit der der Datenbankserver seinen Ausführungsplan während der Ausführung aufgrund von Speichermangel geändert hat
Abfrage: Materialisierte Zeilen/Sek.	Verbindung und Datenbank	Die Rate, mit der Zeilen während der Abfrageverarbeitung in Arbeitstabellen geschrieben werden
Anforderungen: GET DATA/Sek.	Verbindung und Datenbank	Die Rate, mit der eine Verbindung GET DATA-Anforderungen ausgibt
Temporäre Tabellen-seiten	Verbindung und Datenbank	Die Anzahl der Seiten in der temporären Datei, die für temporäre Tabellen verwendet werden
Versionsspeicherseiten	Datenbank	Die Anzahl der Seiten der temporären Datei, die derzeit für die Speicherung der Zeilenversion benutzt werden, wenn die Snapshot-Isolation aktiviert ist

Tipps zum Verbessern der Performance

Performancetipps für Server

- „Tipp: Performance durch den Einsatz des Cachespeichers steigern“ auf Seite 220
- „Tipp: Auf Parallelitätsprobleme prüfen“ auf Seite 228
- „Tipp: Ziel des Optimierers auswählen“ auf Seite 228
- „Tipp: Spaltenstatistiken aktualisieren“ auf Seite 229
- „Tipp: Indizes effizient verwenden“ auf Seite 231
- „Tipp: Verbessern der Index-Performance“ auf Seite 233
- „Tipp: Für gemischte oder OLAP-Arbeitslast optimieren“ auf Seite 233
- „Tipp: Strategisches Sortieren von Abfrageergebnissen“ auf Seite 233
- „Tipp: Den korrekten Cursor-Typ angeben“ auf Seite 233
- „Tipp: Explizite Selektivitätsschätzungen sparsam einsetzen“ auf Seite 234
- „Tipp: Verbessern der Abfrageperformance durch materialisierte Ansichten“ auf Seite 234
- „Tipp: Option WITH EXPRESS CHECK bei der Validierung von Tabellen verwenden“ auf Seite 238
- „Tipp: Arbeitstabellen in der Abfrageverarbeitung verwenden ("Alle Zeilen" als Optimierungsziel verwenden)“ auf Seite 239

Performancetipps für Anwendungen

- „Tipp: Aufbauen effizienter SQL-Abfragen“ auf Seite 240
- „Tipp: Abfrageperformance überwachen“ auf Seite 245
- „Tipp: Kostenträchtige benutzerdefinierte Funktionen reduzieren“ auf Seite 246
- „Tipp: Autocommit-Modus ausschalten“ auf Seite 246
- „Tipp: In-Memory-Modus verwenden“ auf Seite 247

Performancetipps für Datenbanken

- „Tipp: Stets ein Transaktionslog verwenden“ auf Seite 247
- „Tipp: Verzögerte Festschreibungen verwenden“ auf Seite 248
- „Tipp: Statistiken von kleinen Tabellen erfassen“ auf Seite 248
- „Tipp: Fragmentierung reduzieren“ auf Seite 248
- „Tipp: Tabellenstruktur normalisieren“ auf Seite 255
- „Tipp: Kaskadierende referenzielle Aktionen vermeiden“ auf Seite 256
- „Tipp: Integritätsregeln deklarieren“ auf Seite 256
- „Tipp: Verschiedene Dateien auf verschiedene Geräte platzieren“ auf Seite 256
- „Tipp: Ihre Datenbank neu aufsetzen“ auf Seite 257
- „Tipp: Mit Schlüsseln die Abfrageperformance steigern“ auf Seite 258
- „Tipp: Primärschlüssel-Breite vermindern“ auf Seite 258
- „Tipp: Tabellenbreiten vermindern“ auf Seite 259
- „Tipp: Spaltenreihenfolge in Tabellen überprüfen“ auf Seite 259
- „Tipp: Kostenträchtige Trigger ersetzen“ auf Seite 259
- „Tipp: Angemessene Seitengröße verwenden“ auf Seite 260
- „Tipp: Mit AUTOINCREMENT Primärschlüssel erstellen“ auf Seite 261
- „Tipp: Entsprechende Datentypen verwenden“ auf Seite 262
- „Tipp: Massenvorgangsmethoden verwenden“ auf Seite 262
- „Tipp: Ressourcenwächter verwenden“ auf Seite 262

Performancetipps für die Kommunikation

- „Tipp: Anforderungen zwischen Client und Server reduzieren“ auf Seite 263
- „Tipp: Komprimierung mit Umsicht verwenden“ auf Seite 263
- „Tipp: Paketgröße ändern, um die Performance zu steigern“ auf Seite 264

Tipp: Performance durch den Einsatz des Cachespeichers steigern

Der Cache ist ein Speicherbereich, der vom Datenbankserver verwendet wird, um Datenbankseiten für wiederholten schnellen Zugriff zu speichern. Je mehr Seiten im Cache verfügbar sind, desto seltener muss der Datenbankserver Daten von der Festplatte lesen. Die Cachegröße ist daher häufig ein Schlüsselfaktor für die Performance.

SQL Anywhere unterstützt die dynamische Cachedimensionierung, die die Cachegröße angemessen und automatisch optimiert, indem das System als Ganzes überwacht wird. Sie können jedoch auch beim Starten der Datenbank die Option -c in der Befehlszeile des Datenbankservers verwenden, um die Cachegröße für die Datenbank zu steuern.

Im Meldungsfenster des Datenbankservers wird die Cachegröße beim Start angezeigt, aber Sie können auch die folgende Anweisung ausführen, um die aktuelle Cachegröße abzurufen:

```
SELECT PROPERTY( 'CurrentCacheSize' );
```

Verschlüsselte Datenbanken müssen genügend Cache haben, um I/O-Vorgänge zu minimieren, weil diese Vorgänge in verschlüsselten Datenbanken kostenträchtiger als in unverschlüsselten Datenbanken sind, da für jeden Vorgang eine Verschlüsselung und/oder Entschlüsselung durchgeführt werden muss.

Siehe auch

- „Dynamische Cachedimensionierung“ auf Seite 224
- „Datenbankserveroption -c “ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -ca “ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -ch “ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -cl “ [*SQL Anywhere Server - Datenbankadministration*]

Cache und der Speichernutzungswächter

Der SQL Anywhere-Datenbankserver nutzt den Cache (Pufferpool), um Bilder von Datenbankseiten temporär zu speichern (zu puffern). Diese Seiten sind üblicherweise Tabellen- und Indexseiten, auch wenn verschiedene andere Arten von physischen Seiten in einer SQL Anywhere-Datenbank gespeichert werden können. Zusätzlich zu diesen Seiten nutzt der Datenbankserver den Cache für zwei weitere Speicherpools. Einer dieser Pools ist der virtuelle Speicher, der von Datenbankservern für Datenstrukturen verwendet wird, die Verbindungen, Anweisungen und Cursor darstellen. Der zweite Pool besteht aus Cache-Seiten, die als virtueller Speicher für den Abfragespeicher verwendet werden.

Die Abfrageausführung erfordert Speicher für einen effizienten Betrieb. SQL Anywhere verwendet einen Speichernutzungswächter, um festzulegen, wie viel Abfragespeicher jede Anweisung für die Abfrageausführung verwenden darf. Der Speichernutzungswächter ist dafür verantwortlich, einen Pool an Abfragespeicher Anweisungen zuzuweisen, um eine effiziente Ausführung der Systemlast zu gewährleisten.

Der Speichernutzungswächter teilt einzelnen Anweisungen eine ausgewählte Anzahl von Seiten zu, die die Anweisung dann für speicherintensive Abfrageverarbeitungsvorgänge verwenden kann. Der Speicher im Abfragespeicher-Pool steht weiterhin für andere Zwecke (wie Puffern von Tabellen- oder Indexseiten) zur Verfügung, bis der Abfrageprozessor die Seiten verwendet. Zu den speicherintensiven Abfrageverarbeitungsvorgängen, die den Abfragespeicher verwenden, gehören alle Hash-basierten Operatoren, z.B. Hash-Distinct, Hash-Group-By und Hash-Join, sowie Sortierungs- und Fensteroperatoren.

Mithilfe der folgenden Einstellungen, Operatoren und Statistiken können Sie verstehen und steuern, wie der Speichernutzungswächter den Cache verwendet:

- **QueryMemMaxUseful-Operator im grafischen Plan** Wenn eine Anweisung ausgeführt wird, verwendet der Speichernutzungswächter die Schätzungen des Optimierers, um zu bestimmen, wie viel Speicher für die Anweisung sinnvoll (useful) wäre. Diese Schätzung wird im grafischen Plan als QueryMemMaxUseful dargestellt.
- **QueryMemActiveMax-Servereigenschaft** Der Speichernutzungswächter beschränkt die Anzahl der speicherintensiven Anforderungen, die gleichzeitig ausgeführt werden können. Dieser Höchstwert wird basierend auf den Performance-Merkmalen des Computers ermittelt, auf dem der Datenbankserver läuft, und dieses Limit wird mit der Servereigenschaft QueryMemActiveMax angezeigt.
- **QueryMemActiveEst-Systemmonitorstatistik** Der Speichernutzungswächter führt eine laufende Schätzung der Anzahl der gleichzeitigen, speicherintensiven Anforderungen durch. Diese Schätzung ist in Form der Datenbankservereigenschaft bzw. Systemmonitorstatistik QueryMemActiveEst verfügbar.

- **query_mem_timeout-Datenbankoption** Wenn eine speicherintensive Anweisung mit der Ausführung beginnt und bereits die maximale Anzahl von gleichzeitigen speicherintensiven Anforderungen ausgeführt wird, dann warten die eingehenden Anweisungen darauf, dass eine der bestehenden Anforderungen ihren zugewiesenen Speicher freigibt. Die Datenbankoption `query_mem_timeout` steuert, wie lange die eingehende Anforderung auf eine Speicherzuweisung wartet. Bei der Standardeinstellung -1 wartet die Anforderung für eine im Datenbankserver definierte Zeitspanne. Wenn nach der Wartezeit keine Speicherzuweisung verfügbar ist, wird der Zugriffsplan der Anweisung mit wenig Speicher ausgeführt, was zu einer langsamen Ausführung führen kann, möglicherweise unter Verwendung einer Niedrig-Speicher-Ausführungsstrategie, falls es im Plan eine solche für speicherintensive physische Operatoren gibt.
- **QueryMemGrantWaiting-Servereigenschaft und -Systemmonitorstatistik** Die Datenbankservereigenschaft bzw. Systemmonitorstatistik `QueryMemGrantWaiting` zeigt die aktuelle Anzahl der Anforderungen, die auf eine Speicherzuweisung warten.
- **QueryMemGrantWaited-Servereigenschaft und -Systemmonitorstatistik** Die Datenbankservereigenschaft bzw. Systemmonitorstatistik `QueryMemGrantWaited` zeigt, wie viele Male eine Anforderung insgesamt warten musste, bevor eine Speicherzuweisung erteilt wurde.
- **QueryMemNeedsGrant-Operator im grafischen Plan** Im grafischen Plan zeigt der `QueryMemNeedsGrant`-Operator an, ob der Speichernutzungswächter die Anforderung als einfache Anforderung (keine Speicherzuweisung erforderlich) oder als speicherintensive Anforderung (Speicherzuweisung erforderlich) einstuft. Wenn der Speichernutzungswächter eine Anforderung als einfach klassifiziert, d.h. dass keine Speicherzuweisung erforderlich ist, beginnt die Anforderung umgehend mit der Ausführung. Andernfalls fordert die Anforderung einen Prozentsatz des Abfragespeicher-Pools an.
- **QueryMemLikelyGrant-Operator im grafischen Plan** Der `QueryMemLikelyGrant`-Operator im grafischen Plan zeigt eine Schätzung an, wie viele Seiten der Anforderung wahrscheinlich für die Ausführung zugewiesen werden.

Siehe auch

- „Grafischer Plan mit Statistiken“ auf Seite 348
- `QueryMemActiveMax`-Servereigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]
- `QueryMemPages`-Servereigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]
- `QueryMemPercentOfCache`-Servereigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]
- „`query_mem_timeout`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserverkonfiguration der Multiprogramming-Stufe“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption -ch“ [[SQL Anywhere Server - Datenbankadministration](#)]

Cache und der Optimierer

Das Reservieren von Speicher, um zum Beispiel den Inhalt von einem Cursor aufzunehmen, kann kostenträchtig sein. Wenn der Cache voll ist, müssen möglicherweise eine oder mehrere Seiten auf die Festplatte geschrieben werden, um Platz für neue Seiten zu machen. Manche Seiten müssen möglicherweise wiederholt gelesen werden, um den nachfolgenden Vorgang durchzuführen. In diesem Fall rechnet SQL Anywhere mit höheren Kosten bei Ausführungsplänen, die einen zusätzlichen

Puffercache-Overhead erfordern. Diese Kosten bewirken, dass der Optimierer versucht, die Verwendung von Arbeitstabellen zu vermeiden. Der Optimierer achtet jedoch darauf, Speicher dort zu nutzen, wo dies die Performance steigert. Beispielsweise werden die Ergebnisse von Unterabfragen im Cache abgelegt, wenn sie während der Abfrageverarbeitung wiederholt benötigt werden.

Cachespeichernutzung beschränken

Die anfänglichen Werte für Mindest- und Höchstcachegröße können über die Befehlszeile des Datenbankservers gesteuert werden.

- **Anfängliche Cachegröße** Sie können die anfängliche Cachegröße für den Datenbankserver mit der Option `-c` festlegen. Wenn Sie die Option `-c` nicht angeben, berechnet der Datenbankserver die anfängliche Cachezuweisung.
- **Maximale Cachegröße** Den Maximalwert für die Cachegröße können Sie über die Option `-ch` des Datenbankservers steuern. Der Standardwert basiert auf einem heuristischen Wert, der vom physischen Speicher Ihres Computers abhängt. Unter Windows Mobile wird standardmäßig ein Maximalwert benutzt, der der Größe des verfügbaren Programmspeichers minus 4 MB entspricht. Auf anderen Nicht-Unix-Computern ist standardmäßig die Obergrenze etwa der niedrigere Wert der maximal verfügbaren Cachegröße und 90% des physischen Systemspeichers auf dem Computer. Unter Unix wird der Standardwert für die Cachegröße folgendermaßen berechnet:
 - Auf Unix-Plattformen mit 32 Bit ist es der kleinere Wert von 90 % des physischen Speichers oder ein Wert von 1.834.880 kB.
 - Auf Unix-Plattformen mit 64 Bit ist es der kleinere Wert von 90 % des physischen Speichers oder ein Wert von 8.589.672.320 kB.
- **Minimale Cachegröße** Den Minimalwert für die Cachegröße können Sie über die Option `-cl` des Datenbankservers steuern. Standardmäßig ist die minimale Cachegröße identisch mit der anfänglichen Cachegröße, außer unter Windows Mobile. Unter Windows Mobile beträgt die minimale Cachegröße 600 kB.

Wenn Sie die Serveroption `-c` ohne `-cl` angeben, wird die minimale Cachegröße auf die anfängliche Cachegröße festgelegt, die mit der Serveroption `-c` angegeben wurde.

Wenn Sie keine der beiden Serveroptionen `-c` und `-cl` angeben, wird die minimale Cachegröße auf einen sehr niedrigen hartkodierten konstanten Wert festgelegt, sodass der Cache bei Bedarf kleiner werden kann. Unter Windows beträgt dieser Wert 2 MB, unter Unix ist er 8 MB und unter Windows Mobile beträgt er 600 kB.

Wenn Sie versuchen, Ihre anfängliche oder minimale Cachegröße auf einen Wert, der kleiner ist als ein Achtel der maximalen Cachegröße zu setzen, werden die anfängliche bzw. die minimale Cachegröße automatisch um einen in Ansehung der maximalen Cachegröße berechneten Betrag erhöht.

Sie können die dynamische Cachedimensionierung auch mit der Serveroption `"-ca 0"` deaktivieren.

Die folgenden Datenbankservereigenschaften liefern Informationen über den Cache des Datenbankservers:

- **CurrentCacheSize** Gibt die aktuelle Cachegröße in kB zurück.
- **MinCacheSize** Gibt die minimale zulässige Cachegröße in kB zurück
- **MaxCacheSize** Gibt die maximale zulässige Cachegröße in kB zurück
- **PeakCacheSize** Gibt den höchsten Wert zurück, den der Cache in der aktuellen Sitzung erreicht hat, in kB.

Siehe auch

- Cachegröße [\[SQL Anywhere Server - Datenbankadministration\]](#)
- „Liste der Datenbankservereigenschaften“ [\[SQL Anywhere Server - Datenbankadministration\]](#)
- „Datenbankserveroption -c“ [\[SQL Anywhere Server - Datenbankadministration\]](#)
- „Datenbankserveroption -ca“ [\[SQL Anywhere Server - Datenbankadministration\]](#)
- „Datenbankserveroption -ch“ [\[SQL Anywhere Server - Datenbankadministration\]](#)
- „Datenbankserveroption -cl“ [\[SQL Anywhere Server - Datenbankadministration\]](#)

Dynamische Cachedimensionierung

Mithilfe von SQL Anywhere können Sie die Größe des Datenbankcaches automatisch ändern, während der Datenbankserver läuft. Bei voller **dynamischer Cachedimensionierung** wächst der Cache, wenn dem Datenbankserver mehr Speicher zur Verfügung steht, und wird kleiner, wenn der Cachespeicher von anderen Anwendungen benötigt wird. Die Effizienz der dynamischen Cachedimensionierung variiert je nach Betriebssystem und nach Menge des verfügbaren physischen Speichers.

In der Regel werden Cacheanforderungen über die dynamische Cachedimensionierung einmal pro Minute beurteilt. Wenn allerdings eine neue Datenbank gestartet wird oder eine Datei signifikant wächst, kann das Beurteilungsintervall für dreißig Sekunden auf einmal alle fünf Sekunden reduziert werden. Nach den ersten dreißig Sekunden fällt das Beurteilungsintervall zurück auf einmal pro Minute. Das Wachstum einer Datei wird als signifikant angesehen, wenn es um ein Achtel größer ist als das letzte Wachstum, das eine Erhöhung in der Bewertungshäufigkeit ausgelöst hat, oder ein Achtel größer als zu dem Zeitpunkt, zu dem die Datenbank gestartet wurde.

Wenn die dynamische Cachedimensionierung aktiv ist, brauchen Sie den Datenbankcache nicht explizit zu konfigurieren.

Wenn Sie versuchen, Ihre anfängliche oder minimale Cachegröße auf einen Wert, der kleiner ist als ein Achtel der maximalen Cachegröße zu setzen, werden die anfängliche bzw. die minimale Cachegröße automatisch um einen in Ansehung der maximalen Cachegröße berechneten Betrag erhöht.

Siehe auch

- „Datenbankserveroption -ca“ [\[SQL Anywhere Server - Datenbankadministration\]](#)
- „Datenbankserveroption -ch“ [\[SQL Anywhere Server - Datenbankadministration\]](#)
- „Datenbankserveroption -cl“ [\[SQL Anywhere Server - Datenbankadministration\]](#)
- Cachegröße [\[SQL Anywhere Server - Datenbankadministration\]](#)

Dynamische Cachedimensionierung unter Windows

Unter Windows und Windows Mobile beurteilt der Datenbankserver einmal pro Minute die statistischen Daten für den Cache und für das Betriebssystem und ermittelt eine optimale Cachegröße. Der Datenbankserver berechnet eine Zielcachegröße, die den gesamten, derzeit nicht benutzten Speicher nutzt, außer rund 5 MB, die für das System freigelassen werden müssen. Die Zielcachegröße ist nie kleiner als die angegebene bzw. implizite minimale Cachegröße. Die Zielcachegröße übersteigt nie die angegebene bzw. implizite maximale Cachegröße oder die Summe aller offenen Datenbanken und temporären Dateien plus der Größe des Haupt-Heaps.

Um schnelle Schwankungen der Cachegröße zu vermeiden, erhöht der Datenbankserver die Cachegröße inkrementell. Die Cachegröße wird also nicht sofort auf den Zielwert heraufgesetzt, sondern bei jeder Anpassung um 75 % der Differenz zwischen der aktuellen und der Zielcachegröße geändert.

Dynamische Cachedimensionierung unter Unix

Bei Unix verwendet der Datenbankserver Auslagerungsspeicher und Arbeitsspeicher zum Verwalten der Cachegröße. Der Auslagerungsspeicher ist eine systemweite Ressource bei den meisten Unix-Betriebssystemen. In diesem Abschnitt wird die Summe aus Arbeitsspeicher und Auslagerungsspeicher **Systemressourcen** genannt. Weitere Hinweise finden Sie in Ihrer Dokumentation zum Betriebssystem.

Beim Starten weist die Datenbank die angegebene maximale Cachegröße von den Systemressourcen zu. Sie lädt einen Teil in den Speicher (die anfängliche Cachegröße) und behält den Rest als Auslagerungsspeicher.

Die Summe der vom Datenbankserver benutzten Systemressourcen ist konstant, bis der Datenbankserver heruntergefahren wird, aber das Verhältnis der in den Speicher geladenen Daten ändert sich. Jede Minute bewertet der Datenbankserver die statistischen Daten zum Cache und zum Betriebssystem. Wenn der Datenbankserver arbeitet und Speicher anfordert, kann er Cacheseiten vom Auslagerungsspeicher in den Arbeitsspeicher verlegen. Wenn die anderen Prozesse im System Speicher erfordern, entfernt der Datenbankserver möglicherweise Cacheseiten aus dem Speicher, um Speicherplatz auszulagern.

Anfängliche Cachegröße

Standardmäßig ist die anfängliche Cachegröße aufgrund heuristischer Daten in Bezug auf die verfügbaren Systemressourcen zugewiesen. Die anfängliche Cachegröße ist immer niedriger als das 1,1-fache der gesamten Datenbankgröße.

Wenn die anfängliche Cachegröße über drei Viertel der verfügbaren Systemressourcen beträgt, bricht der Datenbankserver mit der Fehlermeldung "Zu wenig Speicher" ab.

Sie können die anfängliche Cachegröße mit der Option -c ändern.

Maximale Cachegröße

Die maximale Cachegröße muss niedriger sein als die verfügbaren Systemressourcen des Computers. Standardmäßig wird die maximale Cachegröße aufgrund heuristischer Daten zugewiesen, die auf den verfügbaren Systemressourcen und dem gesamten Speicher des Computers basieren. Die Cachegröße übersteigt nie die angegebene bzw. implizite maximale Cachegröße oder die Summe aller offenen Datenbanken und temporären Dateien plus der Größe des Haupt-Heaps.

Wenn Sie eine maximale Cachegröße angeben, die größer ist als die verfügbaren Systemressourcen, bricht der Datenbankserver mit der Fehlermeldung "Zu wenig Speicher" ab. Wenn Sie eine maximale Cachegröße angeben, die größer als der verfügbare Speicher ist, warnt der Datenbankserver vor Performance-Verschlechterung, bricht jedoch nicht ab.

Der Datenbankserver weist die gesamte *maximale* Cachegröße von den Systemressourcen zu und gibt sie erst frei, wenn der Datenbankserver beendet wird. Sie sollten darauf achten, eine maximale Cachegröße zu wählen, die eine gute SQL Anywhere-Performance gewährleistet, jedoch Speicherplatz für andere Anwendungen frei lässt. Die Formel für die standardmäßige maximale Cachegröße ist ein heuristischer Wert, der versucht, dieses Gleichgewicht zu erzielen. Sie brauchen lediglich den Wert abzustimmen, wenn der Standardwert für das System ungeeignet ist.

Sie können die Serveroption `-ch` zur Bestimmung der maximalen Cachegröße und zum Begrenzen der automatischen Cachevergrößerung verwenden.

Minimale Cachegröße

Wenn die Option `-c` angegeben ist, ist die Mindestcachegröße identisch mit der anfänglichen Cachegröße. Wenn die Option `-c` nicht angegeben ist, beträgt die minimale Cachegröße unter UNIX 8 MB.

Sie können die Serveroption `-cl` verwenden, um die minimale Cachegröße anzupassen.

Wenn Sie versuchen, Ihre anfängliche oder minimale Cachegröße auf einen Wert, der kleiner ist als ein Achtel der maximalen Cachegröße zu setzen, werden die anfängliche bzw. die minimale Cachegröße automatisch um einen in Ansehung der maximalen Cachegröße berechneten Betrag erhöht.

Siehe auch

- „Datenbankserveroption `-c`“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption `-ch`“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption `-cl`“ [[SQL Anywhere Server - Datenbankadministration](#)]

Statistiken zum Überwachen der Cachegröße

Die folgenden statistischen Daten stehen im Windows-Systemmonitor und den Eigenschaftsfunktionen der Datenbank zur Verfügung.

- **CurrentCacheSize** Die aktuelle Cachegröße in Kilobyte.
- **MinCacheSize** Die minimal zulässige Cachegröße in Kilobyte.
- **MaxCacheSize** Die maximal zulässige Cachegröße in Kilobyte.
- **PeakCacheSize** Die Spitzencachegröße in Kilobyte.

Siehe auch

- [CurrentCacheSize-Servereigenschaft \[SQL Anywhere Server - Datenbankadministration\]](#)
- [MinCacheSize-Servereigenschaft \[SQL Anywhere Server - Datenbankadministration\]](#)
- [MaxCacheSize-Servereigenschaft \[SQL Anywhere Server - Datenbankadministration\]](#)
- [PeakCacheSize-Servereigenschaft \[SQL Anywhere Server - Datenbankadministration\]](#)
- [„Liste der Datenbankservereigenschaften“ \[SQL Anywhere Server - Datenbankadministration\]](#)
- [„Datenbankperformance überwachen“ auf Seite 201](#)

Cachevorwärmung

Die Cachevorwärmung dient dazu, die Ausführungszeiten der ersten Abfragen zu verkürzen, die gegenüber einer Datenbank ausgeführt werden. Das wird erreicht, indem der Cache des Datenbankservers mit Datenbankseiten vorgeladen wird, die das letzte Mal, als die Datenbank gestartet wurde, referenziert wurden. Das Aufwärmen des Caches kann die Performance **nur** verbessern, wenn dieselbe Abfrage oder ähnliche Abfragen gegenüber einer Datenbank jedes Mal ausgeführt werden, wenn sie gestartet wird. Wenn sich die beim Start der Datenbank ausgeführten Anweisungen aber von jenen unterscheiden, die ausgeführt wurden, als die Datenbank zum letzten Mal mit eingeschalteter Cachesammlung gestartet wurde, verbessert die Cachevorwärmung die Performance nicht.

Sie steuern die Einstellung der Cachevorwärmung über die Datenbankserver-Befehlszeile. Es gibt zwei Aktivitäten, die stattfinden können, wenn eine Datenbank gestartet wird und die Cachevorwärmung aktiviert ist: Sammeln von Datenbankseiten und Cache-Zurückladen (Vorwärmen).

Das Sammeln von referenzierten Datenbankseiten wird von der Datenbankserveroption `-cc` gesteuert und ist standardmäßig aktiviert. Wenn das Datenbankseiten-Sammeln aktiviert ist, protokolliert der Datenbankserver jede angeforderte Datenbankseite ab dem Datenbankstart, bis eines der folgenden Ereignisse eintritt: Die maximale Anzahl von Seiten wurde gesammelt (der Wert basiert auf der Cache- und Datenbankgröße), die gesammelte Menge unterschreitet den Minimum-Schwellenwert oder die Datenbank wird heruntergefahren. Der Datenbankserver steuert die maximale Anzahl von Seiten und den Sammel-Schwellenwert. Sobald das Sammeln abgeschlossen ist, werden die referenzierten Seiten in der Datenbank aufgezeichnet, damit sie zum Vorwärmen des Caches verwendet werden können, wenn die Datenbank das nächste Mal gestartet wird.

Die Cachevorwärmung (Zurückladen) ist standardmäßig aktiviert und wird von der Datenbankserveroption `-cr` gesteuert. Um den Cache vorzuwärmen, überprüft der Datenbankserver, ob die Datenbank eine früher aufgezeichnete Sammlung von Seiten enthält. Wenn dies der Fall ist, lädt der Datenbankserver die entsprechenden Seiten in den Cache. Während der Cache Seiten lädt kann die Datenbank weiterhin Anforderungen verarbeiten, aber das Vorwärmen wird möglicherweise beendet, wenn eine signifikante Menge von Eingabe-/Ausgabeaktivitäten in der Datenbank festgestellt wird. Die Cachevorwärmung wird in diesem Fall gestoppt, um eine Performance-Minderung bei Abfragen zu vermeiden, die auf Seiten zugreifen, welche nicht in der Gruppe von in den Cache zurückgeladenen Seiten enthalten sind. Sie können die Option `-cv` angeben, wenn Meldungen über die Cachevorwärmung im Fenster "Datenbankservermeldungen" angezeigt werden sollen.

Siehe auch

- „Datenbankserveroption -cc“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption -cr“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption -cv“ [[SQL Anywhere Server - Datenbankadministration](#)]

Tipp: Auf Parallelitätsprobleme prüfen

Wenn der Datenbankserver eine Transaktion bearbeitet, kann er eine oder mehrere Tabellenzeilen sperren. Die Sperren erhalten die Zuverlässigkeit der in der Datenbank gespeicherten Informationen, indem der gleichzeitige Zugriff von anderen Transaktionen verhindert wird. Sie verbessern auch die Genauigkeit der Abfrageergebnisse, indem sie erkennen lassen, welche Daten gerade aktualisiert werden.

Der Datenbankserver setzt diese Sperren automatisch und bedarf keiner expliziten Instruktion. Alle Sperren, die von einer Transaktion gesetzt werden, bleiben aktiv, bis die Transaktion abgeschlossen ist. Die Transaktion, die auf die Zeile zugreift, hält auch die Sperre. Abhängig vom Typ der Sperre haben andere Transaktionen begrenzten Zugriff auf die gesperrte Zeile oder gar keinen.

Die Performance kann leiden, wenn auf eine oder mehrere Zeilen häufig von mehreren Benutzern gleichzeitig zugegriffen wird. Wenn Sie Sperrenprobleme vermuten, können Sie die Prozedur "sa_locks" verwenden, um Informationen über Sperren in der Datenbank zu erhalten.

Wenn Probleme mit Sperren festgestellt werden, können Sie über die Verbindungseigenschaft "AppInfo" Informationen zu den beteiligten Verbindungsprozessen anzeigen.

Siehe auch

- „sa_locks-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Liste der Verbindungseigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]

Tipp: Ziel des Optimierers auswählen

Die Option "optimization_goal" legt fest, ob SQL Anywhere SQL-Anweisungen in Bezug auf Antwortzeiten ("erste Zeile") oder Ressourcenverbrauch ("alle Zeilen") optimiert. Einfacher gesagt, können Sie wählen, ob die Abfrageverarbeitung dahingehend optimiert werden soll, dass die erste Zeile schnell zurückgegeben wird, oder dass die Kosten für die Rückgabe der gesamten Ergebnismenge minimiert werden sollen.

Wenn die Option auf "First-row" gesetzt ist, wählt SQL Anywhere einen Zugriffsplan, der die Zeitspanne zum Abruf der ersten Zeile der Ergebnismenge reduzieren soll, worunter möglicherweise die Gesamtabrufzeit leidet. Insbesondere wird der Optimierer in der Regel Zugriffspläne vermeiden, die die Materialisierung von Ergebnissen erfordert, um die Zeitspanne zur Darstellung der ersten Zeile zu verkürzen. Mit dieser Einstellung zum Beispiel bevorzugt der Optimierer Zugriffspläne, die für die ORDER BY-Klausel einer Abfrage einen Index benutzen, anstelle von Zugriffsplänen, die einen expliziten Sortiervorgang benötigen.

Das Ziel der Optimierung, das der Optimierer bei einer bestimmten Anweisung verwendet, wird von folgenden Regeln bestimmt:

- Wenn der Hauptabfragenblock eine Tabelle in der FROM-Klausel hat, bei der der Tabellen-Hint auf FASTFIRSTROW gesetzt ist, wird die Anweisung unter Verwendung der First-row-Optimierung optimiert.
- Wenn die Anweisung eine OPTION-Klausel hat, die eine Einstellung für die Option "optimization_goal" enthält, wird die Anweisung unter Verwendung dieser Einstellung optimiert.
- Ansonsten verwendet der Optimierer die aktuelle Einstellung der Option "optimization_goal".

Der Optimierer ist möglicherweise nicht in der Lage, einen Plan zu finden, der die erste Zeile schnell zurückzugeben kann, selbst wenn das Ziel der Optimierung "First-row" ist. Anweisungen, die eine Materialisierung auf Grund von DISTINCT-, GROUP BY- oder ORDER BY-Klauseln erfordern und für die kein relevanter Index existiert, um die benötigte Reihenfolge festzulegen, werden mit dem Ziel "All-rows" optimiert.

Wenn die Option auf "All-rows" (Standardwert) gesetzt ist, wird die SQL Anywhere-Abfrage optimiert, um einen Zugriffsplan mit der geschätzten Mindestabruflzeit zu verwenden. Die Einstellung "All-rows" für "optimization_goal" ist in der Regel für Anwendungen sinnvoll, die die komplette Ergebnismenge verarbeiten sollen, wie z.B. PowerBuilder DataWindow-Anwendungen.

Siehe auch

- „optimization_goal-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „FROM-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Sie können die OPTION-Klausel von SQL-Anweisungen auch wie folgt referenzieren:

- „DELETE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „INSERT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „MERGE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „UPDATE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „UNION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Tipp: Spaltenstatistiken aktualisieren

Spaltenstatistiken werden in der Datenbank in der ISYSCOLSTAT-Systemtabelle permanent gespeichert. Um die Performance des Optimierers ständig zu verbessern, aktualisiert der Optimierer die Spaltenstatistiken automatisch während der Verarbeitung aller SELECT-, INSERT-, UPDATE- und DELETE-Anweisungen, einschließlich Anweisungen für eine einzelne Zeile. Dazu überwacht der Optimierer die Anzahl der Zeilen, die einem Prädikat entsprechen, welches eine Tabelle oder Spalte referenziert, vergleicht diese Anzahl mit der Anzahl der geschätzten Zeilen und aktualisiert dann, falls erforderlich, die vorhandenen Statistiken.

Dann kann der Optimierer auf der Grundlage genauerer Spaltenstatistiken bessere Schätzungen berechnen und die Performance der nachfolgenden Abfragen verbessern.

Sie können über die Datenbankoptionen einstellen, ob die Spaltenstatistiken aktualisiert werden sollen. Mit der Datenbankoption update_statistics wird festgelegt, ob die Spaltenstatistiken während der

Ausführung von Abfragen aktualisiert werden. Mit der Datenbankoption `collect_statistics_on_dml_updates` wird festgelegt, ob die Statistiken während der Ausführung von datenverändernden DML-Anweisungen wie `LOAD`, `INSERT`, `DELETE` und `UPDATE` aktualisiert werden.

Wenn Sie vermuten, dass die Performance beeinträchtigt wird, weil Ihre Statistiken die aktuellen Spaltenwerte falsch wiedergeben, können Sie die Anweisungen `CREATE STATISTICS` oder `DROP STATISTICS` benutzen. Mit `CREATE STATISTICS` werden die alten Statistiken gelöscht und neue erstellt, während mit `DROP STATISTICS` lediglich die alten Statistiken gelöscht werden.

Wenn Sie die Anweisung `CREATE INDEX` ausführen, werden die Statistiken für den Index automatisch erstellt.

Wenn Sie die Anweisung `LOAD TABLE` ausführen, werden die Statistiken für die Tabelle automatisch erstellt.

Siehe auch

- „`SYSCOLSTAT`-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`CREATE STATISTICS`-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`ALTER STATISTICS`-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`DROP STATISTICS`-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`update_statistics`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „`collect_statistics_on_dml_updates`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Verwaltung von Statistiken durch den Statistikwächter

Zusätzlich zur automatischen Anpassung der Spaltenstatistiken, die bei der Ausführung einer Abfrage ausgeführt wird, überwacht der Statistikwächter auch die Funktionsfähigkeit und die Verwendung der Optimiererstatistik. Der Statistikwächter berechnet automatisch die Funktionsfähigkeit und Nützlichkeit aller Statistiken in der Datenbank und führt die erforderlichen Vorgänge aus, damit die Statistiken sich selbst überwachen und korrigieren. Die Statistiken werden im Hintergrund verwaltet, sodass die Performance des Datenbankservers kaum beeinflusst wird.

Der Statistikwächter führt die folgenden Aufgaben aus:

- Aufzeichnung der statistischen Daten und Schätzungsfehler aus Abfragerückmeldungen
- Korrektur oder Neuerstellung von Statistiken mit einer niedrigen Genauigkeit
- Stoppen der automatischen Verwaltung von Statistiken, die nicht auf effiziente Weise verwaltet werden können
- Erstellung potenziell nützlicher statistischer Daten
- Nicht verwendete Statistiken löschen

Die `update_statistics`-Option steuert, ob die angegebene Verbindung Abfragerückmeldungen an den Statistikwächter senden kann. Wenn diese Option auf "Off" gesetzt ist, erhält der Statistikwächter keine Abfragerückmeldungen von der festgelegten Verbindung. Der Statistikwächter kann jedoch weiterhin Abfragerückmeldungen von anderen Verbindungen erhalten und die Verwaltung von Statistiken durchführen.

Der Statistikwächter entscheidet basierend auf ihrer Integrität und Verwendung, wann eine Statistik korrigiert oder erstellt werden muss. Eine Statistik kann entweder durch die Erfassung von Statistiken während der Durchführung einer Abfrage oder durch einen separaten Prozess, der als Statistikaufräumvorgang bezeichnet wird, korrigiert oder erstellt werden. Sie können den Statistikaufräumvorgang mithilfe der StatisticsCleaner-Option für die sa_server_option-Systemprozedur deaktivieren, ohne den Statistikwächter zu deaktivieren. Wenn der Statistikwächter ausgeschaltet ist, werden Statistiken jedoch nur erstellt oder korrigiert, wenn eine Abfrage ausgeführt wird.

Um die Server-Arbeitslast zu verringern, stoppt der Statistikwächter die Verwaltung von schwer zu korrigierenden oder nie benutzten Statistiken. Korrigierte Statistiken, die häufig innerhalb eines kurzen Zeitraums korrigiert wurden und weiterhin schlechte Schätzungen zurückgeben, werden gelöscht und nicht 30 Tage lang verwaltet. Gelöschte Statistiken werden nach 30 Tagen neu erstellt und die reguläre Verwaltung wird wieder aufgenommen. Sie können diese Funktion mit der DropBadStatistics-Option für die sa_server_option-Systemprozedur deaktivieren. Statistiken, die in den letzten 90 Tagen nicht benutzt wurden, werden ebenfalls gelöscht. Verwenden Sie zum Deaktivieren dieser Funktion die DropUnusedStatistics-Option für die sa_server_option-Systemprozedur. Sie können die Verwaltung einer Statistik jederzeit mithilfe der Anweisungen CREATE STATISTICS, DROP STATISTICS, oder ALTER STATISTICS wieder aufnehmen.

Statistiken werden nur für Tabellen überwacht, die in den Speicher geladen wurden, und diese Statistiken werden alle 30 Minuten weggeschrieben. Während des Wegschreibens wird die Funktionsfähigkeit und die Verwendung der Statistiken geprüft und der Statistikwächter führt die Verwaltung der Statistikdaten durch. Die Statusinformationen über eine Statistik (wie etwa Zustand, Verwendung und Informationen darüber, wann eine Statistik aktualisiert oder gelöscht werden muss) werden zwischen Sitzungen nicht gespeichert. Die Statusinformationen werden zurückgesetzt, wenn der Datenbankserver heruntergefahren wird.

Siehe auch

- „CREATE STATISTICS-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ALTER STATISTICS-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DROP STATISTICS-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „update_statistics-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sa_server_option-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Tipp: Indizes effizient verwenden

SQL Anywhere kann Suchbedingungen häufig mithilfe von Indizes auswerten. Indem er einen Index verwendet, kann der Optimierer den Zugriff auf Daten beschleunigen und die Menge der aus den Basistabellen zu lesenden und zu verarbeitenden Daten verringern. Beispiel: Wenn eine Abfrage eine Suchbedingung WHERE column-name=value enthält und die Spalte einen Index aufweist, kann ein Index-Scan verwendet werden, um nur die Zeilen der Tabelle zu lesen, die die Suchbedingung erfüllen. Auch beim Verknüpfen von Tabellen steigern Indizes die Performance dramatisch.

Wenn Sie eine Abfrage ausführen, entscheidet SQL Anywhere, wie auf die jeweilige Tabelle zugegriffen werden soll. Wenn der Datenbankserver keinen passenden Index findet, muss er die Tabelle sequenziell durchsuchen - ein Prozess, der längere Zeit in Anspruch nehmen kann.

Angenommen Sie müssen eine große Datenbank nach Mitarbeitern absuchen, aber kennen nur deren Vor- oder Nachnamen, jedoch nicht beides. Wenn kein Index vorhanden ist, durchsucht SQL Anywhere die gesamte Tabelle. Wenn Sie jedoch zwei Indizes erstellen (einen mit den Nachnamen zuerst und einen mit den Vornamen zuerst), durchsucht SQL Anywhere zunächst die Indizes und kann die Daten normalerweise schneller zurückgeben.

Die richtige Auswahl der Indizes kann sich entscheidend auf die Performance auswirken

Auch wenn SQL Anywhere mit Indizes Daten sehr effizient ermitteln kann, sollten Sie beim Hinzufügen von Indizes vorsichtig sein. Jeder Index sorgt jedes Mal für zusätzliche Arbeit, wenn Sie eine Zeile einfügen, löschen oder aktualisieren, weil SQL Anywhere zusätzlich alle betroffenen Indizes aktualisieren muss.

Indizes sollten hinzugefügt werden, wenn SQL Anywhere damit auf die Daten effizienter zugreifen kann. Fügen Sie insbesondere dann einen Index hinzu, wenn dadurch unnötiger sequenzieller Zugriff auf eine große Tabelle vermieden wird. Wenn Sie jedoch beim Hinzufügen von Zeilen in eine Tabelle eine bessere Performance benötigen und das schnelle Auffinden von Daten nebensächlich ist, verwenden Sie so wenig Indizes wie möglich.

Sie können den Indexberater verwenden, der Sie bei der Auswahl der passenden Indexgruppe für Ihre Datenbank unterstützt.

Abfrageoptimierung

Wann immer es möglich ist, versucht der Optimierer, die Abfrage mithilfe eines reinen Indexabrufs auszuführen. Bei einem reinen Indexabruf verwendet der Datenbankserver nur die Daten in den Indizes, die nötig sind, um die Abfrage auszuführen; so muss er nicht auf Zeilen in der Tabelle zugreifen. Der Optimierer wählt automatisch die Indizes aus, bei denen er festgestellt hat, dass sie die beste Performance liefern. Sie können allerdings auch Index-Hints in Ihrer Abfrage verwenden, um die Indizes festzulegen, die der Optimierer verwenden soll. Wenn einer der angegebenen Indizes nicht verwendet werden kann, wird ein Fehler zurückgegeben. Index-Hints können zu einer Verschlechterung der Performance führen und sollten daher nur von erfahrenen Benutzern verwendet werden. Verwenden Sie den Indexberater, um zu bestimmen, ob zusätzliche Indizes für Ihre Datenbank empfohlen werden.

Clustered-Indizes

Bei der Verwendung von Clustered-Indizes werden Zeilen in einer Tabelle in ungefähr der gleichen Reihenfolge wie im Index gespeichert.

Siehe auch

- „Indizes“ auf Seite 27
- „Indexberater“ auf Seite 164
- „Indizes“ auf Seite 27
- „Clustered-Indizes“ auf Seite 31
- WITH (Tabellen-Hint)-Klausel, FROM-Klausel [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Abfrageprädikate“ auf Seite 292
- „FROM-Klausel“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Tipp: Verbessern der Index-Performance

Wenn Ihr Index eine schlechtere Performance hat als erwartet, können Sie die folgenden Maßnahmen in Betracht ziehen:

- Zusammengesetzte Indizes reorganisieren
- Die Seitengröße erhöhen

Diese Maßnahmen verbessern die Indexselektivität und die Auffächerung des Indexes.

Siehe auch

- [„Fortgeschrittene Aufgaben: Indexselektivität und Auffächerung“](#) auf Seite 40

Tipp: Für gemischte oder OLAP-Arbeitslast optimieren

Mit der Option `optimization_workload` können Sie festlegen, ob die Abfrageverarbeitung für Datenbanken optimiert werden soll, bei denen Aktualisierungen, Löschungen bzw. Einfügungen üblicherweise zusammen mit Abfragen (gemischte Arbeitslast) ausgeführt werden, oder für Datenbanken, bei denen die Hauptform der Aktualisierungsaktivität mit Batches stattfindet, die nur selten zusammen mit Abfragen ausgeführt werden.

Siehe auch

- [„optimization_workload-Option“](#) [*SQL Anywhere Server - Datenbankadministration*]

Tipp: Strategisches Sortieren von Abfrageergebnissen

Reduzieren Sie den Aufwand für das unnötige Sortieren von Daten. Sofern es nicht erforderlich ist, dass die Daten in einer vorhersehbaren Reihenfolge geliefert werden, sollten Sie in `SELECT`-Anweisungen auf die Klausel `ORDER BY` verzichten. Das Sortieren benötigt beim Verarbeiten der Abfrage zusätzliche Zeit und Ressourcen.

Siehe auch

- [„Die ORDER BY-Klausel: Abfrageergebnisse sortieren“](#) auf Seite 481
- [„Die GROUP BY-Klausel: Abfrageergebnisse in Gruppen organisieren“](#) auf Seite 474

Tipp: Den korrekten Cursor-Typ angeben

Das Angeben des korrekten Cursor-Typs kann die Performance steigern. Wenn ein Cursor zum Beispiel schreibgeschützt ist und Sie ihn als schreibgeschützt deklarieren, ermöglicht dies eine schnellere Optimierung und Ausführung, da weniger Material zu erstellen ist (z.B. Prüf-Integritätsregeln usw.). Wenn der Cursor aktualisierbar ist, können einige Neuschreibungen für Abfragen übersprungen werden. Wenn eine Abfrage aktualisierbar ist, muss der Datenbankserver abhängig vom Ausführungsplan, den der Optimierer ausgewählt hat, eine Keyset-gesteuerte Vorgangsweise wählen. Beachten Sie, dass Keyset-gesteuerte Cursor kostenträchtiger sind.

Siehe auch

- „Cursortypen“ [[SQL Anywhere Server - Programmierung](#)]

Tipp: Explizite Selektivitätsschätzungen sparsam einsetzen

Es kann passieren, dass Statistiken ungenau werden. Dieser Fall tritt üblicherweise dann ein, wenn eine große Menge von Daten hinzugefügt, aktualisiert oder gelöscht wurde und dann nur wenige Abfragen ausgeführt wurden. Ungenauere oder nicht verfügbare Statistiken können die Performance beeinträchtigen. Wenn SQL Anywhere zu lange braucht, um die Statistiken zu aktualisieren, sollten Sie CREATE STATISTICS oder DROP STATISTICS ausführen, um sie zu aktualisieren.

SQL Anywhere aktualisiert auch dann einige Statistiken, wenn LOAD TABLE-Anweisungen ausgeführt werden, während der Abfrageausführung, und wenn DML-Anweisungen aktualisiert werden.

Unter ungewöhnlichen Umständen allerdings können sich diese Maßnahmen als nicht effektiv herausstellen. 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.

Auch wenn von Benutzern gelieferte Schätzungen manchmal die Performance steigern können, sollten Sie es vermeiden, explizite Benutzerschätzungen in Anweisungen einzufügen, die laufend verwendet werden. Wenn sich die Daten ändern, verliert die explizite Schätzung ihre Gültigkeit, wodurch der Optimierer möglicherweise ungeeignete Pläne auswählt.

Wenn Sie Selektivitätsschätzungen verwendet haben, die als Behelfslösung für Performance-Probleme nicht ausreichen, bei denen der von der Software ausgewählte Zugriffsplan ungeeignet war, können Sie "user_estimates" auf "Off" setzen, damit die Werte ignoriert werden.

Siehe auch

- „Explizite Selektivitätsschätzungen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Tipp: Verbessern der Abfrageperformance durch materialisierte Ansichten

Sie sollten materialisierte Ansichten für häufig ausgeführte kostenträchtige Abfragen benutzen, etwa für Abfragen mit intensiven Aggregat- und Join-Vorgängen. Materialisierte Ansichten bieten eine abfragbare Struktur, in der aggregierte, durch Joins verknüpfte Daten gespeichert werden. Materialisierte Ansichten werden entwickelt, um die Performance in Umgebungen zu verbessern, in denen die Datenbank groß ist und häufige Abfragen in wiederholten Aggregat- und Join-Vorgängen zu großen Datenmengen führen. Materialisierte Ansichten sind beispielsweise ideal für den Einsatz mit Data Warehousing-Anwendungen geeignet.

Der Optimierer führt eine Liste von materialisierten Ansichten, die bei der Optimierung als Kandidaten für das teilweise oder vollständige Erfüllen einer Abfrage in Betracht kommen. Falls der Optimierer eine mögliche materialisierte Ansicht findet, die eine Abfrage teilweise oder vollständig erfüllt, schließt er die

Ansicht in die Empfehlungen ein, die er für die Enumerationsphase der Optimierung abgibt, wo der beste Plan aufgrund der Kosten ermittelt wird. Das Verfahren, mit dem der Optimierer übereinstimmende Ansichten für Abfragen sucht, wird als **Ansichtenübereinstimmung** bezeichnet. Bevor eine materialisierte Ansicht vom Optimierer in Betracht gezogen wird, muss sie bestimmte Bedingungen erfüllen. Falls der Optimierer feststellt, dass die Verwendung materialisierter Ansichten erlaubt ist, wird jeder Kandidat einer materialisierten Ansicht geprüft. Es gibt keine Garantie, dass der Optimierer eine materialisierte Ansicht verwendet, solange sie nicht explizit von der Abfrage referenziert wird. Sie können jedoch dafür sorgen, dass die Bedingungen erfüllt werden, sodass die Ansicht in Betracht kommt.

Siehe auch

- „Materialisierte Ansichten“ auf Seite 56
- „Aktivieren und Deaktivieren der Verwendung einer materialisierten Ansicht durch den Optimierer“ auf Seite 72

Materialisierte Ansichten und Ansichtenübereinstimmung

Der Optimierer ermittelt mithilfe eines Algorithmus für die Ansichtenübereinstimmung, ob materialisierte Ansichten zum Erfüllen einer Abfrage verwendet werden können. Dazu gehören zwei Bewertungsschritte: einer für die Abfrage und einer für die materialisierte Ansicht.

Prüfung der Abfrage

Während der Abfrageberechnung prüft der Algorithmus für die Ansichtenübereinstimmung die Abfrage. Falls eine der folgenden Bedingungen erfüllt ist, werden **keine** materialisierten Ansichten zum Verarbeiten der Abfrage verwendet.

- Alle von der Abfrage referenzierten Tabellen sind aktualisierbar.

Der Optimierer zieht keine materialisierten Ansichten für eine SELECT-Anweisung in Betracht, die aktualisierbar ist oder die ausdrücklich in einem aktualisierbaren Cursor deklariert wird. Diese Situation kann auftreten, wenn Interactive SQL benutzt wird, in dem standardmäßig aktualisierbare Cursors für SELECT-Anweisungen benutzt werden.

- Bei der Anweisung handelt es sich um eine einfache DML-Anweisung, die den Optimierer umgeht und heuristisch optimiert wird. Sie können jedoch eine kostenbasierte Optimierung jeder SELECT-Anweisung erzwingen, indem Sie die Option FORCE OPTIMIZATION für die OPTION-Klausel benutzen.
- Bei Abfragen innerhalb von gespeicherten Prozeduren und benutzerdefinierten Funktionen wurde der Ausführungsplan der Abfrage im Cache abgelegt. Der Datenbankserver kann möglicherweise die Ausführungspläne für diese Abfragen in den Cache übernehmen, sodass sie erneut verwendet werden können. Für diese Abfrageklasse wird der Abfrageausführungsplan nach der Ausführung im Cache abgelegt. Bei der nächsten Ausführung der Abfrage wird der Plan abgerufen und alle Phasen bis zur Ausführung werden übersprungen.

Prüfung materialisierter Ansichten

Wenn folgende Voraussetzungen erfüllt sind, enthält der Optimierer eine materialisierte Ansicht in der Gruppe der materialisierten Ansichten, die vom Algorithmus für die Ansichtenübereinstimmung geprüft werden:

- Die Ansichtsdefinition enthält nur einen Abfrageblock.
- Die Ansichtsdefinition enthält nur eine FROM-Klausel.
- Die Ansichtsdefinition enthält keine der folgenden Konstrukte oder Spezifikationen:
 - GROUPING SETS
 - CUBE
 - ROLLUP
 - Unterabfrage
 - Abgeleitete Tabelle
 - UNION
 - EXCEPT
 - INTERSECT
 - Materialisierte Ansichten
 - DISTINCT
 - TOP
 - FIRST
 - Selbst-Join
 - Rekursiver Join
 - FULL OUTER JOIN
- (Optional) Die Ansichtsdefinition enthält eine GROUP BY-Klausel und eine HAVING-Klausel, sofern die HAVING-Klausel keine Unterabfrage-Bedingung oder Unterabfragen enthält.

Zusätzlich zu den Kriterien für die Ansichtsdefinition muss Folgendes erfüllt sein:

- Die materialisierte Ansicht ist für die Verwendung durch den Datenbankserver aktiviert.
- Die materialisierte Ansicht ist für die Verwendung in der Optimierung aktiviert.
- Die materialisierte Ansicht wurde initialisiert (mit Daten gefüllt).
- Die Werte einiger entscheidender Optionen, die zum Erstellen der materialisierten Ansichten verwendet werden, stimmen mit den Optionen für die Verbindung überein, die die Abfrage ausführt.
- Die letzte Aktualisierung der materialisierten Ansicht überschreitet nicht den Veraltungsschwellenwert, der mit der materialized_view_optimization-Datenbankoption festgelegt wurde.

Falls die materialisierte Ansicht die obigen Kriterien erfüllt und der Abfrage ganz oder teilweise entspricht, nimmt der Algorithmus für die Ansichtenübereinstimmung die materialisierte Ansicht in seine Empfehlungen für die Enumerationsphase der Optimierung auf, wenn der beste Plan aufgrund der Kosten ausgewählt wird. Dies bedeutet jedoch nicht, dass die materialisierte Ansicht im endgültigen Ausführungsplan tatsächlich benutzt wird. Materialisierte Ansichten, die zum Berechnen des Ergebnisses

einer Abfrage geeignet erscheinen, werden möglicherweise trotzdem nicht benutzt, wenn ein anderer Zugriffsplan, der die materialisierte Ansicht nicht verwendet, als preiswerter geschätzt wird.

Siehe auch

- „Aktivieren und Deaktivieren der Verwendung einer materialisierten Ansicht durch den Optimierer“ auf Seite 72
- „sa_materialized_view_info-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- *Query Processing Based on SQL Anywhere 12.0.1 Architecture* Whitepaper auf <http://www.sybase.com/detail?id=1096047>.
- „Plan-Caching“ auf Seite 337
- „SELECT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ auf Seite 341
- „Fortgeschrittene Aufgaben: Phasen der Abfrageverarbeitung“ auf Seite 329
- „Anwendungsprofilerstellung“ auf Seite 156

Abrufen der Liste möglicher materialisierter Ansichten

In Interactive SQL können Sie eine Liste der materialisierten Ansichten abrufen, die für eine Berücksichtigung durch den Optimierer in Frage kommen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. Führen Sie die folgende Anweisung aus:

```
SELECT * FROM sa_materialized_view_info( ) WHERE AvailForOptimization='Y';
```

Die gelieferte Liste gilt speziell für die anfordernde Verbindung, da der Optimierer Optionseinstellungen berücksichtigt, wenn er die Liste generiert. Eine materialisierte Ansicht wird nicht als Kandidat betrachtet, wenn die Optionen für die Verbindung nicht mit den Optionen übereinstimmen, die beim Erstellen der materialisierten Ansicht in Kraft waren.

2. Sie können eine Liste aller materialisierten Ansichten abrufen, die nicht als Kandidaten für die Verbindung betrachtet werden, da nicht alle Optionseinstellungen übereinstimmen. Führen Sie dazu den folgenden Befehl von der Verbindung aus durch, welche die Abfrage ausführen wird:

```
SELECT * FROM sa_materialized_view_info( ) WHERE AvailForOptimization='O';
```

Ergebnisse

Die Liste der möglichen materialisierten Ansichten wird angezeigt.

Siehe auch

- „Einschränkungen für materialisierte Ansichten“ auf Seite 61

Ermitteln, welche materialisierten Ansichten vom Optimierer in Betracht gezogen wurden

In Interactive SQL können Sie ermitteln, welche materialisierte Ansicht der Optimierer verwendet hat, um eine Abfrage zu erfüllen.

Voraussetzungen

Es gibt keine Voraussetzungen für das Ausführen dieser Aufgabe.

Kontext und Bemerkungen

Hinweis

Wenn die Snapshot-Isolation verwendet wird, zieht der Optimierer keine materialisierten Ansichten in Betracht, die nach dem Start des Snapshots für die aktuelle Transaktion aktualisiert wurden.

Aufgabe

1. Stellen Sie in Interactive SQL eine Verbindung mit der Datenbank her.
2. Klicken Sie auf **Extras » Plananzeige** (oder drücken Sie Umschalt+F5).
3. Geben Sie die Abfrage im Fensterausschnitt **SQL** ein.
4. Wählen Sie **Statistikebene**, einen **Cursortyp** und einen **Update-Status** aus.
5. Klicken Sie auf **Plan abrufen**.
6. In den Fensterausschnitten **Details** und **Erweiterte Details** können Sie sehen, welche materialisierten Ansichten, wenn überhaupt, zum Erfüllen der Abfrage verwendet wurden.

Ergebnisse

Die zum Erfüllen der Abfrage verwendeten materialisierten Ansichten werden angezeigt.

Siehe auch

- „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ auf Seite 341
- „Anwendungsprofilerstellung“ auf Seite 156
- „Benutzerdefinierte Diagnoseprotokollierungsstufen“ auf Seite 175
- „Fortgeschrittene Aufgaben: Phasen der Abfrageverarbeitung“ auf Seite 329

Tipp: Option WITH EXPRESS CHECK bei der Validierung von Tabellen verwenden

Wenn Sie feststellen, dass die Validierung großer Datenbanken mit einem kleinen Cache lange dauert, können Sie eine der beiden Optionen verwenden, damit der Vorgang schneller abläuft. Mit der Option WITH EXPRESS CHECK in Verbindung mit der Anweisung VALIDATE TABLE bzw. mit der Option -

fx in Verbindung mit dem Validierungs-Dienstprogramm (dbvalid) werden Ihre Tabellen bedeutend schneller validiert.

Siehe auch

- „VALIDATE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Validierungs-Dienstprogramm (dbvalid)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Tipp: Arbeitstabellen in der Abfrageverarbeitung verwenden ("Alle Zeilen" als Optimierungsziel verwenden)

Arbeitstabellen sind materialisierte temporäre Ergebnismengen, die während der Ausführung einer Abfrage erstellt werden. Arbeitstabellen werden verwendet, wenn die Kosten niedriger als die von alternativen Strategien sind. Im Allgemeinen dauert es bei Verwendung einer Arbeitstabelle länger, bis die ersten Zeilen abgerufen werden, jedoch können die Kosten für das Abrufen sämtlicher Zeilen deutlich niedriger sein. Aufgrund dieser Differenz wählt SQL Anywhere basierend auf der Einstellung "optimization_goal" verschiedene Strategien. Der Standardwert ist "alle Zeilen". Wenn "alle Zeilen" eingestellt ist, verwendet SQL Anywhere Arbeitstabellen, wenn diese die Gesamtausführungskosten für eine Abfrage verringern. Wenn "erste Zeile" eingestellt ist, versucht SQL Anywhere, Arbeitstabellen zu vermeiden.

Arbeitstabellen werden in den folgenden Fällen verwendet:

- Wenn die Abfrage eine ORDERBY-, GROUPBY- oder DISTINCT-Klausel enthält und SQL Anywhere keinen Index zum Sortieren der Zeilen verwendet. Wenn ein geeigneter Index vorhanden ist und die Einstellung "optimization_goal" auf "erste Zeile" gesetzt ist, vermeidet SQL Anywhere die Verwendung einer Arbeitstabelle. Wenn "optimization_goal" jedoch auf "alle Zeilen" gesetzt wird, kann es kostenträchtiger sein, alle Zeilen einer Abfrage mit einem Index abzurufen, als eine Arbeitstabelle aufzubauen und die Zeilen zu sortieren. SQL Anywhere wählt die kostengünstigere Strategie, wenn das Ziel der Optimierung auf "Alle Zeilen" gesetzt ist. Bei GROUP BY und DISTINCT verwenden zwar die Hash-basierten Algorithmen Arbeitstabellen, sie sind jedoch im Allgemeinen effizienter, wenn alle Zeilen einer Abfrage abgerufen werden.
- Wenn ein Hash Join-Algorithmus gewählt wurde. In diesem Fall werden Arbeitstabellen zum Speichern der Zwischenergebnisse verwendet (wenn die Eingabe nicht in den Speicher passt), und die Ergebnisse des Joins werden ebenfalls in einer Arbeitstabelle gespeichert.
- Wenn ein Cursor mit empfindlichen Werten geöffnet wird. In diesem Fall wird eine Arbeitstabelle erstellt, welche die Zeilenidentifizierer und Primärschlüssel der Basistabellen enthält. Diese Arbeitstabelle wird gefüllt, wenn Zeilen von der Abfrage in Vorwärtsrichtung abgerufen werden. Wenn Sie jedoch die letzte Zeile aus dem Cursor abrufen, wird die gesamte Tabelle gefüllt.
- Wenn ein Cursor mit unempfindlicher Semantik geöffnet wird. In diesem Fall wird eine Arbeitstabelle mit den Ergebnissen der Abfrage gefüllt, wenn die Abfrage geöffnet wird.
- Wenn ein mehrzeiliges UPDATE durchgeführt wird und die aktualisierte Spalte in der WHERE-Klausel der Aktualisierung oder in einem Index verwendet wird, der für die Aktualisierung nötig ist.

- Wenn ein mehrzeiliges UPDATE oder DELETE eine Unterabfrage in der WHERE-Klausel hat, welche die zu ändernde Tabelle referenziert.
- Wenn ein INSERT von einer SELECT-Anweisung durchgeführt wird und die SELECT-Anweisung die INSERT-Tabelle referenziert.
- Wenn ein mehrzeiliges INSERT, UPDATE oder DELETE durchgeführt wird und für die Tabelle ein entsprechender Trigger definiert ist, der während des Vorgangs ausgelöst werden kann.

In diesen Fällen werden die von dem Vorgang betroffenen Datensätze in die Arbeitstabelle eingetragen. Unter bestimmten Umständen, wie z.B. bei Keyset-gesteuerten Cursor, wird ein temporärer Index für die Arbeitstabelle aufgebaut. Das Extrahieren der erforderlichen Datensätze in eine Arbeitstabelle kann recht lange dauern, bevor die Abfrageergebnisse erscheinen. Mithilfe von Indizes, die zum Sortieren verwendet werden können (wie oben im ersten Fall), werden die ersten Zeilen schneller abgerufen. Es kann jedoch sein, dass die Zeit, die insgesamt zum Abrufen aller Zeilen erforderlich ist, kürzer ist, wenn Arbeitstabellen verwendet werden, da diese Abfragealgorithmen ermöglichen, die auf der Hash- und Merge-Sort-Methode basieren. Diese Algorithmen verwenden sequenzielle I/O-Vorgänge, die schneller sind als wahlfreie I/O-Vorgänge, die mit einem Index Scan verwendet werden.

Der Optimierer analysiert jede Abfrage, um festzustellen, ob eine Arbeitstabelle die beste Performance liefert. Für derartige Optimierungen sind keinerlei Eingriffe durch den Benutzer erforderlich.

Hinweise

Die oben genannten Fälle von INSERT, UPDATE und DELETE stellen gewöhnlich kein Performance-Problem dar, da es sich dabei um einmalige Vorgänge handelt. Wenn jedoch Probleme auftreten, können Sie möglicherweise die Anweisung umformulieren und damit den Konflikt sowie den Aufbau von Arbeitstabellen zu vermeiden. Das ist nicht immer möglich.

Siehe auch

- „optimization_goal-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Tipp: Aufbauen effizienter SQL-Abfragen

Um die Performance bei der Abfrageverarbeitung zu verbessern, können Sie anhand der folgenden Tipps effizientere Abfragen aufbauen. Diese Tipps spiegeln Optimierungen wieder, die der Optimierer möglicherweise während der Abfrageverarbeitung wählen würde, um die Abfrage effizienter zu gestalten. Wenn Sie diese Effizienzen in die Abfrage integrieren, muss der Optimierer in der Regel weniger Arbeit verrichten.

Typ	Vorher und nachher	Erklärung
Eliminieren Sie nicht benötigte DISTINCT-Bedingungen.	<p>Vorher:</p> <pre>SELECT DISTINCT p.ID, p.Quantity FROM Products p;</pre> <p>Nachher:</p> <pre>SELECT p.ID, p.Quantity FROM Products p;</pre>	Das DISTINCT-Schlüsselwort in der ersten Anweisung ist unnötig, weil die Products-Tabelle den Primärschlüssel "p.ID" enthält, der Teil der Ergebnismenge ist.
Eliminieren Sie nicht benötigte DISTINCT-Bedingungen.	<p>Vorher:</p> <pre>SELECT DISTINCT * FROM SalesOrders o JOIN Customers c ON o.CustomerID = c.ID WHERE c.State = 'NY';</pre> <p>Nachher:</p> <pre>SELECT * FROM SalesOrders o JOIN Customers c ON o.CustomerID = c.ID WHERE c.State = 'NY';</pre>	Die erste Abfrage enthält die Primärschlüssel beider Tabellen, d.h., jede Zeile im Ergebnis muss unterschiedlich sein.
Entschachteln Sie Unterabfragen.	<p>Vorher:</p> <pre>SELECT s.* FROM SalesOrderItems s WHERE EXISTS (SELECT * FROM Products p WHERE s.ProductID = p.ID AND p.ID = 300 AND p.Quantity > 20);</pre> <p>Nachher:</p> <pre>SELECT s.* FROM Products p JOIN SalesOrderItems s ON p.ID = s.ProductID WHERE p.ID = 300 AND p.Quantity > 20;</pre>	<p>Das Umschreiben von verschachtelten Abfragen in Joins führt häufig zu einer effizienteren Ausführung und effektiveren Optimierung. Allgemein werden korrelierte Unterabfragen mit nicht mehr als einer Tabelle in der FROM-Klausel, die in den Prädikaten ANY, ALL und EXISTS benutzt werden, immer entschachtelt. Eine nicht korrelierte Unterabfrage oder eine Unterabfrage mit mehr als einer Tabelle in der FROM-Klausel wird entschachtelt, wenn aufgrund der Abfragesemantik erkennbar ist, dass die Unterabfrage nicht mehr als eine Zeile liefert.</p> <p>In diesem Beispiel kann die Unterabfrage höchstens eine Zeile für jede Zeile im Außenblock in Übereinstimmung bringen. Weil hier immer nur eine Zeile gefunden wird, ist die Konvertierung in einen Inner-Join möglich.</p>

Tipp	Vorher und nachher	Erklärung
Entschachteln Sie Unterabfragen.	<p>Vorher:</p> <pre>SELECT p.* FROM Products p WHERE EXISTS (SELECT * FROM SalesOrderItems s WHERE s.ProductID = p.ID AND s.ID = 2001);</pre> <p>Nachher:</p> <pre>SELECT DISTINCT p.* FROM Products p JOIN SalesOrderItems s ON p.ID = s.ProductID WHERE s.ID = 2001;</pre>	Die Abfrage unter "Vor" enthält ein konjunktives EXISTS-Prädikat in der Unterabfrage, das mit mehr als einer Zeile übereinstimmen kann. Sie kann mit DISTINCT in der SELECT-Liste in einen Inner-Join konvertiert werden.
Entschachteln Sie Unterabfragen.	<p>Vorher:</p> <pre>SELECT * FROM Products p WHERE p.ID = (SELECT s.ProductID FROM SalesOrderItems s WHERE s.ID = 2001 AND s.LineID = 1);</pre> <p>Nachher:</p> <pre>SELECT p.* FROM Products p, SalesOrderItems s WHERE p.ID = s.ProductID AND s.ID = 2001 AND s.LineID = 1;</pre>	Eliminieren Sie Unterabfragen in Vergleichen, bei denen die Unterabfrage für jede Zeile im Außenblock mit höchstens einer Zeile übereinstimmt.
Ziehen Sie beim Abfragen einer indizierten Spalte in Erwägung, ein IN-Prädikat zu verwenden.	<p>Vorher:</p> <pre>SELECT * FROM SalesOrders WHERE SalesRepresentative = 902 OR SalesRepresentative = 195;</pre> <p>Nachher:</p> <pre>SELECT * FROM SalesOrders WHERE SalesRepresentative IN (195, 902);</pre>	<p>In der umgeschriebenen Form kann das IN-Listen-Prädikat als sargable (als Suchargument nutzbar) behandelt und für den Abruf über den Index genutzt werden. Außerdem kann der Optimierer die IN-Liste sortieren, damit sie mit der Sortierfolge des Indexes übereinstimmt, was einen effizienteren Abruf zur Folge hat.</p> <p>Die IN-Liste darf nur Konstanten oder Werte enthalten, die während einer Ausführung des Abfrageblocks konstant bleiben, z.B. äußere Referenzen.</p>

Tipp	Vorher und nachher	Erklärung
Eliminieren Sie nicht benötigte Joins.	<p>Vorher:</p> <pre>SELECT s.ID, s.LineID, p.ID FROM SalesOrderItems s KEY JOIN Products p FOR READ ONLY;</pre> <p>Nachher:</p> <pre>SELECT s.ID, s.LineID, s.ProductID FROM SalesOrderItems s WHERE s.ProductID IS NOT NULL FOR READ ONLY;</pre>	<p>Ziehen Sie die Eliminierung von Joins in Erwägung, wenn Folgendes zutrifft:</p> <ul style="list-style-type: none"> • Der Join ist ein Primärschlüssel-Fremdschlüssel-Join und in der Abfrage werden lediglich Primärschlüsselspalten aus der Primärtabelle referenziert. In diesem Fall wird die Primärschlüsseltabelle eliminiert, falls sie nicht aktualisierbar ist. • Der Join ist ein Primärschlüssel-Primärschlüssel-Join zwischen zwei Instanzen in derselben Tabelle. In diesem Fall wird eine der Tabellen eliminiert, falls sie nicht aktualisierbar ist. • Der Join ist ein Outer-Join und der nullwertliefernde Tabellenausdruck gibt für jede Zeile der beibehaltenen Seite des Outer-Joins höchstens eine Zeile zurück. Keiner der von dem nullwertliefernden Ausdruck erzeugten Ausdrücke wird im Rest der Abfrage nach dem Outer-Join benötigt. <p>In diesem Fall ist der Join ein Primärschlüssel-Fremdschlüssel-Join, sodass die Primärschlüsseltabelle "Products" eliminiert werden kann. Die zweite Abfrage ist also der ersten semantisch gleichwertig, weil im Ergebnis keine Zeile aus der SalesOrderItems-Tabelle erscheint, die einen NULL-Fremdschlüssel zu "Products" enthält.</p>
Eliminieren Sie nicht benötigte Joins.	<p>Vorher:</p> <pre>SELECT s.ID, s.LineID FROM SalesOrderItems s LEFT OUTER JOIN Products p ON p.ID = s.ProductID WHERE s.Quantity > 5 FOR READ ONLY;</pre> <p>Nachher:</p> <pre>SELECT s.ID, s.LineID FROM SalesOrderItems s WHERE s.Quantity > 5 FOR READ ONLY;</pre>	<p>In der ersten Abfrage kann das Schlüsselwort OUTER JOIN eliminiert werden, weil der nullwertliefernde Tabellenausdruck für jede Zeile der beibehaltenen Seite höchstens eine Zeile erzeugen kann und keine der Spalten von "Products" oberhalb der LEFT OUTER JOIN-Anweisung verwendet wird.</p>

Tipp	Vorher und nachher	Erklärung
<p>Eliminieren Sie unnötige Konvertierungen der Groß- und Kleinschreibung.</p>	<p>Vorher:</p> <pre>SELECT * FROM Customers WHERE UPPER(Surname) = 'SMITH';</pre> <p>Nachher:</p> <pre>SELECT * FROM Customers WHERE Surname = 'SMITH';</pre>	<p>In einer Datenbank, die keine Groß- und Kleinschreibung berücksichtigt, kann die erste Abfrage so umgeschrieben werden, dass der Optimierer die Möglichkeit hat, einen Index für "Customers.Surname" zu verwenden.</p> <p>Standardmäßig führt der Datenbankserver Zeichenfolgenvergleiche ohne Berücksichtigung von Groß- und Kleinschreibung durch, es sei denn, es werden explizite Anweisungen zur Textkonvertierung gegeben (Verwendung von UPPER, UCASE, LOWER oder LCASE). Durch Eliminieren unnötiger Konvertierungen der Groß- und Kleinschreibung können die Prädikate in sargable-Prädikate umgewandelt und dann für Indexabrufe der entsprechenden Tabelle verwendet werden.</p>
<p>Ziehen Sie das Inlining von Funktionen in Erwägung.</p>	<p>Vorher:</p> <pre>CREATE FUNCTION F1(arg1 INT, arg2 INT) RETURNS INT BEGIN RETURN arg1 * arg2 END; SELECT F1(e.EmployeeID, 2.5) FROM Employees e;</pre> <p>Nachher:</p> <pre>SELECT CAST(e.EmployeeID AS INT) * CAST(2.5 AS INT) FROM Employees e;</pre>	<p>Ein Inlining einer benutzerdefinierten Funktion ist möglich, wenn diese eine der folgenden Voraussetzungen erfüllt:</p> <ul style="list-style-type: none"> • Sie enthält eine einzige RETURN-Anweisung. • Sie deklariert eine einzige Variable, weist die Variable zu und gibt einen einzigen Wert zurück. • Sie deklariert eine einzige Variable, führt in dieser Variablen eine SELECT-Anweisung aus und gibt einen einzigen Wert zurück. <p>Dieser Tipp ist nicht anwendbar auf temporäre Funktionen, rekursive Funktionen und Funktionen mit NOT DETERMINISTIC-Klausel.</p> <p>Außerdem ist dieser Tipp nicht anwendbar, wenn die betreffende Funktion mit einer Unterabfrage als Argument oder aus einer temporären Prozedur heraus aufgerufen wird.</p>

Typ	Vorher und nachher	Erklärung
Ziehen Sie das Inlining einfacher gespeicherter Prozeduren in Erwägung.	<p>Vorher:</p> <pre>CREATE PROCEDURE Test1(arg1 INT) BEGIN SELECT * FROM Employees WHERE EmployeeID=arg1 END; SELECT * FROM Test1(200);</pre> <p>Nachher:</p> <pre>SELECT * FROM (SELECT * FROM Employees WHERE EmployeeID=CAST(200 AS INT)) AS Test1;</pre>	Das Inlining einer gespeicherten Prozedur ist möglich, wenn diese beim Aufrufen in der FROM-Klausel einer Abfrage nur als eine einzige SELECT-Anweisung festgelegt wird. Wenn eine Prozedur inline ist, wird sie als abgeleitete Tabelle umgeschrieben. Dieser Tipp gilt nicht für Prozeduren, in denen Standardargumente verwendet werden oder deren Hauptteil etwas anderes enthält als eine einzige SELECT-Anweisung.

Siehe auch

- „Benutzerdefinierte Funktionen“ auf Seite 93
- „CREATE FUNCTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CAST-Funktion [Datentypkonvertierung]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Prozeduren“ auf Seite 82
- „CREATE PROCEDURE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Tipp: Abfrageperformance überwachen

SQL Anywhere bietet eine Reihe von Tools zum Testen der Performance von Abfragen. Diese Tools befinden sich in Unterverzeichnissen unter `%SQLANYSAMPI6%\SQLAnywhere`, wie nachfolgend angegeben. Eine ausführliche Dokumentation zu jedem Tool finden Sie in einer `readme.txt`-Datei in demselben Ordner wie das Tool.

Weitere Hinweise über Systemprozeduren, die die Abfrage-Ausführungszeit messen, finden Sie unter „`sa_get_request_profile`-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)] und „`sa_get_request_times`-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)].

Tool	Funktion	Location
fetchtst	Ermittelt die Zeit, die benötigt wird, um eine Ergebnismenge abzurufen.	<code>%SQLANYSAMPI6%\SQLAnywhere\PerformanceFetch</code>
odbcfet	Ermittelt die Zeit, die benötigt wird, um eine Ergebnismenge abzurufen. Dieses Tool ähnelt "fetchtst", hat aber weniger Funktionalität.	<code>%SQLANYSAMPI6%\SQLAnywhere\PerformanceFetch</code>

Tool	Funktion	Location
instest	Ermittelt die Zeit, die benötigt wird, um Zeilen in eine Tabelle einzufügen	<i>%SQLANYSAMPI6%\SQLAnywhere\PerformanceInsert</i>
trantest	Misst die Last, welche von einer gegebenen Datenbankserverkonfiguration mit einem gegebenen Datenbankdesign und einer gegebenen Reihe von Transaktionen bewältigt werden kann	<i>%SQLANYSAMPI6%\SQLAnywhere\PerformanceTransaction</i>

Siehe auch

- „Praktische Einführung: Langsame Anweisungen diagnostizieren“ auf Seite 272
- „Performance-Probleme beheben“ auf Seite 189

Tipp: Kostenträchtige benutzerdefinierte Funktionen reduzieren

Kostenträchtige benutzerdefinierte Funktionen in Abfragen zu vermindern, in denen sie viele Male ausgeführt werden müssen, kann die Performance verbessern.

Siehe auch

- „Benutzerdefinierte Funktionen“ auf Seite 93

Tipp: Autocommit-Modus ausschalten

Wenn Ihre Anwendung im **Autocommit-Modus** ausgeführt wird, behandelt SQL Anywhere jede Anweisung als separate Transaktion. Die Wirkung ist dieselbe, wie wenn Sie eine COMMIT-Anweisung an das Ende jedes Ihrer Anweisungen anhängen.

Anstatt im Autocommit-Modus zu arbeiten, sollten Sie versuchen, Ihre SQL-Anweisungen so zu gruppieren, dass jede Gruppe eine logische Aufgabe ausführt. Wenn Sie Autocommit deaktivieren, müssen Sie nach jeder logischen Gruppe von SQL-Anweisungen eine explizite COMMIT-Anweisung ausführen. Achten Sie auch darauf, dass bei großen Transaktionen Blockierungen und Deadlocks auftreten können.

Wenn Sie keine Transaktionslogdatei verwenden, sind die Kosten der Verwendung des Autocommit-Modus hoch. Jede Anweisung erzwingt einen Checkpoint – ein Vorgang, bei dem gegebenenfalls mehrere Seiten mit Daten auf den Plattenspeicher geschrieben werden müssen.

Jede Anwendungsschnittstelle setzt ihr autocommit-Verhalten anders. Bei Open Client-, ODBC- und JDBC-Schnittstellen ist "Autocommit" die Standardeinstellung.

Siehe auch

- „Autocommit-Modus und manueller Commit-Modus“ [SQL Anywhere Server - Programmierung]

Tipp: In-Memory-Modus verwenden

Wenn Ihre Anwendung den Verlust von festgeschriebenen Transaktionen nach dem aktuellsten Checkpoint toleriert, könnte die Verwendung des In-Memory-Modus für Ihre Anwendung von Vorteil sein.

Dieser Modus ist bei Anwendungen sinnvoll, bei denen eine verbesserte Performance wünschenswert ist und Sie Ihre Datenbank auf einem System mit einer großen Menge an verfügbarem Speicher ausführen, bei dem üblicherweise alle Datenbankdateien im Cache gehalten werden können.

Sie können zwischen zwei verschiedenen speicherresidenten Modi wählen. Im Modus "Nie-Schreiben" werden festgeschriebene Transaktionen nicht in die Datenbankdatei auf dem Plattenspeicher geschrieben. Wenn Sie den Modus "Nie-Schreiben" angeben, können mehrere gleichzeitige LOAD TABLE-Anweisungen auf einer bzw. auf verschiedenen Tabellen aktiv sein. Alle Änderungen gehen verloren, wenn die Datenbank heruntergefahren oder die Verbindung getrennt wird. Im Modus "Nur-Checkpoint" verwendet der Datenbankserver kein Transaktionslog und Sie können die letzte festgeschriebene Transaktion nicht wiederherstellen. Da jedoch das Checkpoint-Log aktiviert ist, kann die Datenbank bis zum letzten Checkpoint wiederhergestellt werden.

Weitere Hinweise darüber, wie Sie den In-Memory-Modus konfigurieren und ermitteln, ob er für Ihre Anwendung geeignet ist, finden Sie unter „Datenbankserveroption -im“ [[SQL Anywhere Server - Datenbankadministration](#)].

Hinweis

Für den In-Memory-Modus benötigen Sie eine separate Lizenz. Siehe „Getrennt lizenzierte Komponenten“ [[SQL Anywhere 16 - Einführung](#)].

Tipp: Stets ein Transaktionslog verwenden

Die Verwendung eines Transaktionslogs kann Datenschutz bieten und die Performance von SQL Anywhere wesentlich steigern.

Wenn kein Transaktionslog verwendet wird, setzt SQL Anywhere einen Checkpoint am Ende jeder Transaktion, wodurch erhebliche Ressourcen gebunden werden.

Wenn ein Transaktionslog verwendet wird, zeichnet SQL Anywhere nur die Merkmale auf, die diese Änderungen beschreiben, wenn sie auftreten. Er hat die Möglichkeit, neue Datenbankseiten alle auf einmal zum günstigsten Zeitpunkt zu schreiben. **Checkpoints** sorgen dafür, dass Daten in die Datenbankdatei gelangen und dass diese konsistent und aktuell ist.

Sie können die Performance weiter steigern, wenn Sie das Transaktionslog auf einem anderen physischen Gerät als dem speichern, auf dem sich die Haupt-Datenbankdatei befindet. Der Lesekopf des zusätzlichen Datenträgers braucht normalerweise nicht zu suchen, um zum Ende des Transaktionslogs zu gelangen.

Tipp: Verzögerte Festschreibungen verwenden

Wenn die Menge von festgeschriebenen Änderungen bei einer Datenbank hoch ist, können Transaktionslog-Schreibvorgänge der entscheidende Faktor sein, der die Gesamt-Datenbankperformance bestimmt. Wenn Sie versuchen, die Transaktionslog-Performance zu verbessern, können Sie die Option "delayed_commits" aktivieren. Wenn die Option auf "On" gesetzt ist, antwortet der Datenbankserver auf eine COMMIT-Anweisung unmittelbar, anstatt zu warten, bis der Transaktionslogeintrag für COMMIT auf die Festplatte geschrieben wurde. Wenn die Option auf "Off" gesetzt ist, muss die Anwendung warten, bis die COMMIT-Anweisung auf die Festplatte geschrieben wurde. Eine Aktivierung der Option "delayed_commits" führt zu weniger Transaktionslog-Schreibvorgängen, indem mehrfache Neu-Schreibvorgänge auf teilweise vollen Logseiten vermieden werden. Sie können die Option pro Verbindung oder für alle Verbindungen setzen. Wenn die Option "delayed_commits" aktiviert ist, besteht das Risiko, dass festgeschriebene Vorgänge verloren gehen, wenn der Datenbankserver ausfällt, bevor die Transaktionslog-Seiten in den Plattenspeicher geschrieben werden.

Siehe auch

- „delayed_commits-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Tipp: Statistiken von kleinen Tabellen erfassen

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

Standardmäßig erstellt SQL Anywhere Statistiken für alle Tabellen mit fünf oder mehr Zeilen. Wenn Sie Statistiken für eine Tabelle mit weniger als fünf Zeilen erstellen wollen, können Sie dies mit der CREATE STATISTICS-Anweisung tun. Diese Anweisung erstellt Statistiken für alle Tabellen, unabhängig von der Anzahl der Zeilen. Sobald sie erstellt sind, werden Statistiken von SQL Anywhere automatisch verwaltet.

Siehe auch

- „CREATE STATISTICS-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Tipp: Fragmentierung reduzieren

Fragmentierung entsteht, wenn Sie Änderungen in Ihrer Datenbank durchführen. Die Performance kann sich verschlechtern, wenn Ihre Dateien, Tabellen oder Indizes stark fragmentiert sind. Das Verringern der Fragmentierung gewinnt an Bedeutung, wenn Ihre Datenbank an Größe zunimmt. SQL Anywhere enthält gespeicherte Prozeduren, welche Informationen zur Fragmentierung von Dateien, Tabellen und Indizes generieren.

Falls Sie eine beträchtliche Abnahme der Performance feststellen, sollten Sie folgende Aktionen in Betracht ziehen:

- Bauen Sie Ihre Datenbank neu auf, um die Tabellen- bzw. Indexfragmentierung zu reduzieren, besonders wenn Sie umfangreiche Löschen-, Aktualisierungs- oder Einfüge-Aktivitäten in mehreren Tabellen durchgeführt haben.
- Platzieren Sie Ihre Datenbank auf eine eigene Plattenpartition, um die Dateifragmentierung zu vermindern.
- Führen Sie eines der verfügbaren Windows-Dienstprogramme aus, um die Dateifragmentierung zu vermindern.
- Organisieren Sie Ihre Tabellen neu, um die Datenbankfragmentierung zu vermindern.
- Verwenden Sie die REORGANIZE TABLE-Anweisung, um Zeilen in einer Tabelle zu defragmentieren oder Indizes zu komprimieren, die durch Löschvorgänge lückenhaft geworden sind. Durch eine Neuorganisation von Tabellen können Sie die Gesamtzahl der Seiten, die für die Speicherung einer Tabelle und ihrer Indizes verwendet wird, verringern. Außerdem wird eventuell die Anzahl der Ebenen in einem Indexbaum reduziert.

Dateifragmentierung reduzieren

Eine fragmentierte Datenbankdatei kann die Performance Ihres Datenbankservers beeinträchtigen. Der Reduktion der Fragmentierung der Festplatte kommt höhere Bedeutung zu, wenn Ihre Datenbank wächst.

Der Datenbankserver ermittelt die Anzahl der Dateifragmente in jedem DBSpace, wenn Sie eine Datenbank unter Windows starten. Der Datenbankserver zeigt die folgenden Performancewarnungen im Fenster der Datenbankservermeldungen an, sobald mehr als ein Fragment vorhanden ist:
Datenbankdatei "mydatabase.db" besteht aus nnn Festplattenfragmenten.

Sie erhalten die Anzahl der Fragmente der Datenbankdatei auch mit der Datenbankeigenschaft "DBFileFragments".

Um Probleme mit der Dateifragmentierung auszuschalten, sollten Sie die Datenbank in einer eigenen Plattenpartition speichern und dann regelmäßig eines der Windows-Dienstprogramme zur Platten-Defragmentierung ausführen.

Siehe auch

- „Liste der Datenbankeigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Performancewarnung: Datenbankdatei %1 besteht aus %2 Festplattenfragmenten“ [[SQL Anywhere Server - Datenbankadministration](#)]
- DBFileFragments-Datenbankeigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]

Tabellenfragmentierung reduzieren

Tabellen-Fragmentierung tritt auf, wenn Zeilen nicht zusammenhängend gespeichert oder auf mehrere Seiten aufgeteilt werden. Diese Zeilen erfordern zusätzliche Seitenzugriffe und dadurch wird die Performance des Datenbankservers reduziert.

Die Auswirkungen der Fragmentierung auf die Performance sind unterschiedlich. Wenn eine Tabelle zwar stark fragmentiert ist, aber trotzdem in den Arbeitsspeicher passt und die Art und Weise des Zugriffs

es möglich macht, die Seiten in den Cache zu laden, sind die Auswirkungen möglicherweise minimal. Eine fragmentierte Tabelle kann aber auch wesentlich mehr I/O-Vorgänge erfordern und die Performance stark beeinträchtigen, wenn häufig auf getrennte Zeilen zugegriffen wird und die Kosten der zusätzlichen I/O-Vorgänge nicht durch das Caching reduziert werden.

Während die Neuorganisation von Tabellen und der Neuaufbau einer Datenbank die Fragmentierung reduzieren können, kann ein zu häufiges oder zu seltenes Ausführen dieser Vorgänge auch Auswirkungen auf die Performance haben. Sie sollten mit den Tools und Methoden experimentieren, die im folgenden Abschnitt beschrieben werden, um eine akzeptable Fragmentierungsstufe für Ihre Tabellen zu ermitteln.

Falls Sie die Fragmentierung reduzieren und die Performance trotzdem noch schlecht ist, könnte dies an einem anderen Problem liegen, etwa an fehlerhaften Statistikwerten.

Grad der Tabellenfragmentierung ermitteln

Eine einmalige Prüfung der Tabellenfragmentierung reicht nicht aus, um festzustellen, ob die Performance durch eine Defragmentierung verbessert werden kann. Bauen Sie stattdessen die Datenbank neu auf und prüfen Sie die Tabellenfragmentierung, um Baseline-Ergebnisse zu ermitteln. Prüfen Sie die Tabellenfragmentierung dann über einen längeren Zeitraum regelmäßig und ermitteln Sie, ob es einen Zusammenhang zwischen Änderungen der Fragmentierung und Änderungen der Performance-Werte gibt. Mit dieser Methode können Sie ermitteln, wie schnell Tabellen so weit fragmentiert werden, dass die Performance beeinträchtigt wird, und was die optimale Frequenz zur Defragmentierung von Tabellen ist.

Mit folgenden Methoden können Sie ermitteln, wie stark Ihre Datenbanktabellen fragmentiert sind:

- Rufen Sie die `sa_table_fragmentation`-Systemprozedur auf. Beispiel:

```
CALL sa_table_fragmentation( );
```

- Die Registerkarte **Fragmentierung** im SQL Anywhere-Plug-In. Die Registerkarte **Fragmentierung** zeigt eine grafische Darstellung der Ergebnisse der Systemprozedur `sa_table_fragmentation` für Basistabellen.

Siehe auch

- „`sa_table_fragmentation`-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`sa_table_fragmentation`-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „REORGANIZE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Dienstprogramm zum Entladen (`dbunload`)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Tools für Performanceüberwachung und Diagnose“ auf Seite 153
- „Die Registerkarte Fragmentierung (SQL Anywhere-Plug-In)“ auf Seite 251

Methoden zum Reduzieren der Tabellenfragmentierung

Die folgenden Methoden helfen bei der Kontrolle der Tabellenfragmentierung:

- **PCTFREE verwenden** SQL Anywhere reserviert zusätzlichen Platz auf jeder Seite, damit die Zeilen leicht anwachsen können. Wenn die Zeile aufgrund einer Aktualisierung länger wird als der ihr ursprünglich zugewiesene Platz zulässt, wird sie geteilt, und an der anfänglichen Position verweist ein Zeiger auf die Seite, auf der die gesamte Zeile gespeichert ist. Wenn beispielsweise leere Zeilen mit

UPDATE-Anweisungen bzw. durch Einfügen neuer Spalten in eine Tabelle gefüllt werden, kann dies zu zahlreichen geteilten Zeilen führen. Mehr Zeilen werden auf separaten Seiten gespeichert, und mehr Zeit ist für den Zugriff auf die zusätzlichen Seiten erforderlich.

Sie können dafür sorgen, dass Ihre Tabellen nicht zu sehr fragmentiert werden, indem Sie einen Prozentsatz festlegen, der auf Tabellenseiten freigehalten wird, um spätere Aktualisierungen unterzubringen. Diese PCTFREE-Angabe kann mit CREATE TABLE, ALTER TABLE, DECLARE LOCAL TEMPORARY TABLE oder LOAD TABLE festgelegt werden.

- **Tabellen neu organisieren** Sie können bestimmte Tabellen mit der TABLE REORGANIZE-Anweisung defragmentieren oder indem Sie in Sybase Central auf der Registerkarte **Fragmentierung** auf **Reorganisieren** klicken.
- **Datenbank neu aufbauen** Beim Neuaufbau der Datenbank werden alle Tabellen einschließlich der Systemtabellen defragmentiert, *sofern der Neuaufbau als zweistufiger Prozess durchgeführt wird*, das heißt, die Daten werden entladen und auf der Platte gespeichert und dann neu geladen. Der Neuaufbau auf diese Art hat auch den Vorteil, dass die Tabellenzeilen neu angeordnet werden, so dass sie in der Reihenfolge erscheinen, die durch den Clustered-Index und die Primärschlüssel vorgegeben werden. Durch einen Neuaufbau in einem Schritt (z.B. mit den Optionen -ar, -an oder -ac) wird die Tabellenfragmentierung nicht reduziert.

Siehe auch

- PCTFREE-Klausel [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „REORGANIZE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Die Registerkarte Fragmentierung (SQL Anywhere-Plug-In)“ auf Seite 251
- „Neuaufbau von Datenbanken“ auf Seite 763
- „Dienstprogramm zum Entladen (dbunload)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Die Registerkarte Fragmentierung (SQL Anywhere-Plug-In)

Sie können mit der Registerkarte **Fragmentierung** Folgendes durchführen:

- Die Fragmentierung von Basistabellen und Indizes dieser Tabellen anzeigen.
- Tabellen und Indizes reorganisieren.

Innerhalb einer DBSpace-Zuordnung zoomen

Standardmäßig wird die Zoomstufe beim Öffnen einer DBSpace-Zuordnung im unteren Fensterausschnitt auf der Registerkarte **Fragmentierung** auf **An Fenster anpassen** gesetzt. Sie können die Darstellung vergrößern, indem Sie auf die DBSpace-Zuordnung klicken und Sie können die Darstellung verkleinern, indem Sie die Umschalttaste während des Klickens gedrückt halten. Wenn Sie in die DBSpace-Zuordnung klicken oder die Umschalttaste während des Klickens gedrückt halten, wird die betreffende Seite in der Zuordnung nach der Änderung der Zoomstufe zentriert.

Sie können folgende Zoomstufen auch mit den Symbolschaltflächen einstellen:

Symbolschaltfläche	Definition
1 : 1	1 Seite : 1 Pixel
64 KB : 1	1 64 KB-Block: 1 Pixel (ein Datenbank-Lesevorgang)
An Fenster anpassen	Verwendet den gesamten verfügbaren Platz im Fenster

Anzeigen der Fragmentierungsdetails eines Objekts

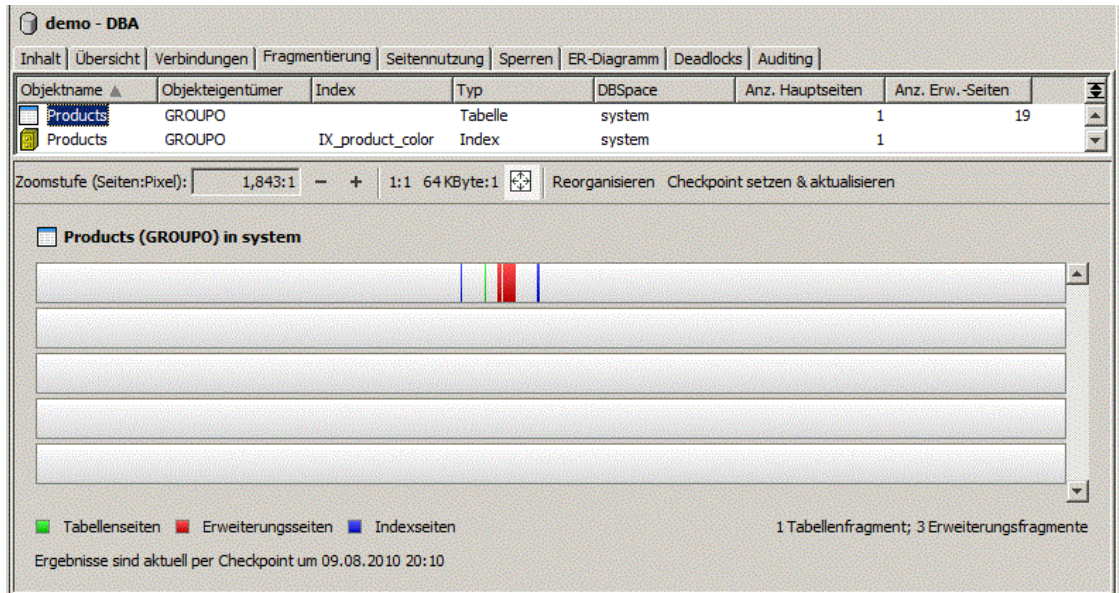
Sie können Fragmentierungsinformationen für ein Objekt in Sybase Central auf der Registerkarte **Fragmentierung** anzeigen.

Voraussetzungen

Sie müssen das CHECKPOINT-Systemprivileg haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Wählen Sie im linken Fensterausschnitt die Datenbank aus. Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **Fragmentierung**.
3. Wählen Sie im oberen Fensterausschnitt ein Objekt aus. Die Fragmentierungsinformationen werden in einer DBSpace-Zuordnung im unteren Fensterausschnitt angezeigt:
 - Wenn Sie eine Basistabelle auswählen, werden die Tabelle, ihre Erweiterungsseiten und zugehörigen Indexseiten in der DBSpace-Zuordnung im unteren Fensterausschnitt angezeigt.
 - Wenn Sie einen Index auswählen, werden seine Indexseiten in der DBSpace-Zuordnung im unteren Fensterausschnitt angezeigt.



4. Klicken Sie auf **Checkpoint setzen & aktualisieren**, um einen Checkpoint auszuführen und die neuesten Fragmentierungsinformationen anzuzeigen.
5. Sehen Sie sich die Seitenindizes an:
 - Führen Sie den Cursor in der DBSpace-Zuordnung im unteren Fensterausschnitt über einen farbigen vertikalen Balken, um den ersten und den letzten Seitenindex an dieser Position anzuzeigen.
 - Wenn Sie die Strg-Taste drücken und halten, während Sie die Maus über einen farbigen vertikalen Balken positionieren, können Sie alle Seitenindizes an dieser Position sehen.

Ergebnisse

Die Fragmentierungsdetails für das angegebene Objekt werden im unteren Fensterausschnitt in der DBSpace-Zuordnung angezeigt.

Reorganisieren von Basistabellen und Indizes

Administratoren können Basistabellen und Indizes in Sybase Central über die Registerkarte **Fragmentierung** reorganisieren.

Voraussetzungen

Sie müssen Eigentümer des Objekts sein oder das REORGANIZE ANY OBJECT-Systemprivileg haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.

2. Wählen Sie im linken Fensterausschnitt die Datenbank aus. Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **Fragmentierung**.
3. Wählen Sie im oberen Fensterausschnitt ein Objekt aus. Die Fragmentierungsinformationen werden in einer DBSpace-Zuordnung im unteren Fensterausschnitt angezeigt.
4. Verwenden Sie eine der folgenden Methoden, um das Objekt zu reorganisieren:
 - Klicken Sie auf **Reorganisieren**, um eine REORGANIZE TABLE-Anweisung für das gewählte Objekt auszuführen.
 - Kopieren Sie ein Objekt aus dem oberen Fensterausschnitt in einen **SQL-Anweisungen**-Fensterausschnitt von Interactive SQL. Eine TABLE REORGANIZE-Anweisung für das Objekt erscheint im Fensterausschnitt **SQL-Anweisungen**. Führen Sie die Anweisung aus.Diese Methode ist nützlich, wenn Sie die Objekte zu einem späteren Zeitpunkt reorganisieren möchten oder wenn Sie mit der Sybase Central-Sitzung fortfahren möchten, während Sie die Objekte reorganisieren.

Ergebnisse

Die angegebene Tabelle bzw. der angegebene Index wird reorganisiert.

Siehe auch

- „REORGANIZE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Indexfragmentierung und -schiefe (Skew) reduzieren

Indizes sollen Suchvorgänge in bestimmten Spalten beschleunigen. Sie können jedoch fragmentiert (weniger dicht) werden oder eine Schiefe zeigen (unausgewogen werden), wenn viele Löschvorgänge in der indizierten Tabelle ausgeführt werden.

Die Indexdichte gibt an, wie voll die Indexseiten durchschnittlich 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.

Um zu ermitteln, ob Ihre Datenbank Indizes enthält, die nicht akzeptable Ausmaße an Fragmentierung oder Schiefe haben, verwenden Sie den **Assistenten für die Anwendungsprofilerstellung**.

Sie können auch die Systemprozedur "sa_index_fragmentation" verwenden, um das Ausmaß von Indexfragmentierung und -schiefe zu überprüfen. Beispiel: Die folgende Anweisung ruft die Systemprozedur "sa_index_density" auf, um Indizes auf der Customer-Tabelle zu überprüfen.

```
CALL sa_index_density( 'Customers' );
```

TableName	TableId	IndexName	IndexID	IndexType	LeafPages	Density	Skew
Customers	718	CustomersKey	0	PKEY	1	0.127686	1.000000

TableName	TableId	IndexName	IndexID	IndexType	LeafPages	Density	Skew
Customers	718	IX_custo- mer_name	1	NUI	1	0.789795	1.000000

SQL Anywhere erstellt automatisch Indizes auf Primärschlüssel. Diese Indizes haben eine IndexID von 0 in den Ergebnissen der sa_index_density-Systemprozedur.

Wenn die Anzahl der Blattseiten niedrig ist, spielen die Dichte- und Schiefewerte keine Rolle. Dichte- und Schiefewerte 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.

Sie können das Ausmaß der Fragmentierung von Indizes, die Basistabellen zugeordnet sind, auch mithilfe der Registerkarte **Fragmentierung** im SQL Anywhere-Plug-In prüfen.

Siehe auch

- „Verwenden des Assistenten für die Anwendungsprofilerstellung“ auf Seite 157
- „REORGANIZE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Die Registerkarte Fragmentierung (SQL Anywhere-Plug-In)“ auf Seite 251
- „Die Registerkarte Fragmentierung (SQL Anywhere-Plug-In)“ auf Seite 251
- „sa_index_density-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Indizes“ auf Seite 27
- „Die Registerkarte Fragmentierung (SQL Anywhere-Plug-In)“ auf Seite 251

Tipp: Tabellenstruktur normalisieren

Eine oder mehrere Datenbanktabellen können mehrere Kopien der gleichen Daten enthalten (z.B. eine Spalte, die in mehreren Tabellen wiederholt wird), und die Tabelle muss möglicherweise normalisiert werden.

Die Normalisierung reduziert das Auftreten von Duplikaten in einer relationalen Datenbank. Nehmen wir z.B. an, dass Ihre Firmenangehörigen an mehreren unterschiedlichen Standorten arbeiten. Für die Normalisierung der Datenbank erwägen Sie die Möglichkeit, die Daten über die jeweiligen Büros, etwa die Adresse und die wichtigsten Telefonnummern, in eine separate Tabelle einzugeben, anstatt diese Angaben bei jedem Mitarbeiter zu wiederholen.

Wenn die Informationen nur selten doppelt vorkommen, ist es für Sie wahrscheinlich günstiger, die Informationen zu wiederholen und ihre Integrität mit Triggern oder anderen Integritätsregeln aufrechtzuerhalten.

Siehe auch

- „Datenbankerstellung“ [*SQL Anywhere Server - Datenbankadministration*]

Tipp: Kaskadierende referenzielle Aktionen vermeiden

Kaskadierende referenzielle Aktionen sind kostenträchtig, da sie bei jeder Transaktion Aktualisierungen in mehreren Tabellen bewirken. Dies wirkt sich auf die Performance aus. Wenn für den Fremdschlüssel aus "Employees" zu "Departments" zum Beispiel mit ON UPDATE CASCADE eine kaskadierende Aktualisierung festgelegt ist, wird bei einer Aktualisierung der Abteilungskennung automatisch auch die Tabelle "Employees" aktualisiert. Kaskadierende referenzielle Aktionen mögen günstig sein, doch ist es möglicherweise effizienter, sie in der Anwendungslogik zu implementieren.

Siehe auch

- „Datenintegrität“ auf Seite 849

Tipp: Integritätsregeln deklarieren

Nicht deklarierte Primärschlüssel/Fremdschlüssel-Beziehungen bestehen zwischen Tabellen, wenn es eine implizite Beziehung zwischen den Spaltenwerten in verschiedenen Tabellen gibt. Es stimmt, dass ein Nicht-Deklariere der Beziehung Zeit bei der Index-Wartung sparen kann, doch kann das Deklarieren der Beziehung die Performance von Abfragen steigern, wenn Joins stattfinden, weil das Kostenmodell eine bessere Schätzung durchführen kann.

Siehe auch

- „Tabellen- und Spalten-Integritätsregeln“ auf Seite 859

Tipp: Verschiedene Dateien auf verschiedene Geräte platzieren

Laufwerke arbeiten viel langsamer als moderne Prozessoren oder RAM. Häufig wird ein Datenbankserver schon verlangsamt, weil er darauf wartet, dass der Plattenspeicher Seiten liest oder schreibt.

Sie können die Datenbankperformance verbessern, indem Sie verschiedene Datenbankdateien auf unterschiedlichen physischen Geräten ablegen. Während zum Beispiel ein Laufwerk mit der Auslagerung von Datenbankseiten im Cache beschäftigt ist, kann auf einem anderen Gerät in die Logdatei geschrieben werden. Damit Sie diese Vorteile nutzen können, müssen die Geräte voneinander unabhängig sein. Eine einzelne Platte, die in kleinere logische Laufwerke partitioniert ist, wird wahrscheinlich nur wenig nützen.

SQL Anywhere verwendet vier Dateitypen: die **Datenbankdatei**, die **Transaktionslogdatei**, den **Transaktionslog-Spiegel** und die **temporäre Datei**. Diese Dateien sollten sich auf separaten Laufwerken befinden.

Zum Schutz gegen Datenträgerfehler ist es empfehlenswert, die Datenbankdatei und die Transaktionslogdatei auf physisch separaten Laufwerken abzulegen.

Wenn Sie die Transaktionslog-Spiegeldatei und die temporäre Datei auf physisch separaten Laufwerken ablegen, kann dies dazu beitragen, dass SQL Anywhere schneller läuft. SQL Anywhere schreibt effizienter in Transaktionslogdatei und Transaktionslog-Spiegeldatei, wenn diese sich auf separaten Laufwerken befinden. Wenn der Datenbankserver die temporäre Datei benutzen muss, hängt die Gesamt-

Datenbankperformance stark von der Geschwindigkeit des Laufwerks ab, das die temporäre Datei enthält. Da viele Vorgänge, für die die temporäre Datei verwendet wird, auch das Abrufen von Informationen aus der Datenbank erfordern, können Vorgänge gleichzeitig ausgeführt werden, wenn die temporäre Datei auf einem separaten Gerät abgelegt ist.

Eine Datenbank kann in bis zu 13 separaten Dateien gespeichert werden (der Hauptdatei und 12 DBSpaces), die sich wiederum auf separaten Laufwerken befinden können. Platzieren Sie Tabellen in separaten DBSpaces, damit gemeinsame Join-Vorgänge die Informationen aus verschiedenen DBSpaces lesen.

Wenn Sie alle Tabellen oder Indizes an einem anderen Speicherort erstellen als dem System-DBSpace, wird der System-DBSpace nur für das Checkpoint-Log und die Systemtabellen benutzt. Dies ist nützlich, wenn das Checkpoint-Log aus Gründen der Performance auf einem Laufwerk getrennt von den anderen Datenbankobjekten gespeichert wird. Um Basistabellen in einem anderen DBSpace zu erstellen, ändern Sie alle CREATE TABLE-Anweisungen so, dass mit der IN DBSPACE-Klausel der alternative DBSpace angegeben wird, oder ändern Sie die Einstellung der default_dbspace-Option, bevor Sie Tabellen erstellen. Temporäre Tabellen können nur im DBSpace TEMPORARY erstellt werden.

Eine ähnliche Methode besteht darin, die temporäre Datei und die Datenbankdateien auf ein RAID-Device oder ein Stripeset zu platzieren. Obwohl solche Devices wie ein logisches Laufwerk funktionieren, steigern sie die Performance deutlich, indem sie Dateien auf viele physische Laufwerke aufteilen und mehrere Leseköpfe für den Zugriff auf die Daten verwenden.

Sie können die Option -fc beim Starten des Datenbankservers angeben, um eine Callback-Funktion zu implementieren, wenn der Datenbankserver Speichermangel im Dateisystem feststellt.

Siehe auch

- „Datenbankdateitypen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Transaktionslog-Dienstprogramm (dblog)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Speicherorte von Transaktionslogs ändern (Sybase Central)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Speicherorte von Transaktionslogs ändern (Befehlszeile)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Tipp: Arbeitstabellen in der Abfrageverarbeitung verwenden ("Alle Zeilen" als Optimierungsziel verwenden)“ auf Seite 239
- „Sicherung und Datenwiederherstellung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „SATMP-Umgebungsvariable“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenträgerfehler“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Zusätzliche Hinweise zu DBSpaces“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „default_dbspace-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption -fc“ [[SQL Anywhere Server - Datenbankadministration](#)]

Tipp: Ihre Datenbank neu aufsetzen

Ihre Datenbank neu aufzusetzen bedeutet, dass Ihre gesamte Datenbank entladen und neu geladen wird. Manchmal wird dieser Vorgang auch als Upgrade des Datenbankdateiformats bezeichnet.

Beim Neuaufbau werden alle Daten aus der Datenbank ausgelesen und danach einheitlich und zusammenhängend wieder eingelesen. Ähnlich wie bei der Defragmentierung der Festplatte wird die Performance gesteigert und der Speicherplatz besser genutzt: Das gibt Ihnen auch die Gelegenheit, bestimmte Einstellungen zu ändern.

Siehe auch

- „Neuaufbau von Datenbanken“ auf Seite 763

Tipp: Mit Schlüsseln die Abfrageperformance steigern

Primärschlüssel und Fremdschlüssel werden vor allem für die Validierung benutzt, können jedoch auch die Performance der Datenbank verbessern.

Beispiel

Das folgende Beispiel veranschaulicht, wie Primärschlüssel dazu beitragen können, dass Abfragen schneller ausgeführt werden.

```
SELECT *  
FROM Employees  
WHERE EmployeeID = 390;
```

Die einfachste Ausführung dieser Abfrage besteht für den Datenbankserver darin, alle 75 Zeilen in der Tabelle "Employee" zu durchsuchen und die Employee-ID in jeder Zeile zu überprüfen, um festzustellen, ob sie 390 ist. Das braucht bei 75 Mitarbeitern nicht sehr lange, aber bei Tabellen mit Tausenden von Einträgen kann eine sequenzielle Suche sehr zeitaufwändig sein.

Die von jedem Primär- und Fremdschlüssel dargestellten Integritätsregeln zur referenziellen Integrität werden von SQL Anywhere mithilfe eines Indexes erzwungen, der implizit mit jeder Primär- bzw. Fremdschlüsseldeklaration erstellt wird. Die Spalte "EmployeeID" ist der Primärschlüssel für die Tabelle "Employees". Mit dem entsprechenden Primärschlüsselindex kann die Mitarbeiternummer 390 sehr rasch abgerufen werden. Die schnelle Suche braucht fast genauso viel Zeit, ganz gleich, ob die Tabelle "Employees" 100 oder 1000000 Zeilen umfasst.

Es werden automatisch separate Indizes für Primär- und Fremdschlüssel erstellt. Dadurch kann SQL Anywhere eine Vielzahl von Vorgängen effizienter ausführen.

Siehe auch

- „Beziehungen zwischen Tabellen“ [*SQL Anywhere 16 - Einführung*]

Tipp: Primärschlüssel-Breite vermindern

Breite Primärschlüssel bestehen aus zwei oder mehr Spalten. Je mehr Spalten Ihr Primärschlüssel benötigt, desto größer ist die Beanspruchung des Datenbankservers. Eine Verminderung der Spaltenanzahl bei Ihren Primärschlüsseln kann die Performance verbessern.

Siehe auch

- „Primärschlüssel“ auf Seite 17
- „Tipp: Mit Schlüsseln die Abfrageperformance steigern“ auf Seite 258

Tipps: Tabellenbreiten vermindern

Tabellen, bei denen die Summe der Breite der Spalten (oder die Größe einer einzelnen Zeile) die Seitengröße der Datenbank überschreiten und die über zwei oder mehr Datenbankseiten aufgeteilt werden müssen, werden "breite Tabellen" genannt. Je mehr Seiten eine Zeile benötigt, desto länger braucht der Datenbankserver, die einzelnen Zeilen zu lesen. Wenn Sie breite Tabellen und eine schlechte Performance haben, kann es sinnvoll sein, Ihre Tabellen weiter zu normalisieren, um die Anzahl der Spalten zu vermindern. Wenn das nicht möglich ist, kann eine größere Seitengröße der Datenbank hilfreich sein, vor allem wenn die meisten Tabellen breit sind.

Siehe auch

- „Datenbankerstellung“ [*SQL Anywhere Server - Datenbankadministration*]

Tipps: Spaltenreihenfolge in Tabellen überprüfen

Die Reihenfolge der Spalten in einer Tabelle wirkt sich auf die Performance aus. Auf die Spalten in einer Zeile wird sequenziell in der Reihenfolge ihrer Erstellung zugegriffen. Um z.B. auf Spalten am Ende der Zeile zuzugreifen, durchläuft der Datenbankserver alle Spalten bis dahin. Sie sollten die Reihenfolge der Spalten so festlegen, dass schmale und/oder häufig benutzte Spalten in der Tabelle vor selten benutzten und/oder breiteren Spalten platziert werden.

Breite Spalten sind Spalten mit einer Größe von über 15 Byte oder mit LONG-Datentypen (z.B. LONG VARCHAR) oder als XML definierte Spalten. Primärschlüsselspalten werden immer am Beginn einer Zeile gespeichert.

Siehe auch

- „Datenbankerstellung“ [*SQL Anywhere Server - Datenbankadministration*]
- „Primärschlüssel“ auf Seite 17

Tipps: Kostenträchtige Trigger ersetzen

Bewerten Sie die Verwendung der Trigger um festzustellen, ob nicht manche der Trigger von Funktionen ersetzt werden können, die im Datenbankserver zur Verfügung stehen. So können zum Beispiel Trigger, die Spalten mit dem Zeitpunkt der letzten Aktualisierung und Benutzerinformationen aktualisieren, durch die entsprechenden Spezialwerte im Datenbankserver ersetzt werden. Überdies kann die Verwendung der Standardeinstellungen bei vorhandenen Triggern die Performance verbessern.

Siehe auch

- „Trigger“ auf Seite 98

Tipp: Angemessene Seitengröße verwenden

Die Seitengröße, die Sie auswählen, kann sich auf die Performance der Datenbank auswirken. Große bzw. kleine Seitengrößen haben sowohl Vor- als auch Nachteile.

SQL Anywhere versucht, Seiten so weit wie möglich anzufüllen. Leerer Speicherplatz nimmt nur dann zu, wenn neue Objekte zu groß sind, um in den leeren Platz auf vorhandenen Seiten zu passen. Daher wirkt sich ein Anpassen der Seitengröße nicht unbedingt signifikant auf die Gesamtgröße Ihrer Datenbank aus.

Es wird dringend empfohlen, beim Wählen einer Seitengröße die Performance zu prüfen (ebenso wie andere Aspekte). Danach wählen Sie die kleinste Seitengröße, die zufriedenstellende Ergebnisse liefert. Es ist wichtig, dass eine korrekte und angemessene Seitengröße gewählt wird, wenn Sie vorhaben, mehrere Datenbanken auf demselben Server zu starten.

Kleinere Seiten enthalten weniger Informationen und führen daher zu einer weniger effizienten Speichernutzung, v.a. wenn Zeilen eingefügt werden, die etwas größer als eine halbe Seite sind: Kleinere Seitengrößen ermöglichen es, dass SQL Anywhere mit weniger Ressourcen laufen kann, weil mehr Seiten in einem Cache mit derselben Größe gespeichert werden können. Kleine Seiten sind nützlich, wenn Ihre Datenbank auf einem kleinen Computer mit beschränktem Speicher ausgeführt wird. Sie können auch sinnvoll sein, wenn Ihre Datenbank hauptsächlich zum Abrufen von kleinen Informationseinheiten von nicht zusammenhängenden Speicherorten verwendet wird.

Größere Seiten helfen SQL Anywhere, Datenbanken effizienter zu lesen. Sie sind meistens auch bei großen Datenbanken nützlich, und bei Abfragen, die sequenzielle Table-Scans durchführen. Häufig ermöglicht es das physische Design von Plattenspeichern, dass wenige große Blöcke effizienter als viele kleine abgerufen werden. Ein weiterer Vorzug großer Seitengrößen ist es, dass die Auffächerung Ihrer Indizes verbessert wird, wodurch die Anzahl der Indexebenen vermindert und es Tabellen ermöglicht wird, mehr Spalten zu enthalten. Falls Sie eine größere Seitengröße wählen, sollten Sie auch die Cachegröße erhöhen, da sonst nur eine geringere Zahl der großen Seiten in den Cache passt. Wenn Ihr Cache nicht genügend Seiten aufnehmen kann, wird die Performance leiden, da der Datenbankserver beginnt, häufig benutzte Seiten auf den Plattenspeicher auszulagern.

Größere Seitengrößen bringen zusätzliche Speicheranforderungen mit sich. Überdies sind extrem umfangreiche Seiten (16 kB oder 32 kB) für die meisten Anwendungen nicht zu empfehlen, außer Sie sind sich sicher, dass jederzeit ein großer Datenbankserver-Cache verfügbar ist.

Die Speicherbeanspruchung des Datenbankservers ist proportional zur Anzahl der geladenen Datenbanken und zur Seitengröße der Datenbanken. Es wird dringend empfohlen, dass Sie die Performance testen (und sonstige Tests durchführen), wenn Sie eine Seitengröße wählen. Danach wählen Sie die kleinste Seitengröße (≥ 4 kB), die zufriedenstellende Ergebnisse liefert. Es ist wichtig, dass eine korrekte (und vernünftige) Seitengröße gewählt wird, wenn Sie vorhaben, eine große Anzahl von Datenbanken auf demselben Server zu starten.

Es ist nicht möglich, die Seitengröße einer vorhandenen Datenbank zu ändern. Stattdessen müssen Sie eine neue Datenbank erstellen und die Option `-p` von `dbinit` verwenden, um die Seitengröße anzugeben. Der folgende Befehl erstellt zum Beispiel eine Datenbank mit 4-kB-Seiten.

```
dbinit -p 4096 new.db
```

Sie können die Anweisung CREATE DATABASE auch mit einer PAGE SIZE-Klausel benutzen, um eine Datenbank mit der neuen Seitengröße zu erstellen.

SQL Anywhere erstellt für jede Tabelle eine Bitmap, welche die Position jeder Tabellenseite in der kompletten DBSpace-Datei reflektiert. Der Datenbankserver verwendet die Bitmap, um große Blöcke (64 kB) von Tabellenseiten anstelle von einzelnen Seiten zu lesen. Diese Effizienz, die auch als **Gruppenlesevorgang** bezeichnet wird, reduziert die Gesamtzahl der I/O-Vorgänge auf der Festplatte und verbessert die Performance. Benutzer können die Kriterien des Datenbankservers für die Erstellung oder Verwendung von Bitmaps nicht beeinflussen.

Seitengröße und Indizes

Die Seitengröße hat auch Einfluss auf Indizes. Jede Indexsuche erfordert einen Seitenlesevorgang pro Indexebene plus einen Seitenlesevorgang für die Tabellenseite, und für eine einzige Abfrage können mehrere Tausend Indexsuchen erforderlich sein. Die Seitengröße kann sich signifikant auf die Auffächerung auswirken, was wiederum Folgen für die Tiefe des für eine Tabelle erforderlichen Indexes hat. Eine hohe Auffächerung bedeutet oftmals, dass weniger Indexebenen benötigt werden, wodurch die Suchvorgänge beachtlich verbessert werden. Für große Datenbanken mit einer beträchtlichen Anzahl von Seiten erzielen Sie mit einer Seitengröße von 8 kB die beste Performance.

Gestreute Lesevorgänge

Wenn Sie mit Windows arbeiten, ermöglicht eine Mindestseitengröße von 4 kB dem Datenbankserver, einen großen zusammenhängenden Bereich von Datenbankseiten direkt in den entsprechenden Platz im Cache einzulesen und den 64 kB-Puffer komplett zu überspringen. Diese Eigenschaft kann die Performance bedeutsam verbessern.

Hinweis

Gestreute Lesevorgänge werden weder für Dateien auf entfernten Computern noch für Dateien verwendet, die durch einen UNC-Namen festgelegt werden, wie etwa `\\meinComputer\meinVerz\meineDB.db`.

Siehe auch

- „Hinweise zur maximalen Seitengröße“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „CREATE DATABASE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Tipp: Mit AUTOINCREMENT Primärschlüssel erstellen

Primärschlüsselwerte müssen eindeutig sein. Auch wenn es eine Reihe von Möglichkeiten gibt, eindeutige Werte für Primärschlüssel zu erstellen, ist es die effizienteste Methode, den Standardspaltenwert auf AUTOINCREMENT einzustellen. Sie können diesen Standardwert bei jeder Spalte verwenden, in der Sie eindeutige Werte aufrechterhalten wollen. Die Erstellung von Primärschlüsselwerten mithilfe der AUTOINCREMENT-Funktion ist schneller als andere Methoden, weil der Wert vom Datenbankserver erstellt wird.

Siehe auch

- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Tipp: Entsprechende Datentypen verwenden

Datentypen speichern Informationen über bestimmte Datensätze, wie den Bereich der Werte, die mit diesen Werten durchführbaren Vorgänge bzw. wie die Werte im Speicher gespeichert werden. Sie können die Performance steigern, indem Sie den Ihren Daten entsprechenden Datentyp verwenden. Vermeiden Sie es zum Beispiel, einen CHAR-Datentyp Werten zuzuordnen, die nur numerische Daten enthalten. Und wo immer möglich, sollten Sie effiziente Datentypen anstelle der kostenträchtigeren numerischen und Zeichenfolgetypen verwenden.

Siehe auch

- „SQL-Datentypen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Tipp: Massenvorgangsmethoden verwenden

Wenn Sie große Datenmengen in Ihre Datenbank laden, können Sie die speziellen Tools nutzen, die für diese Aufgaben zur Verfügung stehen.

Wenn Sie große Dateien laden, ist es effizienter, Indizes für die Tabelle zu erstellen, nachdem die Daten geladen wurden.

Siehe auch

- „Performance-Aspekte von Massenvorgängen“ auf Seite 721

Tipp: Ressourcenwächter verwenden

Wenn Sie einige Benutzer und Rollen einrichten, können Sie die Privilegien für eine Datenbank verwalten. Ein anderer Aspekt von Datenbanksicherheit und Datenbankverwaltung ist die Begrenzung der Ressourcen, die ein einzelner Benutzer verwenden kann.

Zum Beispiel kann es sinnvoll sein, für eine einzelne Verbindung Arbeitsspeicher oder CPU-Ressourcen nur begrenzt zur Verfügung zu stellen. So kann vermieden werden, dass eine einzelne Verbindung die Arbeit anderer Benutzer in der Datenbank verlangsamt.

SQL Anywhere stellt eine Reihe von Datenbankoptionen bereit, die Sie zur Steuerung der Ressourcen verwenden können. Diese Optionen werden **Ressourcenwächter** genannt. Diese Optionen können mithilfe der SET OPTION-Anweisung gesetzt werden.

Ressourcen, die verwaltet werden können

Sie können die folgenden Optionen verwenden, um Ressourcen zu verwalten:

- **max_cursor_count** Mit dieser Option wird die Anzahl der Cursor für eine Verbindung begrenzt.
- **max_statement_count** Begrenzt die Anzahl der vorbereiteten Anweisungen für eine Verbindung.
- **priority** Legt die Prioritätsstufe fest, in der Anforderungen einer Verbindung ausgeführt werden.

- **max_priority** Steuert die maximale Prioritätsstufe für Verbindungen.

Siehe auch

- „max_cursor_count-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „max_statement_count-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „priority-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „max_priority-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankoptionen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „SET OPTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Tipp: Anforderungen zwischen Client und Server reduzieren

Falls Ihr Netzwerk lange Latenzzeiten aufweist oder Ihre Anwendung viele Anforderungen zum Öffnen und Schließen von Cursors sendet, können Sie die Netzwerkverbindungsparameter LazyClose und PrefetchOnOpen verwenden, um die Anzahl der Anforderungen zwischen Client und Server zu verringern und dadurch die Performance zu verbessern.

Siehe auch

- „Verbindungsparameter LazyClose (LCLOSE)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Verbindungsparameter PrefetchOnOpen“ [[SQL Anywhere Server - Datenbankadministration](#)]

Tipp: Komprimierung mit Umsicht verwenden

Komprimierung von Paketen für Verbindungen

Unter bestimmten Umständen erreichen Sie eine bedeutende Verbesserung der Performance, indem Sie die Komprimierung für eine bzw. alle Verbindungen aktivieren und die Mindestgröße, bei welcher Pakete komprimiert werden, anpassen.

Sie können ermitteln, ob die Komprimierung Vorteile bietet, indem Sie eine Performanceanalyse für Ihr Netzwerk durchführen und die entsprechende Anwendung ausführen, bevor Sie die Kommunikationskomprimierung in einer Produktionsumgebung einsetzen.

Die Aktivierung der Komprimierung erhöht die in den Datenpaketen gespeicherte Datenmenge, wodurch die Anzahl der zur Übertragung einer bestimmten Datenmenge erforderlichen Pakete verringert wird. Eine geringere Paketanzahl bedeutet eine schnelle Datenübertragung.

Wenn Sie den Schwellenwert für die Komprimierung angeben, können Sie die Mindestgröße der Datenpakete auswählen, die Sie komprimieren wollen. Der optimale Wert für den Komprimierungsschwellenwert hängt von einer Vielzahl von Faktoren ab, u.a. auch von der Art und der Geschwindigkeit des verwendeten Netzwerks.

Datenbankkomprimierung

Die Verwendung der Komprimierung auf Datei- oder Zeilenebene für Datenbank- und Logdateien wird nicht empfohlen, da die Komprimierungsstufe die Kosten für die I/O-Vorgänge erheblich steigern kann.

Siehe auch

- „Komprimierungseinstellungen für die Kommunikation“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Verbindungsparameter Compress (COMP)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Verbindungsparameter CompressionThreshold (COMPTH)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Tipp: Paketgröße ändern, um die Performance zu steigern

In den meisten Fällen beeinflusst eine Erhöhung der Paketgröße die Performance nicht. In einigen Fällen kann die Performance abnehmen. Die Erhöhung der Paketgröße erhöht auch die Menge des Speichers, der vom Client und dem Datenbankserver verwendet wird. In einer Produktionsumgebung führen Sie eine Performanceanalyse Ihres Netzwerks mithilfe Ihrer Anwendungen durch, bevor Sie Anpassungen vornehmen.

Eine höhere Paketgröße kann die Antwortzeit der Datenbank verbessern, vor allem bei Abfragen, die eine große Datenmenge zwischen einem Client und einem Datenbankserver transportieren. Beispiel: Eine Anwendung, die viele große (mehr als 64 KB) Ergebnismengen hat oder viele große (mehr als 64 KB) BLOBs über sehr schnelle lokale Netzwerke transportiert, kann eine messbare Leistungssteigerung aufweisen, wenn die Paketgröße höher eingestellt wird als der Standardwert.

Sie können die Paketgröße mit der `dbeng16/dbsrv16`-Serveroption `-p` oder durch Setzen des `CommBufferSize (CBSIZE)`-Verbindungsparameters in Ihrer Verbindungszeichenfolge festlegen.

Siehe auch

- „Datenbankserveroption `-p`“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Verbindungsparameter `CommBufferSize (CBSIZE)`“ [[SQL Anywhere Server - Datenbankadministration](#)]
- `PacketSize`-Verbindungseigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]

Praktische Einführungen in die Anwendungsprofilerstellung

In diesen praktischen Einführungen in die Anwendungsprofilerstellung erfahren Sie, wie Sie den **Assistenten für die Anwendungsprofilerstellung** und den **Assistenten für die Datenbankprotokollierung** einsetzen, um verbreitete Performanceprobleme zu analysieren, wie z.B. Deadlocks, langsame Anweisungen, Indexfragmentierung, Tabellenfragmentierung und langsame Prozeduren.

Vorsicht

Die praktischen Einführungen in die Anwendungsprofilerstellung verwenden die Testdatenbank `app_profiling.db` und nicht die Beispieldatenbank (`demo.db`). Verwenden Sie für die praktischen Einführungen nicht die Beispieldatenbank.

Praktische Einführung: Testdatenbank für die praktischen Einführungen in die Anwendungsprofilerstellung erstellen

Erstellen Sie eine Testdatenbank namens *app_profiling.db*, die für die praktischen Einführungen zur Anwendungsprofilerstellung verwendet wird.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. Erstellen Sie das Verzeichnis *C:\AppProfilingTutorial*.
2. Führen Sie die folgende Prozedur aus, um die Testdatenbank *app_profiling.db* zu erstellen, die Daten aus der Beispieldatenbank enthält:

```
newdemo c:\AppProfilingTutorial\app_profiling.db
```

Ergebnisse

Die Datenbank *app_profiling.db* wird erstellt.

Siehe auch

- „Verwenden des Assistenten für die Anwendungsprofilerstellung“ auf Seite 157
- „Diagnoseprotokollierung“ auf Seite 170
- „Neuerstellung der Beispieldatenbank (demo.db)“ [*SQL Anywhere 16 - Einführung*]
- „Praktische Einführung: Verbindung mit der Beispieldatenbank herstellen“ [*SQL Anywhere Server - Datenbankadministration*]

Praktische Einführung: Deadlocks diagnostizieren

In den Lektionen dieser praktischen Einführung erfahren Sie, wie Sie den **Assistenten für die Datenbankprotokollierung** einsetzen, um Deadlocks anzuzeigen, die in Ihrer Datenbank auftreten. Sie können den **Assistenten für die Datenbankprotokollierung** auch verwenden, um die Bedingungen zu untersuchen, unter denen Deadlocks auftreten, sowie die Verbindungen, die sie verursachen.

Privilegien

Für diese praktische Einführung sind keine Privilegien erforderlich, weil in dieser praktischen Einführung die von Ihnen erstellte Testdatenbank *app_profiling.db* verwendet wird. Verbinden Sie sich als Standardbenutzer DBA.

Lektion 1: Datenbank einrichten

Richten Sie die Testdatenbank für die Anwendungsprofilerstellung *app_profiling.db* ein, um einen Deadlock zu erstellen, indem Sie zwei Tabellen und zwei Prozeduren erstellen.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Deadlocks diagnostizieren](#)“.

In dieser praktischen Einführung wird vorausgesetzt, dass Sie die Testdatenbank *app_profiling.db* erstellt haben. Siehe „[Praktische Einführung: Testdatenbank für die praktischen Einführungen in die Anwendungsprofilerstellung erstellen](#)“ auf Seite 265.

Kontext und Bemerkungen

Deadlocks treten auf, wenn zwei oder mehr Transaktionen einander blockieren. Transaktion A erfordert beispielsweise Zugriff auf Tabelle B, aber Tabelle B ist von Transaktion B gesperrt. Transaktion B erfordert Zugriff auf Tabelle A, aber Tabelle A ist von Transaktion A gesperrt. Es entsteht ein zyklischer Blockierungskonflikt.

Ein Anzeichen für Deadlocks ist die Rückgabe von SQLCODE -306 und -307. Um einen Deadlock aufzulösen, setzt SQL Anywhere automatisch die letzte Anweisung zurück, die den Deadlock bewirkt hat. Es entstehen Performanceprobleme, wenn Anweisungen fortwährend zurückgesetzt werden.

Aufgabe

1. Starten Sie Interactive SQL. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Sybase Central**.
2. Stellen Sie in Interactive SQL folgendermaßen eine Verbindung mit *app_profiling.db* her:
 - a. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus, um eine Verbindung mit der Testdatenbank *app_profiling.db* herzustellen:
 - i. Geben Sie im Feld **Authentifizierung** auf **Datenbank** ein.
 - ii. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - iii. Im Feld **Kennwort** geben Sie **sql** ein.
 - iv. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Eine Datenbank auf diesem Computer starten und eine Verbindung herstellen** aus.
 - v. Geben Sie im Feld **Datenbankdatei** Folgendes ein: **C:\AppProfilingTutorial\app_profiling.db**.
 - vi. Geben Sie im Feld **Startzeile** Folgendes ein: **dbeng16 -x tcpip**.
 - b. Klicken Sie auf **Verbinden**.
3. Führen Sie in Interactive SQL die folgenden SQL-Anweisungen aus:
 - a. Erstellen Sie zwei Tabellen:

```
CREATE TABLE "DBA"."deadlock1" (  
    "id" UNSIGNED BIGINT NOT NULL DEFAULT AUTOINCREMENT,  
    "val" CHAR(1) );  
CREATE TABLE "DBA"."deadlock2" (
```

```
"id" UNSIGNED BIGINT NOT NULL DEFAULT AUTOINCREMENT,  
"val" CHAR(1) );
```

- b. Fügen Sie Werte in jede Tabelle ein.

```
INSERT INTO "deadlock1"("val") VALUES('x');  
INSERT INTO "deadlock2"("val") VALUES('x');
```

- c. Erstellen Sie zwei Prozeduren:

```
CREATE PROCEDURE "DBA"."proc_deadlock1"( )  
BEGIN  
    LOCK TABLE "DBA"."deadlock1" IN EXCLUSIVE MODE;  
    WAITFOR DELAY '00:00:20:000';  
    UPDATE deadlock2 SET val='y';  
END;  
CREATE PROCEDURE "DBA"."proc_deadlock2"( )  
BEGIN  
    LOCK TABLE "DBA"."deadlock2" IN EXCLUSIVE MODE;  
    WAITFOR DELAY '00:00:20:000';  
    UPDATE deadlock1 SET val='y';  
END;
```

- d. Schreiben Sie die Änderungen fest, die Sie an der Datenbank durchgeführt haben:

```
COMMIT;
```

4. Schließen Sie Interactive SQL.

Ergebnisse

Sie haben zwei Tabellen und zwei Prozeduren erstellt, mit denen Sie einen Deadlock erstellen.

Nächste Schritte

Gehen Sie weiter zu „[Lektion 2: Deadlock erstellen und Daten erfassen](#)“ auf Seite 267.

Siehe auch

- „Datenbanken auf lokalen Computern starten und Verbindungen herstellen (Sybase Central oder Interactive SQL)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Deadlock erkannt“ [[Fehlermeldungen](#)]
- „Alle Threads sind blockiert“ [[Fehlermeldungen](#)]

Lektion 2: Deadlock erstellen und Daten erfassen

Sie erstellen einen Deadlock und zeichnen Informationen darüber auf, indem Sie den **Assistenten für die Datenbankprotokollierung** verwenden, um eine Sitzung für die Diagnoseprotokollierung zu erstellen.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Deadlocks diagnostizieren](#)“.

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe „[Lektion 1: Datenbank einrichten](#)“ auf Seite 265.

Kontext und Bemerkungen

In der praktischen Einführung in die Anwendungsprofilerstellung werden Protokollierungsinformationen in der Testdatenbank (*app_profiling.db*) gespeichert, bei der es sich um die in den praktischen Einführungen verwendete Datenbank handelt. Wenn Sie allerdings das Profil einer Datenbank mit großer Arbeitslast erstellen, ist es empfehlenswert, Protokollierungsdaten in einer separaten Datenbank zu speichern, um Auswirkungen auf die Performance der Produktionsdatenbank zu vermeiden.

Aufgabe

1. Starten Sie Sybase Central. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Sybase Central**.
2. Stellen Sie in Sybase Central folgendermaßen eine Verbindung mit *app_profiling.db* her:
 - a. Klicken Sie auf **Verbindungen » Verbinden mit SQL Anywhere 16**.
 - b. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus, um eine Verbindung mit der Testdatenbank *app_profiling.db* herzustellen:
 - i. Klicken Sie im Feld **Authentifizierung** auf **Datenbank**.
 - ii. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - iii. Im Feld **Kennwort** geben Sie **sql** ein.
 - iv. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Eine Datenbank auf diesem Computer starten und eine Verbindung herstellen** aus.
 - v. Geben Sie im Feld **Datenbankdatei** Folgendes ein: **C:\AppProfilingTutorial\app_profiling.db**.
 - vi. Geben Sie im Feld **Startzeile** Folgendes ein: **dbeng16 -x tcpip**.
 - c. Klicken Sie auf **Verbinden**.
3. Klicken Sie in Sybase Central auf **Modus » Anwendungsprofil**.

Wenn der **Assistent für die Anwendungsprofilerstellung** eingeblendet wird, klicken Sie auf **Abbrechen**.

4. Starten Sie den **Assistenten für die Datenbankprotokollierung** wie folgt:
 - a. Klicken Sie auf **Datei » Protokollierung**.
 - b. Auf der Seite **Willkommen** klicken Sie auf **Weiter**.
 - c. Klicken Sie auf der Seite **Protokollierungsdetailltiefe** auf **Hohe Detaillierung (bei kurzfristiger, intensiver Überwachung empfohlen)** und anschließend auf **Weiter**.
 - d. Auf der Seite **Protokollierungsstufen bearbeiten** klicken Sie auf **Weiter**.
 - e. Klicken Sie auf der Seite **Externe Datenbank erstellen** auf **Keine neue Datenbank erstellen. Vorhandene Datenbank soll benutzt werden** und anschließend auf **Weiter**.
 - f. Klicken Sie auf der Seite **Protokollierung starten** auf **Protokollierungsdaten in dieser Datenbank speichern**.

- g. Wenn Sie für die Menge der gespeicherten Protokollierungsdaten keine Grenzen festlegen möchten, klicken Sie auf **Keine Beschränkung** und anschließend auf **Fertig stellen**.
5. Erstellen Sie den Deadlock folgendermaßen:
- Vergewissern Sie sich, dass im linken Fensterausschnitt **app_profiling - DBA** ausgewählt ist, und klicken Sie dann auf **Datei » Interactive SQL öffnen**.
Interactive SQL wird gestartet und stellt eine Verbindung mit der Datenbank *app_profiling.db* her.
 - Wiederholen Sie den vorherigen Schritt, um ein zweites Interactive SQL-Fenster zu öffnen.
 - Führen Sie im ersten Interactive SQL-Fenster die folgende SQL-Anweisung aus:

```
CALL "DBA"."proc_deadlock1"();
```
 - Führen Sie im zweiten Interactive SQL Fenster innerhalb von 20 Sekunden nach der Ausführung der SQL-Anweisung im ersten Interactive SQL-Fenster die folgende SQL-Anweisung aus:

```
CALL "DBA"."proc_deadlock2"();
```

Nach ein paar Augenblicken wird ein **ISQL-Fehler**-Fenster angezeigt, das angibt, dass ein Deadlock erkannt wurde.

Der Deadlock ist aufgetreten, da proc_deadlock1 Zugriff auf die Tabelle "deadlock2" benötigt, die von proc_deadlock2 gesperrt wurde. Gleichzeitig benötigt proc_deadlock2 Zugriff auf die Tabelle "deadlock1", die von proc_deadlock1 gesperrt wurde.
 - Klicken Sie auf **OK**.
6. SQL Anywhere hat die Deadlock-Vorgänge gestoppt und Sie können die Interactive SQL-Fenster schließen.
7. Stoppen Sie in Sybase Central die Protokollierungssitzung, indem Sie im linken Fensterausschnitt **app_profiling - DBA** auswählen und anschließend auf **Datei » Protokollierung » Protokollierung mit Speichern stoppen** klicken.

Ergebnisse

Sie haben einen Deadlock erstellt und Informationen darüber erfasst.

Nächste Schritte

Gehen Sie weiter zu [„Lektion 3: Daten der Verbindungsblockierungen überprüfen“ auf Seite 270](#).

Siehe auch

- [„Transaktion blockieren und Deadlock“ auf Seite 901](#)
- [„Benutzerdefinierte Diagnoseprotokollierungsstufen“ auf Seite 175](#)
- [„Deadlocks“ auf Seite 903](#)
- [„Diagnoseprotokollierung“ auf Seite 170](#)

Lektion 3: Daten der Verbindungsblockierungen überprüfen

Verwenden Sie den Modus **Anwendungsprofil**, um eine grafische Darstellung der Verbindungen zu sehen, die an einem Deadlock teilnehmen. Der **Anwendungsprofilerstellung-Modus** stellt außerdem die Registerkarte **Verbindungsblockierungen** zur Verfügung, die zusätzliche Informationen über blockierte Verbindungen enthält.

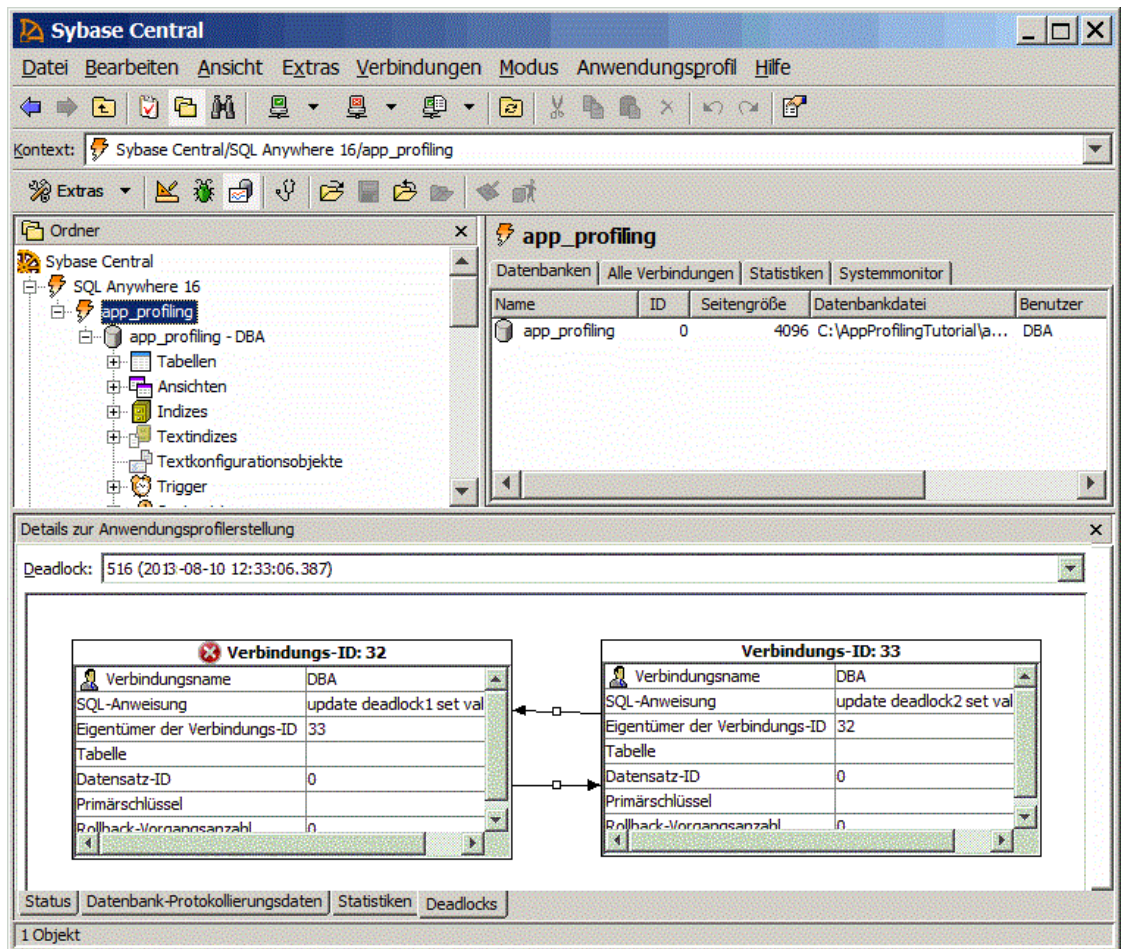
Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Deadlocks diagnostizieren](#)“.

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe „[Lektion 1: Datenbank einrichten](#)“ auf Seite 265.

Aufgabe

1. Öffnen Sie folgendermaßen die Analysedatei, die während der Protokollierungssitzung erstellt wurde:
 - a. Klicken Sie in Sybase Central auf **Anwendungsprofil** » **Analysedatei öffnen oder mit einer Protokollierungsdatenbank verbinden**.
 - b. Klicken Sie auf **In einer Protokollierungsdatenbank**.
 - c. Klicken Sie auf **Open (Öffnen)**.
 - d. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - e. Im Feld **Kennwort** geben Sie **sql** ein.
 - f. Klicken Sie in der Dropdown-Liste **Aktion** auf **Mit einer laufenden Datenbank auf diesem Computer verbinden**.
 - g. Im Feld **Datenbankname** geben Sie **app_profiling** ein.
 - h. Klicken Sie auf **Verbinden**.
2. Zeigen Sie wie folgt die grafische Darstellung des Deadlocks an:
 - a. Im Fensterausschnitt **Details zur Anwendungsprofilerstellung** klicken Sie auf die Registerkarte **Status** und wählen die aktuellste ID in der Liste **Protokollierungssitzungs-ID** aus.
Wenn der Fensterausschnitt **Details zur Anwendungsprofilerstellung** nicht eingeblendet wird, klicken Sie auf **Ansicht** » **Details zur Anwendungsprofilerstellung**.
 - b. Am unteren Rand des Fensterausschnitts **Details zur Anwendungsprofilerstellung** klicken Sie auf die Registerkarte **Deadlocks**. Der aktuellste Deadlock wird angezeigt. Klicken Sie auf die Liste **Deadlock**, um weitere Deadlocks anzuzeigen.
 - c. Die folgende Illustration zeigt, wie die UPDATE-Anweisungen eine Deadlock-Bedingung erzeugt haben.



Jede an einem Deadlock beteiligte Verbindung wird durch eine Tabelle mit den folgenden Feldern dargestellt:

- **Verbindungsname** Die Benutzer-ID, die die Verbindung geöffnet hat
- **SQL-Anweisung** Die an einem Deadlock beteiligte Anweisung. In diesem Fall wurde der Deadlock durch die UPDATE-Anweisungen in den Prozeduren bewirkt, die Sie für jede Instanz von Interactive SQL ausgeführt haben.
- **Eigentümer der Verbindungs-ID** Die ID der Verbindung, die die aktuelle Verbindung blockiert hat
- **Datensatz-ID** Die ID der Zeile, auf der die aktuelle Verbindung blockiert ist
- **Rollback-Vorgangsanzahl** Die Anzahl der Vorgänge, die aufgrund des Deadlocks zurückgesetzt werden müssen. In diesem Fall enthielten die Prozeduren nur die UPDATE-Anweisungen, daher ist die Anzahl 0.

- Um zusätzliche Deadlock-Daten anzuzeigen, z.B. wie häufig die Deadlocks auftreten und wie lange sie dauern, verwenden Sie folgendermaßen die Registerkarte **Verbindungsblockierungen**:

- a. Im Fensterausschnitt **Details zur Anwendungsprofilerstellung** klicken Sie auf die Registerkarte **Datenbank-Protokollierungsdaten**.
- b. Klicken Sie auf die Registerkarte **Verbindungsblockierungen** oberhalb der Registerkarte **Datenbank-Protokollierungsdaten**.

Die Registerkarte **Verbindungsblockierungen** wird eingeblendet und zeigt die Blockierungszeit, die Entblockierungszeit und die Dauer der einzelnen blockierten Verbindungen an.

Ergebnisse

Sie haben das Thema der Diagnose von Deadlocks der praktischen Einführung in die Anwendungsprofilerstellung abgeschlossen.

Siehe auch

- „Transaktion blockieren und Deadlock“ auf Seite 901
- „Benutzerdefinierte Diagnoseprotokollierungsstufen“ auf Seite 175
- „Deadlocks“ auf Seite 903
- „Diagnoseprotokollierung“ auf Seite 170

Praktische Einführung: Langsame Anweisungen diagnostizieren

In den Lektionen dieser praktischen Einführung erfahren Sie, wie Sie den **Assistenten für die Datenbankprotokollierung** einsetzen, um die Ausführungszeit von Anweisungen anzuzeigen, und wie Sie Anweisungen erkennen, die anscheinend nur langsam ausgeführt werden (z.B. Abfragen mit langen Laufzeiten).

Eine langsame Anweisung kommt vor, wenn der Datenbankserver lange braucht, um die Anweisung zu verarbeiten. Lange Verarbeitungszeiten können mehrere Ursachen haben, wie z.B. schlechte Datenbankplanung, falschen Gebrauch von Indizes, Index- und Tabellenfragmentierung oder eine zu geringe Cachegröße. Eine Anweisung kann auch langsam ausgeführt werden, weil sie schlecht formuliert ist oder keine effizienteren Verknüpfungen verwendet, um Ergebnisse zu erzielen.

Diese praktische Einführung beschreibt nicht, wie Sie langsame Anweisungen neu schreiben, da jede Anweisung spezielle Anforderungen haben kann. Dafür zeigt Ihnen diese praktische Einführung, wo Sie nach Ausführungszeiten suchen und wie Sie diese miteinander vergleichen, wenn Sie Abfragen unter Verwendung einer alternativen Syntax neu schreiben.

Privilegien

Für diese praktische Einführung sind keine Privilegien erforderlich, weil in dieser praktischen Einführung die von Ihnen erstellte Testdatenbank *app_profiling.db* verwendet wird. Verbinden Sie sich als Standardbenutzer DBA.

Siehe auch

- „Abfragen“ auf Seite 291
- „Joins: Daten aus mehreren Tabellen abrufen“ auf Seite 492
- „Verwendung von Unterabfragen“ auf Seite 600
- „Benutzerdefinierte Diagnoseprotokollierungsstufen“ auf Seite 175
- „Diagnoseprotokollierung“ auf Seite 170
- „Tipp: Abfrageperformance überwachen“ auf Seite 245
- „Performance-Probleme beheben“ auf Seite 189

Lektion 1: Diagnoseprotokollierungssitzung erstellen

Verwenden Sie den **Assistenten für die Datenbankprotokollierung**, um eine Sitzung für die Diagnoseprotokollierung zu erstellen, die die Daten über die Verarbeitung einer Anweisung, einschließlich der Dauer der Ausführung, erfasst.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: [„Praktische Einführung: Langsame Anweisungen diagnostizieren“](#).

In dieser praktischen Einführung wird vorausgesetzt, dass Sie die Testdatenbank *app_profiling.db* erstellt haben. Siehe [„Praktische Einführung: Testdatenbank für die praktischen Einführungen in die Anwendungsprofilerstellung erstellen“](#) auf Seite 265.

Aufgabe

1. Starten Sie Sybase Central. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Sybase Central**.
2. Stellen Sie in Sybase Central folgendermaßen eine Verbindung mit *app_profiling.db* her:
 - a. Klicken Sie auf **Verbindungen » Verbinden mit SQL Anywhere 16**.
 - b. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus, um eine Verbindung mit der Testdatenbank *app_profiling.db* herzustellen:
 - i. Klicken Sie im Feld **Authentifizierung** auf **Datenbank**.
 - ii. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - iii. Im Feld **Kennwort** geben Sie **sql** ein.
 - iv. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Eine Datenbank auf diesem Computer starten und eine Verbindung herstellen** aus.
 - v. Geben Sie im Feld **Datenbankdatei** Folgendes ein: **C:\AppProfilingTutorial\app_profiling.db**.
 - vi. Geben Sie im Feld **Startzeile** Folgendes ein: **dbeng16 -x tcpip**.
 - c. Klicken Sie auf **Verbinden**.

3. Klicken Sie in Sybase Central auf **Modus » Anwendungsprofil**.

Wenn der **Assistent für die Anwendungsprofilerstellung** eingeblendet wird, klicken Sie auf **Abbrechen**.

4. Starten Sie den **Assistenten für die Datenbankprotokollierung** wie folgt:
 - a. Klicken Sie auf **Datei » Protokollierung**.
 - b. Auf der Seite **Willkommen** klicken Sie auf **Weiter**.
 - c. Klicken Sie auf der Seite **Protokollierungsdetailtiefe** auf **Hohe Detaillierung (bei kurzfristiger, intensiver Überwachung empfohlen)** und anschließend auf **Weiter**.
 - d. Auf der Seite **Protokollierungsstufen bearbeiten** klicken Sie auf **Weiter**.
 - e. Klicken Sie auf der Seite **Externe Datenbank erstellen** auf **Keine neue Datenbank erstellen. Vorhandene Datenbank soll benutzt werden** und anschließend auf **Weiter**.
 - f. Klicken Sie auf der Seite **Protokollierung starten** auf **Protokollierungsdaten in dieser Datenbank speichern**.
 - g. Wenn Sie für die Menge der gespeicherten Protokollierungsdaten keine Grenzen festlegen möchten, klicken Sie auf **Keine Beschränkung** und anschließend auf **Fertig stellen**.
5. Vergewissern Sie sich, dass im linken Fensterausschnitt **app_profiling - DBA** ausgewählt ist, und klicken Sie dann auf **Datei » Interactive SQL öffnen**.

Interactive SQL wird gestartet und stellt eine Verbindung mit der Datenbank *app_profiling.db* her.

6. Führen Sie in Interactive SQL die folgende SQL-Anweisung aus:

```
SELECT SalesOrderItems.ID, LineID, ProductID, SalesOrderItems.Quantity,  
       ShipDate  
FROM SalesOrderItems, SalesOrders  
WHERE SalesOrders.CustomerID = 105 AND  
       SalesOrderItems.ID=SalesOrders.ID;
```

7. Beenden Sie Interactive SQL.
8. Wählen Sie in Sybase Central im linken Fensterausschnitt **app_profiling - DBA** und klicken Sie auf **Datei » Protokollierung » Protokollierung mit Speichern stoppen**, um die Protokollierungssitzung zu stoppen.

Ergebnisse

Sie haben Diagnoseinformationen während einer Protokollierungssitzung erfasst.

Nächste Schritte

Gehen Sie weiter zu „[Lektion 2: Vom Datenbankserver abgearbeitete Anweisungen überprüfen](#)“ auf Seite 275.

Siehe auch

- „Benutzerdefinierte Diagnoseprotokollierungsstufen“ auf Seite 175
- „Diagnoseprotokollierung“ auf Seite 170

Lektion 2: Vom Datenbankserver abgearbeitete Anweisungen überprüfen

Sie können ermitteln, bei welchen Anweisungen der Datenbankserver am längsten für die Prozessverarbeitung braucht, indem Sie die Registerkarten **Zusammenfassung** und **Detail** verwenden, die sich im Fensterausschnitt **Anwendungsprofilerstellung** in Sybase Central befinden.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Langsame Anweisungen diagnostizieren](#)“.

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe „[Lektion 1: Diagnoseprotokollierungssitzung erstellen](#)“ auf Seite 273.

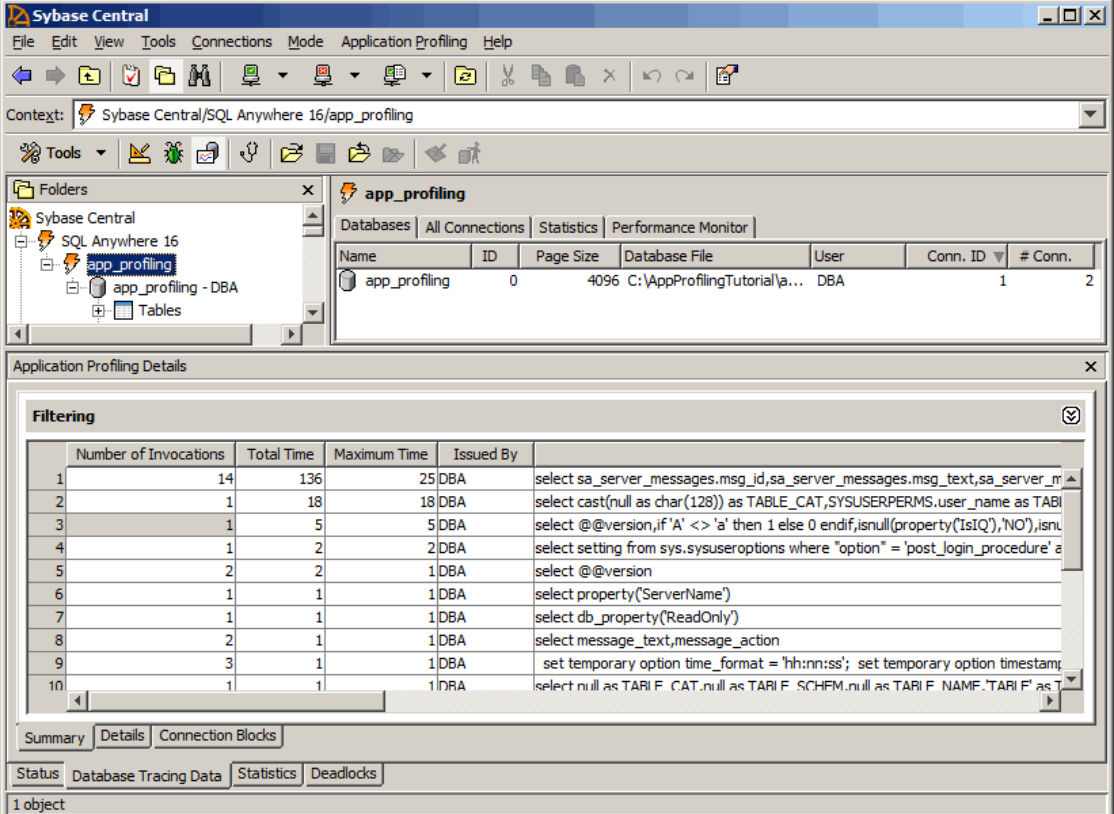
Aufgabe

1. Öffnen Sie die Analysedatei in Sybase Central folgendermaßen:
 - a. Klicken Sie auf **Anwendungsprofil** » **Analysedatei öffnen oder mit einer Protokollierungsdatenbank verbinden**.
 - b. Klicken Sie auf **In einer Protokollierungsdatenbank** und anschließend auf **Öffnen**.
 - c. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - d. Im Feld **Kennwort** geben Sie **sql** ein.
 - e. Klicken Sie in der Dropdown-Liste **Aktion** auf **Mit einer laufenden Datenbank auf diesem Computer verbinden**.
 - f. Im Feld **Datenbankname** geben Sie **app_profiling** ein.
 - g. Klicken Sie auf **Verbinden**.
 - h. Wenn der Fensterausschnitt **Details zur Anwendungsprofilerstellung** nicht am unteren Rand des Fensters eingeblendet wird, klicken Sie auf **Ansicht** » **Details zur Anwendungsprofilerstellung**.
2. Überprüfen Sie wie folgt die Ausführungszeiten von Anweisungen, die während der Protokollierungssitzung abgearbeitet wurden:
 - a. Auf der Registerkarte **Status** im Fensterausschnitt **Details zur Anwendungsprofilerstellung** wählen Sie die aktuellste ID (höchste Nummer) im Feld **Protokollierungssitzungs-ID** und klicken dann auf die Registerkarte **Datenbank-Protokollierungsdaten**.
Auf der Registerkarte **Zusammenfassung** werden die SQL-Anweisungen angezeigt, die Sie während der Sitzung ausgeführt haben. Sie sehen diese zusätzlichen Anweisungen, weil von

Ihnen ausgeführte Anweisungen automatisch die Ausführung von anderen Anweisungen (z.B. eines Triggers) bewirkt haben.

Auf der Registerkarte **Zusammenfassung** werden ähnliche Anweisungen sowie die Gesamtzahl der Aufrufe und die Gesamtzeit für deren Verarbeitung zusammengefasst. Die Anweisungen SELECT, INSERT, UPDATE und DELETE werden gemäß den von ihnen referenzierten Tabellen, Spalten und Ausdrücken in Gruppen zusammengefasst. Andere Anweisungen werden als Ganzes zusammengefasst (z.B. erscheinen alle CREATE TABLE-Anweisungen als einzelner Eintrag auf der Registerkarte **Zusammenfassung**). Eine Anweisung kann auf der Registerkarte **Zusammenfassung** als kostenträchtig angezeigt werden, weil es sich tatsächlich um eine einzelne kostenträchtige Anweisung handelt oder weil sie häufig ausgeführt wird.

Suchen Sie langsam ausgeführte Anweisungen auf Ihrem System, indem Sie die Spalten **Gesamtzeit** und **Maximale Dauer** überprüfen. Diese Spalten liefern für jede Anweisung, die vom Datenbankserver abgearbeitet wird, Angaben über die Ausführungszeiten.



The screenshot shows the Sybase Central interface with the 'Application Profiling Details' window open. The 'Filtering' tab is selected, displaying a table of SQL statements and their execution statistics. The table has columns: Number of Invocations, Total Time, Maximum Time, Issued By, and the SQL statement text.

	Number of Invocations	Total Time	Maximum Time	Issued By	SQL Statement
1	14	136	25	DBA	select sa_server_messages.msg_id,sa_server_messages.msg_text,sa_server_m
2	1	18	18	DBA	select cast(null as char(128)) as TABLE_CAT,SYSUSERPERMS.user_name as TABI
3	1	5	5	DBA	select @@version,if 'A' <> 'a' then 1 else 0 endif,isnull(property('IsIQ'),'NO'),isn
4	1	2	2	DBA	select setting from sys.sysuseroptions where "option" = 'post_login_procedure' a
5	2	2	1	DBA	select @@version
6	1	1	1	DBA	select property('ServerName')
7	1	1	1	DBA	select db_property('ReadOnly')
8	2	1	1	DBA	select message_text,message_action
9	3	1	1	DBA	set temporary option time_format = 'hh:nn:ss'; set temporary option timestam
10	1	1	1	DBA	select null as TABI F. CAT,null as TABI F. SCHFM,null as TABI F. NAME,'TARI F' as T

At the bottom of the window, there are tabs for 'Summary', 'Details', and 'Connection Blocks'. The 'Status' tab is currently selected, showing '1 object'.

- Um Informationen über die Verbindung anzuzeigen, die eine Anweisung ausgeführt hat, klicken Sie auf die Registerkarte **Details**, rechtsklicken Sie auf die Anweisung und klicken Sie auf **Verbindungsdetails für die ausgewählte Anweisung anzeigen**.

4. Um den für eine SQL-Anweisung verwendeten Ausführungsplan anzuzeigen, klicken Sie auf die Registerkarte **Details**, rechtsklicken Sie auf die Anweisung und klicken Sie auf **Weitere SQL-Anweisungsdetails für die ausgewählte Anweisung anzeigen**.

Das Fenster **SQL-Anweisungsdetails** wird eingeblendet, in dem der vollständige Text der Anweisung zusammen mit Details zum Kontext, in dem sie verwendet wurde, angezeigt werden. Der für die Anweisung angezeigte Text stimmt möglicherweise nicht mit der ursprünglichen, von Ihnen ausgeführten SQL-Anweisung überein. Stattdessen zeigt das Fenster **SQL-Anweisungsdetails** die Anweisung in ihrem umgeschriebenen Format an, wie sie vom Datenbankserver abgearbeitet wurde. Abfragen auf Ansichten können beispielsweise voneinander abweichen, weil die Ansichtsdefinitionen häufig vom Optimierer während der Ausführung der Abfrage umgeschrieben werden.

Klicken Sie auf die Registerkarte **Abfrageinformationen** am unteren Rand des Fensters **SQL-Anweisungsdetails**, um den Ausführungsplan anzuzeigen.

Ergebnisse

Sie haben die praktische Einführung in die Diagnose langsam ausführender Anweisungen abgeschlossen.

Siehe auch

- [„Verwendung von Unterabfragen“ auf Seite 600](#)
- [„Anforderungs-Trace-Analysen ausführen“ auf Seite 192](#)
- [„Abfragen“ auf Seite 291](#)
- [„Joins: Daten aus mehreren Tabellen abrufen“ auf Seite 492](#)
- [„Verwendung von Unterabfragen“ auf Seite 600](#)
- [„Benutzerdefinierte Diagnoseprotokollierungsstufen“ auf Seite 175](#)
- [„Diagnoseprotokollierung“ auf Seite 170](#)

Praktische Einführung: Indexfragmentierung diagnostizieren

In den Lektionen dieser praktischen Einführung lernen Sie, wie Sie den **Assistenten für die Anwendungsprofilerstellung** oder Interactive SQL verwenden, um zu ermitteln, ob Ihre Datenbank ein nicht akzeptables Ausmaß an Indexfragmentierung hat, und um ggf. die Indexfragmentierung zu reparieren.

Wenn ein Index erstellt wird, werden Tabellendaten eingelesen und Werte für den Index auf Indexseiten in einer logischen Reihenfolge aufgezeichnet. Wenn sich Daten in der Tabelle ändern, können neue Indexwerte zwischen den bestehenden Werten eingefügt werden. Um die logische Reihenfolge von Indexwerten aufrechtzuerhalten, muss der Datenbankserver möglicherweise neue Indexseiten erstellen, um bestehende Daten aufzunehmen, die verschoben werden. Die neuen Seiten befinden sich üblicherweise nicht neben den Seiten, auf denen die Werte ursprünglich gespeichert wurden. Diese kumulative Verschlechterung in der Reihenfolge der Indexseiten wird Indexfragmentierung genannt.

Wenn häufig ausgeführte Abfragen länger brauchen, besonders bei Tabellen, bei denen große Zeilenblöcke kontinuierlich eingefügt, aktualisiert oder gelöscht werden, stellt dies ein Symptom für die Indexfragmentierung dar.

Privilegien

Für diese praktische Einführung sind keine Privilegien erforderlich, weil in dieser praktischen Einführung die von Ihnen erstellte Testdatenbank *app_profiling.db* verwendet wird. Verbinden Sie sich als Standardbenutzer DBA.

Lektion 1: Indexfragmentierung mit dem Assistenten für die Anwendungsprofilerstellung erkennen und beheben

Verwenden Sie den **Assistenten für die Anwendungsprofilerstellung**, um Indexfragmentierung zu erkennen und zu beheben. Sie sollten Ihre Produktionsdatenbank regelmäßig auf Fragmentierung überprüfen.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Indexfragmentierung diagnostizieren](#)“.

In dieser praktischen Einführung wird vorausgesetzt, dass Sie die Testdatenbank *app_profiling.db* erstellt haben. Siehe „[Praktische Einführung: Testdatenbank für die praktischen Einführungen in die Anwendungsprofilerstellung erstellen](#)“ auf Seite 265.

Aufgabe

1. Starten Sie Sybase Central. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Sybase Central**.
2. Stellen Sie in Sybase Central folgendermaßen eine Verbindung mit *app_profiling.db* her:
 - a. Klicken Sie auf **Verbindungen » Verbinden mit SQL Anywhere 16**.
 - b. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus, um eine Verbindung mit der Testdatenbank *app_profiling.db* herzustellen.
 - i. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - ii. Im Feld **Kennwort** geben Sie **sql** ein.
 - iii. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Eine Datenbank auf diesem Computer starten und eine Verbindung herstellen** aus.
 - iv. Geben Sie im Feld **Datenbankdatei** Folgendes ein: **C:\AppProfilingTutorial\app_profiling.db**.
 - v. Geben Sie im Feld **Startzeile** Folgendes ein: **dbeng16 -x tcpip**.
 - c. Klicken Sie auf **Verbinden**.
3. Klicken Sie auf **Modus » Anwendungsprofil**.

Wenn der **Assistent für die Anwendungsprofilerstellung** nicht eingeblendet wird, klicken Sie auf **Anwendungsprofil » Anwendungsprofil-Assistenten öffnen**.

4. Starten Sie den **Assistenten für die Datenbankprotokollierung** folgendermaßen:
 - a. Auf der Seite **Willkommen** klicken Sie auf **Weiter**.
 - b. Klicken Sie auf der Seite **Profilerstellungsoptionen** auf **Gesamt-Datenbankperformance, basierend auf dem Datenbankschema** und anschließend auf **Weiter**.
 - c. Auf der Seite **Analysedatei** geben Sie im Feld **Die Analyse in folgender Datei speichern** den Dateipfad *C:\AppProfilingTutorial\analysis* ein.
 - d. Klicken Sie auf **Fertig stellen**.

Eine Liste von Empfehlungen wird im Fensterausschnitt **Details zur Anwendungsprofilerstellung** angezeigt.

5. Wenn **Fragmentierte Indizes** angezeigt wird, doppelklicken Sie darauf. Ein Fenster **Empfehlung** wird eingeblendet, das eine SQL-Anweisung enthält, die Sie ausführen können, um die Indexfragmentierung aufzulösen.
6. Schließen Sie Sybase Central.

Ergebnisse

Sie haben den **Assistenten für die Anwendungsprofilerstellung** verwendet, um Indexfragmentierung zu erkennen und zu beheben.

Siehe auch

- „[Neuerstellen eines Indexes](#)“ auf Seite 35
- „[Indexfragmentierung und -schiefe \(Skew\) reduzieren](#)“ auf Seite 254
- „[Anwendungsprofilerstellung](#)“ auf Seite 156
- „[Diagnoseprotokollierung](#)“ auf Seite 170

Lektion 2: Indexfragmentierung mit Interactive SQL erkennen und beheben

Verwenden Sie Interactive SQL, um Indexfragmentierung zu erkennen und zu beheben. Sie sollten Ihre Produktionsdatenbank regelmäßig auf Fragmentierung überprüfen.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Indexfragmentierung diagnostizieren](#)“.

In dieser praktischen Einführung wird vorausgesetzt, dass Sie die Testdatenbank *app_profiling.db* erstellt haben. Siehe „[Praktische Einführung: Testdatenbank für die praktischen Einführungen in die Anwendungsprofilerstellung erstellen](#)“ auf Seite 265.

Achten Sie darauf, dass keine anderen Verbindungen zu *app_profiling.db* vorhanden sind.

Aufgabe

1. Starten Sie Interactive SQL. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Interactive SQL**.
2. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus, um eine Verbindung mit der Testdatenbank *app_profiling.db* herzustellen.
 - a. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - b. Im Feld **Kennwort** geben Sie **sql** ein.
 - c. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Eine Datenbank auf diesem Computer starten und eine Verbindung herstellen** aus.
 - d. Geben Sie im Feld **Datenbankdatei** Folgendes ein: **C:\AppProfilingTutorial\app_profiling.db**.
 - e. Geben Sie im Feld **Startzeile** Folgendes ein: **dbeng16 -x tcpip**.
3. Klicken Sie auf **Verbinden**.
4. Führen Sie in Interactive SQL die folgende SQL-Anweisung aus, um die Indexdichte der Employees-Tabelle zu testen:

```
CALL sa_index_density( 'Employees' );
```

Dichtewerte liegen zwischen 0 und 1. Näher an 1 liegenden Werte zeigen eine geringe Indexfragmentierung an. Werte unter 0,5 zeigen ein Ausmaß an Indexfragmentierung an, das sich auf die Performance auswirken kann.

Hinweis

Die für die Indizes für die Tabelle "Employees" angezeigten Werte weisen auf Fragmentierungsprobleme hin, da die Werte in der Spalte "Dichte" deutlich unter 0,5 liegen. Diese Zahlen sind jedoch übertrieben niedrig, da die Tabelle sehr klein ist.

5. Führen Sie in Interactive SQL die folgende ALTER INDEX ... REBUILD-Anweisung aus, um die Dichte eines Indexes zu verbessern:

```
ALTER INDEX PRIMARY KEY ON Employees REBUILD;
```

6. Schließen Sie Interactive SQL.

Ergebnisse

Sie haben Interactive SQL verwendet, um Indexfragmentierung zu erkennen und zu beheben.

Siehe auch

- „ALTER INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „sa_index_density-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „REORGANIZE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Neuerstellen eines Indexes“ auf Seite 35
- „Indexfragmentierung und -schiefe (Skew) reduzieren“ auf Seite 254
- „Anwendungsprofilerstellung“ auf Seite 156
- „Benutzerdefinierte Diagnoseprotokollierungsstufen“ auf Seite 175
- „Diagnoseprotokollierung“ auf Seite 170

Praktische Einführung: Tabellenfragmentierung diagnostizieren

In den Lektionen dieser praktischen Einführung lernen Sie, wie Sie den **Assistenten für die Anwendungsprofilerstellung** oder Interactive SQL verwenden, um zu ermitteln, ob Ihre Datenbank eine Tabellenfragmentierung aufweist, und um ggf. die Tabellenfragmentierung zu reparieren.

Tabellendaten werden auf Datenbankseiten gespeichert. Wenn Datenmanipulationsanweisungen wie INSERT, UPDATE und DELETE in einer Tabelle ausgeführt werden, werden Zeilen möglicherweise nicht zusammenhängend gespeichert oder auf mehrere Seiten aufgeteilt. Auch wenn die CPU-Aktivität hoch ist, kann sich eine Tabellenfragmentierung auf die Performance von Abfragen auswirken, die einen Scan auf der Tabelle erfordern.

Privilegien

Für diese praktische Einführung sind keine Privilegien erforderlich, weil in dieser praktischen Einführung die von Ihnen erstellte Testdatenbank *app_profiling.db* verwendet wird. Verbinden Sie sich als Standardbenutzer DBA.

Lektion 1: Tabellenfragmentierung mit dem Assistenten für die Anwendungsprofilerstellung erkennen und beheben

Verwenden Sie den **Assistenten für die Anwendungsprofilerstellung**, um Tabellenfragmentierung zu erkennen und zu beheben. Sie sollten Ihre Produktionsdatenbank regelmäßig auf Fragmentierung überprüfen.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Tabellenfragmentierung diagnostizieren](#)“.

In dieser praktischen Einführung wird vorausgesetzt, dass Sie die Testdatenbank *app_profiling.db* erstellt haben. Siehe „[Praktische Einführung: Testdatenbank für die praktischen Einführungen in die Anwendungsprofilerstellung erstellen](#)“ auf Seite 265.

Aufgabe

1. Starten Sie Sybase Central. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Sybase Central**.
2. Stellen Sie in Sybase Central folgendermaßen eine Verbindung mit *app_profiling.db* her:
 - a. Klicken Sie auf **Verbindungen » Verbinden mit SQL Anywhere 16**.
 - b. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus, um eine Verbindung mit der Testdatenbank *app_profiling.db* herzustellen:
 - i. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - ii. Im Feld **Kennwort** geben Sie **sql** ein.
 - iii. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Eine Datenbank auf diesem Computer starten und eine Verbindung herstellen** aus.
 - iv. Geben Sie im Feld **Datenbankdatei** Folgendes ein: **C:\AppProfilingTutorial\app_profiling.db**.
 - v. Geben Sie im Feld **Startzeile** Folgendes ein: **dbeng16 -x tcpip**.
 - c. Klicken Sie auf **Verbinden**.
3. Klicken Sie in Sybase Central auf **Modus » Anwendungsprofil**.

Wenn der **Assistent für die Anwendungsprofilerstellung** nicht eingeblendet wird, klicken Sie auf **Anwendungsprofil » Anwendungsprofil-Assistenten öffnen**.
4. Starten Sie den **Assistenten für die Datenbankprotokollierung** folgendermaßen:
 - a. Klicken Sie auf der Seite **Profileroptionsen** auf **Gesamt-Datenbankperformance, basierend auf dem Datenbankschema**.
 - b. Auf der Seite **Analysedatei** geben Sie im Feld **Die Analyse in folgender Datei speichern** den Dateipfad **C:\AppProfilingTutorial\analysis** ein. Wenn Sie aufgefordert werden, die Datei zu ersetzen, da sie bereits vorhanden ist, klicken Sie auf **Ja**.
 - c. Klicken Sie auf **Fertig stellen**.

Eine Liste von Empfehlungen wird im Fensterausschnitt **Details zur Anwendungsprofilerstellung** angezeigt.
5. Wenn Sie **Fragmentierte Tabellen** sehen, doppelklicken Sie darauf. Ein Fenster **Empfehlung** wird eingeblendet, das eine SQL-Anweisung enthält, die Sie ausführen können, um die Tabellenfragmentierung aufzulösen.
6. Schließen Sie Sybase Central.

Ergebnisse

Sie haben die Anwendungsprofilerstellung zur Ermittlung und Behebung der Tabellenfragmentierung verwendet.

Siehe auch

- „Tabellenfragmentierung reduzieren“ auf Seite 249
- „Anwendungsprofilerstellung“ auf Seite 156
- „Diagnoseprotokollierung“ auf Seite 170

Lektion 2: Tabellenfragmentierung mit Interactive SQL erkennen und beheben

Verwenden Sie Interactive SQL, um Tabellenfragmentierung zu erkennen und zu beheben. Sie sollten Ihre Produktionsdatenbank regelmäßig auf Fragmentierung überprüfen.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Tabellenfragmentierung diagnostizieren](#)“.

In dieser praktischen Einführung wird vorausgesetzt, dass Sie die Testdatenbank *app_profiling.db* erstellt haben. Siehe „[Praktische Einführung: Testdatenbank für die praktischen Einführungen in die Anwendungsprofilerstellung erstellen](#)“ auf Seite 265.

Achten Sie darauf, dass keine anderen Verbindungen zu *app_profiling.db* vorhanden sind.

Aufgabe

1. Starten Sie Interactive SQL. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Interactive SQL**.
2. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus, um eine Verbindung mit der Testdatenbank *app_profiling.db* herzustellen:
 - a. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - b. Im Feld **Kennwort** geben Sie **sql** ein.
 - c. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Eine Datenbank auf diesem Computer starten und eine Verbindung herstellen** aus.
 - d. Geben Sie im Feld **Datenbankdatei** Folgendes ein: **C:\AppProfilingTutorial\app_profiling.db**.
 - e. Geben Sie im Feld **Startzeile** Folgendes ein: **dbeng16 -x tcpip**.
3. Klicken Sie auf **Verbinden**.
4. Führen Sie in Interactive SQL die folgende SQL-Anweisung aus, um die Employees-Tabelle auf Tabellenfragmentierung zu testen:

```
CALL sa_table_fragmentation( 'Employees' );
```

Wenn der Wert in der Spalte "segs_per_row" (die Anzahl der Segmente pro Zeile) größer als 1,1 ist, liegt eine Tabellenfragmentierung vor. Höhere Fragmentierungsgrade können sich negativ auf die Performance auswirken.

5. Führen Sie in Interactive SQL die folgende REORGANIZE TABLE-Anweisung aus, um die Tabellenfragmentierung zu vermindern:

```
REORGANIZE TABLE Employees;
```

6. Schließen Sie Interactive SQL.

Ergebnisse

Sie haben Interactive SQL verwendet, um Tabellenfragmentierung zu diagnostizieren.

Siehe auch

- „sa_table_fragmentation-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REORGANIZE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Tabellenfragmentierung reduzieren“ auf Seite 249
- „Anwendungsprofilerstellung“ auf Seite 156
- „Benutzerdefinierte Diagnoseprotokollierungsstufen“ auf Seite 175
- „Diagnoseprotokollierung“ auf Seite 170

Praktische Einführung: Baselining mit Prozedurprofilerstellung

Verwenden Sie den **Assistenten für die Anwendungsprofilerstellung**, um eine Baseline für Vergleichszwecke zur Verbesserung der Performance zu erhalten.

Die Prozedurprofilerstellung liefert Ausführungszeitmessungen für Prozeduren, benutzerdefinierte Funktionen, Ereignisse, Systemtrigger und Trigger. Sie können Ihre gespeicherten Ergebnisse als Baseline festlegen, inkrementelle Änderungen an einer Prozedur vornehmen und die Prozedur nach jeder durchgeführten Änderung ausführen. Sie können dann die neuen Ergebnisse mit der Baseline vergleichen, um zu ermitteln, ob Ihre Änderungen wie geplant funktioniert haben.

Privilegien

Für diese praktische Einführung sind keine Privilegien erforderlich, weil in dieser praktischen Einführung die von Ihnen erstellte Testdatenbank *app_profiling.db* verwendet wird. Verbinden Sie sich als Standardbenutzer DBA.

Siehe auch

- „Anwendungsprofilerstellung“ auf Seite 156
- „Ergebnisse der Prozedurprofilerstellung“ auf Seite 162
- „Prozedurprofilerstellung im Modus für die Anwendungsprofilerstellung“ auf Seite 158
- „Benutzerdefinierte Diagnoseprotokollierungsstufen“ auf Seite 175
- „Diagnoseprotokollierung“ auf Seite 170

Lektion 1: Baselining-Prozedur erstellen

Erstellen Sie eine Baselining-Prozedur für die Prozedurprofilerstellung.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Baselineing mit Prozedurprofilerstellung](#)“.

In dieser praktischen Einführung wird vorausgesetzt, dass Sie die Testdatenbank *app_profiling.db* erstellt haben. Siehe „[Praktische Einführung: Testdatenbank für die praktischen Einführungen in die Anwendungsprofilerstellung erstellen](#)“ auf Seite 265.

Aufgabe

1. Starten Sie Interactive SQL. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Interactive SQL**.
2. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus, um eine Verbindung mit der Testdatenbank *app_profiling.db* herzustellen:
 - a. Klicken Sie im Feld **Authentifizierung** auf **Datenbank**.
 - b. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - c. Im Feld **Kennwort** geben Sie **sql** ein.
 - d. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Eine Datenbank auf diesem Computer starten und eine Verbindung herstellen** aus.
 - e. Geben Sie im Feld **Datenbankdatei** Folgendes ein: **C:\AppProfilingTutorial\app_profiling.db**.
 - f. Geben Sie im Feld **Startzeile** Folgendes ein: **dbsrv16 -x tcpip**.

3. Klicken Sie auf **Verbinden**.

4. Führen Sie in Interactive SQL die folgenden SQL-Anweisungen aus:

- a. Erstellen Sie eine Tabelle:

```
CREATE TABLE table1 (
  Count INT );
```

- b. Erstellen Sie eine Baselineing-Prozedur:

```
CREATE PROCEDURE baseline( )
BEGIN
  INSERT table1
  SELECT COUNT (*)
  FROM rowgenerator r1, rowgenerator r2,
       rowgenerator r3
  WHERE r3.row_num < 5;
END;
```

- c. Schreiben Sie die Änderungen fest, die Sie an der Datenbank durchgeführt haben:

```
COMMIT;
```

5. Schließen Sie Interactive SQL.

Ergebnisse

Sie haben eine Baselining-Prozedur für Vergleichszwecke erstellt.

Nächste Schritte

Gehen Sie weiter zu „[Lektion 2: Aktualisierte Prozedur gegen die Baselining-Prozedur ausführen](#)“ auf Seite 286.

Lektion 2: Aktualisierte Prozedur gegen die Baselining-Prozedur ausführen

Aktualisieren Sie eine Prozedur und führen Sie sie mit der Baselining-Prozedur aus. Erfassen Sie Diagnoseprotokollierungsdaten über die beiden Prozeduren zur Prozedurprofilerstellung.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Baselining mit Prozedurprofilerstellung](#)“.

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe „[Lektion 1: Baselining-Prozedur erstellen](#)“ auf Seite 284.

Aufgabe

1. Starten Sie Sybase Central. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Sybase Central**.
2. Stellen Sie in Sybase Central folgendermaßen eine Verbindung mit *app_profiling.db* her:
 - a. Klicken Sie auf **Verbindungen » Verbinden mit SQL Anywhere 16**.
 - b. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus, um eine Verbindung mit der Testdatenbank *app_profiling.db* herzustellen:
 - i. Klicken Sie im Feld **Authentifizierung** auf **Datenbank**.
 - ii. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - iii. Im Feld **Kennwort** geben Sie **sql** ein.
 - iv. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Eine Datenbank auf diesem Computer starten und eine Verbindung herstellen** aus.
 - v. Geben Sie im Feld **Datenbankdatei** Folgendes ein: **C:\AppProfilingTutorial\app_profiling.db**.
 - vi. Geben Sie im Feld **Startzeile** Folgendes ein: **dsrv16 -x tcpip**.
 - c. Klicken Sie auf **Verbinden**.
3. Klicken Sie in Sybase Central auf **Modus » Anwendungsprofil**.

Wenn der **Assistent für die Anwendungsprofilerstellung** nicht eingeblendet wird, klicken Sie auf **Anwendungsprofil » Anwendungsprofil-Assistenten öffnen**.

4. Starten Sie den **Assistenten für die Anwendungsprofilerstellung**

- a. Auf der Seite **Willkommen** klicken Sie auf **Weiter**.
- b. Klicken Sie auf der Seite **Profilerstellungsoptionen** auf **Ausführungszeit von gespeicherten Prozeduren, Funktionen, Triggern oder Ereignissen**.
- c. Klicken Sie auf **Fertig stellen**.

Der Datenbankserver beginnt mit der Prozedurprofilerstellung.

5. Doppelklicken Sie im linken Fensterausschnitt von Sybase Central auf **Prozeduren und Funktionen**.

6. Rechtsklicken Sie auf die Baseline-Prozedur und klicken Sie auf **Von Interactive SQL aus ausführen**. Da die Prozedurprofilerstellung aktiviert ist, werden die Ausführungsdetails für die Prozedur erfasst.

7. Schließen Sie Interactive SQL.

8. Zeigen Sie die Ergebnisse der Profilerstellung an.

- a. Wählen Sie im linken Fensterausschnitt von Sybase Central die Baseline-Prozedur aus.
- b. Klicken Sie im rechten Fensterausschnitt auf die Registerkarte **Ergebnisse der Profilerstellung**. Wenn keine Ergebnisse angezeigt werden, klicken Sie auf **Ansicht » Ordner aktualisieren**. Die Ausführungszeit wird für jede Zeile in der Baseline-Prozedur angezeigt.

9. Speichern Sie die Ergebnisse der Profilerstellung:

- a. Rechtsklicken Sie auf den Benutzer und klicken Sie auf **Eigenschaften**.
- b. Klicken Sie auf die Registerkarte **Profilerstellungseinstellungen**.
- c. Wählen Sie **Die aktuellen Profilerstellungsinformationen in der Datenbank in der folgenden Profilerstellungsprotokolldatei speichern** aus und geben Sie dann einen Speicherort und einen Namen für die Profilerstellungslogdatei an, z.B. *C:\AppProfilingTutorial\baseline*.
- d. Klicken Sie auf **Übernehmen**. Schließen Sie nicht das Eigenschaftsfenster.
Die Daten der Prozedurprofilerstellung werden in der angegebenen Profilerstellungslogdatei (.plg) gespeichert.

10. Aktivieren Sie das Baselineing anhand der Profilerstellungslogdatei:

- a. Wählen Sie auf der Registerkarte **Profilerstellungseinstellungen** des Fenster **Eigenschaften der Datenbank 'App_Profiling - DBA** die Option **Die folgenden Profilerstellungsinformationen in der folgenden Profilerstellungslogdatei als Basis für Vergleiche verwenden**.
- b. Navigieren Sie zur erstellten Profilerstellungslogdatei und wählen Sie sie aus.
- c. Klicken Sie auf **Übernehmen**.
- d. Klicken Sie auf **OK**, um das Fenster **App_Profiling - DBA Datenbankeigenschaften** zu schließen.

11. Führen Sie Änderungen an der Baseline-Prozedur durch:

- a. Klicken Sie in Sybase Central auf **Modus » Design**.
- b. Wählen Sie im linken Fensterausschnitt im Ordner **Prozeduren und Funktionen** die Baseline-Prozedur aus.
- c. Ersetzen Sie auf der Registerkarte **SQL** im rechten Fensterausschnitt die bestehende INSERT-Anweisung durch die folgende INSERT-Anweisung:

```
INSERT table1
SELECT COUNT ( * ) FROM rowgenerator r1, rowgenerator r2,
rowgenerator r3
WHERE r3.row_num < 250;
```

- d. Klicken Sie auf **Datei » Speichern**.

12. Rechtsklicken Sie unter **Prozeduren und Funktionen** auf die Baseline-Prozedur und klicken Sie auf **Von Interactive SQL aus ausführen**.

13. Beenden Sie Interactive SQL, wenn die Prozedur ausgeführt wurde.

Ergebnisse

Sie haben die Prozedur aktualisiert und gegen die Baselineprozedur ausgeführt. Dabei haben Sie Daten über die beiden Prozeduren in der Protokollierungssitzung erfasst.

Nächste Schritte

Gehen Sie weiter zu [„Lektion 3: Ergebnisse der Prozedurprofilerstellung vergleichen“](#) auf Seite 288.

Lektion 3: Ergebnisse der Prozedurprofilerstellung vergleichen

Vergleichen Sie die Ergebnisse der Prozedurprofilerstellung, um zu ermitteln, ob die aktualisierte Prozedur die Ausführungszeiten verbessert hat.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: [„Praktische Einführung: Baseline mit Prozedurprofilerstellung“](#).

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe [„Lektion 1: Baseline-Prozedur erstellen“](#) auf Seite 284.

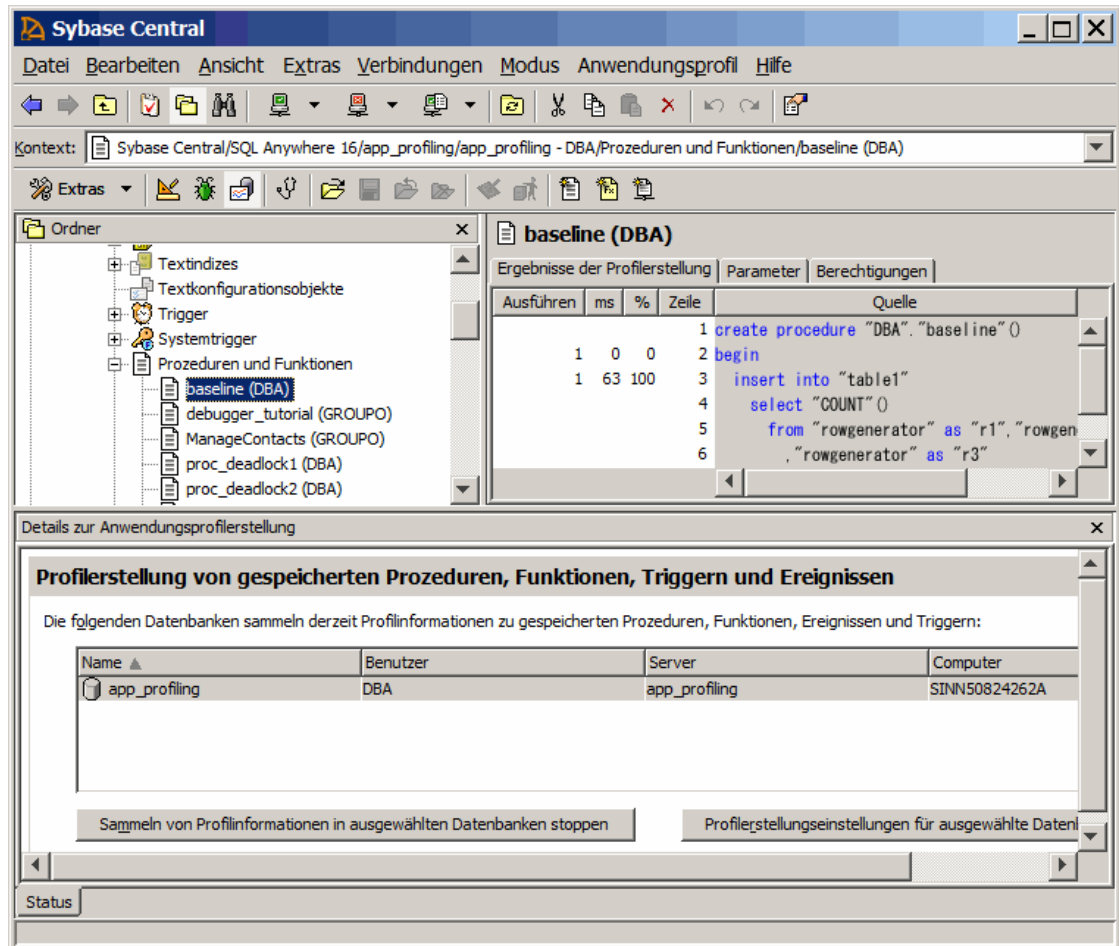
Aufgabe

1. Klicken Sie in Sybase Central auf **Modus » Anwendungsprofil**.

Wenn der **Assistent für die Anwendungsprofilerstellung** eingeblendet wird, klicken Sie auf **Abbrechen**.

2. Im linken Fensterausschnitt von Sybase Central klicken Sie auf der Registerkarte **Prozeduren und Funktionen** auf die Baselining-Prozedur.
3. Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **Ergebnisse der Profilerstellung**.
4. Klicken Sie auf **Ansicht » Ordner aktualisieren**.

Es werden zwei neue Spalten, **Ausf.** +/- und **ms** +/-, eingeblendet.



Die Spalten **Ausf.** +/- und **ms** +/- basieren auf dem Vergleich der Zahlen in der Profilerstellungslogdatei mit den Zahlen, die bei der letzten Ausführung der Prozedur erfasst wurden. Dabei werden die Anzahl der Ausführungen und die Dauer der Ausführung für jede einzelne Codezeile in der Prozedur verglichen.

Die Spalte **ms.** +/- ist von besonderem Interesse, weil sie angibt, ob Sie die Ausführungszeit für Codezeilen in der Prozedur verbessert haben. Schnellere Ausführungszeiten werden durch ein Minuszeichen und rote Schrift gekennzeichnet. Langsamere Ausführungszeiten werden durch ein Pluszeichen und in grüner Schrift dargestellt.

In dieser praktischen Einführung sollte der Wert in der Spalte **ms.** +/- aus einem +-Zeichen und einer Ausführungszeit in grüner Schrift bestehen. Die INSERT-Anweisung in der aktualisierten Prozedur hat eine längere Ausführungszeit als die INSERT-Anweisung in der Baseline-Prozedur.

Ergebnisse

Sie haben die praktische Einführung in das Baseline mit Prozedurprofilerstellung abgeschlossen.

Abfragen und Datenänderung

In diesem Abschnitt wird beschrieben, wie Daten abgefragt und geändert und Joins verwendet werden können. Er enthält einige Kapitel über einfache und komplexe Abfragen sowie Hinweise zum Einfügen, Löschen und Aktualisieren von Daten. Dieses Kapitel beschreibt auch eingehend, wie analytische Abfragen erstellt werden, die mehrdimensionale Ergebnisse zurückgeben.

Abfragen

In einer Abfrage werden Daten aus einer Datenbank abgefragt und Ergebnisse entgegengenommen. Dieser Vorgang wird insgesamt auch als Datenabfrage bezeichnet. Alle SQL-Abfragen werden mit der SELECT-Anweisung ausgedrückt. Sie benutzen die SELECT-Anweisung, um alle oder eine Teilmenge der Zeilen in einer oder mehreren Tabellen abzurufen und um alle oder eine Teilmenge der Spalten in einer oder mehreren Tabellen abzurufen.

Weitere Hinweise zur Abfrageverarbeitung in SQL Anywhere, einschließlich Abfrageoptimierung, Selektivitätsschätzung und Kostenschätzung, finden Sie im *Whitepaper "Query Processing Based on SQL Anywhere 12.0.1 Architecture"* unter <http://www.sybase.com/detail?id=1096047T>.

SELECT-Anweisung und Abfragen

Mit der SELECT-Anweisung werden Informationen aus einer Datenbank abgerufen, die dann von der Clientanwendung verwendet werden. SELECT-Anweisungen werden auch **Abfragen** genannt. Die Informationen werden in Form einer Ergebnismenge an die Clientanwendung gesendet. Der Client kann dann die Ergebnismenge verarbeiten. Interactive SQL zeigt beispielsweise die Ergebnismenge im Fensterausschnitt "Ergebnisse" an. Die Ergebnismengen bestehen aus einer Reihe von Zeilen, genauso wie Tabellen in der Datenbank.

SELECT-Anweisungen enthalten **Klauseln**, die den Bereich der zurückzugebenden Ergebnisse festlegen. In der folgenden SELECT-Syntax ist jede neue Zeile eine separate Klausel. Nur die gebräuchlichen Klauseln werden hier aufgelistet:

```
SELECT select-list
[ FROM table-expression ]
[ WHERE search-condition ]
[ GROUP BY column-name ]
[ HAVING search-condition ]
[ ORDER BY { expression | integer } ]
```

Die Klauseln in der SELECT-Anwendung sind:

- Die SELECT-Klausel legt die Spalten fest, die Sie abfragen wollen. Sie ist die einzige in einer SELECT-Anweisung erforderliche Klausel.
- Die FROM-Klausel legt die Tabellen fest, aus denen die Spalten bezogen werden. Sie ist in allen Abfragen erforderlich, die Daten aus Tabellen abfragen. SELECT-Anweisungen ohne FROM-Klauseln haben eine andere Bedeutung und werden in diesem Abschnitt nicht erläutert.

Obwohl sich die meisten Abfragen auf Tabellen beziehen, können Abfragen auch Daten aus anderen Objekten abrufen, wenn diese aus Spalten und Zeilen bestehen, z.B. Ansichten, andere Abfragen (abgeleitete Tabellen) und Ergebnismengen aus gespeicherten Prozeduren.

- Die WHERE-Klausel legt die Zeilen in der Tabelle fest, die Sie sehen wollen.
- Mit der GROUP BY-Klausel können Sie Daten aggregieren.
- Die HAVING-Klausel legt die Zeilen fest, in denen Aggregatdaten abgefragt werden sollen.
- Die ORDER BY-Klausel sortiert die Zeilen in der Ergebnismenge. (Standardmäßig werden Zeilen aus einer relationalen Datenbank in unbestimmter Reihenfolge zurückgegeben.)

Die meisten Klauseln sind optional, aber wenn sie verwendet werden, müssen sie in der richtigen Reihenfolge erscheinen.

Siehe auch

- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Abfrageergebnisse zusammenfassen, gruppieren und sortieren“ auf Seite 470
- „FROM-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Abfrageprädikate

Ein **Prädikat** ist ein bedingter Ausdruck, aus dem, zusammen mit den logischen Operatoren AND und OR, die Reihe von Bedingungen in einer WHERE-, HAVING- oder ON-Klausel besteht. In SQL wird ein Prädikat, das als UNKNOWN aufgelöst wird, als FALSE interpretiert.

Ein Prädikat, das einen Index verwenden kann, um Zeilen aus einer Tabelle abzurufen, wird **sargable** (als Suchargument nutzbar) genannt. Diese Bezeichnung ist abgeleitet aus *search argument-able* (suchargumentfähig). Prädikate, die Vergleiche einer Spalte mit Konstanten, anderen Spalten oder Ausdrücken beinhalten, können "sargable" sein.

Das Prädikat in der folgenden Anweisung ist "sargable". SQL Anywhere kann es unter Verwendung des primären Indexes der Employees-Tabelle effizient auswerten.

```
SELECT *  
FROM Employees  
WHERE Employees.EmployeeID = 102;
```

Im besten Zugriffsplan erscheint dies folgendermaßen: Employees<Employees>.

Im Gegensatz dazu ist das folgende Prädikat nicht sargable. Auch wenn die EmployeeID-Spalte im primären Index indiziert ist, beschleunigt die Verwendung dieses Indexes die Berechnung nicht, weil das Ergebnis alle Zeilen, oder alle bis auf eine Zeile, enthält.

```
SELECT *  
FROM Employees  
where Employees.EmployeeID <> 102;
```

Im besten Zugriffsplan erscheint dies folgendermaßen: Employees<seq>.

Gleichmaßen kann kein Index eine Suche unterstützen, bei der alle Mitarbeiter gesucht werden, deren Vorname mit k endet. Auch hier besteht die einzige Möglichkeit zum Berechnen dieses Ergebnisses darin, jede Zeile einzeln zu prüfen.

Funktionen

Im Allgemeinen ist ein Prädikat, das eine Funktion für den Spaltennamen hat, nicht "sargable". Beispielsweise würde für die folgende Abfrage kein Index verwendet werden:

```
SELECT *
FROM SalesOrders
WHERE YEAR ( OrderDate ) = '2000';
```

Um die Verwendung einer Funktion zu vermeiden, können Sie eine Abfrage neu schreiben, um sie "sargable" zu machen. Die Abfrage oben können Sie z.B. wie folgt neu schreiben:

```
SELECT *
FROM SalesOrders
WHERE OrderDate > '1999-12-31'
AND OrderDate < '2001-01-01';
```

Eine Abfrage, die eine Funktion verwendet, wird sargable, wenn Sie die Funktionswerte in einer berechneten Spalte speichern und einen Index für diese Spalte erstellen. Eine **berechnete Spalte** ist eine Spalte, deren Werte aus anderen Spalten stammen. Wenn beispielsweise in einer Spalte namens OrderDate das Datum einer Bestellung enthalten ist, können Sie eine berechnete Spalte mit dem Namen OrderYear erstellen, in der die Werte für das Jahr enthalten sind, die aus der Spalte OrderDate extrahiert wurden.

```
ALTER TABLE SalesOrders
ADD OrderYear INTEGER
COMPUTE ( YEAR( OrderDate ) );
```

Sie können dann ganz normal einen Index für die Spalte OrderYear hinzufügen:

```
CREATE INDEX IDX_year
ON SalesOrders ( OrderYear );
```

Wenn Sie dann die folgende Anweisung ausführen, erkennt der Datenbankserver, dass eine Spalte mit Index vorhanden ist, die diese Informationen enthält, und verwendet diesen Index, um auf die Abfrage zu antworten.

```
SELECT * FROM SalesOrders
WHERE YEAR( OrderDate ) = '2000';
```

Die Domäne der berechneten Spalte muss der Domäne des COMPUTE-Ausdrucks entsprechen, damit die Spaltenersetzung vorgenommen werden kann. Wenn im Beispiel oben für YEAR(OrderDate) eine Zeichenfolge anstelle einer Ganzzahl zurückgegeben worden wäre, hätte der Optimierer den Ausdruck nicht durch die berechnete Spalte ersetzt, und der Index "IDX_year" hätte nicht zum Abrufen der erforderlichen Zeilen verwendet werden können.

Beispiele

In jedem dieser Beispiele handelt es sich bei den Attributen x und y um Spalten einer einzigen Tabelle. Das Attribut z ist in einer separaten Tabelle enthalten. Für jedes dieser Attribute gibt es einen Index.

sargable	Nicht sargable
$x = 10$	$x < > 10$
$x \text{ IS NULL}$	
$x \text{ IS NOT NULL}$	
$x > 25$	$x = 4 \text{ OR } y = 5$
$x = z$	$x = y$
$x \text{ IN } (4, 5, 6)$	$x \text{ NOT IN } (4, 5, 6)$
$x \text{ LIKE 'pat\%'}$	$x \text{ LIKE '\%tern'}$
$x = 20 - 2$	$x + 2 = 20$
$X \text{ IS NOT DISTINCT FROM } Y+1$	
$X \text{ IS DISTINCT FROM } Y+1$	

Manchmal ist es nicht offensichtlich, ob ein Prädikat sargable ist. In diesen Fällen könnten Sie das Prädikat neu schreiben, damit es sargable wird. Zum Beispiel könnten Sie das Prädikat $x \text{ LIKE 'pat\%'}$ neu schreiben, unter Berücksichtigung der Tatsache, dass "u" im Alphabet der nächste Buchstabe nach "t" ist: $x \geq \text{'pat'}$ und $x < \text{'pau'}$. In dieser Form hilft ein Index auf dem Attribut x , die Werte im eingeschränkten Bereich aufzufinden. SQL Anywhere führt diese spezielle Transformation automatisch durch.

Ein sargable-Prädikat, das für einen indizierten Abruf in einer Tabelle verwendet wird, nennt man **übereinstimmendes** Prädikat. Eine WHERE-Klausel kann viele übereinstimmende Prädikate haben. Welches das geeignetste Prädikat ist, hängt vom Zugriffsplan ab. Der Optimierer bewertet seine Auswahl von übereinstimmenden Prädikaten neu, wenn er verschiedene Zugriffspläne bewertet.

Siehe auch

- „Berechnete Spalten“ auf Seite 13

SQL-Abfragen

In der gesamten Dokumentation werden SELECT-Anweisungen und andere SQL-Anweisungen mit jeder Klausel in einer eigenen Zeile und mit SQL-Schlüsselwörtern in Großschreibung angezeigt. Dies dient dazu, die Anweisungen leichter lesbar zu machen, es ist aber keine Formatvoraussetzung. Sie können SQL-Schlüsselwörter in beliebiger Groß- und Kleinschreibung eingeben und überall in der Anweisung Zeilenumbrüche einfügen.

Schlüsselwörter und Zeilenumbrüche

Die folgende SELECT-Anweisung holt beispielsweise die Vornamen und Nachnamen von Kontaktpersonen in Kalifornien aus der Tabelle "Contacts".


```
SELECT GivenName, Surname
FROM Contacts
WHERE State = 'CA';
```

Es ist genauso gültig, wenn auch nicht so einfach lesbar, die Anweisung wie folgt einzugeben:

```
SELECT GivenName,
Surname from Contacts
WHERE State
= 'CA';
```

Berücksichtigung von Groß- und Kleinschreibung bei Zeichenfolgen und Bezeichnern

Bei Bezeichnern, wie z.B. Tabellennamen, Spaltennamen etc., wird in SQL Anywhere-Datenbanken zwischen Groß- und Kleinschreibung kein Unterschied gemacht.

Bei Zeichenfolgen wird standardmäßig die Groß- und Kleinschreibung nicht berücksichtigt, sodass 'CA', 'ca', 'cA' und 'Ca' gleichwertig sind. Wenn Sie jedoch eine Datenbank so einrichten, dass die Groß- und Kleinschreibung berücksichtigt wird, ist die Schreibweise relevant. Die SQL Anywhere-Beispieldatenbank berücksichtigt die Groß- und Kleinschreibung nicht.

Qualifizierende Bezeichner

Sie können die Namen von Datenbankbezeichnern qualifizieren, wenn nicht eindeutig feststeht, auf welches Objekt Bezug genommen wird. Die SQL Anywhere-Beispieldatenbank enthält z.B. einige Tabellen mit einer Spalte namens "City", sodass Sie Bezugnahmen auf "City" mit dem Namen der Tabelle qualifizieren müssen. In einer größeren Datenbank müssen Sie eventuell auch den Namen des Eigentümers der Tabelle verwenden, um die Tabelle zu identifizieren.

```
SELECT Contacts.City
FROM Contacts
WHERE State = 'CA';
```

Da die Beispiele in diesem Abschnitt nur Abfragen in einer Tabelle betreffen, werden Spaltennamen in Syntaxmodellen und Beispielen normalerweise nicht mit den Namen der Tabellen oder Eigentümer qualifiziert, zu denen sie gehören.

Diese Elemente werden zur besseren Lesbarkeit ausgelassen. Es ist aber nie falsch, wenn Qualifizierer eingeschlossen werden.

Reihenfolge der Zeilen in der Ergebnismenge

Die Reihenfolge der Zeilen in der Ergebnismenge ist ohne Bedeutung. Die Datenbank gibt Zeilen in beliebiger Reihenfolge zurück. Wenn Sie Zeilen in einer bestimmten Reihenfolge abrufen, müssen Sie die Reihenfolge in der Abfrage festlegen.

Siehe auch

- „Datenbankerstellung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Dienstprogramm Initialisierung (dbinit)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Groß-/Kleinschreibung“ auf Seite 660

Die SELECT-Liste: Angeben von Spalten

Die SELECT-Liste besteht aus einem oder mehreren Objekten, aus denen Daten abgefragt werden. Die SELECT-Liste besteht in der Regel aus einer Reihe von durch Kommata getrennten Spaltennamen oder aus einem Sternchen-Operator, der für alle Spalten steht. Allgemeiner gesagt, kann die SELECT-Liste einen einzelnen oder mehrere durch Kommata getrennte Ausdrücke enthalten. Nach der letzten Spalte in der Liste oder wenn es nur eine Spalte in der Liste gibt, steht kein Komma.

Die allgemeine Syntax für die SELECT-Liste sieht wie folgt aus:

```
SELECT expression [, expression ]...
```

Wenn eine Tabelle oder ein Spaltenname in der Liste den Regeln für gültige Bezeichner nicht entspricht, müssen Sie den Bezeichner in Anführungszeichen setzen.

Die Ausdrücke in der SELECT-Liste können * (alle Spalten), eine Liste von Spaltennamen, Zeichenfolgen, Spaltentitel und Ausdrücke mit arithmetischen Operatoren enthalten. Außerdem können Sie Aggregatfunktionen einbeziehen.

Siehe auch

- „Abfrageergebnisse zusammenfassen, gruppieren und sortieren“ auf Seite 470
- „Ausdrücke“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Auswahl aller Spalten in einer Tabelle

Das Sternchen (*) hat in SELECT-Anweisungen eine spezielle Bedeutung. Es steht für alle Spaltennamen in allen Tabellen, die in der FROM-Klausel bezeichnet werden. Sie können es verwenden, um sich die Zeit für die Eingabe zu sparen und Tippfehler zu vermeiden, wenn ohnedies alle Spalten in einer Tabelle angezeigt werden sollen.

Wenn Sie SELECT * verwenden, werden die Spalten in der Reihenfolge zurückgegeben, in der sie beim Erstellen der Tabelle definiert wurden.

Die Syntax für die Auswahl aller Spalten in einer Tabelle lautet wie folgt:

```
SELECT *  
FROM table-expression;
```

SELECT * findet alle in einer Tabelle befindlichen Spalten, sodass Änderungen in der Struktur einer Tabelle wie Hinzufügen, Entfernen oder Umbenennen von Spalten automatisch die Ergebnisse von SELECT * ändern. Wenn Sie die Spalten einzeln auflisten, haben Sie eine bessere Kontrolle über die Ergebnisse.

Beispiel

Mit der folgenden Anweisung werden alle Spalten in der Tabelle "Departments" ausgelesen. Eine WHERE-Klausel ist nicht vorhanden, daher werden alle Zeilen in der Tabelle abgerufen:

```
SELECT *  
FROM Departments;
```

Die Ergebnisse sehen wie folgt aus:

DepartmentID	DepartmentName	DepartmentHeadID
100	R & D	501
200	Sales	902
300	Finance	1293
400	Marketing	1576
...

Sie erhalten genau dieselben Ergebnisse, wenn Sie alle Spaltennamen in der Tabelle in der Reihenfolge nach dem SELECT-Schlüsselwort eingeben:

```
SELECT DepartmentID, DepartmentName, DepartmentHeadID
FROM Departments;
```

Wie ein Spaltenname kann auch "*" mit einem Tabellennamen qualifiziert werden, wie dies in der folgenden Abfrage der Fall ist:

```
SELECT Departments.*
FROM Departments;
```

Auswahl bestimmter Spalten aus einer Tabelle

Sie können die Anzahl der von einer SELECT-Anweisung zurückgegebenen Spalten begrenzen, indem Sie die Spalten gleich hinter dem Schlüsselwort SELECT aufführen. Diese SELECT-Anweisung hat die folgende Syntax:

```
SELECT column-name [, column-name ]...
FROM table-name
```

In der Syntax müssen *column-name* und *table-name* durch die Namen der Spalten und der Tabelle ersetzt werden, die Sie abfragen möchten.

Beispiel:

```
SELECT Surname, GivenName
FROM Employees;
```

Projektionen und Einschränkungen

Bei einer **Projektion** handelt es sich um eine Teilmenge der Spalten einer Tabelle. Eine **Einschränkung** (auch **Auswahl** genannt) ist eine Teilmenge der Zeilen einer Tabelle, basierend auf bestimmten Bedingungen.

Beispiel: Die folgende SELECT-Anweisung ruft die Namen und Preise aller Artikel in der SQL Anywhere-Beispieldatenbank ab, die über 15 US-Dollar kosten:

```
SELECT Name, UnitPrice
FROM Products
WHERE UnitPrice > 15;
```

Diese Abfrage benutzt sowohl eine Projektion (SELECT Name, UnitPrice) als auch eine Einschränkung (WHERE UnitPrice > 15).

Reihenfolge der Spalten neu anordnen

Die Reihenfolge, in der Sie Spaltennamen angeben, bestimmt die Reihenfolge, in der die Spalten angezeigt werden. In den beiden folgenden Beispielen wird gezeigt, wie eine Spaltenreihenfolge für eine Anzeige festgelegt wird. In beiden Fällen werden die Abteilungsnamen und die ID-Nummern für alle fünf Zeilen in der Tabelle "Departments" angezeigt, allerdings in unterschiedlicher Reihenfolge.

```
SELECT DepartmentID, DepartmentName
FROM Departments;
```

DepartmentID	DepartmentName
100	R & D
200	Sales
300	Finance
400	Marketing
...	...

```
SELECT DepartmentName, DepartmentID
FROM Departments;
```

DepartmentName	DepartmentID
R & D	100
Sales	200
Finance	300
Marketing	400
...	...

Joins

Ein Join verknüpft die Zeilen in zwei oder mehreren Tabellen, indem die Werte in den Spalten jeder Tabelle verglichen werden. Beispiel: Sie wollen die ID-Nummern der Bestellungen und die Artikelbezeichnungen für alle Bestellungen abfragen, bei denen mehr als zwölf Artikel geliefert wurden:

```
SELECT SalesOrderItems.ID, Products.Name
FROM Products JOIN SalesOrderItems
WHERE SalesOrderItems.Quantity > 12;
```

Die Tabellen "Products" und "SalesOrderItems" sind, basierend auf ihrer Fremdschlüsselbeziehung, durch einen Join miteinander verknüpft.

Siehe auch

- „Joins: Daten aus mehreren Tabellen abrufen“ auf Seite 492

Umbenannte Spalten in Abfrageergebnissen

Standardmäßig ist der Titel der einzelnen Spalten einer Ergebnismenge der Name des Ausdrucks, der in der SELECT-Liste bereitgestellt wurde. Bei Ausdrücken, die Spaltenwerte sind, ist der Titel der Spaltenname. In Embedded SQL kann die DESCRIBE-Anweisung verwendet werden, um den Namen der einzelnen von einem Cursor zurückgegeben Ausdrücke zu bestimmen. Andere Anwendungsschnittstellen unterstützen auch die Abfrage der Namen der einzelnen Ergebnismengenspalten durch schnittstellenspezifische Verfahren. Die sa_describe_query-Systemprozedur bietet eine schnittstellenunabhängige Möglichkeit, die Namen der Spalten in der Ergebnismenge für eine beliebige SQL-Abfrage festzustellen.

Sie können den Namen jedes Ausdrucks in der SELECT-Liste einer Abfrage mit einem **Alias** wie folgt überschreiben:

SELECT *column-name* [**AS**] *alias*

Durch einen Alias können Sie die Ergebnisse lesbarer gestalten. Sie können z.B. in einer Abteilungsliste den Spaltentitel "DepartmentName" auf den Alias "Department" setzen:

```
SELECT DepartmentName AS Department,  
       DepartmentID AS "Identifying Number"  
FROM Departments;
```

Department	Identifying Number
R & D	100
Sales	200
Finance	300
Marketing	400
...	...

Verwendung

Hinweis

Die folgenden Zeichen sind in Aliasen nicht zulässig:

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

- **Leerzeichen und Schlüsselwörter in einem Alias verwenden** Im obigen Beispiel ist der Alias "Identifying Number" für "DepartmentID" in Anführungszeichen gesetzt, weil er ein Leerzeichen enthält. Sie können auch Anführungszeichen verwenden, wenn Sie Schlüsselwörter oder Sonderzeichen in Aliasnamen benutzen. Die folgende Anweisung ist beispielsweise ohne Apostrophe ungültig:

```
SELECT DepartmentName AS Department,  
       DepartmentID AS "integer"  
FROM Departments;
```

- **Namespace-Okklusion** Aliasnamen können an einer beliebigen Stelle im SELECT-Block verwendet werden, in dem sie definiert sind, auch in anderen SELECT-Listenausdrücken, die wiederum zusätzliche Aliase definieren. Zyklische Aliasreferenzen sind nicht zulässig. Wenn der für einen Ausdruck angegebene Alias mit dem Namen einer Spalte oder Variablen im Namespace des SELECT-Blocks identisch ist, schließt die Aliasdefinition die Spalte oder Variable aus. Beispiel:

```
SELECT DepartmentID AS DepartmentName  
FROM Departments  
WHERE DepartmentName = 'Marketing'
```

Diese Anweisung gibt den Fehler "Umwandeln von 'Marketing' auf numeric nicht möglich" zurück. Dies ist deshalb der Fall, weil das Gleichheitsprädikat in der WHERE-Klausel der Abfrage versucht, den Zeichenfolgenliteral "Marketing" mit der Ganzzahlspalte "DepartmentID" zu vergleichen, und die Datentypen nicht kompatibel sind.

Hinweis

Bei der Referenzierung von Spaltennamen können Sie die Spaltennamen explizit anhand ihrer Tabellennamen qualifizieren, z.B. "Departments.DepartmentID", um einen Namenskonflikt mit einem Alias zu vermeiden.

- **Transact-SQL-Kompatibilität** Adaptive Server Enterprise unterstützt sowohl das SQL/2008-Schlüsselwort *AS* als auch die Verwendung von Gleichheitszeichen zur Identifikation eines Alias für ein SELECT-Listenelement.

Siehe auch

- „Mit Transact-SQL kompatible Abfragen“ auf Seite 665
- „sa_describe_query-Systemprozedur“ [SQL Anywhere Server - SQL-Referenzhandbuch]
- „DESCRIBE-Anweisung [ESQL]“ [SQL Anywhere Server - SQL-Referenzhandbuch]

Zeichenfolgen in Abfrageergebnissen

Die meisten SELECT-Anweisungen erzeugen Ergebnisse, die ausschließlich aus Daten von Tabellen in der FROM-Klausel bestehen. Allerdings können Zeichenfolgen auch in Abfrageergebnissen angezeigt werden, indem sie in Apostrophe gesetzt und durch Kommata von anderen Elementen in der SELECT-Liste getrennt werden. Um einen Apostroph in eine Zeichenfolge einzuschließen, verdoppeln Sie ihn einfach. Beispiel:

```
SELECT 'The department's name is' AS "Prefix",
       DepartmentName AS Department
FROM Departments;
```

Prefix	Department
The department's name is	R & D
The department's name is	Sales
The department's name is	Finance
The department's name is	Marketing
The department's name is	Shipping

Berechnete Werte in der SELECT-Liste

Die Ausdrücke in der SELECT-Liste können komplizierter sein als nur Spaltennamen oder Zeichenfolgen. Sie können in einer SELECT-Liste beispielsweise Berechnungen mit Daten aus numerischen Spalten vornehmen.

Arithmetische Vorgänge

Um die numerischen Vorgänge zu veranschaulichen, die Sie in der SELECT-Liste ausführen können, beginnen Sie mit einer Liste der Namen, der Lagermengen und der Stückpreise von Produkten aus der SQL Anywhere-Beispieldatenbank.

```
SELECT Name, Quantity, UnitPrice
FROM Products;
```

Name	Quantity	UnitPrice
Tee Shirt	28	9

Name	Quantity	UnitPrice
Tee Shirt	54	14
Tee Shirt	75	14
Baseball Cap	112	9
...

Nehmen wir einmal an, die Regel für den Warenumschlag lautet, dass das Lager aufgefüllt wird, wenn nur noch weniger als 10 Stück vorhanden sind. In der folgenden Abfrage wird die Anzahl der Artikel angezeigt, die verkauft werden müssen, bevor eine Nachbestellung erfolgt ("Sell before reorder"):

```
SELECT Name, Quantity - 10
      AS "Sell before reorder"
FROM Products;
```

Name	Sell before reorder
Tee Shirt	18
Tee Shirt	44
Tee Shirt	65
Baseball Cap	102
...	...

Sie können die Werte auch in Spalten zusammenfassen. In der folgenden Abfrage wird der Gesamtwert der einzelnen Artikel aus dem Lager aufgelistet ("Inventory value"):

```
SELECT Name, Quantity * UnitPrice AS "Inventory value"
FROM Products;
```

Name	Inventory value
Tee Shirt	252.00
Tee Shirt	756.00
Tee Shirt	1050.00
Baseball Cap	1008.00
...	...

Wenn mehr als ein arithmetischer Operator in einem Ausdruck vorhanden ist, werden Multiplikation, Division und Modulo zuerst berechnet, dann Subtraktion und Addition. Wenn alle arithmetischen

Operatoren in einem Ausdruck denselben Vorrang haben, erfolgt die Durchführung von links nach rechts. Ausdrücke in Klammern haben Vorrang vor allen anderen Vorgängen.

Beispiel: Die folgende SELECT-Anweisung berechnet den Gesamtwert der einzelnen Artikel auf Lager und subtrahiert dann fünf Dollar von diesem Wert.

```
SELECT Name, Quantity * UnitPrice - 5
FROM Products;
```

Um korrekte Ergebnisse sicherzustellen, verwenden Sie soweit möglich Klammern. Die folgende Abfrage hat dieselbe Bedeutung und liefert dieselben Ergebnisse wie die vorherige Abfrage, ist aber präziser.

```
SELECT Name, ( Quantity * UnitPrice ) - 5
FROM Products;
```

Arithmetische Vorgänge können einen Überlauf verursachen, wenn sich ihre Ergebnisse nicht im Datentyp darstellen lassen. Wenn ein Überlauf eintritt, wird anstelle eines Werts ein Fehler zurückgegeben.

Zeichenfolgenverarbeitung

Sie können Zeichenfolgen verketteten, indem Sie einen entsprechenden Operator benutzen. Verwenden Sie dafür || (definiert von SQL/2008) oder + (unterstützt von Adaptive Server Enterprise) als Operator für die Verkettung von Zeichenfolgen. Beispiel: Die folgende Anweisung ruft "GivenName"- und "Surname"-Werte ab und verkettet sie in den Ergebnissen:

```
SELECT EmployeeID, GivenName || ' ' || Surname AS Name
FROM Employees;
```

EmployeeID	Name
102	Fran Whitney
105	Matthew Cobb
129	Philip Chin
148	Julie Jordan
...	...

Datums- und Zeitvorgänge

Obwohl Sie Operatoren für Spalten mit Datums- und Zeitangaben verwenden können, sollten dafür normalerweise Funktionen herangezogen werden.

Weitere Hinweise zu berechneten Spalten

- **Spalten kann ein Alias gegeben werden** Standardmäßig ist der Spaltenname der in der SELECT-Liste aufgeführte Ausdruck. Für berechnete Spalten ist der Ausdruck jedoch etwas umständlich und nicht sehr informativ.
- **Andere Operatoren können verwendet werden** Der Multiplikationsoperator kann zum Kombinieren von Spalten verwendet werden. Sie können auch andere Operatoren verwenden,

einschließlich der normalen arithmetischen Operatoren sowie der logischen und Zeichenfolgenoperatoren.

Die folgende Abfrage listet z.B. den vollständigen Namen aller Kunden auf:

```
SELECT ID, (GivenName || ' ' || Surname ) AS "Full name"
FROM Customers;
```

Der Operator `||` verkettet Zeichenfolgen. In dieser Abfrage enthält der Alias für die Spalte Leerstellen und muss daher in Anführungszeichen eingeschlossen werden. Diese Regel gilt nicht nur für Spaltenalias, sondern auch für Tabellennamen und andere Bezeichner in der Datenbank.

- **Funktionen können verwendet werden** Zusätzlich zu den Optionen für die Kombination von Spalten können Sie eine Vielzahl integrierter Funktionen verwenden, um die gewünschten Ergebnisse zu erhalten.

Die folgende Abfrage listet z.B. die Produktnamen in Großbuchstaben auf:

```
SELECT ID, UCASE( Name )
FROM Products;
```

ID	UCASE(Products.name)
300	TEE SHIRT
301	TEE SHIRT
302	TEE SHIRT
400	BASEBALL CAP
...	...

Siehe auch

- „Vorrang der Operatoren“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SQL-Funktionen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Operatoren“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Umbenannte Spalten in Abfrageergebnissen“ auf Seite 299

Eliminierung von doppelten Abfrageergebnissen

Mit dem optionalen `DISTINCT`-Schlüsselwort werden doppelte Zeilen aus den Ergebnissen einer `SELECT`-Anweisung eliminiert. Wenn Sie `DISTINCT` nicht angeben, werden alle Zeilen ausgegeben, auch Duplikatzellen. Optional können Sie `ALL` vor die `SELECT`-Liste setzen, um alle Zeilen abzurufen. Aus Gründen der Kompatibilität mit anderen SQL-Implementierungen ermöglicht die Syntax von SQL Anywhere die Verwendung von `ALL`, um ausdrücklich alle Zeilen abzufragen. `ALL` ist der Standardwert.

Wenn Sie beispielsweise alle Städte der Tabelle "Contacts" ausgeben wollen, ohne `DISTINCT` zu verwenden, erhalten Sie 60 Zeilen:

```
SELECT City  
FROM Contacts;
```

Sie können die Doppelzeilen durch DISTINCT eliminieren. Die folgende Abfrage gibt nur 16 Zeilen zurück:

```
SELECT DISTINCT City  
FROM Contacts;
```

NULL ist nicht "distinct"-fähig

Das Schlüsselwort DISTINCT behandelt NULL als Duplikat. Wenn also DISTINCT in einer SELECT-Anweisung verwendet wird, enthält das Ergebnis nur ein Mal NULL, gleichgültig wie viele Zeilen mit NULL gefunden werden.

Die FROM-Klausel: Tabellen angeben

Die FROM-Klausel ist für jede SELECT-Anweisung erforderlich, die Daten aus Tabellen, Ansichten oder gespeicherten Prozeduren abrufen.

Die FROM-Klausel kann JOIN-Bedingungen enthalten, die Tabellen verbindet sowie Joins zu anderen Abfragen (abgeleiteten Tabellen) herstellen.

Tabellennamen qualifizieren

In der FROM-Klausel ist die volle Namensgebungssyntax für Tabellen und Ansichten immer zulässig, d.h.:

```
SELECT select-list  
FROM owner.table-name;
```

Das Qualifizieren von Tabellen-, Ansichten- und Prozedurnamen ist nur erforderlich, wenn das Objekt einer Benutzer-ID gehört, die sich von der Benutzer-ID der aktuellen Verbindung unterscheidet oder wenn die Benutzer-ID des Eigentümers nicht der Name einer Rolle ist, zu der die Benutzer-ID der aktuellen Verbindung gehört.

Korrelationsnamen verwenden

Sie können einem Tabellennamen einen Korrelationsnamen geben, um die Lesbarkeit zu verbessern und sich das Eingeben des vollständigen Tabellennamens an jeder Stelle, wo die Tabelle referenziert wird, in Zukunft zu ersparen. Sie weisen den Korrelationsnamen in der FROM-Klausel zu, indem Sie ihn nach dem Tabellennamen wie folgt eingeben:

```
SELECT d.DepartmentID, d.DepartmentName  
FROM Departments d;
```

Wenn ein Korrelationsname verwendet wird, *müssen* alle anderen Verweise auf die Tabelle, wie etwa in einer WHERE-Klausel, den Korrelationsnamen statt des Tabellennamens benutzen. Korrelationsnamen müssen den Regeln für gültige Bezeichner entsprechen.

Abgeleitete Tabellen abfragen

Eine abgeleitete Tabelle ist eine Tabelle, die direkt oder indirekt von einer oder mehreren Tabellen durch die Berechnung eines Abfrageausdrucks abgeleitet wird. Abgeleitete Tabellen werden in der FROM-Klausel einer SELECT-Anweisung festgelegt.

Das Abfragen einer abgeleiteten Tabelle funktioniert auf dieselbe Weise wie das Abfragen einer Ansicht. Das heißt, dass die Werte einer abgeleiteten Tabelle zu dem Zeitpunkt bestimmt werden, an dem die Definition der abgeleiteten Tabelle evaluiert wird. Abgeleitete Tabellen unterscheiden sich dadurch von Ansichten, dass die Definition einer abgeleiteten Tabelle nicht in der Datenbank gespeichert wird.

Abgeleitete Tabellen unterscheiden sich von Basis- und temporären Tabellen, indem sie nicht materialisiert sind und nicht von außerhalb der Abfrage, in der sie definiert sind, referenziert werden können.

Die folgende Abfrage verwendet eine abgeleitete Tabelle (my_derived_tbl), in der der Höchstlohn der einzelnen Abteilungen eingefügt wird. Die Daten in der abgeleiteten Tabelle werden dann mit der Tabelle "Employees" verknüpft, um die Nachnamen der Mitarbeiter zu erhalten, die diese Löhne ausgezahlt bekommen.

```
SELECT Surname,
       my_derived_table.maximum_salary AS Salary,
       my_derived_table.DepartmentID
FROM Employees e,
     ( SELECT MAX( Salary ) AS maximum_salary, DepartmentID
       FROM Employees
       GROUP BY DepartmentID ) my_derived_table
WHERE e.Salary = my_derived_table.maximum_salary
AND e.DepartmentID = my_derived_table.DepartmentID
ORDER BY Salary DESC;
```

Surname	Salary	DepartmentID
Shea	138948.00	300
Scott	96300.00	100
Kelly	87500.00	200
Evans	68940.00	400
Martinez	55500.80	500

Das folgende Beispiel erstellt eine abgeleitete Tabelle (MyDerivedTable), die eine Rangfolge der Artikel in der Tabelle "Products" erstellt, und fragt dann die abgeleitete Tabelle ab, um die drei am wenigsten teuren Artikel zurückzugeben.

```
SELECT TOP 3 *
FROM ( SELECT Description,
              Quantity,
              UnitPrice,
              RANK() OVER ( ORDER BY UnitPrice ASC )
              AS Rank
       FROM Products ) AS MyDerivedTable
ORDER BY Rank;
```

Andere Objekte als Tabellen abfragen

Die häufigsten Elemente in einer FROM-Klausel sind Tabellennamen. Es ist allerdings auch möglich, Zeilen aus anderen Datenbankobjekten abzufragen, wenn diese eine tabellenähnliche Struktur haben - also eine definierte Menge von Zeilen und Spalten. Sie können beispielsweise Ansichten oder auch gespeicherte Prozeduren abfragen, die Ergebnismengen zurückgeben.

Beispiel: Die folgende Anweisung fragt die Ergebnismenge einer gespeicherten Prozedur namens "ShowCustomerProducts" ab.

```
SELECT *
FROM ShowCustomerProducts( 149 );
```

Siehe auch

- „FROM-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Joins: Daten aus mehreren Tabellen abrufen“ auf Seite 492

SELECT über eine DML-Anweisung

SQL Anywhere unterstützt die Verwendung einer DML-Anweisung (INSERT, UPDATE, DELETE oder MERGE) als Tabellenausdruck in der FROM-Klausel einer Abfrage.

Wenn Sie einen *dml-derived-table* in einer Anweisung verwenden, wird dieser während des DESCRIBE ignoriert. Zum Zeitpunkt von OPEN wird zunächst die UPDATE-Anweisung ausgeführt und die Ergebnisse werden in einer temporären Tabelle gespeichert. Die temporäre Tabelle verwendet die Spaltennamen der Tabelle, die von der Anweisung geändert wird. Sie können auf die geänderten Werte mit dem Korrelationsnamen aus der REFERENCING-Klausel verweisen. Bei der Angabe von OLD oder FINAL brauchen Sie keine eindeutigen Spaltennamen für die aktualisierte Tabelle, die in der Abfrage referenziert wird. Die *dml-derived-table*-Anweisung kann nur eine einzelne aktualisierbare Tabelle referenzieren. Aktualisierungen mehrerer Tabellen geben einen Fehler zurück.

Die folgende Abfrage verwendet z.B. eine SELECT-Anweisung über eine UPDATE-Anweisung zum Ausführen der unten aufgelisteten Vorgänge:

- Aktualisiert alle Produkte in der Beispieldatenbank mit einer Preiserhöhung von 7 %
- Listet die betroffenen Produkte und ihre Bestellungen auf, die zwischen dem 10. April 2000 und dem 21. Mai 2000 versandt wurden und deren Bestellmenge größer als 36 war

```
SELECT old_products.ID, old_products.name, old_products.UnitPrice AS
OldPrice,
      final_products.UnitPrice AS NewPrice, SOI.ID AS OrderID, SOI.Quantity
FROM
( UPDATE Products SET UnitPrice = UnitPrice * 1.07 )
  REFERENCING ( OLD AS old_products FINAL AS final_products )
  JOIN SalesOrderItems AS SOI ON SOI.ProductID = old_products.ID
WHERE SOI.ShipDate BETWEEN '2000-04-10' AND '2000-05-21'
      AND SOI.QUANTITY > 36
ORDER BY old_products.ID;
```

Die folgende Abfrage verwendet eine MERGE-Anweisung und eine UPDATE-Anweisung. Die Tabelle "modified_employees" repräsentiert eine Sammlung von Mitarbeitern, deren Status geändert wurde, während die MERGE-Anweisung die Bezeichner und Namen der Mitarbeiter, deren Gehalt um 3 %

erhöht wurde, mit Mitarbeitern zusammenführt, die in der Tabelle "modified_employees" enthalten sind. In dieser Abfrage beziehen sich die Optionseinstellungen, die in der OPTION-Klausel angegeben sind, sowohl auf die UPDATE- als auch auf die MERGE-Anweisung.

```
CREATE TABLE modified_employees
( EmployeeID INTEGER PRIMARY KEY, Surname VARCHAR(40), GivenName
  VARCHAR(40) );

MERGE INTO modified_employees AS me
USING (SELECT modified_employees.EmployeeID,
              modified_employees.Surname,
              modified_employees.GivenName
      FROM (
        UPDATE Employees
        SET Salary = Salary * 1.03
        WHERE ManagerID = 501)
        REFERENCING (FINAL AS modified_employees) ) AS dt_e
ON dt_e.EmployeeID = me.EmployeeID
WHEN MATCHED THEN SKIP
WHEN NOT MATCHED THEN INSERT
OPTION( optimization_level=1, isolation_level=2 );
```

Mehrere Tabellen in einer Abfrage verwenden

Bei der Verwendung von mehreren *dml-derived-table*-Anweisungen in einer Abfrage ist die Ausführungsreihenfolge der UPDATE-Anweisung nicht garantiert. Die folgende Anweisung aktualisiert die Tabellen "Products" und "SalesOrderItems" in der Beispieldatenbank und erzeugt dann ein Ergebnis basierend auf einem Join, der folgende Manipulationen enthält:

```
SELECT old_products.ID, old_products.name, old_products.UnitPrice AS
OldPrice,
       final_products.UnitPrice AS NewPrice,
       SalesOrders.ID AS OrderID, SalesOrders.CustomerID,
       old_order_items.Quantity,
       old_order_items.ShipDate AS OldShipDate,
       final_order_items.ShipDate AS RevisedShipDate
FROM
( UPDATE Products SET UnitPrice = UnitPrice * 1.07 )
  REFERENCING ( OLD AS old_products FINAL AS final_products )
JOIN
( UPDATE SalesOrderItems
  SET ShipDate = DATEADD( DAY, 6, ShipDate )
  WHERE ShipDate BETWEEN '2000-04-10' AND '2000-05-21' )
  REFERENCING ( OLD AS old_order_items FINAL AS final_order_items )
  ON (old_order_items.ProductID = old_products.ID)
JOIN SalesOrders ON ( SalesOrders.ID = old_order_items.ID )
WHERE old_order_items.Quantity > 36
ORDER BY old_products.ID;
```

Tabellen ohne Materialisierung von Ergebnissen verwenden

Sie können mit der REFERENCING (NONE)-Klausel auch eine UPDATE-Anweisung einbetten, ohne ihr Ergebnis zu materialisieren. Da das Ergebnis der UPDATE-Anweisung in diesem Fall leer ist, müssen Sie bei der Erstellung der Abfrage sicherstellen, dass sie das gewünschte Ergebnis liefert. Sie können dafür Sorge tragen, dass ein nicht leeres Ergebnis zurückgegeben wird, indem Sie den *dml-derived-table* auf die Nullwert-liefernde Seite eines Outer-Joins positionieren. Beispiel:

```
SELECT 'completed' AS finished, ( SELECT COUNT( * ) FROM Products ) AS
product_total
```

```
FROM SYS.DUMMY LEFT OUTER JOIN
  ( UPDATE Products SET UnitPrice = UnitPrice * 1.07 )
  REFERENCING ( NONE ) ON 1=1;
```

Sie können auch sicherstellen, dass ein nicht leeres Ergebnis zurückgegeben wird, indem Sie den *dml-derived-table* als Teil eines Abfrageausdrucks mit einem der angegebenen Operatoren (UNION, EXCEPT oder INTERSECT) verwenden. Zum Beispiel:

```
SELECT 'completed' AS finished, ( SELECT COUNT( * ) FROM Products ) AS
product_total
FROM SYS.DUMMY
UNION ALL
SELECT 'dummy', 1 /* This query specification returns the empty set */
FROM ( UPDATE Products SET UnitPrice = UnitPrice * 1.07 )
  REFERENCING ( NONE );
```

Siehe auch

- „FROM-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Datenmanipulationsanweisungen“ auf Seite 623

Die WHERE-Klausel: Zeilen angeben

Die WHERE-Klausel gibt in einer SELECT-Anweisung die Suchbedingungen an, die für den Abruf der Zeilen gelten. Suchbedingungen werden auch **Prädikate** genannt. Das allgemeine Format ist:

```
SELECT select-list
FROM table-list
WHERE search-condition
```

Suchbedingungen in der WHERE-Klausel sind:

- **Vergleichsoperatoren** (=, <, >, usw.) Sie können z.B. eine Liste aller Angestellten erstellen, die über 50.000 US-Dollar verdienen:

```
SELECT Surname
FROM Employees
WHERE Salary > 50000;
```

- **Bereiche** (BETWEEN und NOT BETWEEN). Sie können z.B. alle Mitarbeiter auflisten, die zwischen 40.000 und 60.000 US-Dollar verdienen:

```
SELECT Surname
FROM Employees
WHERE Salary BETWEEN 40000 AND 60000;
```

- **Listen** (IN, NOT IN). Sie können z.B. alle Kunden in Ontario, Quebec oder Manitoba auflisten:

```
SELECT CompanyName, State
FROM Customers
WHERE State IN( 'ON', 'PQ', 'MB');
```

- **Zeichenvergleich** (LIKE und NOT LIKE). Sie können z.B. alle Kunden auflisten, deren Telefonnummer mit 415 beginnt (die Telefonnummer wird in der Datenbank als Zeichenfolge gespeichert):

```
SELECT CompanyName, Phone
FROM Customers
WHERE Phone LIKE '415%';
```

- **Unbekannte Werte** (IS NULL und IS NOT NULL). Sie können z.B. alle Abteilungen mit Abteilungsleitern auflisten:

```
SELECT DepartmentName
FROM Departments
WHERE DepartmentHeadID IS NOT NULL;
```

- **Kombinationen** (AND, OR). Sie können z.B. alle Mitarbeiter mit einem Einkommen über 50.000 US-Dollar auflisten, deren Vornamen mit dem Buchstaben A beginnen.

```
SELECT GivenName, Surname
FROM Employees
WHERE Salary > 50000
AND GivenName like 'A%';
```

Siehe auch

- „Suchbedingungen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Vergleichsoperatoren in der WHERE-Klausel

Sie können Vergleichsoperatoren in der WHERE-Klausel verwenden. Die Operatoren haben dabei folgende Syntax:

WHERE *expression comparison-operator expression*

Hinweise zu Vergleichen

- **Sortierreihenfolge** Beim Vergleich von Zeichendaten bedeutet < weiter oben in der Sortierreihenfolge und > weiter unten in der Sortierreihenfolge. Die Sortierreihenfolge wird durch die bei der Erstellung der Datenbank gewählte Kollation bestimmt. Sie können die Kollation ermitteln, indem Sie das Dienstprogramm dbinfo in der Datenbank ausführen:

```
dbinfo -c "uid=DBA;pwd=sql"
```

Sie können die Kollation auch in Sybase Central ermitteln, indem Sie im Fenster **Datenbankeigenschaften** die Registerkarte "Erweiterte Informationen" öffnen.

- **Nachgestellte Leerzeichen** Wenn Sie eine Datenbank erstellen, legen Sie fest, ob nachgestellte Leerzeichen bei Vergleichen ignoriert werden.

Standardmäßig werden Datenbanken so erstellt, dass nachgestellte Leerzeichen nicht ignoriert werden. 'Dirk' ist beispielsweise nicht dasselbe wie 'Dirk '. Sie können Datenbanken erstellen, bei denen ein Auffüllen mit Zeichen erfolgt, sodass nachgestellte Leerzeichen ignoriert werden.

- **Daten vergleichen** Beim Vergleichen von Daten bedeutet < früher und > später.
- **Groß-/Kleinschreibung** Wenn Sie eine Datenbank erstellen, geben Sie an, ob Vergleiche von Zeichenfolgen unter Beachtung der Groß- und Kleinschreibung erfolgen oder nicht.

Standardmäßig werden Datenbanken so erstellt, dass die Groß- und Kleinschreibung nicht berücksichtigt wird. 'Dirk' ist beispielsweise dasselbe wie 'DIRK'. Sie können Datenbanken erstellen, die die Groß-/Kleinschreibung beachten.

Nachstehend finden Sie einige SELECT-Anweisungen mit Vergleichsoperatoren:

```
SELECT *
  FROM Products
 WHERE Quantity < 20;
SELECT E.Surname, E.GivenName
  FROM Employees E
 WHERE Surname > 'McBadden';
SELECT ID, Phone
  FROM Contacts
 WHERE State != 'CA';
```

Der NOT-Operator

Der NOT-Operator negiert einen Ausdruck. Beide nachstehenden Abfragen finden alle T-Shirts und Baseballkappen, die 10 Dollar oder weniger kosten. Beachten Sie jedoch den Unterschied in der Position zwischen dem negativen logischen Operator (NOT) und dem negativen Vergleichsoperator (!>).

```
SELECT ID, Name, Quantity
  FROM Products
 WHERE (name = 'Tee Shirt' OR name = 'BaseBall Cap')
 AND NOT UnitPrice > 10;
SELECT ID, Name, Quantity
  FROM Products
 WHERE (name = 'Tee Shirt' OR name = 'BaseBall Cap')
 AND UnitPrice !> 10;
```

Siehe auch

- „Vergleichsoperatoren“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Ausdrücke“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Bereiche in der WHERE-Klausel

Das Schlüsselwort BETWEEN gibt einen Inklusivbereich an, in dem nach dem niedrigeren Wert und dem höheren Wert sowie den davon umschlossenen Werten gesucht wird.

Sie können NOT BETWEEN verwenden, um alle Zeilen zu finden, die nicht in diesem Bereich liegen.

Beispiel

- Die folgende Abfrage listet alle Produkte mit Preisen von 10 bis 15 Dollar auf.

```
SELECT Name, UnitPrice
  FROM Products
 WHERE UnitPrice BETWEEN 10 AND 15;
```

Name	UnitPrice
Tee Shirt	14

Name	UnitPrice
Tee Shirt	14
Baseball Cap	10
Shorts	15

- Die folgende Abfrage listet alle Preise auf, die unter 10 oder über 15 Dollar liegen.

```
SELECT Name, UnitPrice
FROM Products
WHERE UnitPrice NOT BETWEEN 10 AND 15;
```

Name	UnitPrice
Tee Shirt	9
Baseball Cap	9
Visor	7
Visor	7
...	...

Siehe auch

- „BETWEEN-Suchbedingung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Listen in der WHERE-Klausel

Das IN-Schlüsselwort ermöglicht es Ihnen, Werte auszuwählen, die zu einem der Werte in einer Liste passen. Der Ausdruck kann eine Konstante oder ein Spaltenname sein, und die Liste kann eine Konstantengruppe, bzw. allgemeiner, eine Unterabfrage sein.

Wenn Sie beispielsweise ohne IN eine Liste der Namen und Bundesstaaten mit allen Kunden aus Ontario, Manitoba oder Quebec erhalten wollen, geben Sie folgende Abfrage ein:

```
SELECT CompanyName, State
FROM Customers
WHERE State = 'ON' OR State = 'MB' OR State = 'PQ';
```

Sie erhalten aber dieselben Ergebnisse, wenn Sie IN verwenden. Die Elemente nach dem Schlüsselwort IN müssen durch Kommas getrennt und in Klammern gesetzt werden. Setzen Sie Zeichen-, Datums- und Zeitwerte in Apostrophe. Beispiel:

```
SELECT CompanyName, State
FROM Customers
WHERE State IN( 'ON', 'MB', 'PQ');
```

Der wichtigste Verwendungszweck des IN-Schlüsselworts ist allerdings die verschachtelte Abfrage, die auch als Unterabfrage bezeichnet wird.

Musterübereinstimmung von Zeichenfolgen in der WHERE-Klausel

Die Musterübereinstimmung ist eine vielseitige Methode zur Identifikation von Zeichendaten. In SQL wird das Schlüsselwort LIKE für die Suche nach Mustern verwendet. Bei der Musterübereinstimmung werden mithilfe von Platzhalterzeichen verschiedene Kombinationen von Zeichen gesucht, die übereinstimmen.

Das Schlüsselwort LIKE zeigt, dass die darauf folgende Zeichenfolge ein Vergleichsmuster ist. LIKE wird mit Zeichendaten verwendet.

Die Syntax für LIKE ist:

expression [**NOT**] **LIKE** *match-expression*

Der Ausdruck, der mit dem Übereinstimmungsausdruck identisch sein soll, kann folgende Sonderzeichen enthalten:

Sonderzeichen	Bedeutung
%	Passt zu jeder Zeichenfolge von 0 oder mehr Zeichen
_	Passt zu einem beliebigen Zeichen
[Bezeichner]	<p>Der Bezeichner in eckigen Klammern kann folgende Formate haben:</p> <ul style="list-style-type: none"> • Bereich Ein Bereich hat die Form <i>rangespec1-rangespec2</i>, wobei <i>rangespec1</i> für den Beginn eines Bereichs von Zeichen, der Bindestrich für einen Bereich und <i>rangespec2</i> für das Ende eines Bereichs von Zeichen steht. • Menge Eine Menge kann jede diskrete Wertmenge in jeder Reihenfolge umfassen. Zum Beispiel:[a2bR]. <p>Der Bereich [a-f] sowie die Mengen [abcdef] und [fcbaed] geben dieselbe Wertgruppe zurück.</p>
[^Bezeichner]	Das Einschaltungszeichen (^) vor einem Bezeichner gibt an, dass diese Angabe auszuschließen ist. [^a-f] bedeutet, nicht im Bereich a-f; [^a2bR] bedeutet, nicht a, 2, b oder R.

Sie können die Übereinstimmung der Spaltendaten mit Konstanten, Variablen oder anderen Spalten testen, die die Platzhalter enthalten, die in der Tabelle dargestellt wurden. Wenn Sie Konstanten verwenden, müssen Sie die Vergleichszeichenfolgen und die Zeichenfolge in Apostrophe setzen.

Beispiele

Alle nachfolgenden Beispiele benutzen LIKE mit der Spalte "Surname" in der Tabelle "Contacts". Abfragen haben folgende Form:

```
SELECT Surname
FROM Contacts
WHERE Surname LIKE match-expression;
```

Das erste Beispiel wird eingegeben als:

```
SELECT Surname
FROM Contacts
WHERE Surname LIKE 'Mc%';
```

Vergleichsausdruck	Beschreibung	Rückgabe
'Mc%'	Sucht alle Namen, die mit den Buchstaben Mc beginnen	McEvoy
'%er'	Sucht alle Namen, die mit "er" enden.	Brier, Miller, Weaver, Rayner
'%en%'	Sucht alle Namen, welche die Buchstaben en enthalten.	Pettengill, Lencki, Cohen
'_ish'	Sucht alle vierstelligen Namen, die mit ish enden.	Fish
'Br[iy][ae]r'	Sucht nach Brier, Bryer, Briar oder Bryar.	Brier
'[M-Z]owell'	Sucht alle Namen, die mit owell enden und die mit einem Buchstaben im Bereich M bis Z beginnen.	Powell
'M[^c]%'	Suche nach allen Namen, die mit "M" beginnen und nicht "c" als zweiten Buchstaben haben.	Moore, Mulley, Miller, Masalsky

Platzhalterzeichen benötigen LIKE

Platzhalterzeichen ohne LIKE werden als **Zeichenfolgenlitterale** und nicht als Muster interpretiert: Sie repräsentieren exakt ihre eigenen Werte. Die folgende Abfrage sucht alle Telefonnummern, die nur aus den vier Zeichen "415%" bestehen. Andere Telefonnummern, die mit 415 beginnen, werden nicht gefunden.

```
SELECT Phone
FROM Contacts
WHERE Phone = '415%';
```

LIKE mit Datums- und Zeitwerten verwenden

Sie können LIKE in DATE-, TIME-, TIMESTAMP- und TIMESTAMP WITH TIME ZONE-Feldern verwenden. Allerdings funktioniert das LIKE-Prädikat nur bei Zeichendaten. Wenn Sie LIKE mit Datums- und Zeitwerten verwenden, werden die Werte implizit als CHAR oder VARCHAR festgelegt, wobei die entsprechende Optionseinstellung für DATE-, TIME-, TIMESTAMP- und TIMESTAMP WITH TIME ZONE-Datentypen zum Formatieren des Werts verwendet wird:

Datums-/Zeityp	Umwandlung (CAST) in VARCHAR unter Verwendung von
DATE	date_format

Datums-/Zeittyp	Umwandlung (CAST) in VARCHAR unter Verwendung von
TIME	time_format
TIMESTAMP	timestamp_format
TIMESTAMP WITH TIME ZONE	timestamp_with_time_zone_format

Eine Konsequenz der Verwendung von LIKE bei der Suche nach DATE-, TIME- oder TIMESTAMP-Werten besteht darin, dass das LIKE-Muster sorgfältig geschrieben werden muss, weil Datums- und Zeitwerte verschiedene Datumsteile enthalten können und möglicherweise auf unterschiedliche Weise basierend auf den oben genannten Optionseinstellungen formatiert sind.

Wenn Sie beispielsweise den Wert 9:20 und das aktuelle Datum in eine TIMESTAMP-Spalte namens arrival_time einfügen, wird die folgende Klausel als TRUE ausgewertet, wenn die Option timestamp_format die Zeitangabe mit Doppelpunkten formatiert, um Stunden und Minuten zu trennen:

```
WHERE arrival_time LIKE '%09:20%'
```

Im Gegensatz zu LIKE verwenden Suchbedingungen, die einen einfachen Vergleich zwischen einem Zeichenfolgenliteral und einem DATE-, TIME-, TIMESTAMP- oder TIMESTAMP WITH TIME ZONE-Wert enthalten, den Datums-/Zeitdatentyp als Vergleichsdomäne. In diesem Fall konvertiert SQL Anywhere zuerst den Zeichenfolgenliteral in einen TIMESTAMP-Wert und verwendet dann die erforderlichen Teile dieses Werts, um den Vergleich durchzuführen. SQL Anywhere befolgt den ISO 8601-Standard für die Konvertierung von TIME-, DATE- und TIMESTAMP-Werten und enthält Erweiterungen.

Die nachstehende Klausel beispielsweise ergibt TRUE, weil der Konstanten-Zeichenfolgenwert 9:20 in einen TIMESTAMP-Wert konvertiert wird, wobei 9:20 als Uhrzeitabschnitt und das aktuelle Datum als Datumsteil verwendet wird:

```
WHERE arrival_time = '9:20'
```

NOT LIKE verwenden

Mit NOT LIKE können Sie dieselben Platzhalterzeichen verwenden wie mit LIKE. Um alle Telefonnummern in der Tabelle "Contacts" zu finden, die nicht 415 als Vorwahl haben, können Sie eine der folgenden Abfragen verwenden:

```
SELECT Phone
FROM Contacts
WHERE Phone NOT LIKE '415%';
```

```
SELECT Phone
FROM Contacts
WHERE NOT Phone LIKE '415%';
```

Unterstriche verwenden

Ein weiteres Sonderzeichen, das zusammen mit LIKE verwendet werden kann, ist der Unterstrich . Er steht für exakt ein Zeichen. So stimmt beispielsweise das Muster BR_U% mit allen Namen überein, die mit BR beginnen und deren vierter Buchstabe ein U ist. Im Namen Braun entspricht der Unterstrich () dem Buchstaben A, und das % dem Buchstaben N.

Siehe auch

- „Zeichenfolgenlitterale“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „LIKE-Suchbedingung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Zeichenfolgen und Apostrophe

Wenn Sie Zeichen- und Datumsangaben eingeben oder suchen, müssen Sie sie, wie im nachstehenden Beispiel gezeigt, in Apostrophe setzen.

```
SELECT GivenName, Surname
FROM Contacts
WHERE GivenName = 'John';
```

Wenn die Datenbankoption "quoted_identifier" auf "Off" gesetzt ist (standardmäßig ist sie auf "On"), können Sie die Zeichen- oder Datumsangaben auch in Anführungszeichen setzen, wie im folgenden Beispiel gezeigt wird.

```
SET OPTION quoted_identifier = 'Off';
```

Die Option quoted_identifier wird für die Kompatibilität mit Adaptive Server Enterprise bereitgestellt. Standardmäßig ist bei Adaptive Server Enterprise die Option "quoted_identifier" auf "Off" gesetzt, bei SQL Anywhere hingegen auf "On".

Anführungszeichen in Zeichenfolgen

Es gibt zwei Arten, in einer Zeichenfolge Apostrophe zu verwenden. Die erste Methode besteht darin, die Apostrophe zu verdoppeln. Wenn Sie beispielsweise eine Zeicheneingabe mit einem Apostroph begonnen haben und in der Zeichenfolge nun einen Apostroph verwenden müssen, setzen Sie einen doppelten Apostrophen:

```
'I don''t understand.'
```

Mit Anführungszeichen (quoted_identifier = Off) geben Sie an:

```
"He said, ""It is not really confusing."""
```

Die zweite Methode, die nur funktioniert, wenn die Option "quoted_identifier" auf "Off" gesetzt ist, besteht darin, die eine Art von Anführungsstrich in die andere Art zu setzen. Das heißt also, dass ein Eintrag der Anführungszeichen enthält, in Apostrophe eingeschlossen wird, und umgekehrt. Hier einige Beispiele:

```
'George said, "There must be a better way."'
'Isn't there a better way?'
'George asked, "Isn't there a better way?"'
```

Siehe auch

- „quoted_identifier-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Unbekannte Werte: NULL

NULL in einer Spalte bedeutet, dass der Benutzer oder die Anwendung in dieser Spalte keine Eingabe vorgenommen haben. Das heißt, dass ein Datenwert für die Spalte nicht bekannt oder nicht verfügbar ist.

NULL bedeutet nicht dasselbe wie die Ziffer Null (numerischer Wert) oder eine Leerstelle (Zeichenwert). Vielmehr können Sie mit NULL zwischen einer absichtlichen Eingabe der Ziffer Null in numerischen Spalten oder einer Leerstelle in Zeichenspalten einerseits, bzw. NULL für numerische und Zeichenspalten andererseits unterscheiden.

NULL eingeben

NULL kann nur eingegeben werden, wenn NULL-Werte in der Spalte zulässig sind. Ob eine Spalte NULL-Werte akzeptieren kann, wird beim Erstellen der Tabelle festgelegt. Unter der Annahme, dass eine Spalte NULL-Werte aufnehmen kann, wird NULL eingefügt:

- **Standardvorgabe** Wenn keine Daten eingegeben werden und für die Spalte kein anderer Standardwert definiert wurde.
- **Explizite Eingabe** Sie können explizit den Ausdruck "NULL" ohne Apostrophe einfügen. Ein in einer Zeichenspalte mit Apostrophen eingegebener Ausdruck "NULL" wird als Dateneingabe und nicht als Wert NULL angesehen.

Die Spalte "DepartmentHeadID" der Tabelle "Departments" lässt beispielsweise NULL-Werte zu. Sie können zwei Zeilen für Abteilungen ohne Abteilungsleiter wie folgt eingeben:

```
INSERT INTO Departments (DepartmentID, DepartmentName)
VALUES (201, 'Eastern Sales')
INSERT INTO Departments
VALUES (202, 'Western Sales', NULL);
```

NULL-Werte zurückgeben

NULL-Werte werden an die Clientanwendung wie andere Werte zur Anzeige zurückgegeben. Beispiel: Das folgende Beispiel illustriert, wie NULL-Werte in Interactive SQL angezeigt werden:

```
SELECT *
FROM Departments;
```

DepartmentID	DepartmentName	DepartmentHeadID
100	R & D	501
200	Sales	904
300	Finance	1293
400	Marketing	1576
500	Shipping	703
201	Eastern Sales	(NULL)

DepartmentID	DepartmentName	DepartmentHeadID
202	Western Sales	(NULL)

So vergleichen Sie Spaltenwerte mit NULL

Sie können die IS NULL-Suchbedingungen verwenden, um Spaltenwerte mit NULL zu vergleichen, und um sie auszuwählen bzw. eine bestimmte Aktion je nach Ergebnis des Vergleichs auszuführen. Nur Spalten, die den Wert TRUE zurückgeben, werden ausgewählt oder bewirken die beabsichtigte Aktion, diejenigen mit FALSE oder UNKNOWN nicht.

Im nachfolgenden Beispiel werden nur die Zeilen ausgewählt, für die "UnitPrice" geringer als 15 Dollar oder NULL ist:

```
SELECT Quantity, UnitPrice
FROM Products
WHERE UnitPrice < 15
OR UnitPrice IS NULL;
```

Das Ergebnis des Vergleichs eines Wertes mit NULL ist UNKNOWN, da nicht zu ermitteln ist, ob NULL gleich (oder ungleich) einem gegebenen Wert oder einem anderen NULL-Wert ist.

Es gibt Umstände, die niemals TRUE zurückgeben, sodass Abfragen, die diese Bedingungen verwenden, keine Ergebnismengen zurückgeben. Der folgende Vergleich kann beispielsweise nie als TRUE ermittelt werden, da NULL bedeutet, dass ein unbekannter Wert vorliegt.

```
WHERE column1 > NULL
```

Diese Logik gilt auch, wenn Sie zwei Spaltennamen in einer WHERE-Klausel verwenden, d.h. wenn Sie zwei Tabellen mit einem Join verbinden. Eine Klausel, die die Bedingung `WHERE column1 = column2` enthält, gibt keine Zeilen zurück, bei denen die Spalten NULL enthalten.

Sie können NULL oder Nicht-NULL auch mit diesen Mustern finden:

```
WHERE column_name IS NULL
WHERE column_name IS NOT NULL
```

Zum Beispiel:

```
WHERE advance < $5000
OR advance IS NULL
```

Siehe auch

- „NULL-Spezialwert“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Eigenschaften von NULL

Die folgende Liste enthält ausführlichere Bemerkungen zum NULL-Wert.

- **Der Unterschied zwischen FALSE und UNKNOWN** Obwohl weder FALSE noch UNKNOWN Werte zurückgeben, besteht ein wichtiger logischer Unterschied zwischen FALSE und UNKNOWN, da das Gegenteil von FALSE ("not false") TRUE ist, während das Gegenteil von UNKNOWN ("unbekannt") nicht bedeutet, dass etwas bekannt ist. Beispiel: $1 = 2$ wird als FALSE ausgewertet und $1 \neq 2$ (1 ist ungleich 2) zu TRUE.

Wenn aber NULL in einen Vergleich eingeschlossen wird, können Sie den Ausdruck nicht negieren, um die gegenteilige Gruppe von Zeilen oder den gegenteiligen TRUE-Wert zu erhalten. Ein UNKNOWN-Wert bleibt UNKNOWN.

- **NULL-Werte mit einem Wert ersetzen** Sie können die integrierte Funktion ISNULL verwenden, um NULL durch einen bestimmten Wert zu ersetzen. Die Ersetzung erfolgt nur für die Anzeige, die tatsächlichen Spaltenwerte sind davon nicht betroffen. Hierbei gilt die folgende Syntax:

ISNULL(*expression*, *value*)

Benutzen Sie zum Beispiel folgende Anweisung, um alle Zeilen aus "Departments" auszuwählen und alle NULL-Werte in Spalte "DepartmentHeadID" mit dem Wert -1 anzuzeigen.

```
SELECT DepartmentID,
       DepartmentName,
       ISNULL( DepartmentHeadID, -1 ) AS DepartmentHead
FROM Departments;
```

- **Ausdrücke, die mit NULL aufgelöst werden** Ein Ausdruck mit arithmetischen oder Bit-Operatoren wird auf NULL aufgelöst, wenn einer der Operanden der NULL-Wert ist. Beispiel: $1 + \text{column1}$ wird auf NULL aufgelöst, wenn column1 NULL ist.
- **Zeichenfolgen und NULL verketteten** Wenn Sie eine Zeichenfolge und NULL verketteten, wird der Ausdruck mit der Zeichenfolge aufgelöst. Beispielsweise liefert die folgende Anweisung die Zeichenfolge "abcdef":

```
SELECT 'abc' || NULL || 'def';
```

Siehe auch

- „Arithmetische Operatoren“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Bit-Operatoren“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Logische Operatoren, die Bedingungen verknüpfen

Die logischen Operatoren AND, OR und NOT werden benutzt, um Suchbedingungen in WHERE-Klauseln zu verbinden. Wenn mehr als ein logischer Operator in einer Anweisung verwendet wird, werden AND-Operatoren normalerweise vor den OR-Operatoren aufgelöst. Sie können die Reihenfolge der Auflösung mit Klammern ändern.

AND verwenden

Der Operator AND verbindet zwei oder mehr Bedingungen und gibt nur Ergebnisse zurück, wenn alle Bedingungen TRUE sind. Mit der folgenden Abfrage werden beispielsweise nur die Zeilen gefunden, in denen der Nachname der Kontaktperson Purcell lautet und der Vorname Beth.

```
SELECT *
FROM Contacts
WHERE GivenName = 'Beth'
AND Surname = 'Purcell';
```

OR verwenden

Der Operator OR verbindet zwei oder mehr Bedingungen und gibt Ergebnisse zurück, wenn *irgendeine* dieser Bedingungen TRUE ist. Mit den nachfolgenden Abfragen wird nach Zeilen gesucht, die Varianten von Elizabeth in der Spalte "GivenName" enthalten.

```
SELECT *
FROM Contacts
WHERE GivenName = 'Beth'
OR GivenName = 'Liz';
```

NOT verwenden

Der Operator NOT negiert den Ausdruck, der danach folgt. Die folgende Abfrage listet alle Kontaktpersonen auf, die nicht in Kalifornien leben:

```
SELECT *
FROM Contacts
WHERE NOT State = 'CA';
```

Suchbedingungen, die Datumsangaben vergleichen

Neben Gleichsetzungen können Sie in der WHERE-Klausel auch Operatoren verwenden, um eine Gruppe von Zeilen anzuzeigen, die die Suchbedingung erfüllen. Ungleichheitszeichen (< und >) können zum Vergleichen von Zahlen, Datumsangaben und auch Zeichenfolgen verwendet werden.

Beispiel

Führen Sie in Interactive SQL folgende Abfrage aus, um sämtliche Mitarbeiter aufzulisten, die vor dem 13. März 1964 geboren wurden:

```
SELECT Surname, BirthDate
FROM Employees
WHERE BirthDate < 'March 13, 1964'
ORDER BY BirthDate DESC;
```

Surname	BirthDate
Ahmed	1963-12-12
Dill	1963-07-19
Rebeiro	1963-04-12
Garcia	1963-01-23
Pastor	1962-07-14
...	...

Hinweise

- **Automatische Konvertierung in Datumsangaben** Der SQL Anywhere-Datenbankserver weiß, dass die Geburtsdatumsspalte "BirthDate" Datumsangaben enthält und konvertiert die Zeichenfolge 'March 13, 1964' automatisch in ein Datum.
- **Möglichkeiten zur Angabe eines Datums** Datumsangaben können auf verschiedene Weise dargestellt werden. Beispiel:

```
'March 13, 1964'
'1964/03/13'
'1964-03-13'
```

Sie können die Interpretation von Datumsangaben konfigurieren, indem Sie die Datenbankoption `date_order` setzen.

Datumsangaben im Format `yyyy/mm/dd` oder `yyyy-mm-dd` werden immer eindeutig als Datumsangaben erkannt, ganz gleich, wie die Option "date_order" eingestellt ist.

- **Weitere Vergleichsoperatoren** SQL Anywhere unterstützt mehrere Vergleichsoperatoren.

Siehe auch

- „date_order-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Vergleichsoperatoren“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Zeilenübereinstimmung nach Klang

Mit der Funktion `SOUNDEX` können Sie Zeilen suchen, bei denen der Klang übereinstimmt. Es wurde zum Beispiel eine telefonische Nachricht für einen Namen hinterlassen, der wie **Ms. Brown** klang. Sie könnten die folgende Abfrage ausführen, um nach Mitarbeitern zu suchen, die Namen haben, die ähnlich wie "Brown" klingen.

Hinweis

Der von `SOUNDEX` verwendete Algorithmus ist hauptsächlich für englischsprachige Datenbanken sinnvoll.

Beispiel

Führen Sie in Interactive SQL folgende Abfrage aus, um die Mitarbeiter aufzulisten, deren Nachname wie "Brown" klingt:

```
SELECT Surname, GivenName
FROM Employees
WHERE SOUNDEX( Surname ) = SOUNDEX( 'Brown' );
```

Surname	GivenName
Braun	Jane

Siehe auch

- „SOUNDEX-Funktion [Zeichenfolge]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Die ORDER BY-Klausel: Ergebnisse sortieren

Sofern nicht anders angegeben, gibt der Datenbankserver die Zeilen einer Tabelle in einer beliebigen Reihenfolge zurück. Es ist jedoch häufig hilfreich, die Zeilen in einer bestimmten Reihenfolge anzuzeigen. Sie können die Produkte zum Beispiel in alphabetischer Reihenfolge einsehen.

Sortieren Sie die Zeilen in der Ergebnismenge durch Hinzufügen einer ORDER BY-Klausel am Ende der SELECT-Anweisung. Diese SELECT-Anweisung hat die folgende Syntax:

```
SELECT column-name-1, column-name-2,...  
FROM table-name  
ORDER BY order-by-column-name
```

Dabei müssen *column-name-1*, *column-name-2* und *table-name* durch die Namen der Spalten und Tabelle ersetzt werden, die Sie abfragen möchten, und *order-by-column-name* muss durch eine Spalte in der Tabelle ersetzt werden. Sie können das Sternchen als eine Kurzform für alle Spalten in der Tabelle verwenden.

Hinweise

- **Die Reihenfolge der Klauseln ist wichtig** Die ORDER BY-Klausel muss auf die FROM- und die SELECT-Klausel folgen.
- **Sie können eine aufsteigende oder absteigende Sortierreihenfolge festlegen** Die Standardreihenfolge ist aufsteigend. Sie können eine absteigende Sortierreihenfolge festlegen, indem Sie das Schlüsselwort DESC am Ende der Klausel hinzufügen, wie in der folgenden Abfrage:

```
SELECT ID, Quantity  
FROM Products  
ORDER BY Quantity DESC;
```

ID	Quantity
400	112
700	80
302	75
301	54
600	39
...	...

- **Sie können nach mehreren Spalten sortieren** Die folgende Abfrage sortiert zuerst nach Größe (alphabetisch) und dann nach Namen:

```
SELECT ID, Name, Size
FROM Products
ORDER BY Size, Name;
```

ID	Name	Size
600	Sweatshirt	Large
601	Sweatshirt	Large
700	Shorts	Medium
301	Tee Shirt	Medium
...

- **Die ORDER BY-Spalte muss nicht Bestandteil der SELECT-Liste sein** Die folgende Abfrage sortiert die Produkte nach dem Stückpreis, obgleich der Preis nicht in der Ergebnismenge enthalten ist.

```
SELECT ID, Name, Size
FROM Products
ORDER BY UnitPrice;
```

ID	Name	Size
500	Visor	One size fits all
501	Visor	One size fits all
300	Tee Shirt	Small
400	Baseball Cap	One size fits all
...

- **Wenn Sie keine ORDER BY-Klausel verwenden und eine Abfrage mehr als einmal ausführen, hat es den Anschein, als würden unterschiedliche Ergebnisse abgerufen.** Dies ist darauf zurückzuführen, dass SQL Anywhere dasselbe Ergebnis mit einer anderen Reihenfolge zurückgibt. Ohne die Klausel ORDER BY gibt SQL Anywhere die Zeilen in der jeweils effizientesten Reihenfolge zurück. Das bedeutet, dass das Erscheinungsbild der Ergebnismengen abhängig davon variieren kann, wann Sie zuletzt auf die Zeile zugegriffen haben etc. Die einzige Möglichkeit, sicherzustellen, dass Zeilen in einer bestimmten Reihenfolge zurückgegeben werden, ist die Verwendung von ORDER BY.

Beispiel

Führen Sie in Interactive SQL folgende Abfrage aus, um die Produkte in alphabetischer Reihenfolge aufzulisten:

```
SELECT ID, Name, Description
FROM Products
ORDER BY Name;
```

ID	Name	Description
400	Baseball Cap	Cotton Cap
401	Baseball Cap	Wool cap
700	Shorts	Cotton Shorts
600	Sweatshirt	Hooded Sweatshirt
...

Siehe auch

- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Indizes, die die Performance der ORDER BY-Klausel verbessern

Manchmal gibt es für den SQL Anywhere-Datenbankserver nicht nur einen Weg, eine Abfrage mit einer ORDER BY-Klausel auszuführen. Mithilfe von Indizes können Sie den Datenbankserver in die Lage versetzen, die Tabellen effizienter zu durchsuchen.

Abfragen mit WHERE- und ORDER BY-Klauseln

Ein Beispiel für eine Abfrage, die auf mehrere Weisen ausgeführt werden kann, ist eine Abfrage, in der sowohl eine WHERE- als auch eine ORDER BY-Klausel enthalten ist.

```
SELECT *
FROM Customers
WHERE ID > 300
ORDER BY CompanyName;
```

In diesem Fall muss SQL Anywhere zwischen zwei Vorgehensweisen entscheiden:

1. Die gesamte Tabelle "Customers" anhand des Firmennamens durchsuchen, wobei jede Zeile dahingehend überprüft wird, ob die Kunden-ID größer als 300 ist.
2. Den Schlüssel für die ID-Spalte verwenden, um nur die Firmen zu überprüfen, deren ID größer als 300 ist. Die Ergebnisse müssten dann nach Firmennamen sortiert werden.

Wenn es nur sehr wenige ID-Werte gibt, die größer als 300 sind, ist die zweite Strategie besser, da nur wenige Zeilen abgesucht werden, die rasch sortiert sind. Wenn die meisten ID-Werte größer als 300 sind, ist die erste Strategie vorzuziehen, da kein Sortieren notwendig ist.

Das Problem lösen

Sie können das Problem aus dem vorigen Beispiel lösen, indem Sie einen zweispaltigen Index für ID und "CompanyName" erstellen. SQL Anywhere kann dann diesen Index verwenden, um Zeilen in der richtigen Reihenfolge aus der Tabelle auszuwählen. Vergessen Sie allerdings nicht, dass Indizes Platz in der Datenbankdatei erfordern und Overhead verursachen, um auf dem letzten Stand gehalten zu werden. Daher sollten Sie Indizes nicht wahllos erstellen.

Siehe auch

- Die richtige Auswahl der Indizes kann sich entscheidend auf die Performance auswirken auf Seite 232

Aggregatfunktionen in Abfragen

Einige Abfragen untersuchen Aspekte der Daten in Ihrer Tabelle, die Eigenschaften von Zeilengruppen und nicht einzelner Zeilen widerspiegeln. So möchten Sie vielleicht den Durchschnittsbetrag herausfinden, den Ihre Kunden pro Auftrag bezahlen, oder Sie möchten feststellen, wie viele Mitarbeiter in jeder Abteilung arbeiten. Diese Aufgaben können Sie mithilfe von **Aggregatfunktionen** und der GROUP BY-Klausel erledigen.

Die Funktionen COUNT, MIN und MAX werden als **Aggregatfunktionen** bezeichnet. Aggregatfunktionen fassen Daten zusammen. Andere Aggregatfunktionen sind zum Beispiel Statistikfunktionen, wie etwa AVG, STDDEV und VARIANCE. Alle Aggregatfunktionen außer COUNT benötigen einen Parameter.

Aggregatfunktionen geben für eine Reihe von Zeilen einen einzigen Wert zurück. Wenn keine GROUP BY-Klausel vorhanden ist, gibt die **Aggregatfunktion** einen einzelnen Wert für alle Zeilen zurück, die andere Aspekte der Abfrage erfüllen. Wenn es eine GROUP BY-Klausel gibt, wird die Aggregatfunktionen ein **Vektoraggregat** genannt und gibt einen Wert für jede Gruppe zurück.

SQL Anywhere unterstützt zusätzliche Aggregatfunktionen für Analysen, die manchmal auch OLAP-Funktionen genannt werden. Einige dieser Funktionen können als Fensterfunktionen verwendet werden: sie enthalten RANK, PERCENT_RANK, CUME_DIST, ROW_NUMBER und Funktionen zur Unterstützung von linearen Regressionsanalysen.

Beispiel

- So listen Sie die Anzahl der Mitarbeiter im Unternehmen auf** Führen Sie in Interactive SQL folgende Abfrage aus:

```
SELECT COUNT( * )
FROM Employees;
```

COUNT()
75

Die Ergebnismenge besteht nur aus einer Spalte mit dem Titel COUNT(*) und einer Zeile, die die Anzahl der Mitarbeiter enthält.

- **So listen Sie die Anzahl der Mitarbeiter des Unternehmens und das Geburtsdatum des ältesten und des jüngsten Mitarbeiters auf** Führen Sie in Interactive SQL folgende Abfrage aus:

```
SELECT COUNT( * ), MIN( BirthDate ), MAX( BirthDate )  
FROM Employees;
```

COUNT()	MIN(Employees.BirthDate)	MAX(Employees.BirthDate)
75	1936-01-02	1973-01-18

Siehe auch

- „Aggregatfunktionen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „OLAP-Unterstützung“ auf Seite 552

Wie mit Aggregatfunktionen Daten gruppiert werden

Aggregatfunktionen liefern nicht nur Informationen über eine gesamte Tabelle, sondern können auch auf Gruppen von Zeilen angewendet werden. Die GROUP BY-Klausel ordnet Zeilen in Gruppen an, und Aggregatfunktionen geben einen einzelnen Wert für jede Gruppe von Zeilen zurück.

Semantische Unterschiede bei der leeren Menge

Die SQL-Sprache behandelt die leere Menge bei der Verwendung von Aggregatfunktionen unterschiedlich. Ohne eine GROUP BY-Klausel gibt eine Abfrage mit einer Aggregatfunktion über null Eingabezeilen eine einzelne Zeile als Ergebnis zurück. Im Fall von COUNT ist das Ergebnis der Wert Null, und bei allen anderen Aggregatfunktionen ist das Ergebnis NULL. Wenn die Abfrage jedoch eine GROUP BY-Klausel enthält und die Eingabe für die Abfrage leer ist, ist das Ergebnis der Abfrage leer und es werden keine Zeilen zurückgegeben.

Zum Beispiel gibt die folgende Abfrage eine einzelne Zeile mit dem Wert 0 zurück; gibt es keine Mitarbeiter in Abteilung 103.

```
SELECT COUNT() FROM Employees WHERE DepartmentID = 103;
```

Diese geänderte Abfrage allerdings gibt keine Zeilen aufgrund des Vorhandenseins der GROUP BY-Klausel zurück.

```
SELECT COUNT() FROM Employees WHERE DepartmentID = 103 GROUP BY State;
```

Ein häufiger Fehler bei GROUP BY

Ein häufiger Fehler im Zusammenhang mit GROUP BY besteht darin, dass versucht wird, Informationen abzufragen, die nicht korrekt in einer Gruppe zusammengefasst werden können. Folgende Abfrage erzeugt z.B. eine Fehlermeldung.

```
SELECT SalesRepresentative, Surname, COUNT( * )  
FROM SalesOrders KEY JOIN Employees  
GROUP BY SalesRepresentative;
```


Die Fehlermeldung weist darauf hin, dass eine Referenz der Spalte "Surname" auch in der GROUP BY-Klausel angegeben werden muss. Der Fehler wird gemeldet, da SQL Anywhere nicht prüfen kann, ob alle Ergebniszeilen für einen Mitarbeiter mit einer bestimmten ID den gleichen Nachnamen enthalten.

Diesen Fehler können Sie vermeiden, indem Sie die Spalte in die GROUP BY-Klausel miteinbeziehen.

```
SELECT SalesRepresentative, Surname, COUNT( * )
FROM SalesOrders KEY JOIN Employees
GROUP BY SalesRepresentative, Surname
ORDER BY SalesRepresentative;
```

Wenn dies nicht geeignet ist, können Sie stattdessen eine Aggregatfunktion verwenden, um nur einen einzigen Wert auszuwählen:

```
SELECT SalesRepresentative, MAX( Surname ), COUNT( * )
FROM SalesOrders KEY JOIN Employees
GROUP BY SalesRepresentative
ORDER BY SalesRepresentative;
```

Die Funktion MAX wählt den höchsten (den alphabetisch letzten) Nachnamen aus den Detailzeilen für jede Gruppe. Diese Anweisung ist gültig, da es nur einen eindeutigen Maximalwert geben kann. In diesem Fall wird der Nachname in jeder Detailzeile innerhalb einer Gruppe angezeigt.

Beispiel

Führen Sie in Interactive SQL folgende Abfrage aus, um die Verkäufer und die Anzahl der Aufträge aufzulisten, die jeder von ihnen entgegengenommen hat:

```
SELECT SalesRepresentative, COUNT( * )
FROM SalesOrders
GROUP BY SalesRepresentative
ORDER BY SalesRepresentative;
```

SalesRepresentative	COUNT()
129	57
195	50
299	114
467	56
...	...

Eine GROUP BY-Klausel weist SQL Anywhere an, die Menge aller Zeilen zu partitionieren, die sonst zurückgegeben würden. Alle Zeilen in jeder Partition oder Gruppe haben in den benannten Spalten dieselben Werte. Es gibt für jeden eindeutigen Wert oder jede Wertemenge nur eine einzige Gruppe. In diesem Beispiel haben sämtliche Zeilen in jeder Gruppe denselben Wert für "SalesRepresentative".

Aggregatfunktionen, wie z.B. COUNT, können dann für die Zeilen in jeder Gruppe angewendet werden. Daher wird in dieser Ergebnismenge die gesamte Anzahl der Zeilen in jeder Gruppe zurückgegeben. In den Ergebnissen der Abfrage ist folglich eine Zeile für jede Verkäufer-ID enthalten. Jede Zeile enthält wiederum die Verkäufer-ID sowie die gesamte Anzahl der Bestellungen für diesen Verkäufer.

Wenn die GROUP BY-Klausel verwendet wird, hat die resultierende Tabelle eine Zeile für jede in der GROUP BY-Klausel aufgeführte Spalte oder Spaltengruppe.

Siehe auch

- „Die GROUP BY-Klausel: Abfrageergebnisse in Gruppen organisieren“ auf Seite 474
- „GROUP BY-Klausel“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „GROUP BY mit Aggregatfunktionen“ auf Seite 477
- „SELECT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

HAVING-Klausel

Schränken Sie die Zeilen in Gruppen mithilfe der HAVING-Klausel ein.

Beispiel

Führen Sie in Interactive SQL folgende Abfrage aus, um alle Verkäufer mit mehr als 55 Aufträgen aufzulisten:

```
SELECT SalesRepresentative, COUNT( * ) AS orders
FROM SalesOrders KEY JOIN Employees
GROUP BY SalesRepresentative
HAVING count( * ) > 55
ORDER BY orders DESC;
```

SalesRepresentative	orders
299	114
129	57
1142	57
467	56

Siehe auch

- „Die HAVING-Klausel: Datengruppen auswählen“ auf Seite 480
- „Unterabfragen in der HAVING-Klausel“ auf Seite 607
- „SELECT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Kombination aus WHERE- und HAVING-Klausel

Manchmal können Sie dieselbe Gruppe von Zeilen sowohl mit einer WHERE-Klausel als auch mit einer HAVING-Klausel angeben. In diesen Fällen ist keine der beiden Methoden effektiver als die andere. Der Optimierer analysiert stets automatisch jede Anweisung, die Sie eingeben, und wählt eine effiziente Methode für deren Ausführung. Sie sollten die Syntax verwenden, mit der das gewünschte Ergebnis am deutlichsten beschrieben wird. Im Allgemeinen bedeutet das, dass ungewünschte Zeilen in früheren Klauseln entfernt werden müssen.

Beispiel

Mit der folgenden Anweisung listen Sie alle Verkäufer mit mehr als 55 Aufträgen und einer Kennung auf, die größer als 1.000 ist:

```
SELECT SalesRepresentative, COUNT( * )
FROM SalesOrders
WHERE SalesRepresentative > 1000
GROUP BY SalesRepresentative
HAVING count( * ) > 55
ORDER BY SalesRepresentative;
```

Die nachfolgende Anweisung liefert dieselben Ergebnisse.

```
SELECT SalesRepresentative, COUNT( * )
FROM SalesOrders
GROUP BY SalesRepresentative
HAVING count( * ) > 55 AND SalesRepresentative > 1000
ORDER BY SalesRepresentative;
```

SQL Anywhere stellt fest, dass beide Anweisungen dieselbe Ergebnismenge beschreiben, und führt jede Anweisung effizient aus.

Fortgeschrittene Aufgaben: Phasen der Abfrageverarbeitung

Dieser Abschnitt beschreibt die Phasen, die eine Anweisung durchläuft, beginnend mit der Kommentierungsphase und endend mit der Ausführung. Darüber hinaus werden die der Planung des Optimierers zugrunde liegenden Annahmen beschrieben und die Selektivitätsschätzung, die Kostenschätzung und die Schritte der Abfrageverarbeitung behandelt.

Weitere Hinweise zur Abfrageverarbeitung in SQL Anywhere finden Sie im Whitepaper "*Query Processing Based on SQL Anywhere 12.0.1 Architecture*" unter <http://www.sybase.com/detail?id=1096047>.

Anweisungen, die keine Ergebnismengen haben, wie UPDATE- oder DELETE-Anweisungen, durchlaufen die Phasen der Abfrageverarbeitung

- Kommentierungsphase** Wenn der Datenbankserver eine Abfrage erhält, verwendet er einen Parser, um die Anweisung zu analysieren, und wandelt sie in eine algebraische Darstellung um, die auch als Parse-Baum bezeichnet wird. In diesem Stadium kommt der **Parse-Baum** für semantische und syntaktische Prüfungen zum Einsatz (z.B. die Bestätigung, dass die in der Abfrage referenzierten Objekte im Katalog vorhanden sind) sowie für Privilegienprüfungen, Transformationen von KEY JOINS und NATURAL JOINS unter Verwendung von definierten referenziellen Integritätsregeln und für die Erweiterung von nicht materialisierten Ansichten. Die Ausgabe dieser Phase ist eine umgeschriebene Abfrage in Form eines Parse-Baums, der die Kommentare (erweiterte Informationen wie Datentypen) zu allen Objekten enthält, die in der ursprünglichen Abfrage referenziert wurden.
- Phase der semantischen Transformationen** In dieser Phase durchläuft die Abfrage iterative semantische Transformationen. Während die Abfrage noch als kommentierter Parse-Baum dargestellt wird, werden in dieser Phase Neuschreibungsoptimierungen angewendet wie z.B. Join-Eliminierungen, DISTINCT-Eliminierungen und Prädikatnormalisierung. Die semantischen

Transformationen in dieser Phase werden gemäß den semantischen Transformationsregeln durchgeführt, die heuristisch auf die Darstellung des Parse-Baums angewendet werden.

Abfragen mit Plänen, die bereits vom Datenbankserver im Cache abgelegt wurden, überspringen diese Phase der Abfrageverarbeitung. Einfache Anweisungen können ebenfalls diese Phase der Abfrageverarbeitung überspringen. Viele Anweisungen, die eine heuristische Planauswahl im Bypass-Mechanismus des Optimierers verwenden, werden von der Phase der semantischen Transformationen nicht abgearbeitet. Die Komplexität der SQL-Anweisung legt fest, ob diese Phase an einer Anweisung angewendet wird.

- **Optimierungsphase** In der Optimierungsphase wird eine andere interne Darstellung der Abfrage - die Abfragenoptimierungsstruktur - benutzt, die aus dem Parse-Baum erstellt wird.

Abfragen mit Plänen, die bereits vom Datenbankserver im Cache abgelegt wurden, überspringen diese Phase der Abfrageverarbeitung. Zusätzlich können einfache Anweisungen ebenfalls diese Phase der Abfrageverarbeitung überspringen.

Diese Phase wird in zwei Unterphasen aufgeteilt:

- **Vorphase der Optimierung** In der Vorphase der Optimierung wird die Optimierungsstruktur durch die Informationen vervollständigt, die später in der Enumerationsphase benötigt werden. In dieser Phase wird die Abfrage analysiert, um alle relevanten Indizes und materialisierten Ansichten zu finden, die im Zugriffsplan der Abfrage benutzt werden können. Beispielsweise ermittelt der Algorithmus für die Ansichtenübereinstimmung in dieser Phase alle materialisierten Ansichten, die benutzt werden können, um eine Abfrage ganz oder teilweise zu erfüllen. Zusätzlich erstellt der Optimierer aufgrund der Prädikat-Analyse der Abfrage alternative Join-Methoden, die in der Enumerationsphase benutzt werden können, um einen Join für die Tabellen der Abfrage herzustellen. In dieser Phase wird keine Entscheidung über den besten Zugriffsplan für die Abfrage getroffen. Das Ziel dieser Phase ist die Vorbereitung auf die Enumerationsphase.
- **Enumerationsphase** In dieser Phase enumeriert der Optimierer mögliche Zugriffspläne für die Abfrage und benutzt dabei die Bausteine, die in der Vorphase der Optimierung generiert wurden. Der Suchbereich ist sehr groß und der Optimierer verwendet einen herstellereigenen Enumerationsalgorithmus, um die Zugriffspläne zu generieren und zu kürzen. Für jeden Plan wird eine Kostenschätzung berechnet, um den aktuellen Plan mit dem bisher besten gefundenen Plan zu vergleichen. Bei diesen Vergleichen werden kostenträchtige Pläne verworfen. Bei der Kostenschätzung werden die Ressourcennutzung wie etwa Platten- und Arbeitsspeichervorgänge, die geschätzte Anzahl von Zeilen für die Zwischenergebnisse, das Optimierungsziel, die Cachegröße usw. berücksichtigt. Das Resultat der Enumerationsphase ist der beste Zugriffsplan für die Abfrage.
- **Planerstellungphase** In der Planerstellungphase wird der beste Zugriffsplan verwendet, um die entsprechende endgültige Darstellung des Abfrageausführungsplans zum Ausführen der Abfrage zu erstellen. Sie können eine grafische Version des Plans in der Plananzeige in Interactive SQL einsehen. Der grafische Plan hat eine Baumstruktur, in der jeder Knoten ein physischer Operator ist, der einen spezifischen relationalen, algebraischen Vorgang implementiert. "Hash-Join" und "Ordered-Group-By" sind z.B. physische Operatoren, die einen Join bzw. einen Group-by-Vorgang implementieren.

Abfragen mit Plänen, die bereits vom Datenbankserver im Cache abgelegt wurden, überspringen diese Phase der Abfrageverarbeitung.

- **Ausführungsphase** Das Ergebnis der Abfrage wird durch den Abfrageausführungsplan berechnet, der in der Planerstellungphase erstellt wurde.

Siehe auch

- „Optimierungen während der Abfrageverarbeitung“ auf Seite 340
- „Eignung für ein Überspringen der Abfrageverarbeitungsphase“ auf Seite 331
- „Plan-Caching“ auf Seite 337
- „Grafische Pläne“ auf Seite 347
- „So funktioniert der Optimierer“ auf Seite 333

Eignung für ein Überspringen der Abfrageverarbeitungsphase

Fast alle Anweisungen durchlaufen alle Abfrageverarbeitungsphasen. Es gibt allerdings zwei Ausnahmen: Abfragen, die vom **Plan-Caching** profitieren (Abfragen, die bereits vom Datenbankserver im Cache abgelegt wurden), und **Bypass-Abfragen**.

- **Plan-Caching** Bei Abfragen innerhalb von gespeicherten Prozeduren und benutzerdefinierten Funktionen legt der Datenbankserver die Ausführungspläne eventuell im Cache ab, sodass sie wiederverwendet werden können. Für diese Abfrageklasse wird der Abfrageausführungsplan nach der Ausführung im Cache abgelegt. Bei der nächsten Ausführung der Abfrage wird der Plan abgerufen und alle Phasen bis zur Ausführung werden übersprungen.
- **Bypass-Abfragen** Bypass-Abfragen sind eine Unterklasse der einfachen Abfragen, die bestimmte Merkmale haben, an denen der Datenbankserver erkennt, dass sie für ein Umgehen des Optimierers in Frage kommen. Ein Umgehen der Optimierung kann die erforderliche Zeit zum Erstellen eines Ausführungsplans vermindern.

Wenn eine Abfrage als Bypass-Abfrage erkannt wird, wird eine heuristische Optimierung statt einer kostenbasierten Optimierung verwendet. Dabei werden die semantischen Transformations- und die Optimierungsphase übergangen und der Abfrageausführungsplan wird direkt aus der Darstellung des Parse-Baums der Abfrage erstellt.

Einfache Abfragen

Eine einfache Abfrage ist eine SELECT-, INSERT-, DELETE- oder UPDATE-Anweisung in einem einzelnen Abfrageblock und den folgenden Merkmalen:

- Der Abfrageblock enthält keine Unterabfragen oder zusätzliche Abfrageblöcke wie z.B. für UNION, INTERSECT, EXCEPT und allgemeine Tabellenausdrücke.
- Der Abfrageblock referenziert eine einzige Basistabelle oder materialisierte Ansicht.
- Der Abfrageblock kann die TOP N-, FIRST-, ORDER BY- oder DISTINCT-Klauseln enthalten.
- Der Abfrageblock kann auch Aggregatfunktionen ohne GROUP BY- oder HAVING-Klauseln enthalten.
- Der Abfrageblock enthält keine Fensterfunktionen.

- Die Abfrageblock-Ausdrücke enthalten nicht NUMBER, IDENTITY oder Unterabfragen.
- Die für die Basistabelle festgelegten Integritätsregeln sind einfache Ausdrücke.

Eine komplexe Anweisung kann nach der semantischen Transformationsphase in eine einfache Anweisung umgewandelt werden. Wenn dies passiert, kann die Abfrage vom Optimierer-Bypass abgearbeitet oder ihr Plan vom SQL Anywhere-Server im Cache abgelegt werden.

Optimierung und keine Optimierung erzwingen

Sie können erzwingen, dass Abfragen, die für das Plan-Caching oder für das Umgehen des Optimierers geeignet wären, vom SQL Anywhere-Optimierer abgearbeitet werden. Um dies zu tun, verwenden Sie die FORCE OPTIMIZATION-Klausel mit einer SQL-Anweisung.

Sie können auch erzwingen, dass eine Anweisung den Optimierer umgeht. Um dies zu tun, verwenden Sie die FORCE NO OPTIMIZATION-Klausel in der Anweisung. Wenn die Anweisung zu komplex ist, um den Optimierer zu umgehen - möglicherweise auf Grund von Datenbankoptionseinstellungen oder Merkmalen des Schemas oder der Abfrage -, schlägt die Abfrage fehl und ein Fehler wird zurückgegeben.

Die Klauseln FORCE OPTIMIZATION und FORCE NO OPTIMIZATION sind in der OPTION-Klausel der folgenden Anweisungen zulässig:

- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „UPDATE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „INSERT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DELETE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Siehe auch

- „Plan-Caching“ auf Seite 337

Fortgeschrittene Aufgaben: Abfrageoptimierung

Die Optimierung ist entscheidend für das Erstellen eines geeigneten Zugriffsplans für eine Abfrage. Der **Abfrageoptimierer** (oder einfach der Optimierer) analysiert die Abfrage und entscheidet sich für einen Zugriffsplan, der das Ergebnis mit möglichst geringem Ressourcenverbrauch berechnet. Die Optimierung beginnt direkt vor der Ausführung. Wenn Sie in Ihrer Anwendung Cursor verwenden, beginnt die Optimierung, wenn der Cursor geöffnet wird.

Anders als viele andere kommerzielle Datenbanksysteme optimiert SQL Anywhere normalerweise jede Anweisung kurz vor der Ausführung. Da SQL Anywhere eine "just-in-time"-Optimierung der einzelnen Anweisungen durchführt, hat der Optimierer Zugriff auf die Werte von Hostvariablen und Variablen der gespeicherten Prozeduren, wodurch Analysen der Selektivitätsschätzung verbessert werden können. Außerdem ermöglicht es die "just-in-time"-Optimierung dem Optimierer, seine Auswahl aufgrund der Statistiken anzupassen, die nach vorherigen Abfrageausführungen gespeichert wurden.

Um effizient zu arbeiten, schreibt SQL Anywhere Ihre Abfragen in semantisch äquivalente, aber syntaktisch unterschiedliche Formen um. SQL Anywhere führt viele unterschiedliche Neuschreibungsvorgänge durch. Wenn Sie die Zugriffspläne studieren, werden Sie häufig feststellen, dass sie keine wörtliche Interpretation Ihrer ursprünglichen Anweisung darstellen. Um beispielsweise Ihre

SQL-Anweisungen effizienter zu gestalten, versucht der Optimierer, so weit es geht Unterabfragen mit Joins umzuschreiben.

Weitere Hinweise zur Abfrageverarbeitung in SQL Anywhere finden Sie im Whitepaper "*Query Processing Based on SQL Anywhere 12.0.1 Architecture*" unter <http://www.sybase.com/detail?id=1096047>.

So funktioniert der Optimierer

Die Rolle des Optimierers besteht darin, einen effizienten Weg zum Ausführen von SQL-Anweisungen zu finden. Dazu muss der Optimierer einen Ausführungsplan für eine Abfrage festlegen. Dieser Plan umfasst Entscheidungen über die Zugriffsreihenfolge für Tabellen, die in der Abfrage referenziert werden, über die Join-Operatoren und Zugriffsmethoden für jede Tabelle sowie darüber, ob materialisierte Ansichten, die nicht in der Abfrage referenziert werden, zur Berechnung von Teilen der Abfrage benutzt werden können. Der Optimierer versucht, den besten Plan für die Ausführung der Abfrage während der Join-Enumerationsphase auszuwählen, während der mögliche Zugriffspläne für eine Abfrage generiert und die Kosten berechnet werden. Der beste Zugriffsplan ist derjenige, der nach Einschätzung des Optimierers die gewünschte Ergebnismenge in der kürzesten Zeit und mit den geringsten Kosten liefern wird. Der Optimierer ermittelt die Kosten jeder bei der Enumeration gefundenen Strategie, indem er die Anzahl der erforderlichen Lese- und Schreibvorgänge schätzt.

In Interactive SQL können Sie den besten Zugriffsplan zum Ausführen einer Abfrage anzeigen, indem Sie auf **Extras » Plananzeige** klicken.

Kosten für die Rückgabe der ersten Zeile minimieren

Der Optimierer verwendet ein generisches Kostenmodell für Festplattenzugriffe, um die relativen Performance-Unterschiede zwischen wahlfreien und sequenziellen Abrufen aus der Datenbankdatei zu unterscheiden. Eine Datenbank kann auch mithilfe einer ALTER DATABASE-Anweisung für eine bestimmte Hardware-Konfiguration angepasst werden.

Standardmäßig wird die Abfrageverarbeitung so optimiert, dass die komplette Ergebnismenge schnell zurückgegeben wird. Mit der Option "optimization_goal" können Sie das Standardverhalten ändern, um die Kosten für die schnelle Rückgabe der ersten Zeile zu minimieren. Wenn die Option auf "First-row" gesetzt ist, bevorzugt der Optimierer einen Zugriffsplan, der die Zeitspanne zum Abruf der ersten Zeile der Ergebnismenge reduzieren soll, worunter möglicherweise die Gesamtabrufzeit leidet.

Semantisch äquivalente Syntax verwenden

Die meisten Anweisungen können unter Verwendung der SQL-Sprache auf viele Arten ausgedrückt werden. Diese Ausdrücke sind semantisch gleichwertig, wodurch sie dieselbe Aufgabe durchführen, aber sie können sich in ihrer Syntax grundlegend unterscheiden. Mit wenigen Ausnahmen entwirft der Optimierer einen geeigneten Zugriffsplan, der nur auf der Semantik der einzelnen Anweisungen aufbaut.

Syntaktische Unterschiede haben gewöhnlich keine Auswirkungen, auch wenn sie substanziell erscheinen mögen. So haben zum Beispiel Unterschiede in der Reihenfolge von Prädikaten, Tabellen und Attributen in der Abfrage keine Auswirkungen auf die Auswahl im Zugriffsplan. Auch wird der Optimierer nicht davon beeinflusst, ob eine Abfrage eine nicht materialisierte Ansicht enthält.

Kosten von Optimierungsabfragen vermindern

Idealerweise würde der Optimierer den effizientesten Plan, der zur Verfügung steht, auswählen, doch das ist oft nicht durchführbar. Bei einer komplizierten Abfrage gibt es zahlreiche mögliche Lösungen.

Wie effizient der Optimierer auch sein mag, so benötigt doch das Analysieren der einzelnen Optionen Zeit und Ressourcen. Der Optimierer vergleicht die Kosten weiterer Optimierungen mit den Kosten, die ein Ausführen des bisher besten Plans verursachen würde. Wenn ein Plan mit relativ geringen Kosten entwickelt worden ist, stoppt der Optimierer und ermöglicht es, dass dieser Plan ausgeführt wird. Eine weitere Optimierung könnte mehr Ressourcen als das Ausführen eines bereits gefundenen Zugriffsplans verbrauchen. Sie können kontrollieren, wie viel Aufwand der Optimierer betreibt, indem Sie für die Option "optimization_level" einen hohen Wert festlegen.

Der Optimierer arbeitet länger bei kostenträchtigen und komplexen Abfragen oder wenn eine hohe Optimierungsstufe eingestellt ist. Bei sehr kostenträchtigen Abfragen dauert die Ausführung möglicherweise so lange, dass eine Verzögerung feststellbar ist.

Siehe auch

- „Grafische Pläne“ auf Seite 347
- „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ auf Seite 341
- „ALTER DATABASE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „optimization_level-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „optimization_goal-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Optimiererschätzungen und -statistiken

Der Optimierer wählt eine Strategie zur Verarbeitung einer Anweisung aufgrund von in der Datenbank gespeicherten **Spaltenstatistiken** und **Heuristiken** (fundierte Vermutungen). Für jeden Zugriffsplan, der vom Optimierer in Betracht gezogen wird, muss eine geschätzte Ergebnisgröße (Anzahl von Zeilen) berechnet werden. Es wird z.B. für jede Join-Methode oder für jeden Indexzugriff auf der Basis der Selektivitätsschätzungen für die in der Abfrage benutzten Prädikate eine geschätzte Ergebnisgröße berechnet. Die geschätzten Ergebnisgrößen werden benutzt, um den geschätzten Plattenzugriff und die CPU-Kosten für die einzelnen im Plan benutzten Operatoren wie etwa eine Join-Methode, eine Group By-Methode oder einen sequenziellen Scan zu berechnen. Der Optimierer benutzt Spaltenstatistiken als primäre Daten für die Berechnung von Selektivitätsschätzungen für Prädikate. Die Spaltenstatistiken sind daher ausschlaggebend für die korrekte Schätzung der Kosten eines Zugriffsplans.

Wenn Spaltenstatistiken veraltet sind oder fehlen, kann die Performance nachlassen, da fehlerhafte Statistiken zu ineffizienten Ausführungsplänen führen können. Wenn Sie den Eindruck haben, dass eine schlechte Performance auf fehlerhafte Statistiken zurückzuführen ist, sollten Sie die Statistiken neu erstellen.

Wie der Optimierer Statistiken verwendet

Für den Optimierer sind **Histogramme** die wichtigste Komponente der Spaltenstatistiken. Histogramme speichern Daten über die Verteilung von Werten in einer Spalte. In SQL Anywhere stellt ein Histogramm die Datenverteilung für eine Spalte folgendermaßen dar: Die Domäne der Spalte wird in eine Reihe aufeinander folgender Wertebereiche unterteilt (auch als **Buckets** bezeichnet). Für jeden Wertebereich

(oder Bucket) wird die Anzahl der Zeilen in derjenigen Tabelle festgehalten, für die der Spaltenwert in den Bucket fällt.

SQL Anywhere achtet besonders auf einzelne Spaltenwerte, die in einer großen Anzahl von Zeilen in der Tabelle vertreten sind. Bedeutende Einzelwert-Selektivitäten werden in Einzelwert-Histogramm-Buckets aufrechterhalten (z.B. Buckets, die einen Einzelwert in der Spaltendomäne umgeben). SQL Anywhere versucht, eine Mindestzahl von Einzelwert-Buckets in jedem Histogramm aufrechtzuerhalten, je nach Größe der Tabelle normalerweise zwischen 10 und 100. Außerdem werden alle Einzelwerte mit Selektivitäten über 1 % in Einzelwert-Buckets aufbewahrt. Daher erinnert sich ein Histogramm für eine bestimmte Spalte an die höchsten N Einzelwertselektivitäten für die Spalte, wobei der Wert von N von der Größe der Tabelle und der Anzahl der Einzelwert-Selektivitäten über 1 % abhängt.

Sobald die Mindestzahl der Wertebereiche erreicht wurde, werden Häufigkeiten niedriger Selektivität durch Häufigkeiten großer Selektivität ersetzt, sobald diese auftauchen. Das Histogramm hat nur dann mehr Einzelwert-Wertebereiche als die Mindestzahl, wenn es genügend Werte mit einer Selektivität von über 1 % gesehen hat.

Im Gegensatz zu Basistabellen gibt es zu Prozeduraufrufen, die in der FROM-Klausel ausgeführt werden, keine Spaltenstatistiken. Daher verwendet der Optimierer Standard- oder Schätzwerte bei allen Selektivitätsschätzungen für Daten, die durch einen Prozeduraufruf geliefert werden. Die Ausführungszeit eines Prozeduraufrufs und die Gesamtzahl der Zeilen in einer Ergebnismenge werden aufgrund von Statistikwerten aus vorherigen Aufrufen geschätzt. Diese Statistiken werden in der ISYSPROCEDURE-Systemtabelle in der Spalte "stats" verwaltet.

Wie der Optimierer Heuristiken verwendet

Für jede Tabelle in einem möglichen Ausführungsplan schätzt der Optimierer die Anzahl der Zeilen, die Teil der Ergebnisse sein können. Die Anzahl der Zeilen hängt von der Größe der Tabelle und von den Einschränkungen in der WHERE-Klausel oder der ON-Klausel der Abfrage ab.

Unter Berücksichtigung des Histogramms für eine Spalte schätzt SQL Anywhere die Anzahl der Zeilen, die die Vorgaben eines gegebenen Abfrageprädikats auf die Spalte erfüllen. Hierzu rechnet er die Anzahl der Zeilen in all den Wertebereichen zusammen, welche die dem spezifischen Prädikat entsprechenden Werte überlappen. Für Wertebereiche in den Histogrammen, die teilweise in der Ergebnismenge der Abfrage enthalten sind, verwendet SQL Anywhere eine Interpolation in den Wertebereichen.

Häufig verwendet der Optimierer differenziertere Heuristiken. Der Optimierer verwendet z.B. nur dann Standardschätzungen, wenn keine besseren Statistiken verfügbar sind. Der Optimierer benutzt außerdem Indizes und Schlüssel, um seine Schätzung der Zeilenanzahl zu verbessern. Es folgen einige Beispiele für Einzelspalten:

- Eine Spalte mit einem Wert vergleichen: Die Schätzung ergibt eine Zeile, wenn die Spalte einen eindeutigen Index hat oder Primärschlüssel ist.
- Ein Vergleich einer Spalte im Index mit einer Konstante: Index untersuchen, um den Prozentsatz der Zeilen zu schätzen, für die der Vergleich zutrifft.
- Einen Fremdschlüssel einem Primärschlüssel gleichsetzen (Schlüssel-Join): Relative Tabellengrößen bei der Bestimmung einer Schätzung verwenden. Wenn zum Beispiel eine 5000zeilige Tabelle einen

Fremdschlüssel zu einer 1000-zeiligen Tabelle hat, schätzt der Optimierer, dass es fünf Fremdschlüsselzeilen für jede Primärschlüsselzeile gibt.

Siehe auch

- „Tipp: Spaltenstatistiken aktualisieren“ auf Seite 229
- „Quellen der Selektivitätsschätzung“ auf Seite 336
- „ESTIMATE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ESTIMATE_SOURCE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSPROCEDURE-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_get_histogram-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Histogramm-Dienstprogramm (dbhist)“ [*SQL Anywhere Server - Datenbankadministration*]

Quellen der Selektivitätsschätzung

Der Optimierer kann für jedes Prädikat alle folgenden Quellen für Selektivitätsschätzungen verwenden. Die gewählte Quelle wird im grafischen und im ausführlichen Plan für die Abfrage angezeigt.

- **Statistiken** Der Optimierer kann mithilfe von gespeicherten Spaltenstatistiken Selektivitätsschätzungen berechnen. Wenn Konstanten im Prädikat verwendet werden, sind die gespeicherten statistischen Daten nur verfügbar, wenn die Selektivität einer Konstante eine Zahl mit ausreichender Signifikanz ist, die in den Statistiken gespeichert wird.

Beispiel: Das Prädikat `EmployeeID > 100` kann Spaltenstatistiken als Quelle für die Selektivitätsschätzung verwenden, wenn die Statistik für die Spalte "EmployeeID" vorhanden ist.

- **Join** Der Optimierer kann Integritätsregeln zur Erhaltung der referenziellen Integrität, Eindeutigkeits-Integritätsregeln oder Join-Histogramme zum Berechnen von Selektivitätsschätzungen verwenden. Join-Histogramme werden für ein Prädikat der Form `T.X=R.X` aus den verfügbaren statistischen Daten der Spalten "T.X" und "R.X" berechnet.
- **Spalte-Spalte** Wenn einem Join keine Integritätsregeln zur Erhaltung der referenziellen Integrität, Eindeutigkeits-Integritätsregeln oder Join-Histogramme als Selektivitätsquellen zur Verfügung stehen, kann der Optimierer die geschätzte Anzahl von Zeilen in der Join-Ergebnismenge dividiert durch die Anzahl der Zeilen im kartesischen Produkt der beiden Tabellen verwenden.
- **Spalte** Der Optimierer kann den Durchschnitt aller in der Spaltenstatistiken gespeicherten Werte verwenden.

Beispiel: Die Selektivität des Prädikats `DepartmentName = expression` kann unter Verwendung des Durchschnitts berechnet werden, wenn *expression* keine Konstante ist.

- **Index** Der Optimierer kann Indizes zum Berechnen von Selektivitätsschätzungen untersuchen. Im Allgemeinen wird ein Index für Selektivitätsschätzungen verwendet, wenn keine anderen Quellen von Selektivitätsschätzungen, wie zum Beispiel Spaltenstatistiken, benutzt werden können.

Für das Prädikat `DepartmentName = 'Sales'` kann der Optimierer z.B. einen Index benutzen, der für die Spalte "DepartmentName" definiert wurde, um die Anzahl der Zeilen mit dem Wert "Sales" zu schätzen.

- **Benutzer** Der Optimierer kann vom Benutzer bereitgestellte Selektivitätsschätzungen verwenden, vorausgesetzt die user_estimates-Datenbankoption ist nicht auf "Disabled" gesetzt.
- **Annahme** Der Optimierer kann die beste Annahme für die Berechnung von Selektivitätsschätzungen heranziehen, wenn kein relevanter Index benutzt werden kann, keine Statistiken für die referenzierten Spalten gesammelt wurden oder das Prädikat ein komplexes Prädikat ist. In diesem Fall werden systemeigene Näherungsannahmen für jeden Typ von Prädikat definiert.
- **Berechnet** Beispiel: Für ein sehr komplexes Prädikat wurde die Selektivitätsschätzung auf 100 % und die Selektivitätsquelle auf "Berechnet" festgelegt, als die Selektivitätsschätzung berechnet wurde, zum Beispiel durch Multiplikation oder Addition der Selektivität.
- **Immer** Wenn ein Prädikat immer TRUE ist, ist die Selektivitätsquelle "Immer". Das Prädikat 1=1 ist z.B. immer TRUE.
- **Kombiniert** Wenn die Selektivitätsschätzung durch die Kombination mehrerer der oben genannten Quellen berechnet wird, ist die Selektivitätsquelle "Kombiniert".
- **Beschränkt** Wenn SQL Anywhere eine obere bzw. untere Grenze für die Selektivitätsschätzung festgelegt hat, ist die Selektivitätsquelle "Beschränkt". Beispiel: Grenzwerte sollen sicherstellen, dass eine Schätzung nicht höher als 100 % liegt oder dass die Selektivität nicht weniger als 0 % beträgt.

Siehe auch

- „ESTIMATE-Funktion [Verschiedene]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ESTIMATE_SOURCE-Funktion [Verschiedene]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Selektivitätsinformationen im grafischen Plan“ auf Seite 353
- „Tipp: Explizite Selektivitätsschätzungen sparsam einsetzen“ auf Seite 234
- „sa_get_histogram-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „INDEX_ESTIMATE-Funktion [Verschiedene]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „EXPERIENCE_ESTIMATE-Funktion [Verschiedene]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Tipp: Explizite Selektivitätsschätzungen sparsam einsetzen“ auf Seite 234
- „user_estimates-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Plan-Caching

Normalerweise wählt der Optimierer einen Ausführungsplan für eine Abfrage jedesmal, wenn diese Abfrage ausgeführt wird. Durch die Optimierung zur Ausführungszeit kann der Optimierer den Plan aufgrund des gegenwärtigen Systemzustands sowie der Werte der aktuellen Selektivitätsschätzungen und Schätzungen basierend auf den Werten der Hostvariablen auswählen. Bei Abfragen, die häufig ausgeführt werden, können die Kosten der Abfrageoptimierung die Vorteile einer Optimierung zur Ausführungszeit aufwiegen. Um die Kosten für das wiederholte Optimieren dieser Anweisungen zu reduzieren, zieht der SQL Anywhere-Server für die folgenden Elemente ein Ablegen der Pläne im Cache in Betracht:

- Alle Anweisungen, die in gespeicherten Prozeduren, benutzerdefinierten Funktionen und Triggern ausgeführt werden.
- SELECT-, INSERT-, UPDATE- oder DELETE-Anweisungen, die sich zum Umgehen der Optimierung eignen.

Bei INSERT-Anweisungen sind lediglich INSERT...VALUES-Anweisungen für die Aufnahme in den Cache geeignet. INSERT...ON EXISTING-Anweisungen sind nicht für die Aufnahme in den Cache geeignet.

Bei UPDATE- und DELETE-Anweisungen muss die WHERE-Klausel verwendet werden und Suchbedingungen enthalten, die den Primärschlüssel zur Identifizierung einer Zeile verwenden. Wenn das Plan-Caching gewünscht ist, sind keine zusätzlichen Suchbedingungen erlaubt. Außerdem wird eine UPDATE-Anweisung vom Caching ausgeschlossen, wenn sie eine SET-Klausel mit einer Variablenzuweisung enthält.

Wenn eine dieser Anweisungen mehrmals von einer Verbindung ausgeführt wurde, erstellt der Optimierer einen wieder verwendbaren Plan für die Anweisung, ohne die Hostvariablenwerte zu kennen. Der wieder verwendbare Plan verursacht möglicherweise höhere Kosten, weil Hostvariablenwerte nicht für Selektivitätsschätzungen oder semantische Abfragetransformationen verwendet werden können. Falls der wieder verwendbare Plan die gleiche Struktur hat wie die Pläne, die bei den vorherigen Ausführungen der Anweisung erstellt wurden, nimmt der Datenbankserver den wieder verwendbaren Plan in den Plancache auf. Der Ausführungsplan wird nicht in den Cachespeicher geschrieben, wenn der Nutzen einer Optimierung bei jeder Ausführung die Einsparungen einer Umgehung der Optimierung übersteigt.

Falls ein Ausführungsplan eine materialisierte Ansicht verwendet, die nicht durch die Anweisung referenziert wurde, und die Option "materialized_view_optimization" auf einen anderen Wert als "Stale" gesetzt wurde, wird der Ausführungsplan nicht im Cache abgelegt und die Anweisung wird wieder optimiert, wenn die gespeicherte Prozedur, die benutzerdefinierte Funktion oder der Trigger das nächste Mal aufgerufen wird.

Der Plancache ist ein Cache pro Verbindung für die Datenstrukturen, die zur Ausführung eines Zugriffsplans verwendet werden. Für die Wiederverwendung des im Cache abgelegten Plans wird der Plan im Cache herausgesucht und wieder in seinen anfänglichen Zustand versetzt. Üblicherweise ist dies signifikant schneller als die Anweisung in allen Abfrageverarbeitungsphasen abzuarbeiten. Im Cache abgelegte Pläne können auf der Festplatte gespeichert werden, wenn sie selten benötigt werden, und erhöhen so nicht das Caching. Der Optimierer optimiert Abfragen regelmäßig neu, um zu überprüfen, ob der im Cache abgelegte Plan immer noch effizient ist.

Die maximale Anzahl der im Cache abgelegten Pläne wird mit der Option "max_plans_cached" festgelegt. Standardwert ist "20". Um das Plan-Caching zu deaktivieren, setzen Sie diese Option auf 0.

Sie können die Statistik "QueryCachedPlans" verwenden, um zu zeigen, wie viele Abfrageausführungspläne zurzeit im Cache sind. Diese Eigenschaft können Sie mithilfe der Funktion CONNECTION_PROPERTY abrufen, um festzustellen, wie viele Abfrageausführungspläne für eine bestimmte Verbindung im Cache abgelegt sind. Mit der Funktion DB_PROPERTY kann die Anzahl der Ausführungspläne, die für alle Verbindungen im Cache sind, berechnet werden. Diese Eigenschaft kann mit "QueryCachePages", "QueryOptimized", "QueryBypassed" und "QueryReused" kombiniert werden, um die beste Einstellung für die Option "max_plans_cached" zu ermitteln.

Mit der Datenbank- oder QueryCachePages-Verbindungseigenschaft können Sie die Anzahl der verwendeten Seiten für das Caching von Ausführungsplänen ermitteln. Diese Seiten belegen zwar Speicherplatz in der temporären Datei, sind aber nicht unbedingt speicherresident.

Für Abfragen mit langen Ausführungszeiten wird kein Caching der Ausführungspläne durchgeführt, weil die Vorteile durch das Vermeiden der Abfrageoptimierung im Vergleich zu den Gesamtkosten der

Abfrage klein sind. Außerdem versucht der Datenbankserver nicht, wiederverwendbare Abfragepläne für Abfragen zu rekonstruieren, die sehr empfindlich auf die Werte ihrer Hostvariablen reagieren.

Siehe auch

- „Eignung für ein Überspringen der Abfrageverarbeitungsphase“ auf Seite 331
- „Materialisierte Ansichten“ auf Seite 56
- „materialized_view_optimization-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „DB_PROPERTY-Funktion [System]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CONNECTION_PROPERTY-Funktion [System]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Fortgeschrittene Aufgaben: Phasen der Abfrageverarbeitung“ auf Seite 329
- „Liste der Verbindungseigenschaften“ [*SQL Anywhere Server - Datenbankadministration*]
- „max_plans_cached-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Caching von Unterabfragen und Funktionen

Wenn SQL Anywhere eine Unterabfrage verarbeitet, stellt er das Ergebnis in den Cache. Dieses Caching erfolgt pro Anforderung, wobei die im Cache abgelegten Ergebnisse nie von gleichzeitigen Anforderungen bzw. Verbindungen gemeinsam benutzt werden. Wenn SQL Anywhere die Unterabfrage für dieselbe Menge von Korrelationswerten neu auflösen muss, kann das Ergebnis einfach aus dem Cache abgerufen werden. Auf diese Art vermeidet SQL Anywhere viele Wiederholungen und überflüssige Berechnungen. Wenn die Anforderung abgeschlossen ist (der Cursor der Abfrage ist geschlossen), gibt SQL Anywhere die im Cache abgelegten Werte frei.

Während die Verarbeitung einer Abfrage fortgesetzt wird, überwacht SQL Anywhere die Häufigkeit, mit der Unterabfragenwerte im Cache wieder verwendet werden. Wenn sich die Werte der korrelierenden Variablen selten wiederholen, muss SQL Anywhere die meisten Werte nur einmal berechnen. In diesem Fall erkennt SQL Anywhere, dass es effizienter ist, wenige mehrfach vorhandene Werte erneut zu berechnen, als zahlreiche Einträge in den Cache zu legen, die nur einmal vorkommen. Daher legt der Datenbankserver die Unterabfrage für den Rest der Anweisung nicht mehr im Cachespeicher ab und fährt fort, die Unterabfrage für jede einzelne Zeile im äußeren Abfrageblock neu aufzulösen.

SQL Anywhere verwendet auch keinen Cache, wenn die Größe der abhängigen Spalte mehr als 255 Byte beträgt. In diesen Fällen ist es möglicherweise sinnvoll, Ihre Abfrage umzuschreiben oder eine weitere Spalte Ihrer Tabelle hinzuzufügen, um solche Vorgänge effizienter zu machen.

Caching von Funktionen

Einige integrierte und benutzerdefinierte Funktionen werden auf dieselbe Art und Weise im Cache abgelegt wie die Ergebnisse von Unterabfragen. Das kann zu einer beachtlichen Verbesserung für kostenträchtige Funktionen führen, die während der Abfrageverarbeitung mit denselben Parametern aufgerufen werden. Jedoch kann dies auch bedeuten, dass eine Funktion seltener aufgerufen wird, als anderenfalls zu erwarten wäre.

Damit eine Funktion im Cache abgelegt werden kann, muss sie zwei Bedingungen erfüllen:

- Sie muss jederzeit dasselbe Ergebnis für eine gegebene Menge von Parametern liefern.
- Sie darf keine Auswirkungen auf die zugrunde liegenden Daten haben.

Funktionen, die diese Bedingungen erfüllen, werden als **deterministische** oder **idempotente** Funktionen bezeichnet. SQL Anywhere behandelt alle benutzerdefinierten Funktionen als deterministisch (außer sie werden zur Erstellungszeit ausdrücklich als NOT DETERMINISTIC deklariert). Das heißt, der Datenbankserver nimmt an, dass zwei aufeinander folgende Aufrufe derselben Funktion mit denselben Parametern das gleiche Ergebnis zurückgeben und keine unerwünschten Nebeneffekte auf die Semantik der Abfrage haben.

Mit wenigen Ausnahmen werden integrierte Funktionen als deterministisch behandelt. Die Funktionen RAND, NEWID und GET_IDENTITY werden als nicht deterministisch behandelt und ihre Ergebnisse werden nicht im Cache gespeichert.

Siehe auch

- „CREATE FUNCTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Optimierungen während der Abfrageverarbeitung

In der Neuschreibungsphase für Abfragen führt SQL Anywhere semantische Transformationen durch, um nach effizienteren und sinnvolleren Darstellungen der Abfrage zu suchen. Da die Abfrage möglicherweise in eine semantisch gleichwertige Abfrage umgeschrieben wird, kann sich der Plan erheblich von Ihrer ursprünglichen Abfrage unterscheiden. Übliche Änderungen sind z.B. folgende:

- Unnötige DISTINCT-Bedingungen eliminieren
- Unterabfragen entschachteln
- Prädikat-Pushdown in UNION- oder GROUP-Ansichten und abgeleiteten Tabellen durchführen
- OR- und IN-Listen-Prädikate optimieren
- LIKE-Prädikate optimieren
- Outer-Joins zu Inner-Joins konvertieren
- Outer-Joins und Inner-Joins eliminieren
- Nutzbare Bedingungen durch Prädikat-Inferenz suchen
- Unnötige Konvertierung der Groß- und Kleinschreibung eliminieren
- Unterabfragen als EXISTS-Prädikate umschreiben
- IN-Prädikate in sargable- (als Suchargument nutzbarer), für partielle Index-Scans verwendbarer Form aus OR-Prädikaten ableiten, die nicht in AND-Prädikate umgewandelt werden können

Hinweis

Einige Neuschreibungsoptimierungen für Abfragen können nicht im Hauptabfrageblock durchgeführt werden, wenn der Cursor aktualisierbar ist. Deklarieren Sie den Cursor als schreibgeschützt, um die Optimierungen nutzen zu können.

Einige Neuschreibungsoptimierungen, die während der Neuschreibungsphase für Abfragen durchgeführt werden, sind in den Ergebnissen der Funktion REWRITE einsehbar.

Siehe auch

- „Cursortypen“ [SQL Anywhere Server - Programmierung]
- „DECLARE CURSOR-Anweisung [ESQL] [SP]“ [SQL Anywhere Server - SQL-Referenzhandbuch]
- „REWRITE-Funktion [Verschiedene]“ [SQL Anywhere Server - SQL-Referenzhandbuch]

Fortgeschrittene Aufgaben: Abfrageausführungspläne

Ein Ausführungsplan besteht aus einer Gruppe von Schritten, die der Datenbankserver durchführt, um auf Informationen in der Datenbank zuzugreifen, die in Verbindung mit einer Anweisung stehen. Der Ausführungsplan für eine Anweisung kann gespeichert und angezeigt werden, unabhängig davon, ob er gerade optimiert wurde, ob er den Optimierer umgangen hat oder ob der Plan von vorherigen Ausführungen im Cache abgelegt wurde. Ein Abfrageausführungsplan entspricht möglicherweise nicht exakt der Syntax, die in der ursprünglichen Anweisung verwendet wurde, und verwendet möglicherweise materialisierte Ansichten der Basistabellen, die in der Abfrage explizit angegeben wurden. Die im Ausführungsplan beschriebenen Vorgänge sind allerdings semantisch äquivalent mit der ursprünglichen Abfrage.

Sie können den Ausführungsplan in Interactive SQL oder mithilfe von SQL-Funktionen anzeigen. Der Ausführungsplan kann in verschiedenen Formaten abgerufen werden:

- Kurzer Textplan
- Ausführlicher Textplan
- Grafischer Plan
- Grafischer Plan mit Stammstatistiken
- Grafischer Plan mit allen Statistiken
- UltraLite (kurz, ausführlich oder grafisch)

In Abfrageausführungsplänen gibt es zwei Darstellungsarten für den Text, eine kurze und eine ausführliche Version. Verwenden Sie die SQL-Funktionen, um auf den Textplan zuzugreifen. Außerdem gibt es eine grafische Version des Plans. Sie können auch Pläne für SQL-Abfragen mit einem bestimmten Cursortyp beziehen, indem Sie die Funktionen GRAPHICAL_PLAN und EXPLANATION verwenden.

Weitere Hinweise zur Abfrageverarbeitung in SQL Anywhere finden Sie im Whitepaper *"Query Processing Based on SQL Anywhere 12.0.1 Architecture"* unter <http://www.sybase.com/detail?id=1096047>.

Siehe auch

- „GRAPHICAL_PLAN-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „EXPLANATION-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Fortgeschrittene Aufgaben: Phasen der Abfrageverarbeitung“ auf Seite 329
- „Anzeigen eines grafischen Plans“ auf Seite 355
- „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ auf Seite 341
- „Grafische Pläne“ auf Seite 347

Siehe auch

- „Kurzer Textplan“ auf Seite 342
- „Ausführlicher Textplan“ auf Seite 343
- „Grafische Pläne“ auf Seite 347

Kurzer Textplan

Der kurze Textplan ist dann hilfreich, wenn Sie Pläne schnell miteinander vergleichen wollen. Von allen Planformaten bietet er am wenigsten Informationen, jedoch werden diese in einer einzigen Zeile dargestellt.

Im folgenden Beispiel beginnt der Plan mit `Work[Sort`, da die gesamte Ergebnismenge durch die `ORDER BY`-Klausel sortiert wurde. Der Primärschlüsselindex "CustomersKey" greift auf die Customers-Tabelle zu. Ein Index-Scan wird verwendet, um die Suchbedingung zu erfüllen, da die Spalte Customers.ID ein Primärschlüssel ist. Die Abkürzung JNL gibt an, dass der Optimierer einen Merge-Join gewählt hat, um den Join zwischen "Customers" und "SalesOrders" zu verarbeiten. Schließlich wird mit dem Fremdschlüsselindex "FK_CustomerID_ID" auf die Tabelle "SalesOrders" zugegriffen, um Zeilen zu finden, für die "CustomerID" in der Tabelle "Customers" kleiner als 100 ist.

```
SELECT EXPLANATION ('SELECT GivenName, Surname, OrderDate
FROM Customers JOIN SalesOrders
WHERE CustomerID < 100
ORDER BY OrderDate');
```

```
Work[ Sort[ Customers<CustomersKey> JNL
SalesOrders<FK_CustomerID_ID> ] ]
```

Doppelpunkte trennen Join-Strategien

Die folgende Anweisung enthält zwei **Abfrageblöcke**: Den äußeren SELECT-Block, der die Tabellen "SalesOrders" und "SalesOrderItems" referenziert, und die Unterabfrage, die aus der Products-Tabelle auswählt.

```
SELECT EXPLANATION ('SELECT *
FROM SalesOrders AS o
KEY JOIN SalesOrderItems AS i
WHERE EXISTS
( SELECT *
FROM Products p
WHERE p.ID = 300 )');
```

```
o<seq> JNL i<FK_ID_ID> : p<ProductsKey>
```


Die Join-Strategien der verschiedenen Abfrageblöcke werden durch Doppelpunkte getrennt. Kurze Pläne führen die Join-Strategie für den ersten Block stets zuerst an. Die Join-Strategien für die anderen Abfrageblöcke folgen. Die Reihenfolge der Join-Strategien für diese anderen Abfrageblöcke entspricht möglicherweise weder der Reihenfolge der Abfrageblöcke in Ihrer Anweisung noch der Reihenfolge, in der sie ausgeführt werden.

Siehe auch

- „Komponenten von Ausführungsplänen“ auf Seite 356

Ausführlicher Textplan

Der ausführliche Textplan enthält mehr Informationen als der kurze Textplan und stellt diese so dar, dass sie einfach gedruckt bzw. angezeigt werden können, ohne dass Sie blättern müssen. Ausführliche Pläne enthalten bestimmte Informationen, z.B. den im Cache abgelegten Plan für eine Anweisung, und die bei der Anwendungsprofilerstellung verwendeten Pläne enthalten zusätzliche Informationen darüber, wie eine Abfrage optimiert wurde und welche Prädikate in einem partiellen Index-Scan verwendet wurden.

Beispiel 1

In diesem Beispiel basiert der ausführliche Textplan auf der folgenden Anweisung:

```
SELECT PLAN ('SELECT GivenName, Surname, OrderDate, Region, Country
FROM Customers JOIN SalesOrders ON ( SalesOrders.CustomerID = Customers.ID )
WHERE CustomerID < 100 AND ( Region LIKE ''Eastern''
      OR Country LIKE ''Canada'' )
ORDER BY OrderDate');
```

Der ausführliche Textplan lautet wie folgt:

```
( Plan [ Total Cost Estimate: 6.46e-005, Costed Best Plans: 1, Costed Plans:
10, Optimization Time: 0.0011462,
Estimated Cache Pages: 348 ]
  ( WorkTable
    ( Sort
      ( NestedLoopsJoin
        ( IndexScan Customers CustomersKey[ Customers.ID < 100 : 0.0001%
Index | Bounded ] )
        ( IndexScan SalesOrders FK_CustomerID_ID[ Customers.ID =
SalesOrders.CustomerID : 0.79365% Statistics ]
          [ ( SalesOrders.CustomerID < 100 : 0.0001% Index | Bounded )
            AND ( ( ((Customers.Country LIKE 'Canada' : 100% Computed)
              AND (Customers.Country = 'Canada' : 5% Guess))
              OR ((SalesOrders.Region LIKE 'Eastern' : 100% Computed)
                AND (SalesOrders.Region = 'Eastern' : 5% Guess)) ) : 100%
          Guess ) ] )
        )
      )
    )
  )
)
```

Das Wort "Plan" gibt den Start eines Abfrageblocks an. "Total Cost Estimate" ist die vom Optimierer geschätzte Zeit in Millisekunden für die Ausführung des Plans. "Costed Best Plans", "Costed Plans" und "Optimization Time" sind Statistiken des Optimierungsprozesses, während "Estimated Cache Pages" für die geschätzte aktuelle Cachegröße steht, die zum Verarbeiten der Anweisung verfügbar ist.

Der Plan gibt an, dass die Ergebnisse sortiert sind und dass ein Nested-Loops-Join benutzt wird. In der gleichen Zeile wie der Join-Operator befinden sich die Join-Bedingung und deren Selektivitätsschätzung (die für alle vom Join-Operator erzeugten Zeilen ausgewertet wird). Die IndexScan-Zeilen geben an, dass über die Indizes "CustomersKey" und "FK_CustomerId_ID" auf die Tabellen "Customers" und "SalesOrders" zugegriffen wird.

Beispiel 2

Wenn die folgende Anweisung in einer Prozedur, einem Trigger oder einer Funktion verwendet wird und der Plan für die Anweisung im Cache abgelegt und fünf Mal wiederverwendet wurde, enthält der ausführliche Textplan die Zeichenfolge [R: 5], um anzuzeigen, dass die Anweisung wiederverwendbar ist und nach dem Ablegen im Cache fünf Mal verwendet wurde. Der in der Anweisung verwendete parm1-Parameter hat in diesem Plan einen unbekannten Wert.

```
UPDATE Account SET Account.A = 10 WHERE Account.B =parm1

( Update [ Total Cost Estimate: 1e-006, Costed Best Plans: 1, Costed Plans:
2, Carver pages: 0,
Estimated Cache Pages: 46768 ] [ R: 5 ]
  ( Keyset
    ( TableScan ( Account ) ) [ Account.B = parm1 : 0.39216% Column ]
  )
)
```

Wenn der Plan für die gleiche Anweisung noch nicht im Cache abgelegt wurde, enthält der ausführliche Textplan den Wert für den parm1-Parameter (z.B. 10), was bedeutet, dass der Plan mit dem Wert dieses Parameters optimiert wurde.

```
( Update [ Total Cost Estimate: 1e-006, Costed Best Plans: 1, Costed Plans:
2, Carver pages: 0,
Estimated Cache Pages: 46768 ]
  ( Keyset
    ( TableScan ( Account ) ) [ Account.B = parm1 [ 10 ] : 0.001%
Statistics ]
  )
)
```

Siehe auch

- „Komponenten von Ausführungsplänen“ auf Seite 356

Anzeigen eines kurzen Textplans

Mit einem kurzen Textplan können Sie eine Kurzversion des Abfrageausführungsplans anzeigen.

Voraussetzungen

Sie müssen entweder Eigentümer der Objekte sein, für die die Funktion ausgeführt wird, oder die geeigneten SELECT-, UPDATE-, DELETE- oder INSERT-Privilegien für die Objekte haben.

Aufgabe

1. Stellen Sie eine Verbindung mit einer Datenbank her.
2. Führen Sie die EXPLANATION-Funktion aus.

Ergebnisse

Der kurze Textplan erscheint im Fensterausschnitt **Ergebnisse** in Interactive SQL.

Beispiel

In diesem Beispiel basiert der kurze Textplan auf der folgenden Anweisung:

```
SELECT EXPLANATION ('SELECT GivenName, Surname, OrderDate
FROM GROUPO.Customers JOIN GROUPO.SalesOrders
WHERE CustomerID < 100
ORDER BY OrderDate');
```

Der kurze Textplan lautet wie folgt:

```
Work[ Sort[ Customers<CustomersKey> JNL
SalesOrders<FK_CustomerID_ID> ] ]
```

Der kurze Textplan beginnt mit Work[Sort, da die gesamte Ergebnismenge durch die ORDER BY-Klausel sortiert wird. Der Primärschlüsselindex "CustomersKey" greift auf die Customers-Tabelle zu. Ein Index-Scan wird verwendet, um die Suchbedingung zu erfüllen, da die Spalte Customers.ID ein Primärschlüssel ist. Die Abkürzung JNL gibt an, dass der Optimierer einen Merge-Join gewählt hat, um den Join zwischen "Customers" und "SalesOrders" zu verarbeiten. Schließlich wird mit dem Fremdschlüsselindex "FK_CustomerID_ID" auf die Tabelle "SalesOrders" zugegriffen, um Zeilen zu finden, für die "CustomerID" in der Tabelle "Customers" kleiner als 100 ist.

Siehe auch

- „EXPLANATION-Funktion [Verschiedene]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Komponenten von Ausführungsplänen“ auf Seite 356
- „Kurzer Textplan“ auf Seite 342
- „Anzeigen eines ausführlichen Textplans“ auf Seite 345

Anzeigen eines ausführlichen Textplans

Ein ausführlicher Textplan enthält mehr Informationen als der kurze Textplan, einschließlich des im Cache gespeicherten Plans für eine Anweisung, und stellt diese so dar, dass sie einfach gedruckt bzw. angezeigt werden können, ohne dass Sie blättern müssen.

Voraussetzungen

Sie müssen entweder Eigentümer der Objekte sein, für die die Funktion ausgeführt wird, oder die geeigneten SELECT-, UPDATE-, DELETE- oder INSERT-Privilegien für die Objekte haben.

Aufgabe

1. Stellen Sie eine Verbindung mit einer Datenbank her.
2. Führen Sie die PLAN-Funktion aus.

Ergebnisse

Der ausführliche Textplan erscheint im Fensterausschnitt **Ergebnisse** in Interactive SQL.

Beispiel

In diesem Beispiel basiert der ausführliche Textplan auf der folgenden Anweisung:

```
SELECT PLAN ('SELECT GivenName, Surname, OrderDate, Region, Country
FROM GROUPO.Customers JOIN GROUPO.SalesOrders ON ( SalesOrders.CustomerID =
Customers.ID )
WHERE CustomerID < 100 AND ( Region LIKE ''Eastern''
OR Country LIKE ''Canada'' )
ORDER BY OrderDate');
```

Der ausführliche Textplan lautet wie folgt:

```
( Plan [ Total Cost Estimate: 6.46e-005, Costed Best Plans: 1, Costed Plans:
10, Optimization Time: 0.0011462,
Estimated Cache Pages: 348 ]
  ( WorkTable
    ( Sort
      ( NestedLoopsJoin
        ( IndexScan Customers CustomersKey[ Customers.ID < 100 : 0.0001%
Index | Bounded ] )
        ( IndexScan SalesOrders FK_CustomerID_ID[ Customers.ID =
SalesOrders.CustomerID : 0.79365% Statistics ]
          [ ( SalesOrders.CustomerID < 100 : 0.0001% Index | Bounded )
AND ( ( ((Customers.Country LIKE 'Canada' : 100% Computed)
AND (Customers.Country = 'Canada' : 5% Guess))
OR ((SalesOrders.Region LIKE 'Eastern' : 100% Computed)
AND (SalesOrders.Region = 'Eastern' : 5% Guess)) ) : 100%
Guess ) ] )
        )
      )
    )
  )
)
```

Das Wort "Plan" gibt den Start eines Abfrageblocks an. "Total Cost Estimate" ist die vom Optimierer geschätzte Zeit in Millisekunden für die Ausführung des Plans. "Costed Best Plans", "Costed Plans" und "Optimization Time" sind Statistiken des Optimierungsprozesses, während "Estimated Cache Pages" für die geschätzte aktuelle Cachegröße steht, die zum Verarbeiten der Anweisung verfügbar ist.

Der Plan gibt an, dass die Ergebnisse sortiert sind und dass ein Nested-Loops-Join benutzt wird. In der gleichen Zeile wie der Join-Operator befinden sich die Join-Bedingung und deren Selektivitätsschätzung (die für alle vom Join-Operator erzeugten Zeilen ausgewertet wird). Die IndexScan-Zeilen geben an, dass über die Indizes "CustomersKey" und "FK_CustomerId_ID" auf die Tabellen "Customers" und "SalesOrders" zugegriffen wird.

Siehe auch

- „Komponenten von Ausführungsplänen“ auf Seite 356
- „PLAN-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Ausführlicher Textplan“ auf Seite 343
- „Anzeigen eines kurzen Textplans“ auf Seite 344

Grafische Pläne

Die Funktion "Grafischer Plan" in Interactive SQL zeigt den Ausführungsplan für eine Abfrage im Fenster **Plananzeige** an. Der Ausführungsplan besteht aus einer Baumstruktur mit relationalen Algebra-Operatoren, die, beginnend an den Blättern des Baums, die Basiseingaben der Abfrage verwerten (normalerweise Zeilen aus einer Tabelle) und die Zeilen von unten nach oben verarbeiten, sodass die Wurzeln des Baums das Endergebnis enthalten. Die Knoten in dieser Baumstruktur entsprechen spezifischen Algebra-Operatoren, auch wenn nicht jede Abfrageberechnung, die vom Datenbankserver durchgeführt wurde, durch Knoten dargestellt wird. Die Auswirkungen einer Unterabfrage- oder Funktions-Cachebenutzung beispielsweise werden in einem grafischen Plan nicht direkt dargestellt.

Die Knoten in einem grafischen Plan haben unterschiedliche Formen, die den Typ des durchgeführten Vorgangs anzeigen.

- Sechsecke symbolisieren Vorgänge, die Daten materialisieren.
- Trapeze symbolisieren Index-Scans.
- Rechtecke mit eckigen Ecken symbolisieren Table-Scans.
- Rechtecke mit runden Ecken symbolisieren Vorgänge, die oben noch nicht aufgelistet sind.

Sie können einen grafischen Plan verwenden, um Performanceprobleme bei bestimmten Abfragen zu diagnostizieren. Aufgrund der Informationen im Plan können Sie beispielsweise entscheiden, ob eine Tabelle einen Index benötigt, um die Performance dieser spezifischen Abfrage zu verbessern. Sie können einen grafischen Plan einer Abfrage für die zukünftige Verwendung speichern, indem Sie in der **Plananzeige** auf die Schaltfläche **Speichern** klicken. Grafische Pläne werden in SQL Anywhere mit der Erweiterung *.saplan* gespeichert.

Mögliche Performanceprobleme werden im grafischen Plan durch dicke Linien und rote Umrandungen gekennzeichnet. Beispiel:

- Dickere Linien zwischen Knoten eines Planes zeigen eine entsprechende Zunahme in der Anzahl der verarbeiteten Zeilen an. Das Vorhandensein einer dicken Linie über einem Table-Scan zeigt an, dass die Erstellung eines Index möglicherweise erforderlich ist.
- Die rote Umrandung eines Knotens zeigt an, dass der Vorgang im Vergleich zu anderen Vorgängen im Ausführungsplan kostenträchtig gewesen ist.

Knotenformen und andere grafische Komponenten des Plans können in Interactive SQL angepasst werden.

Sie können entweder einen grafischen Plan, einen grafischen Plan mit Zusammenfassung oder einen grafischen Plan mit detaillierten Statistiken anzeigen. Bei allen drei Plänen ist es möglich, die Teile des

Plans anzuzeigen, die als besonders kostenträchtig eingeschätzt werden. Das Generieren eines grafischen Plans mit Statistik ist kostenträchtiger, weil er tatsächliche Abfrageausführungsstatistiken liefert, wie sie der Datenbankserver während der Ausführung der Abfrage protokolliert hat. Grafische Pläne mit Statistik ermöglichen einen direkten Vergleich zwischen Schätzungen, die vom Abfrageoptimierer beim Erstellen des Zugriffsplans verwendet wurden, und den tatsächlich während der Ausführung protokollierten Statistiken. Beachten Sie jedoch, dass der Optimierer häufig nicht in der Lage ist, die Kosten einer Abfrage präzise einzuschätzen. Daher sollten Sie mit Abweichungen zwischen den geschätzten und den tatsächlichen Kosten rechnen.

Siehe auch

- „Anzeigen eines grafischen Plans“ auf Seite 355
- „Anwendungsprofilerstellung“ auf Seite 156
- „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ auf Seite 341
- „Grafische Pläne anpassen“ [*SQL Anywhere Server - Datenbankadministration*]

Grafischer Plan mit Statistiken

Der grafische Plan bietet mehr Informationen als der kurze bzw. der ausführliche Textplan. Die Generierung des grafischen Plans mit Statistik ist zwar kostenträchtiger, zeigt aber dafür die Statistiken für die Abfrageausführung an, wie sie bei Ausführung der Abfrage vom Datenbankserver aufgezeichnet wurden. Dies ermöglicht einen direkten Vergleich zwischen den vom Optimierer bei der Erstellung des Zugriffsplans verwendeten Schätzungen und den tatsächlichen, während der Ausführung erfassten Statistiken. Signifikante Unterschiede zwischen den tatsächlichen und den geschätzten Statistiken zeigen an, dass der Optimierer nicht genügend Informationen hat, um die Kosten der Abfrage korrekt zu schätzen, was zu einem ineffizienten Ausführungsplan führen kann.

Um einen grafischen Plan mit Statistik zu generieren, muss der Datenbankserver die Anweisung ausführen. Das Generieren eines grafischen Plans für Anweisungen mit langen Laufzeiten kann recht lange dauern. Wenn die Anweisung ein UPDATE, INSERT oder DELETE ist, wird nur der schreibgeschützte Teil der Anweisung ausgeführt. Es werden keine Tabellenmanipulationen durchgeführt. Wenn eine Anweisung allerdings benutzerdefinierte Funktionen enthält, werden sie als Teil der Abfrage ausgeführt. Wenn benutzerdefinierte Funktionen Nebenwirkungen haben (z.B. Zeilen ändern, Tabellen erstellen, Nachrichten an die Konsole senden etc.), werden diese Änderungen durchgeführt, wenn der grafische Plan mit Statistik abgerufen wird. In manchen Fällen können Sie diese Nebenwirkungen auffangen, indem Sie eine ROLLBACK-Anweisung ausgeben, nachdem Sie den grafischen Plan mit Statistik abgerufen haben.

Siehe auch

- „ROLLBACK-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Performanceanalyse mithilfe des grafischen Plans mit Statistik

Sie können den grafischen Plan mit Statistik verwenden, um Datenbank-Performanceprobleme zu identifizieren. Detaillierte Feldbeschreibungen des grafischen Plans mit Statistik finden Sie unter „Komponenten von Ausführungsplänen“ auf Seite 356.

Abfrageausführungsprobleme erkennen

Sie können Datenbankoptionen und andere globale Einstellungen anzeigen, die sich auf die Abfrageausführung bei den Stamm-Operatorknoten auswirken.

Selektivitätsperformance überprüfen

Die Selektivität eines Prädikats (bedingter Ausdruck) ist der Prozentsatz von Zeilen, die die Bedingung erfüllen. Die geschätzte Selektivität von Prädikaten liefert die Informationen, auf denen der Optimierer seine Kostenschätzungen aufbaut. Eine genaue Selektivitätsschätzung ist für die ordnungsgemäße Funktion des Optimierers ausschlaggebend. Wenn der Optimierer beispielsweise versehentlich ein Prädikat für höchst selektiv hält (z.B. eine Selektivität von 5 %), das Prädikat jedoch sehr viel weniger selektiv ist (z.B. 50 %), dann kann die Performance darunter leiden. Auch wenn Selektivitätsschätzungen nicht immer präzise sind, kann ein außergewöhnlich großer Fehler auf ein Problem hinweisen.

Wenn Sie feststellen, dass die Selektivitätsinformationen für einen Schlüsselabschnitt Ihrer Abfrage nicht stimmen, können Sie möglicherweise CREATE STATISTICS verwenden, um eine neue Reihe von Statistiken für die Spalte(n) zu generieren. In seltenen Fällen können Sie explizite Selektivitätsschätzungen vorgeben, obwohl dieses Vorgehen zu Problemen führen kann, wenn die Statistiken später aktualisiert werden.

Selektivitätsstatistiken werden nicht angezeigt, wenn die Abfrage als eine Bypass-Abfrage festgelegt ist.

Indikatoren für eine schlechte Selektivität treten an folgenden Stellen auf:

- **RowsReturned, tatsächlich und geschätzt** **RowsReturned** ist die Anzahl der Zeilen in der Ergebnismenge. Die **RowsReturned**-Statistik wird in der Tabelle für den Stammknoten an der Spitze der Struktur angezeigt. Wenn die geschätzte Zeilenanzahl deutlich von der tatsächlichen Zeilenanzahl abweicht, ist möglicherweise die Selektivität von den diesem Knoten oder dieser Unterstruktur zugeordneten Prädikaten falsch.
- **Prädikat-Selektivität, tatsächlich und geschätzt** Suchen Sie nach dem Subheader **Prädikate**, um die Prädikat-Selektivitäten einzusehen.

Wenn das Prädikat für eine Basisspalte gilt, für die kein Histogramm vorhanden ist, kann das Problem eventuell behoben werden, indem durch die Ausführung einer CREATE STATISTICS-Anweisung ein Histogramm erstellt wird.

Wenn der Selektivitätsfehler weiterhin Probleme bereitet, könnten Sie eine Benutzersелеktivitätsschätzung mit dem Prädikat im Abfragetext angeben.

- **Quelle der Schätzung** Die Quelle der Selektivitätsschätzungen ist ebenso unter dem Prädikat-Subheader im Fensterausschnitt **Statistik** aufgelistet.

Wenn die Quelle einer Prädikat-Selektivitätsschätzung Annahme (**Guess**) ist, hat der Optimierer keine verwertbaren Informationen, um die Filtermerkmale dieses Prädikats zu ermitteln, was möglicherweise auf ein Problem hinweist (wie z.B. ein fehlendes Histogramm). Falls die Quelle für die Schätzung **Index** und die Selektivitätsschätzung falsch ist, kann das Problem durch einen unausgeglichene Index verursacht sein. In so einem Fall kann es vorteilhaft sein, den Index mit der Anweisung REORGANIZE TABLE zu defragmentieren.

Cache-Performance überprüfen

Wenn die Anzahl der Cache-Lesevorgänge (**CacheRead**) und die der Cachetreffer (**CacheHits**) gleich sind, befinden sich alle Objekte, die bei dieser SQL-Anweisung verarbeitet werden, im Cache. Wenn es mehr Cache-Lesevorgänge als Cachetreffer gibt, zeigt dies, dass der Datenbankserver Tabellen- oder Indexseiten vom Plattenspeicher liest, weil sie sich noch nicht im Cache des Datenbankservers befinden. Unter bestimmten Umständen, z.B. bei Hash-Joins, ist dies zu erwarten. In anderen Fällen, wie z.B. bei Nested-Loops-Joins, kann ein schlechtes Cachetreffer-Verhältnis darauf hinweisen, dass nicht genügend Cache (Pufferpool) vorhanden ist, um die Abfrage effizient auszuführen. In diesem Fall ist es sinnvoll, die Cachegröße des Datenbankservers zu erhöhen.

Ineffiziente Indizes identifizieren

Häufig ist auch mit Abfrageausführungsplänen nicht eindeutig ersichtlich, ob die Performance durch Indizes gesteigert werden könnte. Einige der auf Scans basierenden Abfragevorgänge, die in SQL Anywhere eingesetzt werden, bieten eine hervorragende Performance für viele Abfragen, ohne dass Indizes verwendet werden.

Probleme mit der Datenfragmentierung identifizieren

Die tatsächlichen und geschätzten Werte **Runtime** und **FirstRowRunTime** sind in den Stammknoten-Statistiken enthalten. Nur **RunTime** wird im Abschnitt **Verzweigungsstatistiken** angezeigt, wenn für diesen Knoten ein Wert vorhanden ist.

Die Interpretation von **RunTime** hängt vom Statistikabschnitt ab, in dem der Wert angezeigt wird. In **Knotenstatistiken** ist **RunTime** die kumulative Zeit, die der Operator für das Ausführen *nur dieses Knotens* aufgewendet hat. In **Verzweigungsstatistiken** stellt **RunTime** die Gesamt-Ausführungszeit dar, die der Operator für das Ausführen der Unterstruktur unmittelbar unterhalb dieses Knotens aufgewendet hat. Daher sind **RunTime** und **FirstRowRunTime** bei den meisten Operatoren voneinander unabhängige Metriken, die separat analysiert werden sollten:

FirstRowRunTime ist die Zeit, die benötigt wird, um die erste Zeile des Zwischenergebnisses von diesem Knoten zu liefern.

Wenn **RunTime** eines Knotens für einen Table-Scan oder einen Index-Scan höher als erwartet ist, können Sie die Performance möglicherweise verbessern, indem Sie die REORGANIZE TABLE-Anweisung ausführen. Sie können die Systemprozeduren "sa_table_fragmentation()" und "sa_index_density()" verwenden, um festzustellen, ob die Tabelle oder der Index fragmentiert ist.

Siehe auch

- „REORGANIZE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE STATISTICS-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „So funktioniert der Optimierer“ auf Seite 333
- „Explizite Selektivitätsschätzungen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Tabellenfragmentierung reduzieren“ auf Seite 249
- „Komponenten von Ausführungsplänen“ auf Seite 356
- „Selektivitätsinformationen im grafischen Plan“ auf Seite 353
- „Tipp: Indizes effizient verwenden“ auf Seite 231
- „Indexberater“ auf Seite 164
- „Tipp: Performance durch den Einsatz des Cachespeichers steigern“ auf Seite 220

Detaillierte Knoteninformationen in grafischen Plänen

Um detaillierte Knoteninformationen im grafischen Plan anzuzeigen, klicken Sie in der grafischen Darstellung auf den Knoten im linken Fensterausschnitt. Details zu den Knoten werden rechts in den Fensterausschnitten **Details** und **Erweiterte Details** angezeigt. Im Fensterausschnitt **Details** können Statistiken für den Knoten in drei Hauptabschnitten angezeigt werden:

- **Knotenstatistiken**
- **Verzweigungsstatistiken**
- **Statistiken des Optimierers**

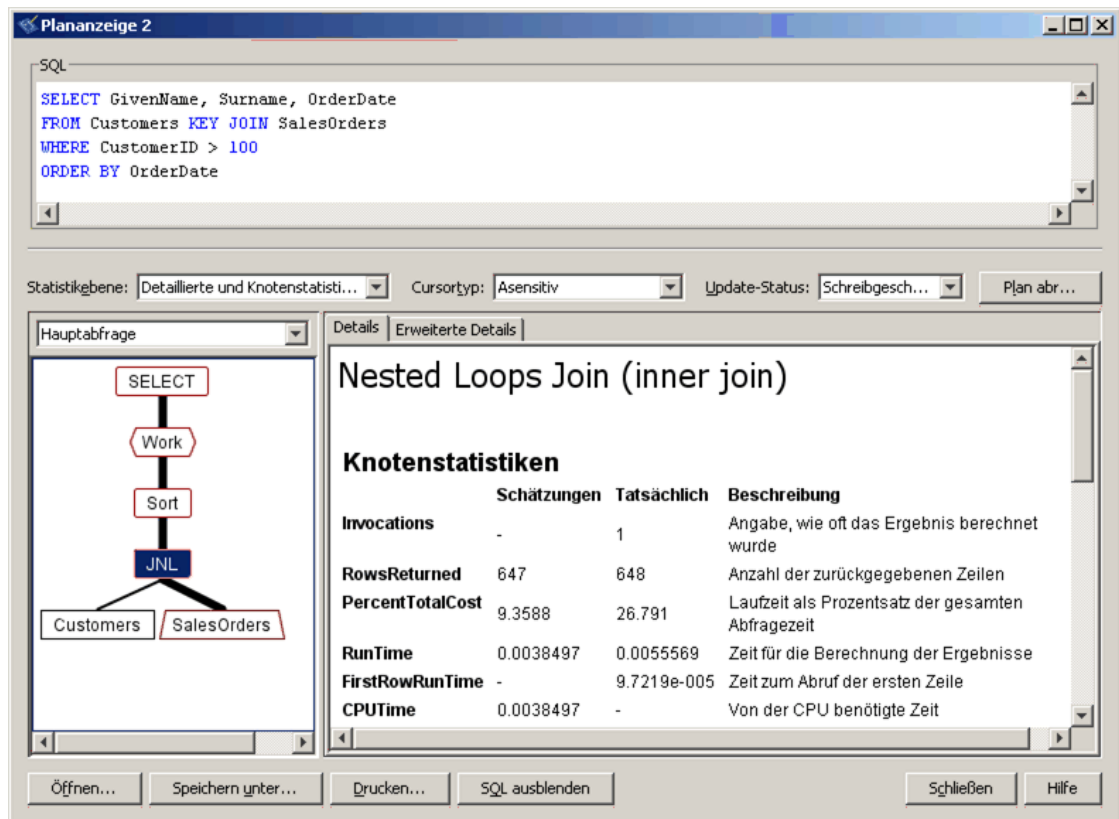
Knotenstatistiken sind Statistiken, die sich auf die Ausführung des spezifischen Knotens beziehen. Planknoten haben einen Fensterausschnitt **Details**, in dem geschätzte, tatsächliche und durchschnittliche Statistiken für den Operator angezeigt werden. Jeder Knoten kann mehrfach ausgeführt werden. Wenn beispielsweise ein Blattknoten rechts neben einem Nested Loops Join-Knoten (Join mit verschachtelten Schleifen) angezeigt wird, können Sie Zeilen mehrmals aus dem Operator des Blattknotens abrufen. In diesem Fall enthält der Fensterbereich **Details** für den Blattknoten (ein sequenzieller, Index- oder RowID-Scan-Knoten) sowohl Statistiken pro Aufruf (Mittelwert) als auch kumulative tatsächliche Statistiken über die Laufzeit.

Wenn ein Knoten kein Blattknoten ist, bezieht er Zwischenergebnisse von anderen Knoten und der Fensterausschnitt **Details** zeigt die geschätzten und tatsächlichen kumulativen Statistiken für die gesamte Unterstruktur dieses Knotens im Abschnitt **Verzweigungsstatistiken** an. Optimierer-Statistikinformationen, die die gesamte SQL-Anforderung umfassen, werden nur für Stammknoten ausgegeben. Statistikwerte des Optimierers beziehen sich spezifisch auf die Optimierung der Anweisung und enthalten Werte wie die Zielsetzung der Optimierung, die Optimierungsstufen-Einstellung, die Anzahl der berücksichtigten Pläne etc.

Sehen Sie sich die folgende Abfrage an, mit der die Kunden nach dem Datum ihrer Bestellung sortiert werden:

```
SELECT GROUPO.Customers.GivenName, GROUPO.Customers.Surname,
GROUPO.SalesOrders.OrderDate
FROM Customers KEY JOIN SalesOrders
WHERE CustomerID > 100
ORDER BY OrderDate
```

Im grafischen Plan für diese Abfrage ist der Knoten **Hash Join (JH)** gewählt und die im rechten Fensterausschnitt angezeigten Informationen beziehen sich nur auf diesen Knoten. Die Beschreibung unter **Prädikate** zeigt an, dass Customers.ID = SalesOrders.CustomerID : 0.79365% Statistics | Join das Prädikat ist, das auf den Hash Join-Knoten angewendet wird. Wenn Sie auf den Customers-Knoten klicken, wird unter "Scan-Prädikate" angegeben, dass Customers.ID > 100 : 100% Index; das Prädikat ist, das auf den Customers-Knoten angewendet wird.



Hinweis

Wenn Sie die Abfrage in dem untenstehenden Beispiel ausführen, erhalten Sie möglicherweise in der **Plananzeige** einen anderen Plan als den gezeigten. Viele Faktoren, wie z.B. Datenbankeinstellungen und zuletzt ausgeführte Abfragen, haben Einfluss darauf, welchen Plan der Optimierer wählt.

Die im Fensterabschnitt **Erweiterte Details** angezeigten Informationen hängen vom jeweiligen Operator ab. Bei Stammknoten enthält der Fensterausschnitt **Erweiterte Details** die Einstellungen, die für die Verbindungsoptionen galten, als die Abfrage optimiert wurde. Bei anderen Knotentypen kann der Fensterabschnitt **Erweiterte Details** Informationen darüber enthalten, welche Indizes oder materialisierten Ansichten für die Prozessverarbeitung des spezifischen Knotens in Betracht gezogen wurden.

Kontextsensitive Hilfe für jeden Knoten im grafischen Plan erhalten Sie, indem Sie auf den Knoten rechtsklicken und auf **Hilfe** klicken.

Hinweis

Wenn eine Abfrage als Bypass-Abfrage erkannt wird, werden einige Optimierungsschritte umgangen und weder der Abschnitt **Abfrageoptimierer** noch der Abschnitt **Prädikat** wird im grafischen Plan angezeigt.

Siehe auch

- [„So funktioniert der Optimierer“ auf Seite 333](#)
- [„Grafische Pläne“ auf Seite 347](#)
- [„Anzeigen eines grafischen Plans“ auf Seite 355](#)
- [„Fortgeschrittene Aufgaben: Abfrageausführungspläne“ auf Seite 341](#)
- [„Komponenten von Ausführungsplänen“ auf Seite 356](#)

Selektivitätsinformationen im grafischen Plan

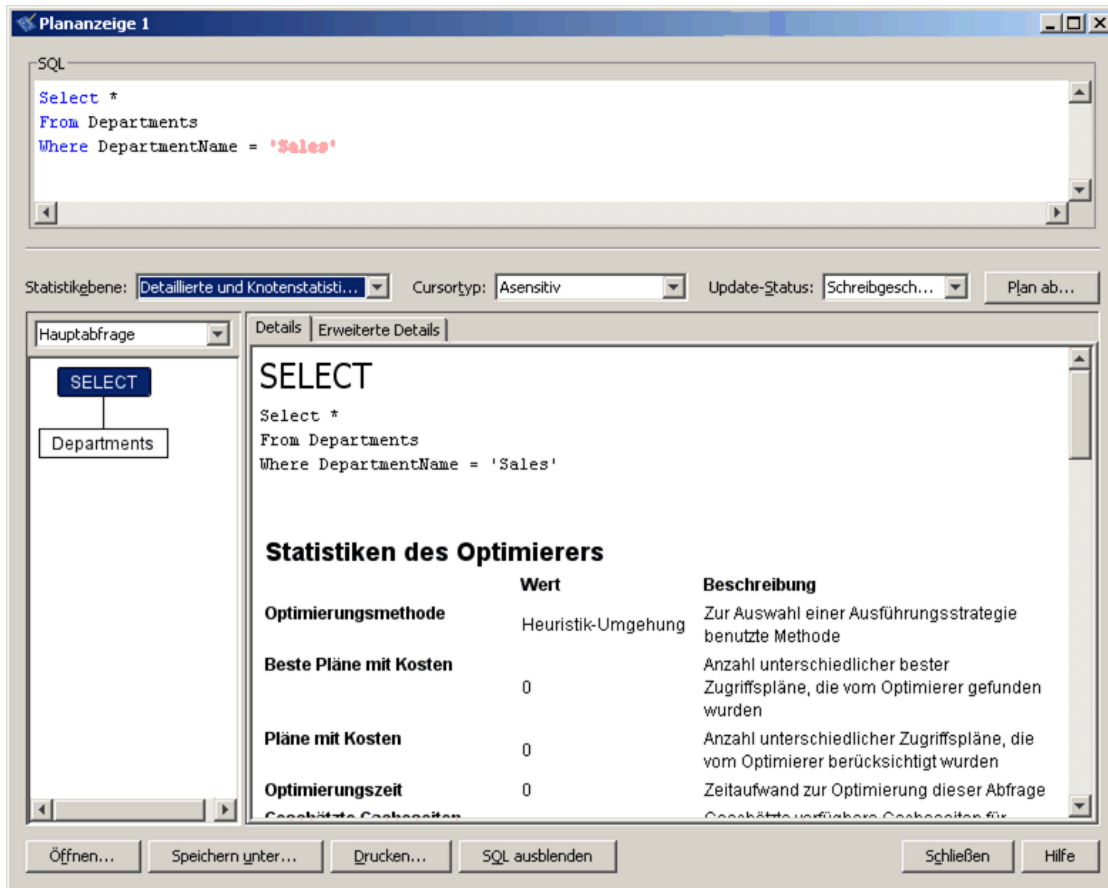
Im folgenden Beispiel stellt der ausgewählte Knoten einen Scan der Departments-Tabelle dar, und im Fensterausschnitt "Statistik" werden das **Prädikat** als Suchbedingung, seine Selektivitätsschätzung und seine tatsächliche Selektivität angezeigt.

Im Fensterausschnitt **Details** werden Statistiken zu einzelnen Knoten in drei Abschnitte aufgeteilt dargestellt: **Knotenstatistiken**, **Verzweigungsstatistiken** und **Statistiken des Optimierers**.

Knotenstatistiken beziehen sich auf die Ausführung dieses spezifischen Knotens. Wenn der Knoten kein Blattknoten im Plan ist und daher Zwischenergebnisse von anderen Knoten bezieht, zeigt der Fensterausschnitt **Details** einen Abschnitt **Verzweigungsstatistiken**, der geschätzte und tatsächliche kumulative Statistiken für die gesamte Unterstruktur dieses Knotens enthält. Statistikinformationen des Optimierers werden nur für Stammknoten ausgegeben, die die gesamte SQL-Anforderung darstellen.

Selektivitätsinformationen werden möglicherweise für Bypass-Abfragen nicht angezeigt. Weitere Hinweise zu Bypass-Abfragen finden Sie unter [„So funktioniert der Optimierer“ auf Seite 333](#).

Der Zugriffsplan hängt von den in der Datenbank verfügbaren Statistiken ab, die wiederum davon abhängen, welche Abfragen davor ausgeführt wurden. Ihre Statistiken und Pläne unterscheiden sich möglicherweise von den hier dargestellten.



Diese Prädikatbeschreibung lautet:

```
Departments.DepartmentName = 'Sales' : 20% Column; true 1/5 20%
```

Es kann wie folgt gelesen werden:

- `Departments.DepartmentName = 'Sales'` ist das Prädikat.
- `20%` ist die Schätzung des Optimierers für die Selektivität. Das bedeutet, dass der Optimierer seine Abfragezugriffsauswahl auf der Schätzung basiert, dass 20 % der Zeilen dem Prädikat entsprechen. Dies ist dieselbe Ausgabe, wie sie auch mit der `ESTIMATE`-Funktion erreicht wird.
- `Column` ist die Quelle der Schätzung. Dies ist dieselbe Ausgabe, wie sie auch mit der `ESTIMATE_SOURCE`-Funktion erreicht wird.
- `true 1/5 20%` ist die tatsächliche Selektivität des Prädikats während der Ausführung. Das Prädikat wurde fünf Mal ausgewertet und war einmal `TRUE`, was eine tatsächliche Selektivität von 20 % ergibt. Wenn sich die tatsächliche Selektivität stark von der Schätzung unterscheidet und das Prädikat sehr häufig ausgewertet wurde, ist es möglich, dass unzutreffende Schätzungen ein signifikantes Problem in

der Abfrageperformance bewirken. Das Sammeln von Prädikat-Statistiken steigert möglicherweise die Performance, indem dem Optimierer bessere Informationen zur Verfügung gestellt werden, auf denen seine Auswahl basiert.

Hinweis

Wenn Sie den grafischen Plan auswählen, aber nicht den grafischen Plan mit Statistik, werden die beiden letzten Statistiken nicht angezeigt.

Siehe auch

- „ESTIMATE-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Quellen der Selektivitätsschätzung“ auf Seite 336

Anzeigen eines grafischen Plans

Verwenden Sie entweder die **Plananzeige** in Interactive SQL oder die Funktion GRAPHICAL_PLAN, um grafische Pläne anzuzeigen. Die GRAPHICAL_PLAN-Funktion zeigt einen grafischen Plan im XML-Format an.

Voraussetzungen

Keine.

Aufgabe

1. Starten Sie Interactive SQL und stellen Sie eine Verbindung zur SQL Anywhere-Datenbank her.
2. Klicken Sie auf **Extras » Plananzeige** (oder drücken Sie Umschalt+F5).
3. Geben Sie eine Anweisung in den Fensterausschnitt **SQL** ein.
4. Wählen Sie **Statistikebene**, einen **Cursortyp** und einen **Update-Status** aus.
5. Klicken Sie auf **Plan abrufen**.

Ergebnisse

Der grafische Plan erscheint im Fensterausschnitt **Ergebnisse**.

Siehe auch

- „GRAPHICAL_PLAN-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Grafische Pläne mit detaillierten und Knotenstatistiken erstellen“ [*SQL Anywhere Server - Datenbankadministration*]
- „Komponenten von Ausführungsplänen“ auf Seite 356
- „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ auf Seite 341
- „Grafische Pläne anpassen“ [*SQL Anywhere Server - Datenbankadministration*]

Komponenten von Ausführungsplänen

Folgende Abkürzungen werden in Ausführungsplänen verwendet:

Weitere Hinweise zu den Algorithmen im Plan finden Sie im Whitepaper *Query Processing Based on SQL Anywhere 12.0.1 Architecture* unter <http://www.sybase.com/detail?id=1096047>.

Kurzer Textplan	Ausführlicher Textplan	Zusätzliche Informationen
	Beste Pläne mit Kosten	Der Optimierer erstellt für eine gegebene Abfrage Zugriffspläne und berechnet die Kosten. Während dieses Vorgangs wird der aktuelle beste Plan möglicherweise durch einen neuen besten Plan ersetzt, bei dem die Kostenschätzung geringer ausfällt. Dieser letzte Plan ist der Ausführungsplan, der zum Ausführen der Anweisung verwendet wird. Kostenberechnete beste Pläne geben die Häufigkeit an, mit der der Optimierer einen besseren Plan als den aktuellen besten Plan gefunden hat. Eine niedrige Zahl gibt an, dass der beste Plan frühzeitig im Enumerationsvorgang festgelegt wurde. Da der Optimierer bei einer gegebenen Anweisung den Enumerationsvorgang zumindest einmal pro Abfrageblock startet, stellen kostenberechnete beste Pläne die kumulative Zählung dar.
	Pläne mit Kosten	Viele vom Optimierer generierte Pläne erweisen sich als zu kostenträchtig, verglichen mit dem bisherigen besten Plan. Pläne mit Kosten geben die Anzahl der partiellen oder vollständigen Pläne an, die der Optimierer bei einer gegebenen Anweisung während des Enumerationsvorgangs in Betracht gezogen hat.
**	**	Ein vollständiger Index-Scan. Der Index-Scan liest alle Zeilen.
DELETE	Delete	Der Stammknoten eines DELETE-Vorgangs.
DistH	HashDistinct	HashDistinct nimmt eine einzelne Eingabe und gibt alle unterschiedlichen Zeilen zurück.
DistO	OrderedDistinct	OrderedDistinct liest jede Zeile und vergleicht sie mit der vorherigen Zeile. Wenn sie gleich ist, wird sie ignoriert, andernfalls wird sie ausgegeben.
DP	DecodePostings	DecodePostings dekodiert Positionsdaten für die Begriffe im Textindex.

Kurzer Textplan	Ausführlicher Textplan	Zusätzliche Informationen
DT	DerivedTable	DerivedTable erscheint möglicherweise in einem Plan, entweder aufgrund von Neuschreibungsoptimierungen einer Abfrage oder aus verschiedenen anderen Gründen, besonders wenn die Abfrage einen oder mehrere Outer-Joins umfasst.
EAH	HashExceptAll	Gibt an, dass eine Hash-basierte Implementierung des SQL-Mengendifferenzenoperators EXCEPT verwendet wurde.
EAM	MergeExceptAll	Gibt an, dass eine sortierungsbasierte Implementierung des SQL-Mengendifferenzenoperators EXCEPT verwendet wurde.
EH	HashExcept	Gibt an, dass eine Hash-basierte Implementierung des SQL-Mengendifferenzenoperators EXCEPT verwendet wurde.
EM	MergeExcept	Gibt an, dass eine sortierungsbasierte Implementierung des SQL-Mengendifferenzenoperators EXCEPT verwendet wurde.
Exchange	Exchange	Gibt an, dass beim Verarbeiten einer SELECT-Anweisung abfrageinterne Parallelität verwendet wurde.
Filter	Filter	Gibt an, dass Suchbedingungen angewendet werden. Dazu gehören alle Arten von Prädikaten, Vergleiche unter Verwendung von Unterabfrage-Bedingungen sowie EXISTS- und NOT EXISTS-Unterabfragen (und andere Formen quantifizierter Unterabfragen).
GrByH	HashGroupBy	HashGroupBy baut eine speicherresidente Hash-Tabelle auf, die eine Zeile pro Gruppe enthält. Beim Lesen der Eingabezeilen werden die zugeordneten Gruppen in der Arbeitstabelle gesucht. Die Aggregatfunktionen werden aktualisiert, und die Gruppenzeile wird erneut in die Arbeitstabelle geschrieben. Wenn kein Gruppendatensatz vorhanden ist, wird ein neuer Gruppendatensatz initialisiert und in die Arbeitstabelle eingefügt.
GrByHClust	HashGroupByClustered	In manchen Fällen sind die Werte in den Gruppenspalten der Eingabetabelle gebündelt (clustered), d.h. ähnliche Werte treten nahe beieinander auf. ClusteredHashGroupBy nutzt diese Clusterbildung.
GrByHP	ParallelHashGroupBy	Eine Variante von HashGroupBy.

Kurzer Textplan	Ausführlicher Textplan	Zusätzliche Informationen
GrByHSets	HashGroupBySets	Als eine Variante von HashGroupBy wird HashGroupBySets verwendet, wenn GROUPING SETS-Abfragen durchgeführt werden sollen.
GrByO	OrderedGroupBy	OrderedGroupBy liest eine Eingabe, die nach Gruppierungsspalten sortiert ist. Jede einzelne Zeile wird gelesen und mit der vorhergehenden Zeile verglichen. Wenn die gruppierten Spalten übereinstimmen, wird die gegenwärtige Gruppe aktualisiert; anderenfalls wird die gegenwärtige Gruppe ausgegeben, und es wird mit einer neuen Gruppe begonnen.
GrByOSets	OrderedGroupBy-Sets	Als eine Variante von OrderedGroupBy wird OrderedGroupBySets verwendet, wenn GROUPING SETS-Abfragen durchgeführt werden sollen.
GrByS	SingleRowGroupBy	Wenn kein GROUP BY angegeben ist, wird SingleRowGroupBy verwendet, um ein einzelnes Zeilenaggregat zu erzeugen. Eine einzelne Gruppenzeile verbleibt im Speicher und wird bei jeder Eingabezeile aktualisiert.
GrBySSets	SortedGroupBySets	SortedGroupBySets wird bei der Verarbeitung von OLAP-Abfragen verwendet, die GROUPING SETS enthalten.
HF	HashFilter	Gibt an, dass ein Hash-Filter (oder Bloom-Filter) verwendet wurde.
HFP	ParallelHashFilter	Gibt an, dass ein Hash-Filter (oder Bloom-Filter) verwendet wurde.
HTS	HashTableScan	Gibt an, dass ein Hash-Table-Scan verwendet wurde.
IAH	HashIntersectAll	Gibt an, dass eine Hash-basierte Implementierung des SQL-Mengendifferenzenoperators INTERSECT verwendet wurde.
IAM	MergeIntersectAll	Gibt an, dass eine sortierungsbasierte Implementierung des SQL-Mengendifferenzenoperators INTERSECT verwendet wurde.
IH	HashIntersect	Gibt an, dass eine Hash-basierte Implementierung des SQL-Mengendifferenzenoperators INTERSECT verwendet wurde.
IM	MergeIntersect	Gibt an, dass eine sortierungsbasierte Implementierung des SQL-Mengendifferenzenoperators INTERSECT verwendet wurde.

Kurzer Textplan	Ausführlicher Textplan	Zusätzliche Informationen
IN	InList	InList wird in Fällen verwendet, bei denen ein IN-Listenprädikat unter Verwendung eines Indexes erfüllt werden kann.
<i>table-name</i> < <i>index-name</i> >	IndexScan, ParallelIndexScan	In einem grafischen Plan wird ein Index-Scan als Indexname in einem Trapez dargestellt.
INSENSITIVE	Insensitive	
INSERT	Insert	Stammknoten eines INSERT-Vorgangs.
IO	IndexOnlyScan, ParallelIndexOnlyScan	Gibt an, dass der Optimierer einen Index verwendet hat, der alle zum Erfüllen der Abfrage erforderlichen Daten enthält.
JH	HashJoin	Der HashJoin-Algorithmus erstellt eine speicherresidente Hash-Tabelle der kleineren der beiden Eingaben. Dann liest er die größere Eingabe und untersucht die Hash-Tabelle im Speicher auf Übereinstimmungen, die dann in eine Arbeitstabelle geschrieben werden. Wenn die kleinere Eingabe nicht in den Speicher passt, teilt HashJoin beide Eingaben in kleinere Arbeitstabellen auf. Diese kleineren Arbeitstabellen werden rekursiv abgearbeitet, bis die kleinere Eingabe in den Speicher passt.
JHS	HashSemijoin	HashSemijoin führt einen Semijoin zwischen der linken und der rechten Seite durch.
JHSP	ParallelHashSemijoin	Eine Variante von HashJoin.
JHFO	Full Outer HashJoin	Eine Variante von HashJoin.
JHA	HashAntisemijoin	HashAntisemijoin führt einen Anti-Semijoin zwischen der linken und der rechten Seite durch.
JHAP	ParallelHashAntisemijoin	Eine Variante von HashJoin.
JHO	Left Outer HashJoin	Eine Variante von HashJoin.
JHP	ParallelHashJoin	Eine Variante von HashJoin.
JHPO	ParallelLeftOuterHashJoin	Eine Variante von HashJoin.

Kurzer Textplan	Ausführlicher Textplan	Zusätzliche Informationen
JHR	RecursiveHashJoin	Eine Variante von HashJoin.
JHRO	RecursiveLeftOuterHashJoin	Eine Variante von HashJoin.
JM	MergeJoin	MergeJoin liest zwei Eingaben, die beide nach den Join-Attributen sortiert sind. Für jede Zeile in der linken Eingabe liest der Algorithmus alle übereinstimmenden Zeilen der rechten Eingabe, indem er auf die Zeilen in sortierter Reihenfolge zugreift.
JMFO	Full Outer MergeJoin	Eine Variante von MergeJoin.
JMO	Left Outer MergeJoin	Eine Variante von MergeJoin.
JNL	NestedLoopsJoin	NestedLoopsJoin berechnet den Join zwischen seiner rechten und linken Seite, indem er die rechte Seite für jede Zeile der linken Seite vollständig liest.
JNLA	NestedLoopsAntisemijoin	NestedLoopsAntisemijoin verknüpft seine Eingaben, indem er die rechte Seite nach jeder Zeile der linken Seite durchsucht.
JNLFO	Full Outer NestedLoopsJoin	Eine Variante von NestedLoopsJoin.
JNLO	Left Outer NestedLoopsJoin	Eine Variante von NestedLoopsJoin.
JNLS	NestedLoopsSemi-join	NestedLoopsSemijoin verknüpft seine Eingaben, indem er die rechte Seite nach jeder Zeile der linken Seite durchsucht.
KEYSET	Keyset	Gibt einen Keyset-gesteuerten Cursor an.
LOAD	Load	Stammknoten eines LOAD-Vorgangs.
MultiIdx	MultipleIndexScan	Die MultipleIndexScan-Methode wird verwendet, wenn ein Index verwendet werden kann oder muss, um eine Abfrage zu erfüllen, die einen Satz von Suchbedingungen enthält, die mit den logischen Operatoren AND oder OR verbunden sind.
OpenString	OpenString	OpenString wird in Fällen verwendet, bei denen die FROM-Klausel einer SELECT-Anweisung eine OPENSTRING-Klausel enthält.

Kurzer Textplan	Ausführlicher Textplan	Zusätzliche Informationen
	Optimierungszeit	Die Gesamtzeit, die der Optimierer bei einer gegebenen Anweisung während aller Enumerationsvorgänge verbraucht hat.
PC	ProcCall	Procedure call (Prozeduraufruf, Tabellenfunktion).
PreFilter	PreFilter	Filter wenden Suchbedingungen an. Dazu gehören alle Arten von Prädikaten, Vergleiche zwischen beteiligten Unterabfrage-Bedingungen sowie EXISTS- und NOT EXISTS-Unterabfragen (und andere Formen quantifizierter Unterabfragen).
R	R	Ein umgekehrter Index-Scan. Der Index-Scan liest die Zeilen aus dem Index in umgekehrter Reihenfolge.
RL	RowLimit	RowLimit gibt die ersten n Zeilen seiner Eingabe zurück und ignoriert die restlichen Zeilen. Zeilenbegrenzungen werden mit der TOP n- bzw. FIRST-Klausel der SELECT-Anweisung gesetzt.
ROWID	RowIdScan	In einem grafischen Plan wird ein RowID Scan als Tabellenname in einem Rechteck dargestellt.
ROWS	RowConstructor	RowConstructor ist ein spezialisierter Operator, der eine virtuelle Zeile erstellt, die als Eingabe für andere Algorithmen verwendet werden kann.
RR	RowReplicate	RowReplicate wird während der Ausführung von Mengenoperationen wie etwa EXCEPT ALL und INTERSECT ALL verwendet.
RT	RecursiveTable	Gibt an, dass als Ergebnis einer WITH-Klausel in einer Abfrage eine rekursive Tabelle benutzt wurde, wobei die WITH-Klausel für rekursive Vereinigungsabfragen verwendet wurde.
RU	RecursiveUnion	RecursiveUnion wird während der Ausführung von rekursiven Vereinigungsabfragen verwendet.
SELECT	Select	Stammknoten eines SELECT-Vorgangs.
seq	TableScan, Parallel-TableScan	In einem grafischen Plan werden Table-Scans als Tabellennamen in einem Rechteck dargestellt.
Sortieren	Sortieren	Indizierte Sortierung oder Zusammenführungssortierung.

Kurzer Textplan	Ausführlicher Textplan	Zusätzliche Informationen
SrtN	SortTopN	SortTopN wird für Abfragen verwendet, die eine TOP N-Klausel und eine ORDER BY-Klausel enthalten.
TermBreak	TermBreak	Der TermBreaker-Algorithmus für Volltextsuche.
UA	UnionAll	UnionAll liest die Zeilen aus jeder seiner Eingaben und gibt sie wieder aus, gleichgültig ob Duplikate vorhanden sind oder nicht. Dieser Algorithmus wird verwendet, um UNION- und UNION ALL-Anweisungen zu implementieren.
UPDATE	Update	Der Stammknoten eines UPDATE-Vorgangs.
Window	Window	Window wird bei der Berechnung von OLAP-Abfragen verwendet, die Fensterfunktionen verwenden.
Work	Work table	Ein interner Knoten, der ein Zwischenergebnis darstellt.

Optimiererstatistiken-Feldbeschreibungen

Nachstehend finden Sie die Beschreibungen der Felder, die in den Abschnitten **Optimierer-Statistiken**, **Lokale Optimierer-Statistiken** und **Globale Optimierer-Statistiken** eines grafischen Plans angezeigt werden. Diese Statistiken enthalten Informationen über den Status des Datenbankservers und den Optimierungsprozess.

Feld	Beschreibung
Optimierungsmethode	<p>Der zur Auswahl einer Ausführungsstrategie verwendete Algorithmus. Rückgabewerte:</p> <ul style="list-style-type: none"> ● Umgehung mit Kostenrechnung ● Umgangen mit Kosten (einfach) ● Heuristik-Umgehung ● Umgangen, dann optimiert ● Optimiert ● Wiederverwendet ● Wiederverwendet (einfach)

Feld	Beschreibung
Beste Pläne mit Kosten	<p>Wenn der Abfrageoptimierer verschiedene Abfrageausführungsstrategien aufzählt, protokolliert er die Häufigkeit, mit der er eine Strategie findet, deren geschätzte Kosten günstiger sind als die der besten Strategie, die vor der aktuellen gefunden wurde. Es ist schwierig vorherzusagen, wie oft dies bei einer bestimmten Abfrage auftreten wird, aber eine niedrigere Zahl zeigt eine signifikante Beschneidung des Suchplatzbedarfs durch den Algorithmus des Optimierers und bezeichnenderweise kürzere Optimierungszeiten an. Da der Optimierer bei einer gegebenen Anweisung den Enumerationsvorgang zumindest einmal pro Abfrageblock startet, stellen Beste Pläne mit Kosten die kumulative Zählung dar.</p> <p>Wenn die Werte für Beste Pläne mit Kosten, Pläne mit Kosten und Optimierungszeit 0 sind, dann wurde die Anweisung vom SQL Anywhere-Optimierer nicht optimiert. Stattdessen umging der Datenbankserver die Anweisung und generierte den Ausführungsplan, ohne die Anweisung zu optimieren, oder der Plan für die Anweisung wurde im Cache abgelegt.</p>
Pläne mit Kosten	<p>Die Anzahl der unterschiedlichen Zugriffspläne, die vom Optimierer für diese Anforderung in Betracht gezogen wurden und deren Kosten teilweise oder vollständig geschätzt wurden. Wie bei den Besten Plänen mit Kosten, zeigen niedrigere Werte üblicherweise kürzere Optimierungszeiten und höhere Werte komplexere SQL-Abfragen an.</p> <p>Wenn die Werte für Beste Pläne mit Kosten, Pläne mit Kosten und Optimierungszeit 0 sind, dann wurde die Anweisung nicht optimiert. Stattdessen hat der Datenbankserver die Anweisung übergangen und einen Ausführungsplan ohne Optimierung der Anweisung generiert.</p>

Feld	Beschreibung
Optimierungszeit	<p>Die verstrichene Zeit, die zur Optimierung der Anweisung aufgewendet wurde.</p> <p>Wenn die Werte für Beste Pläne mit Kosten, Pläne mit Kosten und Optimierungszeit 0 sind, dann wurde die Anweisung nicht optimiert. Stattdessen hat der Datenbankserver die Anweisung übergangen und einen Ausführungsplan ohne Optimierung der Anweisung generiert.</p>
Geschätzte Cacheseiten	<p>Die geschätzte aktuelle Cachegröße, die für die Prozessverarbeitung der Anweisung verfügbar ist.</p> <p>Um die Anzahl der ineffizienten Zugriffspläne zu vermindern, nimmt der Optimierer an, dass die Hälfte der aktuellen Cachegröße für die Prozessverarbeitung der ausgewählten Anweisung zur Verfügung steht.</p>
CurrentCacheSize	Die Cachegröße des Datenbankservers in kB zum Zeitpunkt der Optimierung.
QueryMemMaxUseful	Die Anzahl der Abfragespeicherseiten, die bei dieser Anforderung sinnvoll sind. Wenn die Anzahl Null ist, enthält der Ausführungsplan der Anweisung keine speicherintensiven Operatoren und wird nicht vom Speichernutzungswächter des Datenbankservers gesteuert.
QueryMemNeedsGrant	Gibt an, ob der Speichernutzungswächter Speicher an einen oder mehrere speicherintensive Abfrageausführungs-Operatoren zuweisen muss, die in der Ausführungsstrategie dieser Anforderung aufscheinen.
QueryMemLikelyGrant	Die geschätzte Anzahl von Seiten aus dem Abfragespeicher-Pool, die dieser Anweisung zugewiesen würden, wenn sie sofort ausgeführt würde. Diese Schätzung kann unterschiedlich ausfallen, abhängig von der Anzahl der speicherintensiven Operatoren im Plan, der Multiprogramming-Stufe des Datenbankservers und der Anzahl der parallel ausführenden speicherintensiven Anforderungen.

Feld	Beschreibung
QueryMemPages	Die Gesamtmenge von Speicher im Abfragespeicher-Pool, die für speicherintensive Algorithmen zur Abfrageausführung für alle Verbindungen zur Verfügung steht, ausgedrückt als eine Anzahl von Seiten.
QueryMemActiveMax	Die maximale Anzahl von Aufgaben, die aktiv den Abfragespeicher zu jedem beliebigen Zeitpunkt nutzen können.
QueryMemActiveEst	Die Schätzung des Datenbankservers zum Steady-state-Mittelwert der Anzahl der Aufgaben, die aktiv den Abfragespeicher nutzen.
isolation_level	Die Isolationsstufe der Anweisung. Die Isolationsstufe der Anweisung unterscheidet sich möglicherweise von der anderer Anweisungen in derselben Transaktion und kann überdies bei spezifischen Basistabellen durch die Verwendung eines Hints in der FROM-Klausel aufgehoben werden.
optimization_goal	Gibt an, ob die Abfrageverarbeitung dahingehend optimiert ist, die erste Zeile schnell zurückzugeben, oder ob die Kosten für die Ausgabe der vollständigen Ergebnismenge minimiert werden sollen.
optimization_level	Steuert, wie intensiv sich der Abfrageoptimierer bemühen soll, einen Zugriffsplan zu finden.
optimization_workload	Der Mixed - oder OLAP -Wert der Einstellung "optimization_workload".
max_query_tasks	Maximale Anzahl der Aufgaben, die von einem parallelen Ausführungsplan für eine Abfrage benutzt werden können.
user_estimates	Steuert, ob Benutzerschätzungen, die in einzelnen Prädikaten im Abfragetext angegeben sind, berücksichtigt oder ignoriert werden.
Geschätzte Aufgaben	Die Anzahl der geschätzten Aufgaben für abfrageinterne Parallelität
Maximale Anzahl von Aufgaben	Die maximale Anzahl von Aufgaben für die abfrageinterne Parallelität

Feld	Beschreibung
Zeit für die Build-Optimierung	Die Menge der verstrichenen Zeit beim Erstellen der Optimierungsinterna
Voroptimierungszeit	Die Menge der verstrichenen Zeit für die Voroptimierung
Optimierungszeit	Die Menge der verstrichenen Zeit für die Optimierung
Zeit ohne Optimierung	Die Menge der verstrichenen Zeit in Phasen ohne Optimierung
Nachoptimierungszeit	Die Menge der verstrichenen Zeit für die Nachoptimierung
Erstellungsdauer für den endgültigen Plan	Die Menge der verstrichenen Zeit beim Erstellen des endgültigen Plans
Geschätzte Maximalkosten	Geschätzte Maximalkosten für diese Optimierung
Algorithmus für die Join-Auflistung	<p>Der für die Join-Auflistung verwendete Algorithmus. Die möglichen Werte sind:</p> <ul style="list-style-type: none"> • Unbeschränkte Bäume 1 • Unbeschränkte Bäume 2 • Unbeschränkte Bäume mit Voroptimierung • Unbeschränkte Bäume mit Pruning • Parallele unbeschränkte Bäume • Links-tiefe Bäume • Unbeschränkte Bäume 3 • Links-tiefe Bäume mit Memoisation
Anzahl der Voroptimierungen	Gültig für unbeschränkte Bäume mit Voroptimierung-Join-Aufzählungsalgorithmus
Für Voroptimierung verwendete Seiten	Die Anzahl der Speicherseiten, die während der Vorphase der Optimierung benutzt wurden
Während der Join-Auflistung verwendete Seiten	Die Anzahl der Speicherseiten, die während der Join-Auflistung benutzt wurden
Anzahl der berücksichtigten Voroptimierungen	Die Anzahl der Speicherseiten, die während untersuchter Voroptimierungen benutzt wurden
Während der Join-Auflistung verwendete Speicherseiten	Die Anzahl der Speicherseiten, die während der Join-Auflistungsphase benutzt wurden

Feld	Beschreibung
Während der Join-Auflistung verwendete zusätzliche Seiten	Die Anzahl der zusätzlichen Speicherseiten, die von der Join-Auflistung mit Beschneidung verwendet wurden
Laufzeit der Join-Auflistung	Die Menge der während der Join-Auflistungsphase verstrichenen Zeit
Initialisierungslaufzeit	Die Menge der verstrichenen Zeit während der Initialisierungsphase
Laufzeit für geschätzte Maximalkosten	Die Menge der verstrichenen Zeit während der geschätzten maximalen Kostenphase
Pruning-Laufzeit	Die Menge der verstrichenen Zeit während der Pruning-Phase.
Kostenlaufzeit	Die Menge der verstrichenen Zeit während der Kostenbewertungsphase
Laufzeit für die Generierung des physischen Plans	Die Menge der verstrichenen Zeit während der Phase der Generierung des physischen Plans
Parallellaufzeit	Die Menge der verstrichenen Zeit während der parallelen Phase
Laufzeit für die Generierung der links-tiefen Bäume	Die Menge der verstrichenen Zeit während der Generierungsphase der links-tiefen Bäume
Laufzeit für die Generierung des logischen Plans	Die Menge der verstrichenen Zeit während der Generierungsphase des logischen Plans
Voroptimierungslaufzeit	Die Menge der verstrichenen Zeit während der Voroptimierungsphase
Bereinigungslaufzeit	Die Menge der verstrichenen Zeit während der Bereinigungsphase
Partitionslaufzeit	Die Menge der verstrichenen Zeit während der Partitionierungsphase
Protokollierungslaufzeit	Die Menge der verstrichenen Zeit während der Protokollierungsphase
Sonstige Laufzeit	Die Menge der verstrichenen Zeit während sonstiger Phasen

Feld	Beschreibung
Anzahl der aufgelisteten logischen Joins	Die Anzahl der bei der Auflistung gefundenen logischen Joins
Anzahl der Joins mit Kosten	Die Anzahl der Joins mit Kosten
Joins mit Pruning	Die Anzahl der Joins mit Pruning basierend auf lokalen und globalen Kosten
Vorgänge in der Memoisationstabelle	Die Vorgänge in der Memorisationstabelle (Einfügen, Ersetzen, Suchen).

Knotenstatistiken-Feldbeschreibungen

Im Folgenden finden Sie die Beschreibungen der Felder, die im Abschnitt **Knotenstatistiken** eines grafischen Plans angezeigt werden.

Feld	Beschreibung
CacheHits	Die Gesamtzahl von Cache-Leseanforderungen durch diesen Operator, die vom Pufferpool erfüllt wurden und keinen Festplatten-Lesezugriff für diese Seite erforderten.
Cache-Read	Die Gesamtzahl der von diesem Operator vorgenommenen Versuche, um eine Seite der Datenbankdatei zu lesen, üblicherweise für Tabellen- bzw. Indexseiten.
CPUTime	Die CPU-Zeit, die durch den Verarbeitungsalgorithmus angefallen ist, der durch diesen Knoten dargestellt wird.
DiskRead	Die kumulative Anzahl der Seiten, die vom Plattenspeicher als Ergebnis der Prozessverarbeitung dieses Knotens gelesen wurden.
DiskRead-Time	Die kumulativ verstrichene Zeit, die erforderlich war, um Festplatten-Lesevorgänge für die Datenbankseiten auszuführen, die von diesem Knoten für die Prozessverarbeitung benötigt wurden.
DiskWrite	Die kumulative Anzahl der Seiten, die auf den Plattenspeicher als Ergebnis der Prozessverarbeitung dieses Knotens geschrieben wurden.
DiskWriteTime	Die kumulativ verstrichene Zeit, die erforderlich war, um Festplatten-Schreibvorgänge für die Datenbankseiten auszuführen, die vom Verarbeitungsalgorithmus dieses Knotens benötigt wurden.
FirstRow-RunTime	Der Wert FirstRowRunTime gibt die Zeit wieder, die tatsächlich vergangen ist, um die erste Zeile des Zwischenergebnisses von diesem Knoten zu liefern.

Feld	Beschreibung
Invocations	Die Häufigkeit, mit der der Knoten aufgerufen wurde, um ein Ergebnis zu berechnen und dieses Ergebnis an den übergeordneten Knoten zurückzugeben. Die meisten Knoten werden nur einmal aufgerufen. Wenn jedoch das übergeordnete Objekt eines Scan-Knotens ein Nested Loops Join (Join mit verschachtelten Schleifen) ist, wird der Knoten möglicherweise mehrfach ausgeführt und könnte nach jedem Aufruf eine andere Zeilenmenge zurückgeben.
Percent-TotalCost	Die RunTime , die zur Berechnung des Ergebnisses in diesem spezifischen Knoten aufgewendet wurde, ausgedrückt als Prozentsatz der gesamten Laufzeit für diese Anweisung.
QueryMemMaxUseful	Die geschätzte Menge an Abfragespeicher, die voraussichtlich von diesem spezifischen Operator benötigt wird. Wenn sich die tatsächliche Menge an Abfragespeicher, die in der Tatsächlich -Statistik angezeigt wird, signifikant von der geschätzten Menge unterscheidet, kann dies auf ein Problem mit der Ergebnismenge-Größenschätzung durch den Abfrageoptimierer hinweisen. Ein möglicher Grund für diesen Schätzfehler sind nicht akkurate oder fehlende Prädikat-Selektivitätsschätzungen.
RowsReturned	<p>Die Anzahl der an den übergeordneten Knoten zurückgegebenen Zeilen als Ergebnis der Prozessverarbeitung der Anforderung. RowsReturned ist häufig, aber nicht zwingend identisch mit der Anzahl der Zeilen im (möglicherweise abgeleiteten) Objekt, das durch diesen Knoten dargestellt wird. Nehmen wir einen Blattknoten, der einen Basistabellen-Scan darstellt. Es ist möglich, dass der Wert RowsReturned kleiner oder größer als die Anzahl der Zeilen der Tabelle ist. RowsReturned ist kleiner, wenn der übergeordnete Knoten beim Berechnen des endgültigen Ergebnisses nicht alle Zeilen der Tabelle anfordert. RowsReturned kann im Fall einer GROUP BY GROUPING SETS-Abfrage größer sein, bei der der übergeordnete Knoten "Group By Hash Grouping Sets " mehrere Durchgänge in Bezug auf die Eingabe benötigt, um die verschiedenen Gruppen zu berechnen.</p> <p>Ein signifikanter Unterschied zwischen den geschätzten, zurückgegebenen Zeilen und der tatsächlichen, zurückgegebenen Anzahl könnte bedeuten, dass der Optimierer mit ungeeigneten Selektivitätsinformationen arbeitet.</p>

Feld	Beschreibung
RunTime	<p>Dieser Wert ist ein Uhrzeit-Messwert und umfasst Wartezeiten für Eingabe/Ausgabe, Zeilensperren, Tabellensperren, interne Datenbankserver-Kontrollmechanismen im Mehrbenutzerbetrieb und die tatsächliche Laufzeit-Prozessverarbeitung. Die Interpretation von RunTime hängt vom Statistikabschnitt ab, in dem der Wert angezeigt wird. In Knotenstatistiken ist RunTime die kumulative Zeit, die der entsprechende Operator des Knotens für das Ausführen nur dieses einen Knotens aufgewendet hat. Sowohl die geschätzten als auch die tatsächlichen Werte für diese Statistik werden im Abschnitt "Knotenstatistiken" angezeigt.</p> <p>Wenn RunTime eines Knotens bei einem Table-Scan oder einem Index-Scan größer als erwartet ist, sind weitere Analysen angebracht, um das Problem zu ermitteln. Die Abfrage muss möglicherweise um gemeinsame Ressourcen konkurrieren und könnte aus diesem Grund blockiert werden. Blockierte Verbindungen können Sie mit der Systemprozedur <code>sa_locks()</code> überwachen. Es wäre auch möglich, dass das Datenbankseiten-Layout im Plattenspeicher suboptimal ist oder eine Tabelle eine interne Seitenfragmentierung aufweist. Sie können die Performance verbessern, indem Sie die Anweisung "REORGANIZE TABLE" ausführen. Sie können die Systemprozeduren "sa_table_fragmentation()" und "sa_index_density()" verwenden, um festzustellen, ob die Tabelle oder der Index fragmentiert ist.</p>

Häufig im Plan verwendete Statistiken

Die folgenden Statistiken sind tatsächliche, gemessene Werte.

Statistik	Erklärung
CacheHits	Gibt die Anzahl der Suchvorgänge nach Datenbankseiten zurück, die durch eine im Cache gefundene Seite belohnt wurden.
CacheRead	Gibt die Anzahl der Datenbankseiten zurück, die im Cache nachgeschlagen wurden.
CacheReadTable	Gibt die Anzahl der Tabellenseiten zurück, die aus dem Cache gelesen wurden.
CacheReadIndLeaf	Gibt die Anzahl der Indexblattseiten zurück, die aus dem Cache gelesen wurden.
CacheReadIndInt	Gibt die Anzahl der internen Knotenseiten des Indexes zurück, die aus dem Cache gelesen wurden.
DiskRead	Gibt die Anzahl der Seiten zurück, die von der Festplatte gelesen wurden.
DiskReadTable	Gibt die Anzahl der Tabellenseiten zurück, die von der Festplatte gelesen wurden.
DiskReadIndLeaf	Gibt die Anzahl der Indexblattseiten zurück, die von der Festplatte gelesen wurden.

Statistik	Erklärung
DiskReadIndInt	Gibt die Anzahl der internen Knotenseiten des Indexes zurück, die von der Festplatte gelesen wurden.
DiskWrite	Gibt die Anzahl der geänderten Seiten zurück, die auf die Festplatte geschrieben wurden.
IndAdd	Gibt die Anzahl der Einträge zurück, die Indizes hinzugefügt wurden.
IndLookup	Gibt die Anzahl der Einträge zurück, nach denen in Indizes gesucht wurde.
FullCompare	Gibt die Anzahl der Vergleiche zurück, die über den Hash-Wert hinaus in einem Index durchgeführt wurden.

Häufig im Plan verwendete Schätzungen

Statistik	Erklärung
EstRowCount	Geschätzte Anzahl der Zeilen, die vom Knoten jedesmal zurückgegeben werden, wenn er aufgerufen wird
AvgRowCount	Mittlere Anzahl der Zeilen, die bei jedem Aufruf zurückgegeben werden. Diese Zahl wird nicht geschätzt, sondern berechnet als RowsReturned / Invocations. Wenn dieser Wert deutlich von EstRowCount abweicht, ist eventuell die Selektivitätsschätzung schlecht.
EstRunTime	Geschätzte Ausführungszeit (Summe von EstDiskReadTime, EstDiskWriteTime und EstCpuTime)
AvgRunTime	Mittlere Ausführungszeit (gemessen)
EstDiskReads	Geschätzte Anzahl der Lesevorgänge von der Festplatte
AvgDiskReads	Mittlere Anzahl der Lesevorgänge von der Festplatte (gemessen)
EstDiskWrites	Geschätzte Anzahl der Schreibvorgänge auf die Festplatte
AvgDiskWrites	Mittlere Anzahl der Schreibvorgänge auf die Festplatte (gemessen)
EstDiskReadTime	Geschätzte Zeit zum Lesen von Zeilen von der Festplatte
EstDiskWriteTime	Geschätzte Zeit zum Schreiben von Zeilen auf die Festplatte
EstCpuTime	Geschätzte Prozessorzeit, die für die Ausführung erforderlich ist

Elemente im Plan in Bezug auf SELECT, INSERT, UPDATE und DELETE

Element	Erklärung
Optimierungsziel (optimization_goal)	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.
Optimierungsarbeitslast (optimization_workload)	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.
ANSI-Aktualisierungsbeschränkungen (ansi_update_constraints)	Steuert den Bereich der Aktualisierungen, die zulässig sind (Optionen sind: Off, Cursors und Strict).
Optimierungsstufe (optimization_level)	Reserviert
Select-Liste	Liste mit Ausdrücken, die von der Abfrage ausgewählt wurden

Element	Erklärung
Materialisierte Ansichten	<p>Liste der materialisierten Ansichten, die vom Optimierer in Betracht gezogen werden. Jeder Listeneintrag ist ein Tupel im folgenden Format: <code>view-name [view-matching-outcome] [table-list]</code>, wobei gilt: <i>view-matching-outcome</i> zeigt die Nutzung einer materialisierten Ansicht. Falls der Wert <i>COSTED</i> lautet, wurde die Ansicht während der Auflistung verwendet. Die <i>table-list</i> ist eine Liste mit Abfragetabellen, die möglicherweise durch diese Ansicht ersetzt wurden.</p> <p>Zu den Werten für das <i>view-matching-outcome</i> gehören:</p> <ul style="list-style-type: none"> • Basistabelle stimmt nicht überein • Privilegien stimmen nicht überein • Prädikat stimmt nicht überein • Auswahlliste stimmt nicht überein • Kosten • Ansicht veraltet • Snapshot veraltet • Kann von Optimierer nicht verwendet werden • Kann intern von Optimierer nicht verwendet werden • Definition kann nicht erstellt werden • Zugriff nicht möglich • Deaktiviert • Optionen stimmen nicht überein • Schwellenwert für Ansichtenübereinstimmung erreicht • Verwendete Ansicht

Elemente im Plan in Bezug auf Sperren

Element	Erklärung
Gesperrte Tabellen	Liste aller gesperrten Tabellen und ihrer Isolationsstufen

Elemente im Plan in Bezug auf Scans

Element	Erklärung
Tabellenname	Tatsächlicher Name der Tabelle

Element	Erklärung
Korrelationsname	Alias für die Tabelle
Geschätzte Zeilen	Geschätzte Anzahl der Zeilen in der Tabelle
Geschätzte Seiten	Geschätzte Anzahl der Seiten in der Tabelle
Geschätzte Zeilengröße	Geschätzte Zeilengröße für die Tabelle
Seitenzuordnung	JA, wenn eine Seitenzuordnung zum Lesen mehrerer Seiten verwendet wird

Elemente im Plan in Bezug auf Index-Scans

Element	Erklärung
Selektivität	Geschätzte Anzahl der Zeilen, die innerhalb der Bereichsgrenzen liegen
Indexname	Name des Indexes
Schlüsseltyp	Kann PRIMARY KEY, FOREIGN KEY, CONSTRAINT (unique constraint) oder UNIQUE (unique index) sein. Der Schlüsseltyp erscheint nicht, wenn der Index ein nicht eindeutiger Sekundärindex ist.
Tiefe	Höhe des Indexes.
Geschätzte Blattseiten	Geschätzte Anzahl der Blattseiten
Sequenzielle Transitionen	Statistiken für jeden physischen Index, die angeben, wie gebündelt (clustered) der Index ist.
Zufällige Transitionen	Statistiken für jeden physischen Index, die angeben, wie gebündelt (clustered) der Index ist.
Schlüsselwerte	Die Anzahl der eindeutigen Einträge im Index
Kardinalität	Die Kardinalität des Indexes, wenn er sich von der geschätzten Anzahl der Zeilen unterscheidet. Dies gilt nur für SQL Anywhere-Datenbanken der Version 6.0.0 und früher.
Richtung	FORWARD oder BACKWARD.
Bereichsgrenzen	Bereichsgrenzen werden als eine Liste dargestellt (col_name=value) oder col_name IN [low, high]

Element	Erklärung
Primärschlüsseltabelle	Der Name der Primärschlüsseltabelle für einen Fremdschlüsselindex-Scan..
Geschätzte Zeilen in Primärschlüsseltabelle	Die Anzahl der Zeilen in der Primärschlüsseltabelle für einen Fremdschlüsselindex-Scan.
Primärschlüsselspalte	Die Namen der Primärschlüsselspalten für einen Fremdschlüsselindex-Scan.

Elemente im Plan in Bezug auf Joins, Filter und Vorfilter

Element	Erklärung
Prädikate	Die Suchbedingung, die in diesem Knoten aufgelöst wird, sowie Selektivitätsschätzungen und –messungen.

Elemente im Plan in Bezug auf Hash-Filter

Element	Erklärung
Erzeugungswerte	Geschätzte Anzahl von unterschiedlichen Werten in der Eingabe.
Prüfwerte	Geschätzte Anzahl von unterschiedlichen Werten in der Eingabe beim Prüfen des Prädikats
Bits	Anzahl der Bits, die zum Aufbau der Hash-Map ausgewählt wurde
Seiten	Anzahl der Seiten, die zum Speichern der Hash-Map erforderlich sind

Elemente im Plan in Bezug auf Vereinigung

Element	Erklärung
Union-Liste	Spalten, die in eine UNION-Anweisung eingebunden sind

Elemente im Plan in Bezug auf GROUP BY

Element	Erklärung
Aggregate	Alle Aggregatfunktionen
Group by-Liste	Alle Spalten in der Group by-Klausel

Elemente im Plan in Bezug auf DISTINCT

Element	Erklärung
Distinct-Liste	Alle Spalten in der distinct-Klausel

Elemente im Plan in Bezug auf die IN-Liste

Element	Erklärung
In-Liste	Alle Ausdrücke in der angegebenen Menge
SQL-Ausdruck	Ausdrücke, die mit der Liste verglichen werden müssen

Elemente im Plan in Bezug auf SORT

Element	Erklärung
Order-by	Listet alle Ausdrücke, nach denen sortiert werden soll

Elemente im Plan in Bezug auf Zeilenbegrenzung

Element	Erklärung
Zeilenlimit gesamt	Maximale Anzahl von zurückgegebenen Zeilen, wie durch FIRST oder TOP <i>n</i> angegeben

Siehe auch

- „So funktioniert der Optimierer“ auf Seite 333
- „Cache und der Speichernutzungswächter“ auf Seite 221
- „Selektivitätsinformationen im grafischen Plan“ auf Seite 353
- „isolation_level-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*]
- „max_query_tasks-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „user_estimates-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „ansi_update_constraints-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Einschränkungen für materialisierte Ansichten“ auf Seite 61

Fortgeschrittene Aufgaben: Parallelität während der Abfrageausführung

SQL Anywhere unterstützt zwei unterschiedliche Arten von Parallelität bei der Abfrageausführung: abfrageinterne und abfrageübergreifende Parallelität. Abfrageübergreifende Parallelität (Inter-Query-Parallelität) bedeutet, dass verschiedene Abfragen gleichzeitig auf separaten CPUs ausgeführt werden.

Jede Anforderung (Task) läuft in einem einzelnen Thread und wird auf einem einzelnen Prozessor ausgeführt.

Abfrageinterne Parallelität (Intra-Query-Parallelität) bedeutet, dass mehrere CPUs gleichzeitig eine einzelne Anforderung verarbeiten, sodass Teile der Abfrage parallel durch Multi-Prozessor-Hardware berechnet werden. Die Verarbeitung dieser Teile wird vom Exchange-Algorithmus durchgeführt.

Die abfrageinterne Parallelität kann die Systemlast verbessern, wenn die Anzahl der gleichzeitig ausgeführten Abfragen normalerweise geringer ist als die Anzahl der verfügbaren Prozessoren. Der Höchstgrad der Parallelität wird durch die `max_query_tasks`-Option gesteuert.

Der Optimierer schätzt die Extrakosten der Parallelität (zusätzliches Kopieren von Zeilen, Zusatzkosten für Koordination der Aufgaben) und wählt parallele Pläne nur aus, wenn sie die Performance verbessern können.

Die abfrageinterne Parallelität wird nicht bei Verbindungen angewendet, für die die `Priority`-Option auf `"background"` gesetzt wurde.

Die abfrageinterne Parallelität wird nicht angewendet, wenn die Anzahl der Server-Threads, die gerade eine Anforderung bearbeiten (Servereigenschaft `"ActiveReq"`), die Anzahl von CPU-Kernen des Computers überschreitet, für dessen Benutzung der Datenbankserver lizenziert ist. Über die genaue Zeitspanne entscheidet der Server. Dabei handelt es sich normalerweise um ein paar Sekunden.

Parallelausführung

Ob eine Abfrage die Vorteile der parallelen Ausführung nutzen kann, hängt von einer Vielzahl von Faktoren ab:

- den verfügbaren Ressourcen im System zur Zeit der Optimierung (Arbeitsspeicher, Datenmenge im Cache usw.)
- der Anzahl von logischen Prozessoren auf dem Computer
- der Anzahl der Plattenmedien für die Speicherung der Datenbank sowie deren Geschwindigkeit relativ zum Prozessor und der I/O-Architektur des Computers.
- den spezifischen algebraischen Operatoren, die für die Anforderung erforderlich sind. SQL Anywhere unterstützt fünf algebraische Operatoren, die parallel ausgeführt werden können:
 - Paralleler sequenzieller Scan (Table-Scan)
 - Paralleler Index-Scan
 - Paralleler Hash-Join sowie parallele Versionen von Hash-Semijoin und Anti-Semijoin
 - Parallele Nested Loops Joins (Joins mit verschachtelten Schleifen) sowie parallele Versionen von Nested Loops Semijoins (Semijoins mit verschachtelten Schleifen) und Anti-Semijoins
 - Paralleler Hash-Filter
 - Paralleler Hash-Group-By

Eine Abfrage, die nicht unterstützte Operatoren verwendet, kann trotzdem parallel ausgeführt werden, jedoch müssen die unterstützten Operatoren im Plan unterhalb der nicht unterstützten Operatoren erscheinen (beim Anzeigen in Interactive SQL). Eine Abfrage, bei der die meisten nicht unterstützten Operatoren weit oben erscheinen können, wird mit größerer Wahrscheinlichkeit Parallelität verwenden.

Ein Sort-Operator kann beispielsweise nicht parallel ausgeführt werden, aber eine Abfrage, die ORDER BY im äußersten Block verwendet, kann parallel ausgeführt werden, indem der Sort-Operator an den Anfang des Plans gesetzt wird und alle parallelen Operatoren darunter. Im Gegensatz dazu wird eine Abfrage die Parallelität weniger wahrscheinlich benutzen, wenn sie TOP *n* und ORDER BY in einer abgeleiteten Tabelle verwendet, da der Sort-Operator an einer anderen Stelle als am Anfang des Plans erscheinen muss.

Standardmäßig geht SQL Anywhere davon aus, dass sich alle DBSpaces auf einem Platten-Subsystem mit einer einzelnen Platte befinden. Obwohl es Vorteile haben kann, in einer solchen Umgebung Abfragen parallel auszuführen, macht es das I/O-Kostenmodell des Optimierers für ein einzelnes Gerät schwierig, einen parallelen Table- oder Index-Scan zu wählen, wenn sich die Tabellendaten nicht vollständig im Cache befinden. Wenn Sie jedoch ein Platten-Subsystem mit der ALTER DATABASE CALIBRATE PARALLEL READ-Anweisung kalibrieren, kann der Optimierer die Kostenvorteile der parallelen Ausführung mit größerer Genauigkeit berechnen. Der Optimierer wählt mit hoher Wahrscheinlichkeit Ausführungspläne mit Parallelität, wenn das Platten-Subsystem über mehrere Platten verfügt.

Wenn für einen Zugriffsplan abfrageinterne Parallelität benutzt wird, enthält der Plan einen Exchange-Operator, der die Ergebnisse der parallelen Berechnung jeder Unterstruktur zusammenführt (Union). Die Anzahl der Unterstrukturen unterhalb des Exchange-Operators ist der Grad der Parallelität. Jede Unterstruktur oder Plankomponente ist eine Aufgabe für den Datenbankserver. Der Kernel des Datenbankservers plant die Ausführung dieser Aufgaben so, als wären es einzelne SQL-Anforderungen, basierend auf der Verfügbarkeit von Ausführungsthreads (oder Fibers). Diese Architektur bedeutet, dass die parallele Berechnung eines Zugriffsplans weitgehend selbstoptimierend ist, da die Arbeit für die parallele Ausführung einer Aufgabe in einem Thread (Fiber) geplant wird, wenn der Server-Kernel dies erlaubt, und die Ausführung der Plankomponenten gleichmäßig durchgeführt wird.

Siehe auch

- „Datenbankserveroption -gn“ [SQL Anywhere Server - Datenbankadministration]
- „max_query_tasks-Option“ [SQL Anywhere Server - Datenbankadministration]
- „SQL Anywhere-Threading“ [SQL Anywhere Server - Datenbankadministration]
- „Datenbankserveroption -gtc“ [SQL Anywhere Server - Datenbankadministration]
- „Datenbankserverkonfiguration der Multiprogramming-Stufe“ [SQL Anywhere Server - Datenbankadministration]
- „Fortgeschrittene Aufgaben: Abfrageausführungspläne“ auf Seite 341
- „ALTER DATABASE-Anweisung“ [SQL Anywhere Server - SQL-Referenzhandbuch]
- *Whitepaper Query Processing Based on SQL Anywhere 12.0.1 Architecture* unter <http://www.sybase.com/detail?id=1096047>
- „priority-Option“ [SQL Anywhere Server - Datenbankadministration]
- „Liste der Datenbankserveigenschaften“ [SQL Anywhere Server - Datenbankadministration]

Parallelität in Abfragen

Eine Abfrage wird mit größerer Wahrscheinlichkeit Parallelität verwenden, wenn die Abfrage wesentlich mehr Zeilen verarbeitet als zurückgegeben werden. In diesem Fall umfasst die Anzahl der verarbeiteten Zeilen die Größe aller gescannten Zeilen plus die Größe aller Zwischenergebnisse. Sie umfasst nur Zeilen, die gescannt wurden, weil ein Index zum Überspringen der meisten Teile der Tabelle verwendet wird. Ein idealer Fall ist ein GROUP BY für eine Einzelzeile über eine große Tabelle, wobei viele Zeilen gescannt

werden und nur eine zurückgegeben wird. Abfragen für mehrere Gruppen kommen ebenfalls in Frage, falls die Gruppen groß sind. Auch Prädikate oder Join-Bedingungen, die viele Zeilen löschen, eignen sich für die parallele Verarbeitung.

In der folgenden Liste werden die Situationen aufgeführt, in denen eine Abfrage die Vorteile der Parallelität nicht nutzen kann, weder bei der Optimierung noch während der Ausführungszeit:

- Der Computer verfügt nicht über mehrere Prozessoren.
- Der Servercomputer ist nicht für die Verwendung mehrerer Prozessoren lizenziert. Sie können dies anhand der Servereigenschaft "NumLogicalProcessorsUsed" prüfen. Hyperthread-Prozessoren werden für abfrageinterne Parallelität jedoch nicht gezählt. Wenn es sich also um einen Hyperthread-Computer handelt, müssen Sie den Wert von "NumLogicalProcessorsUsed" durch 2 teilen.
- Die Option "max_query_tasks" ist auf 1 gesetzt.
- Die Option "priority" ist auf "background" gesetzt.
- Die Anweisung, die die Abfrage enthält, ist keine SELECT-Anweisung.
- Der Wert von "ActiveReq" war kürzlich größer oder gleich dem Wert von "NumLogicalProcessorsUsed" (teilen Sie die Anzahl der Prozessoren durch 2, wenn es sich um einen Hyperthread-Computer handelt).
- Es gibt nicht genug verfügbare Aufgaben.

Siehe auch

- „Fortgeschrittene Aufgaben: Parallelität während der Abfrageausführung“ auf Seite 376
- „SQL Anywhere-Threading“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „max_query_tasks-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „priority-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- Eigenschaften "max_query_tasks", "priority", "NumLogicalProcessorsUsed" und "ActiveReq": „Liste der Datenbankservereigenschaften“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „CREATE DATABASE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Volltextsuche

Was ist eine Volltextsuche?

Die Volltextsuche stellt eine erweiterte Methode zum Durchsuchen einer Datenbank dar. Die Volltextsuche findet schnell alle Instanzen eines Begriffs (Worts) in einer Tabelle, ohne die Zeilen durchsuchen zu müssen und ohne wissen zu müssen, in welcher Spalte ein Begriff gespeichert ist. Bei der Volltextsuche werden **Textindizes** verwendet. Ein Textindex speichert Positionsinformationen für alle in den Spalten, für die Sie den Textindex erstellt haben, gefundenen Begriffe. Die Verwendung eines Textindexes kann bei der Suche nach Zeilen, die einen bestimmten Wert enthalten, gegenüber der Verwendung eines normalen Indexes schneller sein.

Die Funktion der Volltextsuche in SQL Anywhere unterscheidet sich von einer Suche mit Prädikaten wie LIKE, REGEXP und SIMILAR TO, weil die Übereinstimmung begriffsbasiert und nicht musterbasiert ist.

Zeichenfolgenvergleiche in der Volltextsuche verwenden alle üblichen Kollationseinstellungen für die Datenbank. Wenn die Datenbank beispielsweise dahingehend konfiguriert ist, die Groß-/Kleinschreibung nicht zu berücksichtigen, wird sie auch von der Volltextsuche nicht berücksichtigt.

Abgesehen von den angeführten Ausnahmen nutzt eine Volltextsuche alle internationalen Funktionen, die von SQL Anywhere unterstützt werden.

Hinweise zur Volltextsuche in einer Datenbank, die chinesische, japanische oder koreanische Daten (CJK) enthält, finden Sie im Whitepaper *"Performing Full Text Searches on Chinese, Japanese, and Korean Data in SQL Anywhere 11"* unter <http://www.sybase.com/detail?id=1061814>.

Zwei Methoden zum Durchführen einer Volltextsuche

Sie können eine Volltextabfrage durchführen, indem Sie entweder eine CONTAINS-Klausel in der FROM-Klausel einer SELECT-Anweisung oder eine CONTAINS-Suchbedingung (Prädikat) in einer WHERE-Klausel verwenden. Beide geben dieselben Zeilen zurück. Die Verwendung der CONTAINS-Klausel in einer FROM-Klausel liefert jedoch auch Treffer für übereinstimmende Zeilen.

Die folgenden Beispiele zeigen, wie die CONTAINS-Klausel und Suchbedingungen in einer Abfrage verwendet werden. Diese Beispiele verwenden den Beispiel-Textindex "MarketingInformation.Description" aus der Beispieldatenbank:

```
SELECT *
  FROM MarketingInformation CONTAINS ( Description, 'cotton' );

SELECT *
  FROM MarketingInformation
 WHERE CONTAINS ( Description, 'cotton' );
```

Hinweise zur Verwendung der Volltextsuche

Bei der Entscheidung, Volltextindizes statt regulärer Indizes zu verwenden, müssen Sie einige Faktoren berücksichtigen:

- Sie können in einer CONTAINS-Klausel oder einer CONTAINS-Suchbedingung keine Aliase verwenden.
- Bei der Verwendung doppelter Korrelationsnamen in einer Abfrage wird eine CONTAINS-Klausel (FROM CONTAINS()) nur bei der ersten Instanz des Korrelationsnamens unterstützt. Zum Beispiel gibt die folgende Syntax einen Fehler zurück, weil das zweite CONTAINS-Prädikat A involviert:

```
SELECT *
  FROM CONTAINS(A contains-query-string) JOIN B ON A.x = B.x,
       CONTAINS(A contains-query-string) JOIN C ON A.y = C.y;
```

Bei Verwendung externer Begriffsegmentierer und Vorfilterbibliotheken gibt es mehrere zusätzliche Dinge, die berücksichtigt werden müssen:

- **Abfragen und aktualisieren** Die externe Bibliothek muss für alle Vorgänge, die eine Aktualisierung oder Abfrage erfordern oder die den unter Verwendung der Bibliotheken erstellten Textindizes ändern, verfügbar bleiben.
- **Entladen und neu laden** Die externe Bibliothek muss während des Entladens und Neuladens von mit dem Volltextindex verknüpften Daten verfügbar sein.
- **Datenbankwiederherstellung** Die externe Bibliothek muss für die Wiederherstellung der Datenbank verfügbar sein. Dies liegt daran, dass die Datenbank nicht wiederhergestellt werden kann, wenn im Transaktionslog Vorgänge vorhanden sind, an denen seit dem letzten Checkpoint die externe Bibliothek beteiligt war.

Siehe auch

- „CONTAINS-Suchbedingung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Fortgeschrittene Aufgaben: Externe Begriffsegmentierer- und Vorfilterbibliotheken“ auf Seite 448
- „Konzepte und Referenz zu Textindizes“ auf Seite 423
- „Internationale Sprachen und Zeichensätze“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Kollationen“ [[SQL Anywhere Server - Datenbankadministration](#)]

Textkonfigurationsobjekte erstellen (Sybase Central)

Textkonfigurationsobjekte werden verwendet, wenn Sie Textindizes erstellen und aktualisieren. Sie können ein Textkonfigurationsobjekt in Sybase Central erstellen, indem Sie den **Assistenten zum Erstellen eines Textkonfigurationsobjekts** verwenden.

Voraussetzungen

Wenn Sie Textkonfigurationen für Objekte erstellen möchten, deren Eigentümer Sie sind, müssen Sie das CREATE TEXT CONFIGURATION-Systemprivileg haben.

Wenn Sie Textkonfigurationen für Objekte erstellen möchten, deren Eigentümer andere Benutzer sind, müssen Sie das CREATE ANY TEXT CONFIGURATION-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie auf **Textkonfigurationsobjekte** und klicken Sie auf **Neu » Textkonfigurationsobjekt**.
3. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen eines Textkonfigurationsobjekts**.
4. Klicken Sie auf den Fensterausschnitt **Textkonfigurationsobjekte**.

Ergebnisse

Das neue Textkonfigurationsobjekt wird im Fensterausschnitt **Textkonfigurationsobjekte** angezeigt.

Siehe auch

- „CREATE TEXT CONFIGURATION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten“ auf Seite 406
- „Beispiel-Textkonfigurationsobjekte“ auf Seite 414
- „Anzeigen eines Textkonfigurationsobjekts in der Datenbank“ auf Seite 383
- „ALTER TEXT CONFIGURATION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- Standard-Textkonfigurationsobjekte auf Seite 414

Ändern eines Textkonfigurationsobjekts

In Sybase Central können Administratoren Eigenschaften von Textkonfigurationsobjekten ändern, z.B. den Begriffsegmentiertyp, die Stoppliste und Optionseinstellungen.

Voraussetzungen

Sie müssen das CREATE EXTERNAL REFERENCE-Privileg haben.

Kontext und Bemerkungen

Ein Textindex hängt vom Textkonfigurationsobjekt ab, das zu seiner Erstellung verwendet wurde. Sie müssen daher abhängige Textindizes entweder kürzen oder löschen. Beachten Sie außerdem Folgendes: Wenn Sie vorhaben, die für das Textkonfigurationsobjekt gespeicherten Datums- oder Zeitformatoptionen zu ändern, müssen Sie eine Verbindung zur Datenbank herstellen und für die Optionen müssen dabei die gewünschten Einstellungen festgelegt sein.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie im linken Fensterausschnitt auf **Textkonfigurationsobjekt**.
3. Rechtsklicken Sie auf das Textkonfigurationsobjekt und klicken Sie auf **Eigenschaften**.
4. Bearbeiten Sie die Textkonfigurationsobjekt-Eigenschaften und klicken Sie auf **OK**.

Ergebnisse

Das Textkonfigurationsobjekt wird geändert.

Siehe auch

- „Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten“ auf Seite 406
- „Beispiel-Textkonfigurationsobjekte“ auf Seite 414
- „Anzeigen eines Textkonfigurationsobjekts in der Datenbank“ auf Seite 383
- „CREATE TEXT CONFIGURATION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TEXT CONFIGURATION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- Standard-Textkonfigurationsobjekte auf Seite 414

Anzeigen eines Textkonfigurationsobjekts in der Datenbank

Zeigen Sie die Einstellungen und andere Eigenschaften eines Textkonfigurationsobjekts in Sybase Central an.

Voraussetzungen

Sie müssen entweder Eigentümer des Textkonfigurationsobjekts sein oder das ALTER ANY TEXT CONFIGURATION-Systemprivileg oder das ALTER ANY OBJECT-Systemprivileg haben.

- ALTER ANY TEXT CONFIGURATION
- ALTER ANY OBJECT

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie im linken Fensterausschnitt auf **Textkonfigurationsobjekt**.
3. Rechtsklicken Sie auf das Textkonfigurationsobjekt und klicken Sie auf **Eigenschaften**.

Ergebnisse

Die Einstellungen für das Textkonfigurationsobjekt werden angezeigt.

Siehe auch

- „Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten“ auf Seite 406
- „SYSTEXTCONFIG-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Textindizes erstellen

Erstellen Sie Textindizes für Spalten jedes Typs. Spalten, die nicht vom Typ VARCHAR oder NVARCHAR sind, werden während der Indexierung in Zeichenfolgen konvertiert.

Voraussetzungen

Wenn Sie einen Textindex für eine Tabelle erstellen möchten, müssen Sie der Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- CREATE ANY INDEX-Systemprivileg
- CREATE ANY OBJECT-Systemprivileg
- REFERENCES-Privileg für die Tabelle und entweder das COMMENT ANY OBJECT-Systemprivileg, das ALTER ANY INDEX-Systemprivileg oder das ALTER 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

Sie können keinen Textindex erstellen, wenn mit der WITH HOLD-Klausel geöffnete Cursor vorhanden sind, die entweder Anweisungs- oder Transaktions-Snapshots verwenden.

Sie können keinen Textindex für eine reguläre Ansicht oder eine temporäre Tabelle erstellen. Sie können für eine deaktivierte materialisierte Ansicht keinen Textindex erstellen.

Kontext und Bemerkungen

Textindizes verbrauchen Plattenspeicher und müssen aktualisiert werden. Erstellen Sie sie nur für Spalten, die benötigt werden, um Ihre Abfragen zu unterstützen.

Wenn Sie mehr als einen, eine Spalte referenzierenden Textindex erstellen, kann dies zu unerwarteten Ergebnissen führen.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie auf die Registerkarte **Textindizes**.
3. Klicken Sie auf **Datei » Neu » Textindex**.
4. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen von Indizes**.

Der neue Textindex wird auf der Registerkarte **Textindizes** angezeigt. Er wird außerdem im Ordner **Textindizes** angezeigt.

Ergebnisse

Der Textindex wird erstellt. Wenn Sie einen Textindex mit sofortiger Aktualisierung erstellt haben, wird er automatisch mit Daten gefüllt. Bei den anderen Aktualisierungstypen müssen Sie den Textindex neu aufbauen.

Siehe auch

- „CREATE TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Textindizes aktualisieren“ auf Seite 385
- „Textindex-Aktualisierungstypen“ auf Seite 424
- „Datentypkonvertierungen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „REFRESH TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Textindizes aktualisieren

Bauen Sie Textindizes neu auf, um die Daten im Textindex zu aktualisieren. Der Neuaufbau eines Textindexes bewirkt, dass dieser alle Datenänderungen widerspiegelt, die in der Basistabelle vorgenommen wurden.

Voraussetzungen

Wenn Sie einen Textindex aktualisieren möchten, müssen Sie Eigentümer der Basistabelle sein oder eines der folgenden Privilegien haben:

- CREATE ANY INDEX-Systemprivileg
- CREATE ANY OBJECT-Systemprivileg
- ALTER ANY INDEX-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg
- REFERENCES-Privileg für die Tabelle

Sie können nur Textindizes aktualisieren, die als AUTO REFRESH und MANUAL REFRESH definiert sind. Sie können Textindizes nicht aktualisieren, die als IMMEDIATE definiert sind.

Kontext und Bemerkungen

Textindizes für materialisierte Ansichten werden aktualisiert, wenn die materialisierte Ansicht aktualisiert wird.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie im linken Fensterausschnitt auf **Textindizes**.
3. Rechtsklicken Sie auf den Textindex und klicken Sie auf **Daten aktualisieren**.
4. Wählen Sie eine Isolationsstufe für die Aktualisierung und klicken Sie auf **OK**.

Ergebnisse

Der Textindex wird aktualisiert.

Siehe auch

- „REFRESH TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Textindex-Aktualisierungstypen“ auf Seite 424

Einen Textindex ändern

Sie können die folgenden Merkmale eines Textindexes ändern:

- **Aktualisierungstyp** Sie können den Aktualisierungstyp von AUTO REFRESH auf MANUAL REFRESH bzw. umgekehrt ändern. Verwenden Sie die REFRESH-Klausel der ALTER TEXT INDEX-Anweisung, um den Aktualisierungstyp zu ändern.

Sie können einen Textindex nicht von bzw. auf IMMEDIATE REFRESH ändern. Um das durchzuführen, müssen Sie den Textindex löschen und neu erstellen.

- **Name** Sie können den Textindex umbenennen, indem Sie die RENAME-Klausel der Anweisung ALTER TEXT INDEX verwenden.
- **Inhalt** Mit Ausnahme der Spaltenliste werden Einstellungen, die die Indizierung steuern, in einem Textkonfigurationsobjekt gespeichert. Zum Ändern der indizierten Objekte ändern Sie das Textkonfigurationsobjekt, das ein Textindex referenziert. Sie müssen abhängige Textindizes kürzen, bevor Sie das Textkonfigurationsobjekt ändern können, und den Textindex aktualisieren, nachdem Sie das Textkonfigurationsobjekt geändert haben. Bei Textindizes mit sofortiger Aktualisierung müssen Sie den Textindex löschen und ihn wieder erstellen, nachdem Sie das Textkonfigurationsobjekt geändert haben.

Sie können einen Textindex nicht dahingehend ändern, dass er ein anderes Textkonfigurationsobjekt referenziert. Wenn der Textindex ein anderes Textkonfigurationsobjekt referenzieren soll, löschen Sie den Textindex und erstellen ihn unter Angabe des neuen Textkonfigurationsobjekts erneut.

Siehe auch

- „Textindex-Aktualisierungstypen“ auf Seite 424
- „Konzepte und Referenz zu Textkonfigurationsobjekten“ auf Seite 406
- „TRUNCATE TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ALTER TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „REFRESH TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DROP TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „sa_refresh_text_indexes-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Konzepte und Referenz zu Textindizes“ auf Seite 423

Textindizes ändern

Sie können den Namen eines Textindexes oder seinen Aktualisierungstyp ändern.

Voraussetzungen

Wenn Sie einen Textindex für eine Tabelle ändern möchten, müssen Sie der Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- ALTER ANY INDEX-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg
- REFERENCES-Privileg für die Tabelle und entweder das COMMENT ANY OBJECT-Systemprivileg, das CREATE ANY INDEX-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg

Wenn Sie einen Textindex 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

Sie können einen Textindex nicht dahingehend ändern, dass er ein anderes Textkonfigurationsobjekt referenziert. Wenn der Textindex ein anderes Textkonfigurationsobjekt referenzieren soll, löschen Sie den Textindex und erstellen ihn unter Angabe des neuen Textkonfigurationsobjekts erneut.

Sie können einen Textindex nicht von bzw. auf IMMEDIATE REFRESH ändern. Um das durchzuführen, müssen Sie den Textindex löschen und neu erstellen.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie im linken Fensterausschnitt auf **Textindizes**.
3. Rechtsklicken Sie auf den Textindex und klicken Sie auf **Eigenschaften**.
4. Bearbeiten Sie die Textindex-Eigenschaften.

Sie können den Textindex auf der Registerkarte **Allgemein** umbenennen.

5. Klicken Sie auf **OK**.

Ergebnisse

Der Textindex wird angezeigt.

Siehe auch

- „Textindex-Aktualisierungstypen“ auf Seite 424
- „Konzepte und Referenz zu Textkonfigurationsobjekten“ auf Seite 406
- „TRUNCATE TEXT INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TEXT INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REFRESH TEXT INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_refresh_text_indexes-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Konzepte und Referenz zu Textindizes“ auf Seite 423

Begriffe und Einstellungen für Textindizes anzeigen (Sybase Central)

Zeigen Sie die Begriffe und Einstellungen eines Textindexes in Sybase Central an.

Voraussetzungen

Wenn Sie vollständige Informationen zu einem Textindex anzeigen möchten, müssen Sie der Eigentümer der Tabelle oder materialisierten Ansicht sein oder eines der folgenden Systemprivilegien haben:

- CREATE ANY INDEX
- CREATE ANY OBJECT
- ALTER ANY INDEX
- ALTER ANY OBJECT
- DROP ANY INDEX
- DROP ANY OBJECT
- MANAGE ANY STATISTICS

Um Informationen in der Registerkarte **Vokabular** anzeigen zu können, benötigen Sie außerdem eines der folgenden Privilegien:

- SELECT-Privileg für die Tabelle oder materialisierte Ansicht, für die der Textindex erstellt wird
- SELECT ANY TABLE-Systemprivileg

Der Textindex muss initialisiert werden.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie im linken Fensterausschnitt auf **Textindizes**.
3. Um die Begriffe im Textindex anzuzeigen, doppelklicken Sie im linken Fensterausschnitt auf den Textindex und klicken Sie dann im rechten Fensterausschnitt auf die Registerkarte **Vokabular**.
4. Um die Textindex-Einstellungen wie den Aktualisierungstyp oder das vom Index referenzierte Textkonfigurationsobjekt anzuzeigen, rechtsklicken Sie auf den Textindex und klicken Sie auf **Eigenschaften**.

Ergebnisse

Die Begriffe und Einstellungen des Textindexes werden angezeigt.

Siehe auch

- „sa_text_index_stats-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSMVOPTION-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSMVOPTIONNAME-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSTAB-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Begriffe und Einstellungen für Textindizes anzeigen (SQL)

Zeigen Sie Begriffe und Einstellungen eines Textindexes in Interactive SQL an.

Voraussetzungen

Wenn Sie Einstellungen und statistische Informationen für einen Textindex anzeigen möchten, müssen Sie eines der folgenden Systemprivilegien haben:

- MANAGE ANY STATISTICS
- CREATE ANY INDEX
- ALTER ANY INDEX
- DROP ANY INDEX
- CREATE ANY OBJECT
- ALTER ANY OBJECT
- DROP ANY OBJECT

Wenn Sie Begriffe für einen Textindex anzeigen möchten, benötigen Sie außerdem eine der folgenden Privilegien:

- SELECT-Privileg für die Tabelle oder materialisierte Ansicht
- SELECT ANY TABLE-Systemprivileg

Der Textindex muss initialisiert werden.

Aufgabe

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Führen Sie die sa_text_index_stats-Systemprozedur aus, um statistische Informationen für einen Textindex anzuzeigen:

```
CALL sa_text_index_stats( );
```

3. Führen Sie die sa_text_index_vocab-Systemprozedur aus, um Begriffe für einen Textindex anzuzeigen:

```
CALL sa_text_index_vocab( );
```

Ergebnisse

Die statistischen Informationen und Begriffe für den Textindex werden angezeigt.

Nächste Schritte

Wenn ein Textindex erstellt wurde, werden die aktuellen Datenbankoptionen mit dem Textindex gespeichert. Zum Abrufen der während der Textindexerstellung verwendeten Optionseinstellungen führen Sie die folgende Anweisung aus:

```
SELECT b.object_id, b.table_name, a.option_id, c.option_name, a.option_value
FROM SYSMVOPTION a, SYSTAB b, SYSMVOPTIONNAME c
WHERE a.view_object_id=b.object_id
AND b.table_type=5;
```

Ein table_type von 5 in der SYSTAB-Ansicht ist ein Textindex.

Siehe auch

- „sa_text_index_stats-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSMVOPTION-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSMVOPTIONNAME-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSTAB-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Arten der Volltextsuche

Mit der Volltextsuche können Sie nach **Begriffen**, **Phrasen** (Begriffssequenzen) oder **Präfixen** suchen. Sie können auch mehrere Begriffe, Phrasen oder Präfixe in Boolesche Ausdrücke kombinieren oder mit einer Nachbarschaftssuche nach Ausdrücken suchen, die nahe beieinander auftreten.

Sie führen eine Volltextsuche aus, indem Sie eine CONTAINS-Klausel entweder in einer WHERE-Klausel oder einer FROM-Klausel einer SELECT-Anweisung benutzen. Sie können auch eine Volltextsuche als Teil der IF-Suchbedingung (z.B. SELECT IF CONTAINS . . .) ausführen.

Die folgenden Abschnitte zeigen, wie Sie die verschiedenen Typen von Volltextsuche durchführen, die in SQL Anywhere zur Verfügung stehen.

Begriffs- und Phrasensuche

Wenn Sie eine Volltextsuche mit einer Liste von Begriffen durchführen, ist die Reihenfolge der Begriffe ohne Bedeutung, außer sie sind Teil einer Phrase. Wenn Sie Begriffe in einer Phrase einfügen, sucht der Datenbankserver nach diesen Begriffen in genau derselben Reihenfolge und an denselben relativen Positionen, wie Sie sie eingegeben haben.

Wenn Begriffe bei einer Begriffs- oder Phrasensuche aus der Abfrage gelöscht werden, weil sie die Einstellungen für die Begriffslänge überschreiten oder sich in der Stoppliste befinden, können Sie eine andere als die erwartete Zeilenanzahl erhalten. Der Grund hierfür ist, dass ein Entfernen der Begriffe aus der Abfrage einer Änderung der Suchkriterien entspricht. Wenn Sie beispielsweise nach der Phrase ' "grown cotton" ' suchen und "grown" in der Stoppliste steht, erhalten Sie jede indizierte Zeile, die "cotton" enthält.

Sie können nach Begriffen suchen, die als Schlüsselwörter in der Grammatik der CONTAINS-Klausel angesehen werden, solange sie sich innerhalb von Phrasen befinden.

Begriffssuche

In der Beispieldatenbank wurde ein Textindex namens MarketingTextIndex für die Spalte "Description" der Tabelle "MarketingInformation" erstellt. Die folgende Anweisung fragt die Spalte "MarketingInformation.Description" ab und gibt die Zeilen zurück, bei denen der Wert der Spalte "Description" den Begriff **cotton** enthält.

```
SELECT ID, Description
FROM MarketingInformation
WHERE CONTAINS ( Description, 'cotton' );
```

ID	Description
906	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton construction. Shields against sun and precipitation.cotton Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor.</p></body></html>
908	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage.</p></body></html>
909	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% cotton/20% polyester blend makes it easy to keep them clean.</p></body></html>
910	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US><p>These quick-drying cotton shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist.</p></body></html>

Das folgende Beispiel fragt die Tabelle "MarketingInformation" ab und gibt einen einzelnen Wert für jede Zeile zurück, wobei angegeben wird, ob der Wert in der Spalte "Description" den Begriff **cotton** enthält.

```
SELECT ID, IF CONTAINS ( Description, 'cotton' )
THEN 1
```

```
ELSE 0
ENDIF AS Results
FROM MarketingInformation;
```

ID	Results
901	0
902	0
903	0
904	0
905	0
906	1
907	0
908	1
909	1
910	1

Das folgende Beispiel fragt die Tabelle "MarketingInformation" nach Einträgen ab, die den Begriff **cotton** in der Spalte "Description" haben, und zeigt den Punktwert für jede Übereinstimmung an.

```
SELECT ID, ct.score, Description
FROM MarketingInformation CONTAINS ( MarketingInformation.Description,
'cotton' ) as ct
ORDER BY ct.score DESC;
```

ID	score	Description
908	0.9461597363521859	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage.</p></body></html>

ID	score	Description
910	0.9244136988525732	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US><p>These quick-drying cotton shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist.</p></body></html>
906	0.9134171046194403	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor.</p></body></html>
909	0.8856420222728282	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% cotton/20% polyester blend makes it easy to keep them clean.</p></body></html>

Phrasensuche

Wenn Sie eine Volltextsuche mit einer Phrase durchführen, setzen Sie die Phrase in Anführungszeichen. Eine Spalte stimmt überein, wenn sie die Begriffe in der angegebenen Reihenfolge bzw. an den relativen Positionen enthält.

Sie können keine CONTAINS-Schlüsselwörter, wie z.B. AND oder FUZZY, als Begriffe angeben, nach denen gesucht werden soll, außer Sie setzen sie in eine Phrase (Ein-Begriff-Phrasen sind zulässig).
Beispiel: Die untenstehende Anweisung ist annehmbar, obwohl NOT ein CONTAINS-Schlüsselwort ist.

```
SELECT * FROM table-name CONTAINS ( Remarks, ' "NOT" ' );
```

Mit der Ausnahme des Sternchens werden Sonderzeichen nicht als Sonderzeichen interpretiert, wenn sie sich innerhalb einer Phrase befinden.

Phrasen können nicht als Argumente in Nachbarschaftssuchen verwendet werden.

Die folgende Anweisung fragt "MarketingInformation.Description" nach der Phrase "grown cotton" ab und zeigt den Punktwert für jede Übereinstimmung an:

```
SELECT ID, ct.score, Description
FROM MarketingInformation CONTAINS ( MarketingInformation.Description,
'grown cotton' ) as ct
ORDER BY ct.score DESC;
```

ID	score	Description
908	1.6619019465461564	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage.</p></body></html>
906	1.6043904700786786	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor.</p></body></html>

Siehe auch

- „Punktwerte für Ergebnisse einer Volltextsuche“ auf Seite 403
- „Präfixsuche“ auf Seite 394
- „CONTAINS-Suchbedingung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Präfixsuche

Die Volltextsuchfunktion ermöglicht es Ihnen, nach dem Anfang eines Begriffs zu suchen. Dies wird **Präfixsuche** genannt. Um eine Präfixsuche auszuführen, geben Sie das Präfix an, nach dem Sie suchen wollen, gefolgt von einem Sternchen. Dies wird als **Präfixbegriff** bezeichnet.

Schlüsselwörter für die CONTAINS-Klausel können nicht bei einer Präfixsuche verwendet werden, außer sie sind Teil einer Phrase.

Sie können auch mehrere Präfixausdrücke in einer Abfragezeichenfolge angeben, einschließlich innerhalb von Phrasen (z.B. ' "shi* fab" ').

Das folgende Beispiel fragt die Tabelle "MarketingInformation" nach Einträgen ab, die mit dem Präfix **shi** beginnen:

```

SELECT ID, ct.score, Description
FROM MarketingInformation CONTAINS ( MarketingInformation.Description,
'shi*' ) AS ct
ORDER BY ct.score DESC;

```

ID	score	Description
906	2.295363835537917	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor.</p></body></html>
901	1.6883275743936228	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>We've improved the design of this perennial favorite. A sleek and technical shirt built for the trail, track, or sidewalk. UPF rating of 50+.</p></body></html>
903	1.6336529491832605	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>A sporty, casual shirt made of recycled water bottles. It will serve you equally well on trails or around town. The fabric has a wicking finish to pull perspiration away from your skin.</p></body></html>
902	1.6181703448678983	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester fabric is gentle on the earth and soft against your skin.</p></body></html>

ID 906 hat die höchste Punktezahl, weil der Begriff "shield" im Textindex weniger häufig als "shirt" vorkommt.

Präfixsuchen in GENERIC-Textindizes

Bei GENERIC-Textindizes ist das Verhalten von Präfixsuchen folgendermaßen:

- Wenn ein Präfixbegriff länger als MAXIMUM TERM LENGTH ist, wird er aus der Abfragezeichenfolge gelöscht, weil es keine Begriffe im Textindex geben kann, die MAXIMUM TERM LENGTH überschreiten. Daher ist bei einem Textindex mit MAXIMUM TERM LENGTH 3 die Suche nach 'red appl*' gleichwertig mit der Suche nach 'red'.
- Wenn der Präfixbegriff kürzer als MINIMUM TERM LENGTH und nicht Teil einer Phrasensuche ist, wird die Präfixsuche normal durchgeführt. Daher gibt bei einem GENERIC-Textindex mit MINIMUM TERM LENGTH 5 die Suche nach 'macintosh a*' indizierte Zeilen zurück, die "macintosh" enthalten, sowie alle Begriffe, deren Länge 5 oder größer ist, die mit "a" beginnen.
- Wenn ein Präfixbegriff kürzer als MINIMUM TERM LENGTH, aber Teil einer Phrasensuche ist, wird der Präfixbegriff aus der Abfrage gelöscht. Daher ist bei einem GENERIC-Textindex, bei dem MINIMUM TERM LENGTH 5 ist, die Suche nach '"macintosh appl* turnover"' gleichwertig mit einer Suche nach "macintosh", gefolgt von einem beliebigen Begriff, dem "turnover" nachfolgt. Eine Zeile, die "macintosh turnover" enthält, wird nicht gefunden, da es einen Begriff zwischen "macintosh" und "turnover" geben muss.

Präfixsuchen in NGRAM-Textindizes

Bei NGRAM-Textindizes kann eine Präfixsuche unerwartete Ergebnisse zurückgeben, weil ein NGRAM-Textindex nur N-Gramme und keine Informationen über den Anfang von Begriffen enthält.

Abfragebegriffe werden ebenfalls in N-Gramme aufgeteilt und die Suche wird unter Verwendung der N-Gramme, und nicht der Abfragebegriffe, durchgeführt. Aus diesem Grund sollte das folgende Verhalten beachtet werden:

- Wenn ein Präfixbegriff kürzer als die N-Gramm-Länge (MAXIMUM TERM LENGTH) ist, gibt die Abfrage alle indizierten Zeilen zurück, die N-Gramme enthalten, welche mit dem Präfixbegriff beginnen. Bei einem 3-grams-Textindex gibt die Suche nach 'ea*' beispielsweise alle indizierten Zeilen zurück, die N-Gramme enthalten, welche mit "ea" beginnen. Wenn daher die Begriffe "weather" und "fear" indiziert wurden, würden die Zeilen als Übereinstimmungen angesehen werden, da ihre N-Gramme "eat" bzw. "ear" enthalten.
- Wenn ein Präfixbegriff länger als die N-Gramme-Länge und weder Teil einer Phrase noch ein Argument in einer Nachbarschaftssuche ist, wird der Präfixbegriff in einen N-Gramm-Phrase umgewandelt und das Sternchen gelöscht. Bei einem 3-gram-Textindex ist die Suche nach 'purple blac*' beispielsweise gleichwertig mit der Suche nach '"pur urp rpl ple" AND "blac" '.
- Bei Phrasen ist folgendes Verhalten zu beobachten:
 - Wenn der Präfixbegriff der einzige Begriff in der Phrase ist, wird er in eine N-Gramm-Phrase umgewandelt und das Sternchen gelöscht. Bei einem 3-gram-Textindex ist die Suche nach '"purpl*"' beispielsweise gleichwertig mit der Suche nach '"pur urp rpl"'.

- Wenn sich der Präfixbegriff an der letzten Position der Phrase befindet, wird das Sternchen gelöscht und die Begriffe werden zu einer Phrase aus N-Grammen umgewandelt. Bei einem 3-gram-Textindex ist die Suche nach `'"purple blac*"'` beispielsweise gleichwertig mit der Suche nach `'"pur urp rpl ple bla lac"'`.
- Wenn sich der Präfixbegriff nicht an der letzten Position der Phrase befindet, wird die Phrase in Phrasen aufgeteilt, die durch AND miteinander verbunden sind. Bei einem 3-gram-Textindex ist die Suche nach `'"purp* blac*"'` beispielsweise gleichwertig mit der Suche nach `'"pur urp" AND "bla lac"'`.
- Wenn ein Präfixbegriff ein Argument in einer Nachbarschaftssuche ist, wird die Nachbarschaftssuche zu einem AND umgewandelt. Bei einem 3-gram-Textindex ist die Suche nach `'red NEAR[1] appl*'` beispielsweise gleichwertig mit der Suche nach `'red AND "app ppl"'`.

Siehe auch

- [Zulässige Syntax für den Sternchen-Platzhalter \(*\) \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- [„Konzepte und Referenz zu Textindizes“ auf Seite 423](#)
- [„CONTAINS-Suchbedingung“ \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)

Nachbarschaftssuche

Die Volltext-Suchfunktion ermöglicht es Ihnen, nach Begriffen zu suchen, die in einer einzigen Spalte nahe beieinander liegen. Dies wird **Nachbarschaftssuche** genannt. Um eine Nachbarschaftssuche durchzuführen, geben Sie zwei Begriffe an, zwischen denen entweder das Schlüsselwort NEAR oder eine Tilde (~) steht.

Sie können ein Ganzzahlargument mit dem NEAR-Schlüsselwort verwenden, um die maximale Entfernung anzugeben. Beispiel: `term1 NEAR[5] term2` findet Instanzen von `term1`, die innerhalb von fünf Begriffen um `term2` auftreten. Die Reihenfolge der Begriffe ist nicht wichtig. `'term1 NEAR term2'` entspricht `'term2 NEAR term1'`.

Wenn Sie keine Entfernung angeben, verwendet der Datenbankserver 10 als Standardentfernung.

Sie können auch anstelle des NEAR-Schlüsselworts eine Tilde (~) verwenden. Beispiel: `'term1 ~ term2'`. Sie können allerdings keine Entfernung angeben, wenn Sie die Tildenform verwenden, und es wird der Standardwert von zehn Begriffen verwendet.

Sie können nicht eine Phrase als Argument in Nachbarschaftssuchen angeben.

Wenn Sie in einer Nachbarschaftssuche mit einem NGRAM-Textindex einen Präfixbegriff als Argument angeben, wird die Nachbarschaftssuche in einen AND-Ausdruck umgewandelt. Bei einem 3-gram-Textindex ist die Suche nach `'red NEAR[1] appl*'` beispielsweise gleichwertig mit der Suche nach `'red AND "app ppl"'`. Da dies nicht länger eine Nachbarschaftssuche ist, ist die Suche nicht mehr auf eine einzige Spalte beschränkt, falls mehrere Spalten in der CONTAINS-Klausel enthalten sind.

Beispiele

Angenommen, Sie wollen "MarketingInformation.Description" nach dem Begriff "fabric" innerhalb von 10 Begriffen um den Begriffs "skin" suchen. Sie können die folgende Anweisung ausführen.

```
SELECT ID, "contains".score, Description
FROM MarketingInformation CONTAINS ( Description, 'fabric ~ skin' );
```

ID	score	Description
902	1.5572371866083279	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester fabric is gentle on the earth and soft against your skin.</p></body></html>

Da die Standardentfernung 10 Begriffe ist, müssen Sie keine Entfernung angeben. Wenn Sie jedoch die Entfernung um einen Begriff erweitern, wird eine andere Zeile zurückgegeben:

```
SELECT ID, "contains".score, Description
FROM MarketingInformation CONTAINS ( Description, 'fabric NEAR[11]
skin' );
```

ID	score	Description
903	1.5787803210404958	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>A sporty, casual shirt made of recycled water bottles. It will serve you equally well on trails or around town. The fabric has a wicking finish to pull perspiration away from your skin.</p></body></html>
902	2.163125855043747	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester fabric is gentle on the earth and soft against your skin.</p></body></html>

Die Punktezahl für ID 903 ist höher, weil die Begriffe näher beieinander liegen.

Boolesche Suche

Sie können bei einer Volltextsuche mehrere Begriffe angeben, die durch Boolesche Operatoren getrennt sind. SQL Anywhere unterstützt die folgenden Booleschen Operatoren, wenn eine Volltextsuche durchgeführt wird: AND, OR und AND NOT.

Den AND-Operator in der Volltextsuche verwenden

Der AND-Operator findet eine Zeile, wenn sie beide Begriffe enthält, die links bzw. rechts des AND-Operators angegeben sind. Sie können auch ein kaufmännisches Und (&) für den AND-Operator verwenden. Wenn Begriffe ohne Operator angegeben sind, wird AND angenommen.

Die Reihenfolge, in der die Begriffe aufgelistet sind, ist unwichtig.

Beispiel: Jede der folgenden Anweisungen findet Zeilen in "MarketingInformation.Description", die den Begriff **fabric** und einen Begriff enthalten, der mit **ski** beginnt:

```
SELECT *
  FROM MarketingInformation
 WHERE CONTAINS ( MarketingInformation.Description, 'ski* AND fabric' );
SELECT *
  FROM MarketingInformation
 WHERE CONTAINS ( MarketingInformation.Description, 'fabric & ski*' );
SELECT *
  FROM MarketingInformation
 WHERE CONTAINS ( MarketingInformation.Description, 'ski* fabric' );
```

Den OR-Operator in der Volltextsuche verwenden

Der OR-Operator findet eine Zeile, wenn sie zumindest einen der links oder rechts von OR angegebenen Suchbegriffe enthält. Sie können auch einen Senkrechtrich (|) anstelle des OR-Operators verwenden. Die beiden sind äquivalent.

Die Reihenfolge, in der die Begriffe aufgelistet sind, ist unwichtig.

Beispiel: Beide der folgenden Anweisungen finden Zeilen in "MarketingInformation.Description", die entweder den Begriff **fabric** oder einen Begriff enthalten, der mit **ski** beginnt:

```
SELECT *
  FROM MarketingInformation
 WHERE CONTAINS ( MarketingInformation.Description, 'ski* OR fabric' );
SELECT *
  FROM MarketingInformation
 WHERE CONTAINS ( MarketingInformation.Description, 'fabric | ski*' );
```

Den AND NOT-Operator in der Volltextsuche verwenden

Der AND NOT-Operator findet Ergebnisse, die mit dem linken Argument, aber nicht dem rechten Argument übereinstimmen. Sie können auch einen Bindestrich (-) anstelle des AND NOT-Operators verwenden. Die beiden sind äquivalent.;

Beispiel: Die folgenden Anweisungen sind gleichwertig und geben Zeilen zurück, die den Begriff **fabric**, aber keine Begriffe enthalten, die mit **ski** beginnen.

```
SELECT *
  FROM MarketingInformation
 WHERE CONTAINS ( MarketingInformation.Description, 'fabric AND NOT
```

```
ski*' );
SELECT *
  FROM MarketingInformation
 WHERE CONTAINS ( MarketingInformation.Description, 'fabric -ski*' );
SELECT *
  FROM MarketingInformation
 WHERE CONTAINS ( MarketingInformation.Description, 'fabric & -ski*' );
```

Verschiedene Boolesche Operatoren kombinieren

Die Booleschen Operatoren können in einer Abfragezeichenfolge kombiniert werden. Beispiel: Die folgenden Anweisungen sind gleichwertig und durchsuchen die Spalte "MarketingInformation.Description" nach Einträgen, die **fabric** und **skin**, aber nicht **cotton** enthalten:

```
SELECT *
  FROM MarketingInformation
 WHERE CONTAINS ( MarketingInformation.Description, 'skin fabric -
cotton' );
SELECT *
  FROM MarketingInformation
 WHERE CONTAINS ( MarketingInformation.Description, 'fabric -cotton AND
skin' );
```

Die folgenden Anweisungen sind gleichwertig und durchsuchen die Spalte "MarketingInformation.Description" nach Einträgen, die **fabric** oder sowohl **cotton** als auch **skin** enthalten:

```
SELECT *
  FROM MarketingInformation
 WHERE CONTAINS ( MarketingInformation.Description, 'fabric | cotton AND
skin' );
SELECT *
  FROM MarketingInformation
 WHERE CONTAINS ( MarketingInformation.Description, 'cotton skin OR
fabric' );
```

Begriffe und Phrasen gruppieren

Begriffe und Ausdrücke können mit Klammern gruppiert werden. Beispiel: Die folgende Anweisung durchsucht die Spalte "MarketingInformation.Description" nach Einträgen, die **cotton** oder **fabric** sowie Begriffe enthalten, die mit **ski** beginnen.

```
SELECT ID, Description FROM MarketingInformation
 WHERE CONTAINS( MarketingInformation.Description, '( cotton OR fabric )
AND shi*' );
```

ID	Description
902	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester fabric is gentle on the earth and soft against your skin.</p></body></html>

ID	Description
903	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>A sporty, casual shirt made of recycled water bottles. It will serve you equally well on trails or around town. The fabric has a wicking finish to pull perspiration away from your skin.</p></body></html>
906	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>Light-weight 100% organically grown cotton construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor.</p></body></html>

Mehrere Spalten durchsuchen

Sie können eine Volltextsuche über mehrere Spalten in einer einzigen Abfrage durchführen, sofern die Spalten Teil desselben Textindexes sind.

```
SELECT *
  FROM t
 WHERE CONTAINS ( t.c1, t.c2, 'term1|term2' );
```

```
SELECT *
  FROM t
 WHERE CONTAINS( t.c1, 'term1' )
    OR CONTAINS( t.c2, 'term2' );
```

Die erste Abfrage findet Übereinstimmungen, wenn *t1.c1 term1* oder wenn *t1.c2 term2* enthält

Die zweite Abfrage findet Übereinstimmungen, wenn entweder *t1.c1* oder *t1.c2* entweder *term1* oder *term2* enthält. Diese Verwendung von CONTAINS gibt auch Punktwerte für die Übereinstimmungen zurück.

Siehe auch

- „CONTAINS-Suchbedingung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Punktwerte für Ergebnisse einer Volltextsuche“ auf Seite 403

Fuzzy-Suche

Die Fuzzy-Suche kann verwendet werden, um nach falschen Schreibweisen oder Variationen eines Worts zu suchen. Um dies zu tun, verwenden Sie den FUZZY-Operator, gefolgt von einer Zeichenfolge in Anführungszeichen, um eine angenäherte Übereinstimmung für die Zeichenfolge zu finden. Beispiel: CONTAINS (Products.Description, 'FUZZY "cotton"') gibt **cotton** und falsche Schreibweisen wie **coton** oder **cotten** zurück .

Hinweis

Sie können Fuzzy-Suchen nur auf Textindizes ausführen, die mit dem NGRAM-Begriffsegmentierer erstellt wurden. Weitere Hinweise über den NGRAM-Begriffsegmentierer und wie er mit Fuzzy-Suchen verwendet wird, finden Sie unter [„Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten“ auf Seite 406](#).

Die Verwendung des FUZZY-Operators ist gleichwertig damit, die Zeichenfolge manuell in Unterzeichenfolgen der Länge n aufzuteilen und diese mit OR-Operatoren voneinander zu trennen. Beispiel: Angenommen, Sie haben einen Textindex, der mit dem NGRAM- Begriffsegmentierer konfiguriert ist, und eine Begriff-Höchstlänge (MAXIMUM TERM LENGTH) von 3. Die Angabe 'FUZZY "500 main street"' ist äquivalent zu der Angabe '500 OR mai OR ain OR str OR tre OR ree OR eet'.

Der FUZZY-Operator ist in einer Volltextsuche nützlich, die einen Punktwert zurückgibt. Dies ist so, weil viele angenäherte Übereinstimmungen zurückgegeben werden können, aber gewöhnlich sind nur die Übereinstimmungen mit den höchsten Punktwerten sinnvoll.

Ansichtensuche

Um eine Volltextsuche auf eine Ansicht oder eine abgeleitete Tabelle anzuwenden, müssen Sie einen Textindex für die Spalten in der Basistabelle erstellen, in denen Sie die Volltextsuche ausführen möchten. Die folgenden Anweisungen erstellen eine Ansicht für die Tabelle "MarketingInformation" in der Beispieldatenbank, die bereits einen Textindexnamen besitzt, und führen dann eine Volltextsuche für diese Ansicht durch.

Um eine Ansicht für die Basistabelle "MarketingInformation" zu erstellen, führen Sie die folgende Anweisung aus:

```
CREATE VIEW MarketingInfoView AS
SELECT MI.ProductID AS ProdID,
       MI."Description" AS "Desc"
FROM GROUPO.MarketingInformation AS MI
WHERE MI."ID" > 3
```

Mit der folgenden Anweisung können Sie die Ansicht abfragen und dabei den Textindex für die Basistabelle verwenden.

```
SELECT *
FROM MarketingInfoView
WHERE CONTAINS ( "Desc", 'Cap OR Tee*' )
```

Sie können auch die folgende Anweisung ausführen, um eine abgeleitete Tabelle mit dem Textindex für die Basistabelle abzufragen.

```
SELECT *
FROM (
  SELECT MI.ProductID, MI."Description"
  FROM MarketingInformation AS MI
  WHERE MI."ID" > 4 ) AS dt ( P_ID, "Desc" )
WHERE CONTAINS ( "Desc", 'Base*' )
```

Hinweis

Die Spalten, für die Sie die Volltextsuche ausführen möchten, müssen in die SELECT-Liste der Ansicht bzw. der abgeleiteten Tabelle einbezogen werden.

Die Suche in einer Ansicht mithilfe eines Textindexes für die Basistabelle ist folgenden Einschränkungen unterworfen:

- Die Ansicht darf weder die Klauseln TOP, FIRST, DISTINCT, GROUP BY, ORDER BY, UNION, INTERSECT, EXCEPT noch eine Fensterfunktion enthalten.
- Die Ansicht darf keine Aggregatfunktionen enthalten.
- Eine CONTAINS-Abfrage kann sich auf eine Basistabelle in einer Ansicht beziehen, aber nicht auf eine Basistabelle in einer Ansicht in einer anderen Ansicht.

Siehe auch

- „CONTAINS-Suchbedingung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Punktwerte für Ergebnisse einer Volltextsuche

Wenn Sie eine CONTAINS-Klausel in eine FROM-Klausel einer Abfrage einbeziehen, hat jede Übereinstimmung einen ihr zugeordneten Punktwert. Der Punktwert zeigt an, wie genau die Übereinstimmung ist, und Sie können die Punktwertinformationen verwenden, um die Daten zu sortieren.

Die Bewertung basiert auf zwei Kriterien:

- **Die Häufigkeit, mit der ein Begriff in der indizierten Zeile erscheint.** Je häufiger ein Begriff in der indizierten Zeile auftritt, desto höher ist der Punktwert.
- **Die Häufigkeit, mit der ein Begriff im Textindex erscheint.** Je häufiger ein Begriff im Textindex erscheint, desto niedriger ist der Punktwert. In Sybase Central können Sie sehen, wie häufig ein Begriff im Textindex erscheint, indem Sie die Registerkarte **Vokabular** für den Textindex verwenden. Klicken Sie auf die Begriffsspalte, um die Begriffe alphabetisch zu sortieren. Die Häufigkeitsspalte zeigt an, wie oft der Begriff im Textindex erscheint.

Abhängig vom Typ der Volltextsuche wirken sich auch andere Kriterien auf die Bewertung aus. In Nachbarschaftssuchen wirkt sich beispielsweise die Nähe von Suchbegriffen zueinander auf die Bewertung aus.

Punktwerte anwenden

Standardmäßig hat die Ergebnismenge einer CONTAINS-Klausel den Korrelationsnamen **contains** und enthält eine einzelne Spalte namens **score**. Sie können "contains".score in der SELECT-Liste, in der ORDER BY-Klausel oder in anderen Teilen der Abfrage referenzieren. Da "contains" ein reserviertes SQL-Wort ist, dürfen Sie allerdings nicht vergessen, es in Anführungszeichen zu setzen. Als Alternative können Sie einen anderen Korrelationsnamen angeben (z.B. CONTAINS (expression) AS ct). In den Dokumentationsbeispielen für die Volltextsuche wird die Punktwertspalte als ct.score referenziert.

Die folgende Anweisung durchsucht "MarketingInformation.Description" nach Begriffen, die mit **stretch** oder mit **comfort** beginnen:

```
SELECT ID, ct.score, Description
      FROM MarketingInformation CONTAINS ( MarketingInformation.Description,
      'stretch* | comfort*' ) AS ct
      ORDER BY ct.score DESC;
```

ID	score	Description
910	5.570408968026068	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US><p>These quick-drying cotton shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist.</p></body></html>
907	3.658418186470189	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>A polycarbonate visor with an abrasion-resistant coating on the outside. Great for jogging in the spring, summer, and early fall. The elastic headband has plenty of stretch to give you a snug yet comfortable fit every time you wear it.</p></body></html>

ID	score	Description
905	1.6750365447462499	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US><p>A lightweight wool cap with mesh side vents for breathable comfort during aerobic activities. Moisture-absorbing headband liner.</p></body></html>

Artikel 910 hat den höchsten Punktwert, weil er zwei Instanzen des Präfixbegriffs **comfort** enthält, während andere nur eine Instanz haben. Außerdem hat Artikel 910 auch eine Instanz des Präfixbegriffs **stretch**.

Beispiel 2: Mehrere Spalten durchsuchen

Das folgende Beispiel zeigt, wie Sie eine Volltextsuche über mehrere Spalten ausführen und die Ergebnisse bewerten:

1. Erstellen Sie folgendermaßen einen sofortigen Textindex für die Products-Tabelle

```
CREATE TEXT INDEX scoringExampleMult
ON Products ( Description, Name );
```

2. Führen Sie wie folgt eine Volltextsuche auf den Spalten "Description" und "Name" für die Begriffe **cap** oder **visor** aus. Das Ergebnis der CONTAINS-Klausel wird dem Korrelationsnamen "ct" zugewiesen und in der SELECT-Liste referenziert, um in die Ergebnisse aufgenommen zu werden. Außerdem wird die Spalte "ct.score" in der ORDER BY-Klausel referenziert, um die Ergebnisse in absteigender Sortierfolge anhand der Punktwerte zu sortieren.

```
SELECT Products.Description, Products.Name, ct.score
FROM Products CONTAINS ( Products.Description, Products.Name, 'cap
OR visor' ) ct
ORDER BY ct.score DESC;
```

Description	Name	score
Cloth Visor	Visor	3.5635154905713042
Plastic Visor	Visor	3.4507856451176244
Wool cap	Baseball Cap	3.2340501745357333

Description	Name	score
Cotton Cap	Baseball Cap	3.090467108972918

Die Punktwerte bei einer Suche über mehrere Spalten werden berechnet, als ob die Spaltenwerte miteinander verkettet und als einzelner Wert indiziert wären. Beachten Sie allerdings, dass Phrasen und NEAR-Operatoren nie über mehrere Spaltengrenzen übereinstimmen, und ein Suchbegriff, der in mehr als einer Spalte erscheint, den Punktwert mehr erhöht, als er es in einem einzelnen verketteten Wert tun würde.

3. Damit die Beispiele in der Dokumentation richtig funktionieren, müssen Sie den Textindex löschen, den Sie für die Products-Tabelle erstellt haben. Um dies zu tun, führen Sie die folgende Anweisung aus:

```
DROP TEXT INDEX scoringExampleMult ON Products;
```

Konzepte und Referenz zu Textkonfigurationsobjekten

Ein Textkonfigurationsobjekt steuert, welche Begriffe in einen Textindex aufgenommen werden, wenn er erstellt oder aktualisiert wird, und wie eine Volltextabfrage interpretiert wird. Die Einstellungen für jedes Textkonfigurationsobjekt werden in einer Zeile in der ISYTEXTCONFIG-Systemtabelle gespeichert.

Wenn der Datenbankserver einen Textindex erstellt oder aktualisiert, verwendet er die Einstellungen für das Textkonfigurationsobjekt, die angegeben wurden, als der Textindex erstellt wurde. Wenn Sie bei der Erstellung des Textindexes kein Textkonfigurationsobjekt angegeben haben, wählt der Datenbankserver eines der Standard-Textkonfigurationsobjekte aus, basierend auf dem Typ der Daten in den Spalten, die indiziert werden. SQL Anywhere stellt zwei Standard-Textkonfigurationsobjekte bereit.

Um Einstellungen für vorhandene Textkonfigurationsobjekte anzuzeigen, führen Sie eine Abfrage in der SYSTEXTCONFIG-Systemansicht durch.

Siehe auch

- „Beispiel-Textkonfigurationsobjekte“ auf Seite 414
- „SYSTEXTCONFIG-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten

SQL Anywhere stellt zwei Standard-Textkonfigurationsobjekte bereit, default_char für die Verwendung mit CHAR-Daten und default_nchar für die Verwendung mit NCHAR- und CHAR-Daten. default_nchar kann zwar nicht mit beliebigen Daten verwendet werden, die Zeichensatzkonvertierung wird jedoch dennoch durchgeführt.

Sie können testen, wie sich ein Textkonfigurationsobjekt auf die Begriffsegmentierung auswirkt, indem Sie die Systemprozeduren sa_char_terms und sa_nchar_terms verwenden.

Siehe auch

- „sa_nchar_terms-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „sa_char_terms-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Beispiel-Textkonfigurationsobjekte“ auf Seite 414
- Standard-Textkonfigurationsobjekte auf Seite 414
- „Textkonfigurationsobjekte erstellen (Sybase Central)“ auf Seite 381
- „Ändern eines Textkonfigurationsobjekts“ auf Seite 382
- „Fuzzy-Suche“ auf Seite 401
- „Konzepte und Referenz zu Textindizes“ auf Seite 423
- „CONTAINS-Suchbedingung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE TEXT CONFIGURATION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ALTER TEXT CONFIGURATION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DROP TEXT CONFIGURATION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSTEXTIDX-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

TERM BREAKER-Klausel - Angeben des Begriffsegmentiereralgorithmus

Die TERM BREAKER-Einstellung gibt den Algorithmus an, der für die Aufteilung von Zeichenfolgen in Begriffe verwendet wird. Zur Auswahl stehen GENERIC, um Begriffe, und NGRAM, um N-Gramme zu speichern. Bei GENERIC können Sie den integrierten Begriffsegmentiereralgorithmus oder einen externen Begriffsegmentierer verwenden.

Die folgende Tabelle beschreibt die Auswirkungen des Werts von TERM BREAKER auf die Textindizierung und darauf, wie Abfragezeichenfolgen verarbeitet werden:

Textindizes	Abfragezeichenfolgen
<ul style="list-style-type: none"> GENERIC-Textindex Die Performance von GENERIC-Textindizes kann schneller sein als diejenige von NGRAM-Textindizes. Sie können allerdings keine Fuzzy-Suche in GENERIC-Textindizes durchführen. Beim Erstellen eines GENERIC-Textindex mit dem integrierten Algorithmus werden Gruppen von alphanumerischen Zeichen, die zwischen nicht-alphanumerischen Zeichen auftreten, vom Datenbankserver als Begriffe verarbeitet und es werden ihnen Positionen zugewiesen. Beim Erstellen eines GENERIC-Textindex mit einer externen Begriffsegmentierbibliothek, werden Begriffe und ihre jeweilige Position durch die externe Bibliothek definiert. Sobald die Begriffe durch den Begriffsegmentierer identifiziert wurden, wird jeder Begriff, der die Einschränkungen für die Begriffslänge nicht erfüllt oder in der Stoppliste enthalten ist, zwar gezählt, aber nicht in den Textindex eingefügt. NGRAM-Textindex Ein N-Gramm ist eine Gruppe von Zeichen mit der Länge n, wobei n der Wert von MAXIMUM TERM LENGTH ist. Wenn er einen NGRAM-Textindex aufbaut, behandelt der Datenbankserver jede Gruppe von alphanumerischen Zeichen zwischen nicht-alphanumerischen Zeichen wie einen Begriff. Wenn die Begriffe definiert sind, teilt der Datenbankserver die Begriffe in N-Gramme auf. Während dieses Vorgangs werden Begriffe, die kürzer als n sind, und N-Gramme, die in der Stoppliste enthalten sind, verworfen. Beispiel: Bei einem NGRAM-Textindex mit MAXIMUM TERM LENGTH 3 wird die Zeichenfolge 'my red table' im Textin- 	<p>Bei der syntaktischen Analyse einer CONTAINS-Abfrage extrahiert der Datenbankserver Schlüsselwörter und Sonderzeichen aus der Abfragezeichenfolge und wendet anschließend den Begriffsegmentieralgorithmus auf die verbleibenden Begriffe an. Wenn beispielsweise die Abfragezeichenfolge 'ab_cd* AND b*' ist, werden das Sternchen * und das Schlüsselwort AND extrahiert und die Zeichenfolgen ab_cd und b werden dem jeweiligen Begriffsegmentieralgorithmus zur getrennten syntaktischen Analyse übergeben.</p> <ul style="list-style-type: none"> GENERIC-Textindex Bei der Abfrage eines GENERIC-Textindex werden Begriffe in der Abfragezeichenfolge so abgearbeitet, als ob sie indiziert würden. Eine Übereinstimmung wird festgestellt, indem Abfragebegriffe mit Begriffen im Textindex verglichen werden. NGRAM-Textindex Bei der Abfrage eines NGRAM-Textindex werden Begriffe in der Abfragezeichenfolge so abgearbeitet, als ob sie indiziert würden. Eine Übereinstimmung wird festgestellt, indem N-Gramme aus den Abfragebegriffen mit N-Grammen aus den indizierten Begriffen verglichen werden.

Textindizes	Abfragezeichenfolgen
<p>dex als folgendes N-Gramm dargestellt: red tab abl ble.</p> <p>Bei N-Grammen werden die Positionsinformationen der N-Gramme gespeichert, und nicht die Positionsinformationen für die ursprünglichen Begriffe.</p>	

Sofern nicht anders definiert, wird der Standardwert für TERM BREAKER von der Einstellung im Standard-Textkonfigurationsobjekt bezogen. Wenn kein Begriffsegmentierer im Standard-Textkonfigurationsobjekt definiert ist, wird der interne Begriffsegmentierer verwendet.

Siehe auch

- „Präfixsuche“ auf Seite 394
- Standard-Textkonfigurationsobjekte auf Seite 414
- „CONTAINS-Suchbedingung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- TERM BREAKER-Klausel, ALTER TEXT CONFIGURATION-Anweisung [*SQL Anywhere Server - SQL-Referenzhandbuch*]

MINIMUM TERM LENGTH-Klausel - Festlegen der Begriff-Mindestlänge

Die MINIMUM TERM LENGTH-Einstellung gibt die Mindestlänge (in Zeichen) für Begriffe an, die in den Index eingefügt werden oder nach denen in einer Volltextabfrage gesucht wird. MINIMUM TERM LENGTH hat für NGRAM-Textindizes keine Relevanz.

MINIMUM TERM LENGTH hat spezielle Auswirkungen bei der Präfixsuche.

Der Wert von MINIMUM TERM LENGTH muss größer als 0 sein. Wenn Sie den Wert größer als MAXIMUM TERM LENGTH setzen, wird MAXIMUM TERM LENGTH automatisch angepasst, um gleich groß wie MINIMUM TERM LENGTH zu sein.

Sofern nicht anders definiert, wird der Standardwert für MINIMUM TERM LENGTH von der Einstellung im Standard-Textkonfigurationsobjekt bezogen, die üblicherweise 1 ist.

Die folgende Tabelle beschreibt die Auswirkungen des Werts von MINIMUM TERM LENGTH auf die Textindizierung und darauf, wie Abfragezeichenfolgen verarbeitet werden:

Textindizes	Abfragezeichenfolgen
GENERIC-Textindex Bei GENERIC-Textindizes enthält der Textindex keine Wörter, die kürzer als MINIMUM TERM LENGTH sind.	GENERIC-Textindex Wenn ein GENERIC-Textindex abgefragt wird, werden Abfragebegriffe, die kürzer als MINIMUM TERM LENGTH sind, ignoriert, weil sie nicht im Textindex vorkommen können.
NGRAM-Textindex Bei NGRAM-Textindizes wird diese Einstellung ignoriert.	NGRAM-Textindex Die MINIMUM TERM LENGTH-Einstellung hat bei Volltextabfragen in NGRAM-Textindizes keine Auswirkung.

Siehe auch

- [MINIMUM TERM LENGTH-Klausel, ALTER TEXT CONFIGURATION-Anweisung \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- [„Präfixsuche“ auf Seite 394](#)
- [Standard-Textkonfigurationsobjekte auf Seite 414](#)

MAXIMUM TERM LENGTH-Klausel - Festlegen der Begriffshöchstlänge

Die MAXIMUM TERM LENGTH-Einstellung wird, abhängig vom Begriffsegmentieralgorithmus, unterschiedlich verwendet.

Der Wert von MAXIMUM TERM LENGTH muss kleiner oder gleich 60 sein. Wenn Sie den Wert kleiner als MINIMUM TERM LENGTH setzen, wird MINIMUM TERM LENGTH automatisch angepasst, um gleich groß wie MAXIMUM TERM LENGTH zu sein.

Sofern nicht anders definiert, wird der Standardwert für MAXIMUM TERM LENGTH von der Einstellung im Standard-Textkonfigurationsobjekt bezogen, die üblicherweise 20 ist.

Die folgende Tabelle beschreibt die Auswirkungen des Werts von MAXIMUM TERM LENGTH auf die Textindizierung und darauf, wie Abfragezeichenfolgen verarbeitet werden:

Textindizes	Abfragezeichenfolgen
<p>GENERIC-Textindizes Bei GENERIC-Textindizes gibt MAXIMUM TERM LENGTH die Höchstlänge (in Zeichen) für Begriffe an, die in den Textindex eingefügt werden.</p> <p>NGRAM-Textindex Bei NGRAM-Textindizes legt MAXIMUM TERM LENGTH die Länge der N-Gramme fest, in die Begriffe aufgeteilt werden. Die entsprechende Längenauswahl für N-Gramme hängt von der Sprache ab. Üblich sind 4 oder 5 Zeichen für Englisch und 2 oder 3 Zeichen für Chinesisch.</p>	<p>GENERIC-Textindizes Bei GENERIC-Textindizes werden Abfragebegriffe ignoriert, die länger als MAXIMUM TERM LENGTH sind, weil sie im Textindex nicht vorkommen können.</p> <p>NGRAM-Textindex Bei NGRAM-Textindizes werden Abfragebegriffe in N-Gramme der Länge n aufgeteilt, wobei n dasselbe wie MAXIMUM TERM LENGTH ist. Anschließend verwendet der Datenbankserver die N-Gramme, um den Textindex zu durchsuchen. Begriffe, die kürzer als MAXIMUM TERM LENGTH sind, werden ignoriert, weil sie nicht mit den N-Grammen im Textindex übereinstimmen werden. Aus diesem Grund funktionieren Nachbarschaftssuchen nicht, es sei denn, die Argumente sind Präfixe der Länge n.</p>

Siehe auch

- [MAXIMUM TERM LENGTH-Klausel, ALTER TEXT CONFIGURATION-Anweisung \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- [Standard-Textkonfigurationsobjekte auf Seite 414](#)

STOPLIST-Klausel - Konfigurieren der Stoppliste

Eine Stoppliste gibt die Begriffe an, die beim Erstellen des Textindexes ignoriert werden sollen.

Sofern nicht anders definiert, wird der Standardwert für diese Einstellung von der Einstellung im Standard-Textkonfigurationsobjekt bezogen, die üblicherweise eine leere Stoppliste ist.

STOPLIST-Auswirkung auf Textindizes	STOPLIST-Auswirkung auf Abfragebegriffe
<p>GENERIC-Textindizes Bei GENERIC-Textindizes werden Begriffe, die in der Stoppliste enthalten sind, nicht in den Textindex eingefügt.</p> <p>NGRAM-Textindex Bei NGRAM-Textindizes enthält der Textindex nicht die N-Gramme, die aus den Begriffen in der Stoppliste geformt wurden.</p>	<p>GENERIC-Textindizes Bei GENERIC-Textindizes werden Abfragebegriffe ignoriert, die in der Stoppliste enthalten sind, weil sie im Textindex nicht vorkommen können.</p> <p>NGRAM-Textindex Begriffe in der Stoppliste werden in N-Gramme aufgeteilt und die N-Gramme werden für das Filtern von Begriffen verwendet. Gleichmaßen werden Abfragebegriffe in N-Gramme aufgeteilt und jene, die mit N-Grammen in der Stoppliste übereinstimmen, werden gelöscht, weil sie im Textindex nicht vorkommen können.</p>

Die Einstellungen im Textkonfigurationsobjekt werden auf die Stoppliste angewendet, wenn diese syntaktisch analysiert wird. Das heißt, der angegebene Begriffsegmentierer und die Einstellungen für minimale bzw. maximale Länge werden angewendet.

Bei NGRAM-Textindizes können Stopplisten zu unerwarteten Ergebnissen führen, weil die Stoppliste in N-Gramm-Form gespeichert wird, und nicht in Form der angegebenen Stopplistenbegriffe. Beispiel: Wenn Sie bei einem NGRAM-Textindex, bei dem MAXIMUM TERM LENGTH 3 ist, STOPLIST 'there' eingeben, werden die folgenden N-Gramme als Stoppliste gespeichert: the her ere. Dies wirkt sich auf die Fähigkeit aus, Begriffe abzufragen, die die N-Gramme "the" "her" und "ere" enthalten.

Hinweis

Die Einschränkungen, die für die Angabe von Zeichenfolgenliteralen gelten, gelten auch für Stopplisten. So müssen beispielsweise die Apostrophe mit Escapezeichen versehen werden, usw.

Das Beispielverzeichnis enthält Beispielcode, der Stopplisten für mehrere Sprachen lädt. Es wird empfohlen, diese Beispiel-Stopplisten nur bei GENERIC-Textindizes zu verwenden. Den Speicherort des Beispielverzeichnisses finden Sie unter [Beispielverzeichnis \[SQL Anywhere Server - Datenbankadministration\]](#).

Siehe auch

- [Standard-Textkonfigurationsobjekte auf Seite 414](#)
- [STOPLIST-Klausel, ALTER TEXT CONFIGURATION-Anweisung \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- [„Zeichenfolgenliterale“ \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)

PREFILTER-Klausel - Angeben des externen Vorfilter-Algorithmus

Vorfiltern ist der Prozess des Extrahierens von Textdaten aus Dateitypen wie z.B. Word, PDF, HTML und XML. Bei der Textindizierung, können Sie mit Vorfiltern nur die zu indizierenden Daten extrahieren und damit vermeiden, dass nicht benötigte Inhalte wie etwa HTML-Tags auch indiziert werden. Für bestimmte Typen von Dokumente (z.B. Microsoft Word-Dokumente) ist ein Vorfiltern erforderlich, damit Volltextindizes sinnvoll genutzt werden können.

SQL Anywhere bietet keine integrierte Vorfilter-Funktion. Sie können jedoch eine externe Vorfilterbibliothek zum Vorfiltern entsprechend Ihrer Anforderungen verwenden und dann Ihr Textkonfigurationsobjekt so ändern, dass es die Bibliothek benutzt.

Die folgende Tabelle beschreibt die Auswirkungen des Werts von PREFILTER EXTERNAL NAME auf die Textindizierung und darauf, wie Abfragezeichenfolgen verarbeitet werden:

Textindizes	Abfragezeichenfolgen
GENERIC- und NGRAM-Textindizes Ein externer Vorfilter erhält einen Eingabewert (ein Dokument) und filtert es gemäß den durch die Vorfilterbibliothek definierten Regeln. Der sich daraus ergebende Text wird dann vor dem Erstellen oder Aktualisieren des Textindex an den Begriffsegmentierer übergeben.	GENERIC- und NGRAM-Textindizes Abfragezeichenfolgen werden nicht durch einen Vorfilter geleitet, sodass die Einstellung der PREFILTER EXTERNAL NAME-Klausel keine Auswirkungen auf Abfragezeichenfolgen hat.

Das Verzeichnis *ExternalLibrariesFullText* in Ihrem SQL Anywhere-Installationsverzeichnis enthält Codebeispiele für Vorfilter und Begriffsegmentierer. Dieses Verzeichnis befindet sich im Verzeichnis *Samples*. Weitere Informationen zum Speicherort des Verzeichnisses *Samples* finden Sie unter [Beispielverzeichnis \[SQL Anywhere Server - Datenbankadministration\]](#).

Siehe auch

- „Externe Vorfilterbibliotheken“ auf Seite 451
- PREFILTER EXTERNAL NAME-Klausel, ALTER TEXT CONFIGURATION-Anweisung [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Formateinstellungen für Datum, Uhrzeit und Zeitstempel

Wenn ein Textkonfigurationsobjekt erstellt wird, werden die Werte der Optionen `date_format`, `time_format`, `timestamp_format` und `timestamp_with_time_zone_format` für die aktuelle Verbindung mit dem Textkonfigurationsobjekt gespeichert. Diese Optionswerte steuern, wie die Spalten DATE, TIME und TIMESTAMP für mit diesem Textkonfigurationsobjekt erstellte Textindizes formatiert werden. Sie können diese Optionswerte für das Textkonfigurationsobjekt nicht explizit einstellen. Die Einstellungen spiegeln die aktiven Einstellungen für die Verbindung wider, von der das Textkonfigurationsobjekt erstellt wurde. Sie können die Einstellungen jedoch ändern.

Siehe auch

- „Ändern eines Textkonfigurationsobjekts“ auf Seite 382
- „ALTER TEXT CONFIGURATION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Datenbankoptionen mit Auswirkungen auf Textkonfigurationsobjekte

Wenn ein Textkonfigurationsobjekt erstellt wird, werden die aktuellen Einstellungen für die Datenbankoptionen `date_format`, `time_format` und `timestamp_format` mit dem Textkonfigurationsobjekt gespeichert. Der Grund hierfür ist, dass diese Einstellungen sich auf Zeichenfolgenkonvertierungen beim Erstellen und bei der Aktualisierung von Textindizes auswirken, die vom Textkonfigurationsobjekt abhängen.

Das Speichern der Einstellungen mit dem Textkonfigurationsobjekt ermöglicht es Ihnen, die Einstellungen für diese Datenbankoptionen zu ändern, ohne eine Änderung am Format der Daten zu bewirken, die in den abhängigen Textindizes gespeichert sind.

Um das Format der Zeichenfolgen zu ändern, die in einem Textindex Datumsangaben und Uhrzeiten darstellen, müssen Sie Folgendes durchführen:

1. Löschen Sie den Textindex, das Textkonfigurationsobjekt und alle seine abhängigen Textindizes.
2. Löschen Sie das Standard-Textkonfigurationsobjekt, das Sie zum Erstellen des Textkonfigurationsobjekts und aller seiner abhängigen Textindizes verwendet haben.
3. Ändern Sie die Datenbankoptionen auf das gewünschte Format.
4. Erstellen Sie ein Textkonfigurationsobjekt.
5. Erstellen Sie einen Textindex mithilfe des neuen Textkonfigurationsobjektes.

Hinweis

Die Option `conversion_error` muss auf ON gesetzt sein, wenn Sie einen Textindex erstellen oder aktualisieren.

Siehe auch

- „Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten“ auf Seite 406
- „date_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „time_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „timestamp_format-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „conversion_error-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel-Textkonfigurationsobjekte

Um eine Liste aller Textkonfigurationsobjekte in der Datenbank und ihrer Einstellungen zu erhalten, führen Sie eine Abfrage in der Systemansicht SYSTEXTCONFIG durch (z.B. `SELECT * FROM SYSTEXTCONFIG`).

Sie können testen, wie ein Textkonfigurationsobjekt eine Zeichenfolge mit den Systemprozeduren `sa_char_terms` und `sa_nchar_terms` in Begriffe segmentiert.

Standard-Textkonfigurationsobjekte

SQL Anywhere stellt zwei Standard-Textkonfigurationsobjekte, **default_nchar** und **default_char**, für die Verwendung mit NCHAR- bzw. Nicht-NCHAR-Daten bereit. Diese Konfigurationen werden erstellt, wenn Sie erstmals versuchen, ein Textkonfigurationsobjekt oder einen Textindex zu erstellen.

Die Einstellungen für `default_char` und `default_nchar` zum Zeitpunkt der Installation werden in der untenstehenden Tabelle gezeigt. Diese Einstellungen wurden ausgewählt, weil sie für die meisten zeichenbasierten Sprachen besonders geeignet sind. Es wird ausdrücklich empfohlen, dass Sie diese Einstellungen in den Standard-Textkonfigurationsobjekten nicht ändern.

Einstellung	Installierter Wert
TERM BREAKER	0 (GENERIC)
MINIMUM TERM LENGTH	1
MAXIMUM TERM LENGTH	20
STOPLIST	(leer)

Wenn Sie ein Standard-Textkonfigurationsobjekt löschen, wird es automatisch wieder erstellt, wenn Sie das nächste Mal einen Textindex oder ein Textkonfigurationsobjekt erstellen.

Wenn ein Standard-Textkonfigurationsobjekt vom Datenbankserver erstellt wird, werden die Datenbankoptionen, die sich darauf auswirken, wie Datums- und Zeitwerte in Zeichenfolgen konvertiert werden, aus der aktuellen Verbindung heraus im Textkonfigurationsobjekt gespeichert.

Siehe auch

- „SYSTEXTCONFIG-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „sa_char_terms-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „sa_nchar_terms-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Datenbankoptionen mit Auswirkungen auf Textkonfigurationsobjekte“ auf Seite 413
- „Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten“ auf Seite 406

Beispiel-Textkonfigurationsobjekte

Die folgende Tabelle zeigt die Einstellungen für verschiedene Textkonfigurationsobjekte und wie sich die Einstellungen darauf auswirken, was indiziert wird und wie eine Volltext-Abfragezeichenfolge interpretiert wird. Alle Beispiele verwenden die Zeichenfolge 'I'm not sure I understand'.

Konfigurationseinstellungen	Begriffe, die indiziert werden	Abfrageinterpretation
TERM BREAKER GENERIC MINIMUM TERM LENGTH 1 MAXIMUM TERM LENGTH 20 STOPLIST "	I m not sure I understand	("I m" AND NOT sure) AND I AND understand' 'not' wird in der ursprünglichen Zeichenfolge als Operator, nicht als das Wort 'not' (nicht) interpretiert.

Konfigurationseinstellungen	Begriffe, die indiziert werden	Abfrageinterpretation
TERM BREAKER GENERIC MINIMUM TERM LENGTH 2 MAXIMUM TERM LENGTH 20 STOPLIST 'not and'	sure under- stand	'understand'. 'sure' wird gelöscht, da 'not' interpretiert als Operator (AND NOT) zwischen der Phrase "i am" und "sure" interpretiert wird. Da die Phrase "i am" Begriffe enthält, die zu kurz sind und daher gelöscht werden, wird die rechte Seite der AND NOT-Bedingung ('sure') ebenfalls gelöscht. Auf diese Weise bleibt nur 'understand' übrig.
TERM BREAKER NGRAM MAXIMUM TERM LENGTH 3 STOPLIST 'not and'	sur ure und nde der ers rst sta tan	'und AND nde AND der AND ers AND rst AND sta AND tan'. Bei einer Fuzzy Suche: 'und OR nde OR der OR ers OR rst OR sta OR tan'
TERM BREAKER GENERIC MINIMUM TERM LENGTH 1 MAXIMUM TERM LENGTH 20 STOPLIST 'not and'	I m sure I understand	' ("I m" AND NOT sure) AND I AND understand'.

Konfigurationseinstellungen	Begriffe, die indiziert werden	Abfrageinterpretation
TERM BREAKER NGRAM MAXIMUM TERM LENGTH 20 STOPLIST 'not and'	<p>Nichts wird indiziert, weil kein Begriff gleich oder länger als 20 Zeichen ist.</p> <p>Dies veranschaulicht, wie unterschiedlich sich MAXIMUM TERM LENGTH auf GENERIC- und NGRAM-Textindizes auswirkt. Bei NGRAM-Textindizes legt MAXIMUM TERM LENGTH die Länge der N-Gramme fest, die in die Textindizes eingefügt werden.</p>	Die Suche gibt eine leere Ergebnismenge zurück, weil kein N-Gramm mit 20 Zeichen aus der Abfragezeichenfolge geformt werden kann.

Beispiele für die Interpretation von CONTAINS-Zeichenfolgen

Die folgende Tabelle enthält Beispiele dafür, wie die Einstellungen der Textkonfigurationsobjekt-Zeichenfolgen interpretiert werden.

Die Zahlen in Klammern in der Spalte "Interpretierte Zeichenfolge" geben die Positionsinformationen wieder, die für jeden Begriff gespeichert sind. Die Zahlen dienen zur Illustration in der Dokumentation. Die tatsächlich gespeicherten Begriffe enthalten keine Zahlen in Klammern.

Konfigurationseinstellungen	Zeichenfolge	Interpretierte Zeichenfolge
TERM BREAKER GENERIC MINIMUM TERM LENGTH 3 MAXIMUM TERM LENGTH 20	'w*'	' "w*(1) " '
	'we*'	' "we*(1) " '
	'wea*'	' "wea*(1) " '
	'we* -the'	' "we*(1) " - "the(1) " '

Konfigurationseinstellungen	Zeichenfolge	Interpretierte Zeichenfolge
	'we* the'	"we*(1)" & "the(1)"
	'for* wonderl*'	'"for*(1)" "wonderl*(1)"'
	'wonderlandwonderland-wonderland*'	' '
	'"tr* weather"'	'"weather(1)"'
	'"tr* the weather"'	'"the(1) weather(2)"'
	'"wonderlandwonderland-wonderland* wonderland"'	'"wonderland(1)"'
	'"wonderlandwonderland-wonderland* weather"'	'"weather(1)"'
	'"the_wonderlandwonderlandwonderland* weather"'	'"the(1) weather(3)"'
	'the_wonderlandwonderlandwonderland* weather'	'"the(1)" & "weather(1)"'
	'"light_a* the end" & tunnel'	'"light(1) the(3) end(4)" & "tunnel(1)"'
	light_b* the end" & tunnel'	'"light(1) the(3) end(4)" & "tunnel(1)"'
	'"light_at_b* end"'	'"light(1) end(4)"'

Konfigurationseinstellungen	Zeichenfolge	Interpretierte Zeichenfolge
	'and_te*'	'"and(1) te*(2) "'
	'a_long_and_t* & journey'	'"long(2) and(3) t*(4) " & "jour- ney(1) "'
	'weather -is'	'"wea- ther(1) "'
TERM BREAKER NGRAM MAXIMUM TERM LENGTH 3	'w*'	'"w*(1) "'
	'we*'	'"we*(1) "'
	'wea*'	'"wea(1) "'
	'we* -the'	'"we*(1) " -"the(1) "'
	'we* the'	'"we*(1) " & "the(1) "'
	'for la*'	'"for(1) " "la*(1) "'
	'weath*'	'"wea(1) eat(2) ath(3) "'
	'"ful weat*"'	'"ful(1) wea(2) eat(3) "'
	'"wo* la*"'	'"wo*(1) " & "la*(2) "'
	'"la* won* "'	'"la*(1) " & "won(2) "'
	'"won* weat*"'	'"won(1) " & "wea(2) eat(3) "'

Konfigurationseinstellungen	Zeichenfolge	Interpretierte Zeichenfolge
	' "won* weat" '	' "won(1)" & "wea(2) eat(3)" '
	' "wo* weat*" '	' "wo*(1)" & "wea(2) eat(3)" '
	' "weat* wo* " '	' "wea(1) eat(2)" & "wo*(3)" '
	' "wo* weat" '	' "wo*(1)" & "wea(2) eat(3)" '
	' "weat wo* " '	' "wea(1) eat(2) wo*(3)" '
	'w* NEAR[1] f*'	' "w*(1)" & "f*(1)" '
	'weat* NEAR[1] f*'	' "wea(1) eat(2)" & "f*(1)" '
	'f* NEAR[1] weat*'	' "f*(1)" & "wea(1) eat(2)" '
	'weat NEAR[1] f*'	' "wea(1) eat(2)" & "f*(1)" '
	'f* NEAR[1] weat'	' "f*(1)" & "wea(1) eat(2)" '
	'for NEAR[1] weat*'	' "for(1)" & "wea(1) eat(2)" '

Konfigurationseinstellungen	Zeichenfolge	Interpretierte Zeichenfolge
	'weat* NEAR[1] for'	'"wea(1) eat(2)" & "for(1)"'
	'and_tedi*'	'"and(1) ted(2) edi(3)"'
	'and_t*'	'"and(1) t*(2)"'
	'"and_tedi*"'	'"and(1) ted(2) edi(3)"'
	'"and-t*"'	'"and(1) t*(2)"'
	'"ligh* at_the_end of_the tun* nel"'	'"lig(1) igh(2)" & ("the(4) end(5) the(7) tun(8)" & "nel(9)")'
	'"ligh* at_the_end_of_the_tun* nel"'	'"lig(1) igh(2)" & ("the(4) end(5) the(7) tun(8)" & "nel(9)")'
	'"at_the_end of_the tun* ligh* nel"'	'"the(2) end(3) the(5) tun(6)" & ("lig(7) igh(8)" & "nel(9)")'
	'l* NEAR[1] and_t*'	'"l*(1)" & "and(1) t*(2)"'

Konfigurationseinstellungen	Zeichenfolge	Interpretierte Zeichenfolge
	'long NEAR[1] and_t*'	'"lon(1) ong(2)" & "and(1) t*(2)"'
	'end NEAR[3] tunne*'	'"end(1)" & "tun(1) unn(2) nne(3)"'
TERM BREAKER NGRAM MAXIMUM TERM LENGTH 3 SKIPPED TOKENS IN TABLE AND IN QUERIES	'"cat in a hat"'	'"cat(1) hat(4)"'
	'"cat in_a hat"'	'"cat(1) hat(4)"'
	'"cat_in_a_hat"'	'"cat(1) hat(4)"'
	'"cat_in a_hat"'	'"cat(1) hat(4)"'
	'cat in a hat'	'"cat(1)" & "hat(1)"'
	'cat in_a hat'	'"cat(1)" & "hat(1)"'
	'"ice hat"'	'"ice(1) hat(2)"'
	'ice NEAR[1] hat'	'"ice(1)" NEAR[1] "hat(1)"'
	'ear NEAR[2] hat'	'"ear(1)" NEAR[2] "hat(1)"'
	'"ear a hat"'	'"ear(1) hat(3)"'
	'"cat hat"'	'"cat(1) hat(2)"'

Konfigurationseinstellungen	Zeichenfolge	Interpretierte Zeichenfolge
	'cat NEAR[1] hat'	'"cat(1)" NEAR[1] "hat(1)"'
	'ear NEAR[1] hat'	'"ear(1)" NEAR[1] "hat(1)"'
	'"ear hat"'	'"ear(1) hat(2)"'
	'"wear a a hat"'	'"wea(1) ear(2) hat(5)"'
	'weather -is'	'"wea(1) eat(2) ath(3) the(4) her(5)"'

Konzepte und Referenz zu Textindizes

Wenn Sie eine Volltextsuche ausführen, durchsuchen Sie einen **Textindex** (und nicht Tabellenzeilen). Daher müssen Sie, bevor Sie eine Volltextsuche durchführen können, einen Textindex für die Spalten erstellen, die Sie durchsuchen wollen. Ein Textindex speichert Positionsinformationen für Begriffe in den Spalten mit Index. Abfragen, die Textindizes verwenden, können schneller sein als solche, die alle Werte in der Tabelle durchsuchen müssen.

Wenn Sie einen Textindex erstellen, können Sie angeben, welches **Textkonfigurationsobjekt** für den Aufbau und die Aktualisierung des Textindex verwendet werden soll. Ein Textkonfigurationsobjekt enthält Einstellungen, die sich darauf auswirken, wie ein Index erstellt wird. Wenn Sie kein Textkonfigurationsobjekt angeben, verwendet der Datenbankserver ein Standard-Konfigurationsobjekt.

Sie können auch einen **Aktualisierungstyp** für den Textindex angeben. Der Aktualisierungstyp legt fest, wie oft der Textindex aktualisiert wird. Je aktueller der Textindex ist, desto akkurater sind seine Ergebnisse. Die Aktualisierung braucht allerdings Zeit und kann sich auf die Performance auswirken. Häufige Aktualisierungen einer Tabelle mit Index beispielsweise können sich auf die Performance auswirken, wenn der Textindex so konfiguriert ist, dass er jedes Mal neu aufgebaut wird, sobald sich die zugrunde liegenden Daten ändern.

Sie können mit der `VALIDATE TEXT INDEX`-Anweisung überprüfen, ob die Positionsdaten für die Begriffe im Textindex intakt sind. Falls die Positionsdaten beschädigt sind, wird ein Fehler generiert.

Um Einstellungen für vorhandene Textindizes anzuzeigen, verwenden Sie die Systemprozedur `sa_text_index_stats`.

Siehe auch

- „[VALIDATE-Anweisung](#)“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „[Textindex-Aktualisierungstypen](#)“ auf Seite 424
- „[Konzepte und Referenz zu Textkonfigurationsobjekten](#)“ auf Seite 406
- „[sa_text_index_stats-Systemprozedur](#)“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Textindex-Aktualisierungstypen

Wenn Sie einen Textindex erstellen, müssen Sie auch einen **Aktualisierungstyp** auswählen. Es gibt drei Aktualisierungstypen, die bei Textindizes unterstützt werden: Sofort, Automatisch und Manuell. Sie legen den Aktualisierungstyp für einen Textindex während seiner Erstellung fest. Abgesehen von sofortigen Textindizes können Sie den Aktualisierungstyp ändern, nachdem der Textindex erstellt wurde.

- **IMMEDIATE REFRESH** IMMEDIATE REFRESH-Textindizes werden aktualisiert, wenn sich die Daten in den zugrunde liegenden Tabellen oder materialisierten Ansichten ändern, und sind für Basistabellen nur zu empfehlen, wenn die Daten stets auf dem letzten Stand sein müssen, die Spalten mit Index verhältnismäßig kurz sind oder nur selten Datenänderungen vorgenommen werden.

Der Standard-Aktualisierungstyp für Textindizes ist IMMEDIATE REFRESH. Textindizes für materialisierte Ansichten unterstützen nur IMMEDIATE REFRESH.

Wenn Sie einen AUTO REFRESH- oder MANUAL REFRESH-Textindex verwenden, können Sie ihn nicht zu einem IMMEDIATE REFRESH-Textindex ändern. Stattdessen müssen Sie ihn löschen und ihn als IMMEDIATE REFRESH-Textindex neu erstellen.

IMMEDIATE REFRESH-Textindizes unterstützen alle Isolationsstufen. Sie werden zum Zeitpunkt der Erstellung mit Daten gefüllt und während dieses erstmaligen Neuaufbaus gilt eine Exklusivsperrung für die Tabelle oder die materialisierte Ansicht.

- **AUTO REFRESH** AUTO REFRESH-Textindizes werden automatisch in von Ihnen angegebenen Intervallen aktualisiert und sind zu empfehlen, wenn eine gewisse Datenveraltung akzeptabel ist. Eine Abfrage in einem veralteten Index gibt Daten zurück, die seit der letzten Aktualisierung nicht geändert wurden. Entsprechend werden Zeilen, die seit der letzten Aktualisierung eingefügt, gelöscht oder aktualisiert wurden, von einer Abfrage nicht zurückgegeben.

AUTO REFRESH-Textindizes werden möglicherweise auch häufiger als mit dem angegebenen Intervall aktualisiert, wenn mindestens eine der folgenden Bedingungen erfüllt ist:

- Die Zeitspanne seit der letzten Aktualisierung ist länger als das Aktualisierungsintervall.
- Die Gesamtlänge aller ausstehenden Zeilen (`pending_length` wie von der Systemprozedur `sa_text_index_stats` zurückgegeben) überschreitet 20 % der Gesamtgröße des Indexes (`doc_length` wie von `sa_text_index_stats` zurückgegeben).
- Die gelöschte Länge stellt über 50 % der Gesamtgröße des Indexes dar (`doc_length`). In diesem Fall wird immer ein vollständiger Neuaufbau ausgeführt und keine inkrementelle Aktualisierung.

AUTO REFRESH-Textindizes werden unter Verwendung der Isolationsstufe 0 aktualisiert.

Ein AUTO REFRESH-Textindex enthält keine Daten zum Zeitpunkt seiner Erstellung und steht erst nach der ersten Aktualisierung zur Verfügung, die üblicherweise innerhalb der ersten Minute nach der Erstellung des Textindexes stattfindet. Sie können einen AUTO REFRESH-Textindex auch manuell mit der REFRESH TEXT INDEX-Anweisung aktualisieren.

AUTO REFRESH-Textindizes werden während eines Neuladens nur aktualisiert, wenn die -g-Option für dbunload angegeben ist.

- MANUAL REFRESH** MANUAL REFRESH-Textindizes werden nur aktualisiert, wenn Sie sie aktualisieren, und sind zu empfehlen, wenn Daten in der zugrunde liegenden Tabelle selten geändert werden, ein höherer Grad von Datenverwaltung akzeptabel ist oder Sie eine Aktualisierung nach einem Ereignis oder einer erfüllten Bedingung durchführen. Eine Abfrage in einem veralteten Index gibt Daten zurück, die seit der letzten Aktualisierung nicht geändert wurden. Entsprechend werden Zeilen, die seit der letzten Aktualisierung eingefügt, gelöscht oder aktualisiert wurden, von einer Abfrage nicht zurückgegeben.

Sie können Ihre eigene Strategie zur Aktualisierung von MANUAL REFRESH-Textindizes festlegen. Im folgenden Beispiel werden alle MANUAL REFRESH-Textindizes unter Verwendung eines als Argument übergebenen Aktualisierungsintervalls und von Regeln, die den für AUTO REFRESH-Textindizes verwendeten Regeln ähneln, aktualisiert.

```
CREATE PROCEDURE refresh_manual_text_indexes(
    refresh_interval UNSIGNED INT )
BEGIN
    FOR lpl AS c1 CURSOR FOR
        SELECT ts.*
        FROM SYS.SYSTEXTIDX ti JOIN sa_text_index_stats( ) ts
        ON ( ts.index_id = ti.index_id )
        WHERE ti.refresh_type = 1 -- manual refresh indexes only
    DO
        BEGIN
            IF last_refresh_utc IS null
            OR cast(pending_length as float) / (
                IF doc_length=0 THEN NULL ELSE doc_length ENDIF) > 0.2
            OR DATEDIFF( MINUTE, CURRENT UTC TIMESTAMP, last_refresh_utc )
                > refresh_interval THEN
                EXECUTE IMMEDIATE 'REFRESH TEXT INDEX ' || text-index-name || ' ON "'
                || table-owner || '."' || table-name || '"';
            END IF;
        END;
    END FOR;
END;
```

Sie können die Systemprozedur sa_text_index_stats jederzeit verwenden, um zu entscheiden, ob eine Aktualisierung erforderlich ist und ob die Aktualisierung ein vollständiger Neuaufbau oder ein inkrementelles Aktualisieren sein soll.

Ein MANUAL REFRESH-Textindex enthält zum Zeitpunkt seiner Erstellung keine Daten und steht erst zur Verfügung, wenn Sie ihn aktualisiert haben. Um einen MANUAL REFRESH-Textindex zu aktualisieren, verwenden Sie die Anweisung REFRESH TEXT INDEX.

MANUAL REFRESH-Textindizes werden während des Neuladens nur aktualisiert, wenn die -g-Option für dbunload angegeben ist.

Siehe auch

- „Dienstprogramm zum Entladen (dbunload)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Textindizes erstellen“ auf Seite 383
- „Angaben beim Erstellen oder Ändern von Textkonfigurationsobjekten“ auf Seite 406
- „sa_text_index_stats-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE TEXT INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TEXT INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REFRESH TEXT INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „isolation_level-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „sa_text_index_stats-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Praktische Einführung: Volltextsuche in einem GENERIC-Textindex durchführen

Führen Sie eine Volltextsuche in einem Textindex aus, der einen GENERIC-Begriffsegmentierer verwendet.

Voraussetzungen

Sie benötigen die Systemprivilegien CREATE TEXT CONFIGURATION und CREATE TABLE. Außerdem benötigen Sie das SELECT ANY TABLE-Systemprivileg oder das SELECT-Privileg für die Tabelle "MarketingInformation".

Aufgabe

1. Starten Sie Interactive SQL. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Interactive SQL**.
2. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus:
 - a. Wählen Sie in der Dropdown-Liste **Authentifizierung** die Option **Datenbank** aus.
 - b. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - c. Im Feld **Kennwort** geben Sie **sql** ein.
 - d. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Mit einer ODBC-Datenquelle verbinden** aus.
3. Führen Sie die folgende Anweisung aus, um ein Textkonfigurationsobjekt namens myTxtConfig zu erstellen. Sie müssen die FROM-Klausel aufnehmen, um das Textkonfigurationsobjekt anzugeben, das als Vorlage verwendet werden soll.

```
CREATE TEXT CONFIGURATION myTxtConfig FROM default_char;
```

4. Führen Sie die folgende Anweisung aus, um das Textkonfigurationsobjekt anzupassen, indem Sie eine Stoppliste hinzufügen, die die Begriffe "because", "about", "therefore" und "only" enthält. Anschließend setzen Sie die Begriff-Höchstlänge auf 30.

```
ALTER TEXT CONFIGURATION myTxtConfig  
  STOPLIST 'because about therefore only';
```

```
ALTER TEXT CONFIGURATION myTxtConfig
  MAXIMUM TERM LENGTH 30;
```

5. Starten Sie Sybase Central. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Sybase Central**.
6. Klicken Sie auf **Verbindungen » Verbinden mit SQL Anywhere 16**.
7. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus:
 - a. Wählen Sie in der Dropdown-Liste **Authentifizierung** die Option **Datenbank** aus.
 - b. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - c. Im Feld **Kennwort** geben Sie **sql** ein.
 - d. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Mit einer ODBC-Datenquelle verbinden** aus.
8. Erstellen Sie eine Kopie der MarketingInformation-Tabelle.
 - a. Erweitern Sie den Ordner **Tabellen**.
 - b. Rechtsklicken Sie auf **MarketingInformation** und klicken Sie auf **Kopieren**.
 - c. Rechtsklicken Sie auf den Ordner **Tabellen** und klicken Sie auf **Einfügen**.
 - d. Im Feld **Name** geben Sie **MarketingInformation1** ein.
 - e. Klicken Sie auf **OK**.
9. In Interactive SQL führen Sie die folgende Anweisung aus, um die neue Tabelle mit Daten anzufüllen:

```
INSERT INTO MarketingInformation1
  SELECT * FROM GROUPO.MarketingInformation;
```

10. Für die Spalte "Description" der Tabelle "MarketingInformation1" in der Beispieldatenbank erstellen Sie einen Textindex, der das Textkonfigurationsobjekt "myTxtConfig" referenziert. Setzen Sie das Aktualisierungsintervall auf 24 Stunden.

```
CREATE TEXT INDEX myTxtIndex ON MarketingInformation1 ( Description )
  CONFIGURATION myTxtConfig
  AUTO REFRESH EVERY 24 HOURS;
```

11. Führen Sie die folgende Anweisung aus, um den Textindex zu aktualisieren:

```
REFRESH TEXT INDEX myTxtIndex ON MarketingInformation1;
```

12. Führen Sie die folgenden Anweisungen aus, um den Textindex zu testen:

- a. Diese Anweisung durchsucht den Textindex nach den Begriffen **cotton** oder **cap**. Die Ergebnisse werden anhand der Punktwerte in absteigender Sortierfolge sortiert. **Cap** hat eine höhere Bewertung als **cotton**, weil **cap** weniger häufig im Textindex vorkommt.

```
SELECT ID, Description, ct.*
  FROM MarketingInformation1
  CONTAINS ( Description, 'cotton | cap' ) ct
 ORDER BY score DESC;
```

ID	Description	Score
905	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US><p>A lightweight wool cap with mesh side vents for breathable comfort during aerobic activities. Moisture-absorbing headband liner.</p></body></html>	2.2742084275032632
904	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US><p>This fashionable hat is ideal for glacier travel, sea-kayaking, and hiking. With concealed draw cord for windy days.</p></body></html>	1.6980426550094467
908	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage.</p></body></html>	0.9461597363521859
910	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US><p>These quick-drying cotton shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist.</p></body></html>	0.9244136988525732

ID	Description	Score
906	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor.</p></body></html>	0.9134171046194403
909	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% cotton/20% polyester blend makes it easy to keep them clean.</p></body></html>	0.8856420222728282

- b. Die folgende Anweisung durchsucht den Textindex nach dem Begriff "cotton". Zeilen, die auch das Wort "visor" enthalten, werden verworfen. Die Ergebnisse werden nicht als Treffer betrachtet, da die CONTAINS-Klausel ein Prädikat verwendet.

```
SELECT ID, Description
FROM MarketingInformation1
WHERE CONTAINS( Description, 'cotton -visor' );
```

ID	Description
908	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage.</p></body></html>

ID	Description
909	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% cotton/20% polyester blend makes it easy to keep them clean.</p></body></html>
910	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US><p>These quick-drying cotton shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist.</p></body></html>

- c. Die folgende Anweisung testet jede Zeile auf den Begriff **cotton**. Wenn eine Zeile den Begriff enthält, erscheint eine 1 in der Spalte "Ergebnisse", ansonsten wird 0 zurückgegeben.

```
SELECT ID, Description, IF CONTAINS ( Description, 'cotton' )
      THEN 1
      ELSE 0
      ENDIF AS Results
FROM MarketingInformation1;
```

ID	Description	Results
901	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>We've improved the design of this perennial favorite. A sleek and technical shirt built for the trail, track, or sidewalk. UPF rating of 50+.</p></body></html>	0
902	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester fabric is gentle on the earth and soft against your skin.</p></body></html>	0

ID	Description	Results
903	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>A sporty, casual shirt made of recycled water bottles. It will serve you equally well on trails or around town. The fabric has a wicking finish to pull perspiration away from your skin.</p></body></html>	0
904	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US><p>This fashionable hat is ideal for glacier travel, sea-kayaking, and hiking. With concealed draw cord for windy days.</p></body></html>	0
905	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US><p>A lightweight wool cap with mesh side vents for breathable comfort during aerobic activities. Moisture-absorbing headband liner.</p></body></html>	0
906	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor.</p></body></html>	1
907	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>A polycarbonate visor with an abrasion-resistant coating on the outside. Great for jogging in the spring, summer, and early fall. The elastic headband has plenty of stretch to give you a snug yet comfortable fit every time you wear it.</p></body></html>	0

ID	Description	Results
908	<code><html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweat-shirt</title></head><body lang=EN-US><p>Light-weight 100% organically grown cotton hooded sweat-shirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage.</p></body></html></code>	1
909	<code><html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweat-shirt</title></head><body lang=EN-US><p>Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% cotton/20% polyester blend makes it easy to keep them clean.</p></body></html></code>	1
910	<code><html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US><p>These quick-drying cotton shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist.</p></body></html></code>	1

13. Schließen Sie Interactive SQL und Sybase Central.

Ergebnisse

Sie haben eine Volltextsuche in einem GENERIC-Textindex durchgeführt.

Nächste Schritte

(Optional) Setzen Sie die Beispieldatenbank (*demo.db*) in ihren ursprünglichen Zustand zurück. Siehe „[Neuerstellung der Beispieldatenbank \(demo.db\)](#)“ [*SQL Anywhere 16 - Einführung*].

Siehe auch

- „Konzepte und Referenz zu Textkonfigurationsobjekten“ auf Seite 406
- „CREATE TEXT CONFIGURATION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TEXT CONFIGURATION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Konzepte und Referenz zu Textindizes“ auf Seite 423
- „CREATE TEXT INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TEXT INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Praktische Einführung: Eine Fuzzy-Volltextsuche durchführen

Führen Sie eine Fuzzy-Volltextsuche in einem Textindex durch, der einen NGRAM-Begriffsegmentierer verwendet.

Voraussetzungen

Sie benötigen die Systemprivilegien CREATE TEXT CONFIGURATION und CREATE TABLE. Außerdem benötigen Sie das SELECT ANY TABLE-Systemprivileg oder das SELECT-Privileg für die Tabelle "MarketingInformation".

Aufgabe

1. Starten Sie Interactive SQL. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Interactive SQL**.
2. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus:
 - a. Wählen Sie in der Dropdown-Liste **Authentifizierung** die Option **Datenbank** aus.
 - b. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - c. Im Feld **Kennwort** geben Sie **sql** ein.
 - d. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Mit einer ODBC-Datenquelle verbinden** aus.
3. Führen Sie die folgende Anweisung aus, um ein Textkonfigurationsobjekt namens myFuzzyTextConfig zu erstellen. Sie müssen die FROM-Klausel aufnehmen, um das Textkonfigurationsobjekt anzugeben, das als Vorlage verwendet werden soll.

```
CREATE TEXT CONFIGURATION myFuzzyTextConfig FROM default_char;
```

4. Führen Sie die folgenden Anweisungen aus, um den Begriffsegmentierer auf NGRAM zu ändern und die Begriff-Höchstlänge auf 3 festzulegen. Fuzzy-Suchen werden unter Verwendung von N-Grammen durchgeführt.

```
ALTER TEXT CONFIGURATION myFuzzyTextConfig
  TERM BREAKER NGRAM;
ALTER TEXT CONFIGURATION myFuzzyTextConfig
  MAXIMUM TERM LENGTH 3;
```

5. Starten Sie Sybase Central. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Sybase Central**.
6. Klicken Sie auf **Verbindungen » Verbinden mit SQL Anywhere 16**.
7. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus:
 - a. Wählen Sie in der Dropdown-Liste **Authentifizierung** die Option **Datenbank** aus.
 - b. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - c. Im Feld **Kennwort** geben Sie **sql** ein.
 - d. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Mit einer ODBC-Datenquelle verbinden** aus.
8. Erstellen Sie eine Kopie der MarketingInformation-Tabelle.
 - a. In Sybase Central erweitern Sie den Ordner **Tabellen**.
 - b. Rechtsklicken Sie auf **MarketingInformation** und klicken Sie auf **Kopieren**.
 - c. Rechtsklicken Sie auf den Ordner **Tabellen** und klicken Sie auf **Einfügen**.
 - d. Im Feld **Name** geben Sie **MarketingInformation2** ein. Klicken Sie auf **OK**.
9. Führen Sie in Interactive SQL die folgende Anweisung aus, um der Tabelle "MarketingInformation2" Daten hinzuzufügen:

```
INSERT INTO MarketingInformation2
SELECT * FROM GROUPO.MarketingInformation;
```

10. Führen Sie die folgende Anweisung aus, um einen Textindex für die Spalte "MarketingInformation2.Description" zu erstellen, der das Textkonfigurationsobjekt "myFuzzyTextConfig" referenziert:

```
CREATE TEXT INDEX myFuzzyTextIdx ON MarketingInformation2 ( Description )
CONFIGURATION myFuzzyTextConfig;
```

11. Führen Sie die folgende Anweisung aus, um auf Begriffe zu prüfen, die **coten** ähnlich sind:

```
SELECT MarketingInformation2.Description, ct.*
FROM MarketingInformation2 CONTAINS
( MarketingInformation2.Description, 'FUZZY "coten"' ) ct
ORDER BY ct.score DESC;
```

Description	Score
<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage.</p></body></html>	0.9461597363521859

Description	Score
<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US><p>These quick-drying cotton shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist.</p></body></html>	0.9244136988525732
<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor.</p></body></html>	0.9134171046194403
<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% cotton/20% polyester blend makes it easy to keep them clean.</p></body></html>	0.8856420222728282
<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US><p>This fashionable hat is ideal for glacier travel, sea-kayaking, and hiking. With concealed draw cord for windy days.</p></body></html>	0

Description	Score
<pre><html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US><p>A lightweight wool cap with mesh side vents for breathable comfort during aerobic activities. Moisture-absorbing headband liner.</p></body></html></pre>	0
<pre><html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>We've improved the design of this perennial favorite. A sleek and technical shirt built for the trail, track, or sidewalk. UPF rating of 50+.</p></body></html></pre>	0
<pre><html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>A sporty, casual shirt made of recycled water bottles. It will serve you equally well on trails or around town. The fabric has a wicking finish to pull perspiration away from your skin.</p></body></html></pre>	0
<pre><html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester fabric is gentle on the earth and soft against your skin.</p></body></html></pre>	0

Description	Score
<pre><html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>A polycarbonate visor with an abrasion-resistant coating on the outside. Great for jogging in the spring, summer, and early fall. The elastic headband has plenty of stretch to give you a snug yet comfortable fit every time you wear it.</p></body></html></pre>	0

Hinweis

Die letzten sechs Zeilen haben Begriffe, die übereinstimmende N-Gramme enthalten. Es wurden ihnen allerdings keine Punktwerte zugeteilt, weil alle Zeilen in der Tabelle diese Begriffe enthalten.

12. Schließen Sie Interactive SQL und Sybase Central.

Ergebnisse

Sie haben eine Fuzzy-Volltextsuche durchgeführt.

Nächste Schritte

(Optional) Setzen Sie die Beispieldatenbank (*demo.db*) in ihren ursprünglichen Zustand zurück. Siehe „[Neuerstellung der Beispieldatenbank \(demo.db\)](#)“ [*SQL Anywhere 16 - Einführung*].

Siehe auch

- „Fuzzy-Suche“ auf Seite 401
- „Konzepte und Referenz zu Textkonfigurationsobjekten“ auf Seite 406
- „CREATE TEXT CONFIGURATION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TEXT CONFIGURATION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Konzepte und Referenz zu Textindizes“ auf Seite 423
- „CREATE TEXT INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TEXT INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Punktwerte für Ergebnisse einer Volltextsuche“ auf Seite 403

Praktische Einführung: Nicht-Fuzzy-Volltextsuche in einem NGRAM-Textindex durchführen

Führen Sie eine Nicht-Fuzzy-Volltextsuche in einem Textindex durch, der einen NGRAM-Begriffsegmentierer verwendet. Dieses Verfahren kann auch verwendet werden, um eine Volltextsuche von chinesischen, japanischen oder koreanischen Daten zu erstellen.

Voraussetzungen

Sie benötigen die Systemprivilegien `CREATE TEXT CONFIGURATION` und `CREATE TABLE`. Außerdem benötigen Sie das `SELECT ANY TABLE`-Systemprivileg oder das `SELECT`-Privileg für die Tabelle "MarketingInformation".

Kontext und Bemerkungen

In Datenbanken mit Mehrbyte-Zeichensätzen werden möglicherweise einige Satzzeichen und Leerzeichen wie Kommas und Leerstellen mit voller Breite als alphanumerische Zeichen behandelt.

Aufgabe

1. Starten Sie Interactive SQL. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Interactive SQL**.
2. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus:
 - a. Wählen Sie in der Dropdown-Liste **Authentifizierung** die Option **Datenbank** aus.
 - b. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
 - c. Im Feld **Kennwort** geben Sie **sql** ein.
 - d. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Mit einer ODBC-Datenquelle verbinden** aus.

3. Führen Sie die folgende Anweisung aus, um ein NCHAR-Textkonfigurationsobjekt namens myNcharNGRAMTextConfig zu erstellen.

```
CREATE TEXT CONFIGURATION myNcharNGRAMTextConfig FROM default_nchar;
```

4. Führen Sie die folgende Anweisung aus, um den TERM BREAKER-Algorithmus auf NGRAM zu ändern und MAXIMUM TERM LENGTH (N) auf 2 zu setzen:

```
ALTER TEXT CONFIGURATION myNcharNGRAMTextConfig  
  TERM BREAKER NGRAM;  
ALTER TEXT CONFIGURATION myNcharNGRAMTextConfig  
  MAXIMUM TERM LENGTH 2;
```

Bei chinesischen, japanischen und koreanischen Daten ist der empfohlene Wert für N 2 oder 3. Bei Suchen, die auf ein oder zwei Zeichen beschränkt sind, setzen Sie den N-Wert auf 1. Das Festlegen des N-Werts auf 1 kann eine langsamere Ausführung von langen Abfragen bewirken.

5. Starten Sie Sybase Central. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Sybase Central**.
6. Klicken Sie auf **Verbindungen » Verbinden mit SQL Anywhere 16**.
7. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus:
 - a. Wählen Sie in der Dropdown-Liste **Authentifizierung** die Option **Datenbank** aus.
 - b. Im Feld **Benutzer-ID** geben Sie **DBA** ein.

- c. Im Feld **Kennwort** geben Sie **sql** ein.
 - d. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Mit einer ODBC-Datenquelle verbinden** aus.
8. Erstellen Sie eine Kopie der MarketingInformation-Tabelle.
- a. Erweitern Sie den Ordner **Tabellen**.
 - b. Rechtsklicken Sie auf **MarketingInformation** und klicken Sie auf **Kopieren**.
 - c. Rechtsklicken Sie auf den Ordner **Tabellen** und klicken Sie auf **Einfügen**.
 - d. Im Feld **Name** geben Sie **MarketingInformationNgram** ein.
 - e. Klicken Sie auf **OK**.
9. Führen Sie in Interactive SQL die folgende Anweisung aus, um der Tabelle "MarketingInformationNgram" Daten hinzuzufügen:

```
INSERT INTO MarketingInformationNgram
SELECT * FROM GROUPO.MarketingInformation;
COMMIT;
```

10. Führen Sie die folgende Anweisung aus, um einen IMMEDIATE REFRESH-Textindex für die Spalte "MarketingInformationNgram.Description" unter Verwendung des Textkonfigurationsobjekts "myNcharNGRAMTextConfig" zu erstellen:

```
CREATE TEXT INDEX ncharNGRAMTextIndex
ON MarketingInformationNgram( Description )
CONFIGURATION myNcharNGRAMTextConfig;
```

11. Testen Sie den Textindex.

- a. Die folgende Anweisung durchsucht den 2-Gramm-Textindex nach Begriffen, die **sw** enthalten. Die Ergebnisse werden anhand der Punktwerte in absteigender Sortierfolge sortiert.

```
SELECT M.Description, ct.*
FROM MarketingInformationNgram AS M
CONTAINS( M.Description, 'sw' ) ct
ORDER BY ct.score DESC;
```

Description	Score
<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton hooded Sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage.</p></body></html>	2.262071918398649

Description	Score
<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% cotton/20% polyester blend makes it easy to keep them clean.</p></body></html>	1.5556043490424176

- b. Die folgende Anweisung sucht nach Begriffen, die **ams** enthalten. Die Ergebnisse werden anhand der Punktwerte in absteigender Sortierfolge sortiert.

```
SELECT M.Description, ct.*
FROM MarketingInformationNgram AS M
CONTAINS( M.Description, 'ams' ) ct
ORDER BY ct.score DESC;
```

Beim 2-gram-Textindex ist die vorherige Anweisung äquivalent zu:

```
SELECT M.Description, ct.*
FROM MarketingInformationNgram AS M
CONTAINS( M.Description, '"am ms"' ) ct
ORDER BY ct.score DESC;
```

Beide Anweisungen geben die folgenden Ergebnisse zurück:

Description	Score
<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage.</p></body></html>	1.6619019465461564

Description	Score
<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% cotton/20% polyester blend makes it easy to keep them clean.</p></body></html>	1.5556043490424176

- c. Die folgende Anweisung sucht nach Begriffen mit **v** gefolgt von einem beliebigen alphanumerischen Zeichen. Da **ve** in den indizierten Daten häufiger vorkommt, erhalten Zeilen, die das 2-GRAM **ve** enthalten, eine niedrigere Bewertung als Zeilen, die **vi** enthalten. Die Ergebnisse werden anhand der Punktwerte in absteigender Sortierfolge sortiert.

```
SELECT M.ID, M.Description, ct.*
FROM MarketingInformationNgram AS M
CONTAINS( M.Description, 'v*' ) ct
ORDER BY ct.score DESC;
```

ID	Description	Score
901	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>We've improved the design of this perennial favorite. A sleek and technical shirt built for the trail, track, or sidewalk. UPF rating of 50+.</p></body></html>	3.3416789108071976
907	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>A polycarbonate visor with an abrasion-resistant coating on the outside. Great for jogging in the spring, summer, and early fall. The elastic headband has plenty of stretch to give you a snug yet comfortable fit every time you wear it.</p></body></html>	2.1123084896159376

ID	Description	Score
905	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US><p>A lightweight wool cap with mesh side vents for breathable comfort during aerobic activities. Moisture-absorbing headband liner.</p></body></html>	1.6750365447462499
910	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US><p>These quick-drying cotton shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist.</p></body></html>	0.9244136988525732
906	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor.</p></body></html>	0.9134171046194403
904	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US><p>This fashionable hat is ideal for glacier travel, sea kayaking, and hiking. With concealed draw cord for windy days.</p></body></html>	0.7313071661212746

ID	Description	Score
903	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>A sporty, casual shirt made of recycled water bottles. It will serve you equally well on trails or around town. The fabric has a wicking finish to pull perspiration away from your skin.</p></body></html>	0.6799436746197272

- d. Die folgenden Anweisungen durchsuchen jede Zeile nach Begriffen, die "v" enthalten. Nach der zweiten Anweisung enthält die Variable die Zeichenfolge av OR ev OR iv OR ov OR rv OR ve OR vi OR vo. Die Ergebnisse werden anhand der Punktwerte in absteigender Sortierfolge sortiert. Wenn ein N-Gramm in allen indizierten Zeilen vorkommt, wird ihm ein Punktwert von null zugewiesen.

Nur diese Methode ermöglicht es, ein einzelnes Zeichen aufzufinden, wenn es vor einer Leerstelle oder einem nicht-alphanumerischen Zeichen auftritt.

```
CREATE VARIABLE query NVARCHAR (100);
SELECT LIST (term, ' OR ' )
INTO query
FROM sa_text_index_vocab_nchar( 'ncharNGRAMTextIndex',
'MarketingInformationNgram', 'dba' )
WHERE term LIKE '%v%';
SELECT M.ID, M.Description, ct.*
FROM MarketingInformationNgram AS M
CONTAINS( M.Description, query ) ct
ORDER BY ct.score DESC;
```

ID	Description	Score
901	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>We've improved the design of this perennial favorite. A sleek and technical shirt built for the trail, track, or sidewalk. UPF rating of 50+.</p></body></html>	6.654350268810443

ID	Description	Score
907	<html><head><meta http-equiv=Content-Type content="text/html; charset=wind-ows-1252"><title>Visor</title></head><body lang=EN-US><p>A polycarbonate visor with an abrasion-resistant coating on the outside. Great for jogging in the spring, summer, and early fall. The elastic headband has plenty of stretch to give you a snug yet comfortable fit every time you wear it.</p></body></html>	4.265623837817126
903	<html><head><meta http-equiv=Content-Type content="text/html; charset=wind-ows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>A sporty, casual shirt made of recycled water bottles. It will serve you equally well on trails or around town. The fabric has a wicking finish to pull perspiration away from your skin.</p></body></html>	2.9386676702799504
910	<html><head><meta http-equiv=Content-Type content="text/html; charset=wind-ows-1252"><title>Shorts</title></head><body lang=EN-US><p>These quick-drying cotton shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist.</p></body></html>	2.5481193655722336
904	<html><head><meta http-equiv=Content-Type content="text/html; charset=wind-ows-1252"><title>Baseball Cap</title></head><body lang=EN-US><p>This fashionable hat is ideal for glacier travel, sea-kayaking, and hiking. With concealed draw cord for windy days.</p></body></html>	2.4293498211307214

ID	Description	Score
905	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US><p>A lightweight wool cap with mesh side vents for breathable comfort during aerobic activities. Moisture-absorbing headband liner.</p></body></html>	1.6750365447462499
906	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor.</p></body></html>	0.9134171046194403
902	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US><p>This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester fabric is gentle on the earth and soft against your skin.</p></body></html>	0
908	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Lightweight 100% organically grown cotton hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage.</p></body></html>	0

ID	Description	Score
909	<html><head><meta http-equiv=Content-Type content="text/html; charset=wind-ows-1252"><title>Sweatshirt</title></head><body lang=EN-US><p>Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% cotton/20% polyester blend makes it easy to keep them clean.</p></body></html>	0

- e. Die folgende Anweisung durchsucht die Spalte Description nach Zeilen, die **ea**, **ka** und **ki** enthalten.

```
SELECT M.ID, M.Description, ct.*
FROM MarketingInformationNgram AS M
CONTAINS( M.Description, 'ea ka ki' ) ct
ORDER BY ct.score DESC;
```

ID	Description	Score
904	<html><head><meta http-equiv=Content-Type content="text/html; charset=wind-ows-1252"><title>Baseball Cap</title></head><body lang=EN-US><p>This fashionable hat is ideal for glacier travel, sea-kayaking, and hiking. With concealed draw cord for windy days.</p></body></html>	3.4151032739119733

- f. Die folgende Anweisung durchsucht die Spalte "Description" nach Zeilen, die **ve** und **vi**, aber nicht **gg** enthalten.

```
SELECT M.ID, M.Description, ct.*
FROM MarketingInformationNgram AS M
CONTAINS( M.Description, 've & vi -gg' ) ct
ORDER BY ct.score DESC;
```


ID	Description	Score
905	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Baseball Cap</title></head><body lang=EN-US><p>A lightweight wool cap with mesh side vents for breathable comfort during aerobic activities. Moisture-absorbing headband liner.</p></body></html>	1.6750365447462499

12. Schließen Sie Interactive SQL und Sybase Central.

Ergebnisse

Sie haben eine Volltextsuche in einem NGRAM-Textindex durchgeführt.

Nächste Schritte

(Optional) Setzen Sie die Beispieldatenbank (*demo.db*) in ihren ursprünglichen Zustand zurück. Siehe „[Neuerstellung der Beispieldatenbank \(demo.db\)](#)“ [*SQL Anywhere 16 - Einführung*].

Siehe auch

- „Konzepte und Referenz zu Textkonfigurationsobjekten“ auf Seite 406
- „CREATE TEXT CONFIGURATION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TEXT CONFIGURATION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Konzepte und Referenz zu Textindizes“ auf Seite 423
- „CREATE TEXT INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TEXT INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Fortgeschrittene Aufgaben: Löschen von Begriffen in einer Volltextsuche

Textindizes werden entsprechend der Einstellungen erstellt, die für die Textkonfigurationsobjekte festgelegt sind, die zum Erstellen des Textindexes verwendet werden. Ein Begriff erscheint nicht in einem Textindex, wenn eine oder mehrere der folgenden Bedingungen zutreffen:

- Der Begriff ist in der Stoppliste enthalten.
- Der Begriff ist kürzer als die Begriff-Mindestlänge (nur GENERIC).
- Der Begriff ist länger als die Begriff-Höchstlänge.

Dieselbe Regel gilt für Abfragezeichenfolgen. Der gelöschte Begriff kann mit null oder mehr Begriffen am Ende oder am Anfang einer Phrase übereinstimmen. Beispiel: Angenommen, der Begriff 'the' steht in der Stoppliste:

- Wenn der Begriff auf einer der Seiten von AND, OR oder NEAR erscheint, werden sowohl der Operator als auch der Begriff entfernt. Beispiel: Die Suche nach 'the AND apple', 'the OR apple' oder 'the NEAR apple' ist gleichwertig mit der Suche nach 'apple'.
- Wenn der Begriff auf der rechten Seite eines AND NOT-Operators erscheint, werden sowohl AND NOT als auch der Begriff gelöscht. Beispiel: Die Suche nach 'apple AND NOT the' ist gleichwertig mit der Suche nach 'apple'.

Wenn der Begriff auf der linken Seite eines AND NOT-Operators erscheint, wird der gesamte Ausdruck gelöscht und es werden keine Zeilen zurückgegeben. Zum Beispiel: 'orange and the AND NOT apple' = 'orange'

- Wenn der Begriff in einer Phrase erscheint, kann die Phrase mit jedem anderen Begriff an der Position des gelöschten Begriffs übereinstimmen. Beispiel: Die Suche nach 'feed the dog' findet 'feed the dog', 'feed my dog', 'feed any dog' ect.

Wenn keiner der Begriffe, nach denen Sie suchen, im Textindex enthalten ist, werden keine Zeilen zurückgegeben. Beispiel: Angenommen, sowohl 'the' als auch 'a' stehen in der Stoppliste. Die Suche nach 'a OR the' gibt keine Zeilen zurück.

Siehe auch

- „Konzepte und Referenz zu Textkonfigurationsobjekten“ auf Seite 406

Fortgeschrittene Aufgaben: Externe Begriffsegmentierer- und Vorfilterbibliotheken

Gründe für die Verwendung einer externen Begriffsegmentierer- oder Vorfilterbibliothek

Die Volltextsuche in SQL Anywhere wird mithilfe eines Textindexes durchgeführt. Jeder Wert in einer Spalte, für die ein Textindex erstellt wurde, wird als ein **Dokument** bezeichnet. Wenn ein Textindex erstellt wurde, wird jedes Dokument von einem integrierten, in der Textkonfiguration für den Textindex angegebenen Begriffsegmentierer verarbeitet, um die im Dokument enthaltenen **Begriffe** (auch als **Token** bezeichnet) und der Position der Begriffe im Dokument zu ermitteln. Der integrierte Begriffsegmentierer wird auch dazu verwendet, die Begriffe für die Dokumente (Textkomponenten) einer Abfragezeichenfolge zu segmentieren. Beispiel: Die Abfragezeichenfolge 'rain or shine' besteht aus zwei Dokumenten, 'rain' und 'shine', verbunden durch den Operator OR. Außerdem wird der integrierte, in der Textkonfiguration angegebene Begriffsegmentierer-Algorithmus verwendet, um die Stoppliste in Begriffe zu segmentieren und um die Eingabe der sa_char_terms-Systemprozedur in Begriffe zu segmentieren.

Abhängig von den Anforderungen Ihrer Anwendung stellen Sie möglicherweise fest, dass einige Verhaltensweisen des integrierten GENERIC-Begriffsegmentierers unerwünschte oder einschränkende Auswirkungen haben und dass NGRAM-Begriffsegmentierer für die Anforderungen der Anwendung

nicht geeignet sind. Beispielsweise bietet der integrierte GENERIC-Begriffsegmentierer keine sprachspezifische Begriffsegmentierung. Im Folgenden finden Sie einige Gründe, die dafür sprechen, einen benutzerdefinierten **Begriffsegmentierer** zu implementieren:

- **Keine sprachspezifische Begriffsegmentierung** Linguistische Regeln in Bezug darauf, was einen Begriff darstellt, unterscheiden sich je nach Sprache. Aus diesem Grund sind auch Begriffsegmentierungsregeln von Sprache zu Sprache verschieden. Die integrierten Begriffsegmentierer bieten keine sprachspezifischen Begriffsegmentierungsregeln.
- **Umgang mit Wörtern mit Apostroph** Das Wort "they'll" wird vom integrierten GENERIC-Begriffsegmentierer als "they ll" behandelt. Sie können jedoch einen benutzerdefinierten GENERIC-Begriffsegmentierer konfigurieren, der den Apostroph als Teil des Worts behandelt.
- **Keine Unterstützung für die Ersetzung von Begriffen** Sie können nicht festlegen, durch welche anderen Begriffe ein Begriff ersetzt werden soll. Beispielsweise bei der Indizierung des Worts "they' ll" ist es möglicherweise angebracht, diese als zwei Begriffe zu speichern: "they" und "will". Auf ähnliche Weise kann es sinnvoll sein, die Ersetzung von Begriffen zu verwenden, um eine Suche ohne Berücksichtigung der Groß- und Kleinschreibung in einer Datenbank durchzuführen, in der die Groß- und Kleinschreibung berücksichtigt wird.

In SQL Anywhere können Sie außerdem externe Vorfilterbibliotheken für die Durchführung von **Vorfilterungen** an Daten vor ihrer Indizierung verwenden. Das Vorfiltern erlaubt Ihnen, ausschließlich den zu indizierenden Textinhalt eines Dokuments zu extrahieren. Beispiel: Angenommen Sie möchten einen Textindex für eine Spalte mit XML-Werten erstellen. Mit einem Vorfilter können Sie die XML-Tags herausfiltern, damit diese nicht gemeinsam mit dem Inhalt indiziert werden.

SQL Anywhere stellt eine API bereit, die Sie verwenden können, um auf benutzerdefinierte Vorfilter und Begriffsegmentierer-Bibliotheken bzw. auf Bibliotheken von Drittherstellern zuzugreifen, wenn Sie Volltextindizes erstellen und aktualisieren. Das bedeutet, Sie können externe Bibliotheken verwenden, um bei Dokumentformaten wie XML, PDF und Word unerwünschte Begriffe und Inhalte vor der Indizierung des Inhalts zu entfernen.

In Ihrem Beispielverzeichnis (*Samples*) werden einige Beispielvorfiler- und Begriffsegmentierer-Bibliotheken bereitgestellt, die Sie bei der Konfiguration eigener Bibliotheken unterstützen. Sie können aber auch die API verwenden, um auf Bibliotheken von Drittherstellern zuzugreifen. Wenn Microsoft Office auf dem System installiert ist, auf dem der Datenbankserver ausgeführt wird, sind IFilters for Office-Dokumente, wie z.B. Word und Excel, verfügbar. Wenn auf dem Server Acrobat Reader installiert ist, dann steht wahrscheinlich ein PDF IFilter zur Verfügung.

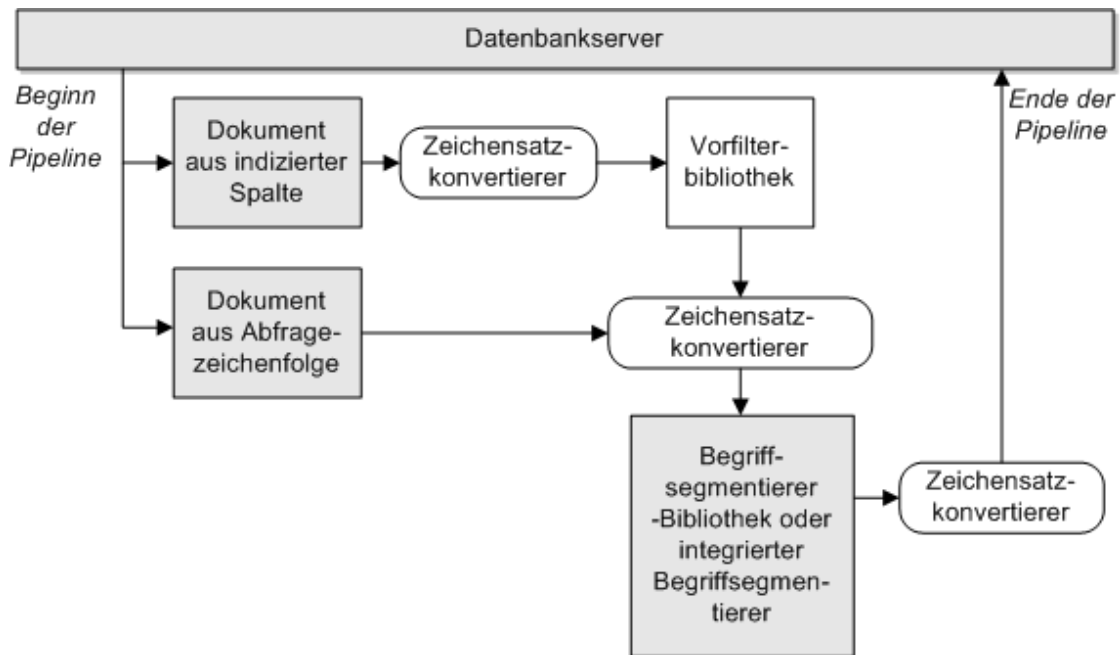
Hinweis

Externe NGRAM-Begriffsegmentierer werden nicht unterstützt.

Flusssteuerung für Volltext-Pipeline

Das folgende Diagramm zeigt, wie Daten in SQL Anywhere von einem Dokument in einen Stream von zu indizierenden Begriffen konvertiert werden. Die Flusssteuerung für die Erstellung, Aktualisierung und Abfrage eines Textindexes wird als **Pipeline** bezeichnet. Die obligatorischen Teile der Pipeline sind

hellgrau dargestellt. Pfeile zeigen den Fluss der Daten durch die Pipeline an. Funktionsaufrufe werden in umgekehrter Richtung weitergegeben.



Gesamtübersicht über die Funktionsweise der Pipeline

1. Die Verarbeitung jedes einzelnen Dokuments wird durch den Datenbankserver initiiert, der die Methode `begin_document` am Beginn einer Pipeline aufruft, entweder am Begriffsegmentierer oder am Zeichensatzkonvertierer. Jede Komponente in der Pipeline ruft `begin_document` für ihren eigenen Producer auf, bevor Ergebnisse von dieser Methode zurückgegeben werden.
2. Der Datenbankserver ruft, nachdem die Methode `begin_document` erfolgreich abgeschlossen wurde, am Ende der Pipeline `get_words` auf.
 - Während der Ausführung von `get_words` ruft der Begriffsegmentierer `get_next_piece` für seinen Producer auf, damit die Daten verarbeitet werden. Wenn ein Vorfilter in der Pipeline vorhanden ist, werden die Daten während des nächsten Aufrufs der Methode `get_next_piece` gefiltert.
 - Der Begriffsegmentierer segmentiert die von seinem Producer erhaltenen Daten entsprechend seiner Begriffsegmentierregeln in Begriffe.
3. Der Datenbankserver wendet die Einstellungen für die minimale und maximale Begriffslänge sowie die Stopplisteneinschränkungen auf die durch Aufruf der Methode `get_words` zurückgegebenen Begriffe an.
4. Der Datenbankserver fährt mit dem Aufrufen von `get_words` fort, bis keine weiteren Begriffe mehr zurückgegeben werden. Zu diesem Zeitpunkt ruft der Datenbankserver die Methode `end_document` auf. Dieser Aufruf wird auf dieselbe Weise in der Pipeline weitergegeben wie der Aufruf der Methode `begin_document`.

Hinweis

Zeichensatzkonvertierer werden durch den Datenbankserver, wo erforderlich, der Pipeline auf transparente Weise hinzugefügt.

Codebeispiele für Vorfilter und Begriffsegmentierer

Das Verzeichnis *ExternalLibrariesFullText* in Ihrem SQL Anywhere-Installationsverzeichnis enthält Codebeispiel für Vorfilter und Begriffsegmentierer. Dieses Verzeichnis befindet sich im Verzeichnis *Samples*. Weitere Informationen zum Speicherort des Verzeichnisses *Samples* finden Sie unter [Beispielverzeichnis \[SQL Anywhere Server - Datenbankadministration\]](#).

Siehe auch

- [Flusssteuerung für externe Vorfilterbibliothek auf Seite 452](#)
- [Flusssteuerung für externe Begriffsegmentiererbibliothek auf Seite 455](#)

Externe Vorfilterbibliotheken**SQL Anywhere für die Verwendung eines externen Vorfilters konfigurieren**

SQL Anywhere bietet keinen integrierten Vorfilter-Algorithmus. Damit Daten eine externe Vorfilterbibliothek durchlaufen, geben Sie mit der ALTER TEXT CONFIGURATION-Anweisung die Bibliothek und ihre Eintrittspunktfunktion an, ähnlich wie im folgenden Beispiel:

```
ALTER TEXT CONFIGURATION my_text_config
  PREFILTER EXTERNAL NAME 'my_prefilter@myprefilterLibrary.dll'
```

In diesem Beispiel wird der Datenbankserver angewiesen, die Eintrittspunktfunktion my_prefilter in der Bibliothek *myprefilterLibrary.dll* zu verwenden, um eine Vorfilter-Instanz zu erhalten, die beim Erstellen oder Aktualisieren eines Textindexes mit dem Textkonfigurationsobjekt my_text_config verwendet wird.

Siehe auch

- „ALTER TEXT CONFIGURATION-Anweisung“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- „a_text_source-Schnittstelle“ auf Seite 459

Externe Vorfilterbibliothek entwerfen

Die Vorfilterbibliothek muss in C/C++ implementiert werden und folgende Voraussetzungen erfüllen:

- Sie muss die Header-Datei für die Vorfilter-Schnittstellendefinition, *extpfapiv1.h*, enthalten.
- Sie muss die Schnittstelle *a_text_source* implementieren.
- Sie muss eine Eintrittspunktfunktion bieten, die eine Instanz von *a_text_source* (Vorfilter) initialisiert und mit dem Label des vom Vorfilter unterstützten Zeichensatzes zurückgibt.

Aufrufsequenz für den Vorfilter

Die folgende Aufrufsequenz wird für jedes verarbeitete Dokument durch den Client des Vorfilters ausgeführt:

```
begin_document(a_text_source*)
get_next_piece(a_text_source*, buffer**, len*)
get_next_piece(a_text_source*, buffer**, len*)
...
end_document(a_text_source*)
```

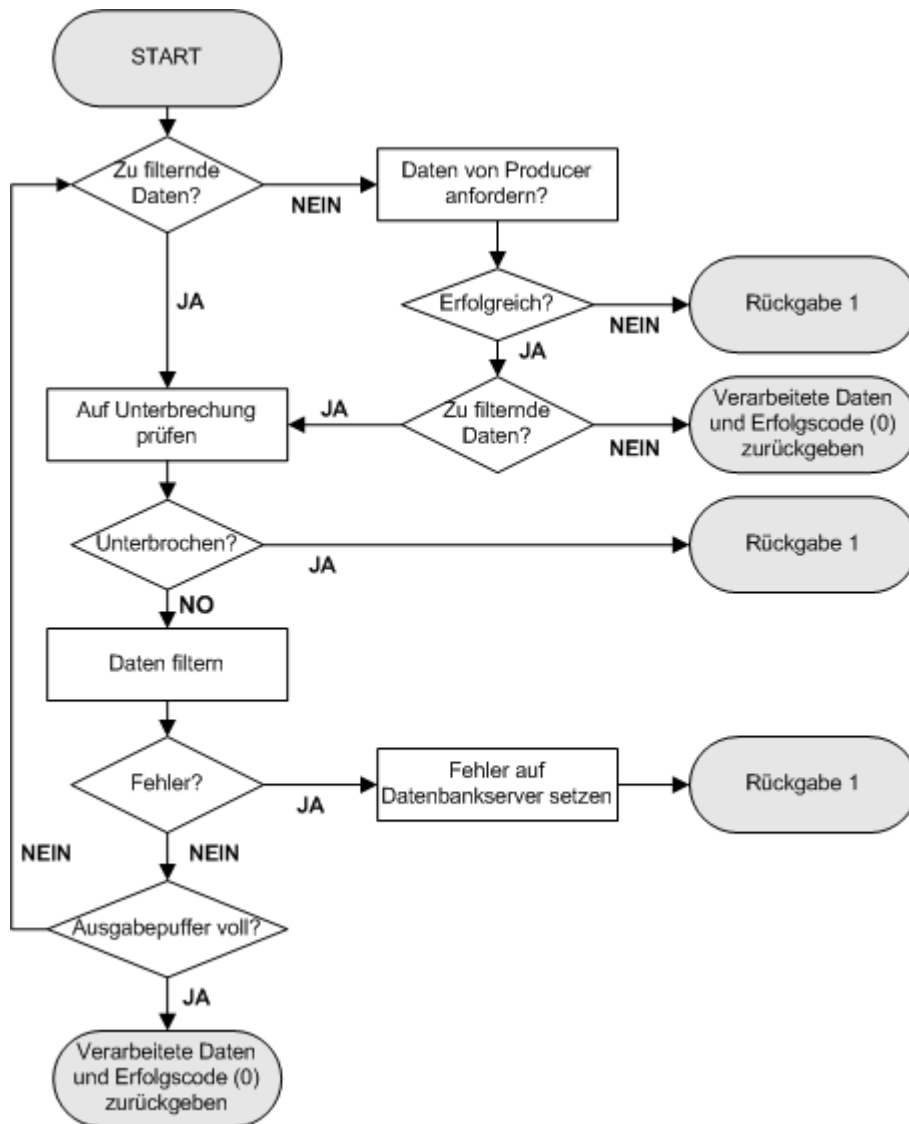
Hinweis

end_document kann mehrere Male aufgerufen werden, ohne dass ein begin_document-Aufruf dazwischen kommt. Beispiel: Wenn eines der zu indizierenden Dokumente leer ist, ruft der Datenbankserver möglicherweise die Methode end_document für dieses Dokument auf, ohne vorher begin_document aufzurufen.

Die get_next_piece-Funktion ist dafür vorgesehen, nicht benötigte Daten wie etwa Formatinformation und Grafiken aus dem eintreffenden Bytestrom herauszufiltern und den nächsten Abschnitt gefilterter Daten in einem sich selbst zugewiesenen Puffer zurückzugeben.

Flusssteuerung für externe Vorfilterbibliothek

Das folgende Flussdiagramm veranschaulicht die Ablauflogik beim Aufrufen der get_next_piece-Funktion:



Siehe auch

- [„a_text_source-Schnittstelle“ auf Seite 459](#)
- [„Flusssteuerung für Volltext-Pipeline“ auf Seite 449](#)
- [Codebeispiele für Vorfilter und Begriffsegmentierer auf Seite 451](#)

Externe Begriffsegmentiererbibliotheken

SQL Anywhere für die Verwendung eines externen Begriffsegmentierers konfigurieren

Standardmäßig wird beim Erstellen eines Textkonfigurationsobjekts ein integrierter Begriffsegmentierer für Daten verwendet, die mit diesem Textkonfigurationsobjekt verknüpft sind. Damit Daten stattdessen eine externe Begriffsegmentiererbibliothek durchlaufen, geben Sie mit der ALTER TEXT CONFIGURATION-Anweisung die Bibliothek und ihre Eintrittspunktfunktion an, ähnlich wie im folgenden Beispiel:

```
ALTER TEXT CONFIGURATION my_text_config
  TERM BREAKER GENERIC EXTERNAL NAME 'my_termbreaker@termbreaker'
```

In diesem Beispiel wird der Datenbankserver angewiesen, die Eintrittspunktfunktion my_termbreaker in der Begriffsegmentiererbibliothek zu verwenden, um bei der syntaktischen Analyse der Stoppliste des Textkonfigurationsobjekts und der Verarbeitung der erhaltenen Daten in der Systemprozedur sa_char_terms eine Begriffsegmentierer-Instanz zur Verwendung beim Erstellen, Aktualisieren oder Abfragen eines mit dem Textkonfigurationsobjekt my_text_config verknüpften Textindexes zu erhalten.

Siehe auch

- „ALTER TEXT CONFIGURATION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „a_word_source-Schnittstelle“ auf Seite 463

Externe Begriffsegmentiererbibliothek entwerfen

Die externe Begriffsegmentiererbibliothek muss in C/C++ implementiert werden und folgende Voraussetzungen erfüllen:

- Sie muss die Header-Datei für die Begriffsegmentierer-Schnittstellendefinition, *exttbapiv1.h*, enthalten.
- Sie muss die Schnittstelle a_word_source implementieren.
- Sie muss eine Eintrittspunktfunktion bieten, die eine Instanz von a_word_source (Begriffsegmentierer) initialisiert und mit dem Label des vom Begriffsegmentierer unterstützten Zeichensatzes zurückgibt.

Aufrufsequenz für den Begriffsegmentierer

Die folgende Aufrufsequenz wird für jedes verarbeitete Dokument durch den Client des Begriffsegmentierers ausgeführt:

```
begin_document(a_word_source*, asql_uint32);
get_words(a_word_source*, a_term**, uint32 *num_words)
get_words(a_word_source*, a_term**, uint32 *num_words)
...
end_document(a_word_source*)
```

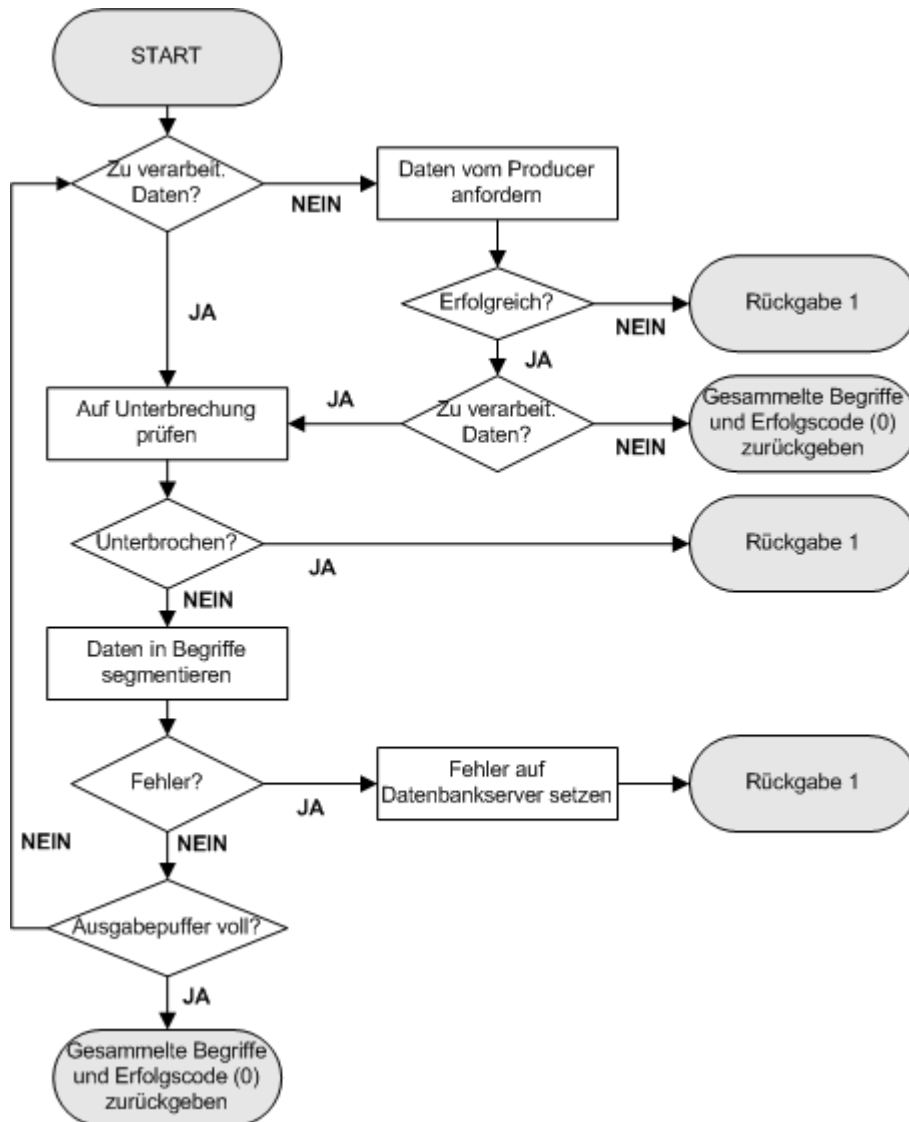
Die get_words-Funktion muss, damit Daten in Begriffe segmentiert werden, get_next_piece für seinen Producer aufrufen bis das Array von a_term-Strukturen gefüllt ist oder keine weiteren Daten mehr zur Verarbeitung verfügbar sind.

Hinweis

end_document kann mehrere Male aufgerufen werden, ohne dass ein begin_document-Aufruf dazwischen kommt. Beispiel: Wenn eines der zu indizierenden Dokumente leer ist, ruft der Datenbankserver möglicherweise die Methode end_document für dieses Dokument auf, ohne vorher begin_document aufzurufen.

Flusssteuerung für externe Begriffsegmentiererbibliothek

Das folgende Flussdiagramm veranschaulicht die Ablauflogik beim Aufrufen der get_words-Funktion:



Siehe auch

- „a_word_source-Schnittstelle“ auf Seite 463
- „a_text_source-Schnittstelle“ auf Seite 459
- „a_term-Struktur“ auf Seite 465
- „Flusssteuerung für Volltext-Pipeline“ auf Seite 449
- Codebeispiele für Vorfilter und Begriffsegmentierer auf Seite 451

Fortgeschrittene Aufgaben: API für externe Volltext-Bibliotheken

Folgende Schritte müssen zur Erstellung und Verwendung einer externen Vorfilter- oder Begriffsegmentiererbibliothek mit Textindizes durchgeführt werden:

- Implementieren Sie die SQL Anywhere-C/C++-Schnittstellen.
- Erstellen einer dynamisch ladbaren Bibliothek durch Kompilieren und Verknüpfen des im oben beschriebenen Schritt geschriebenen Codes.
- Erstellen des Textkonfigurationsobjekts in der Datenbank und dann Ändern des Textkonfigurationsobjekts zur Angabe der Eintrittspunktfunktion in der externen Bibliothek für Vorfilter und/oder Begriffsegmentierer.

Die Eintrittspunktfunktionen werden zum Abrufen der Vorfilter- und Begriffsegmentierer-Objekte verwendet, die bei Einfügen/Löschen von Textindex-Einträgen verwendet werden sollen, wenn zugrunde liegende Dokumente (Spaltenwerte) geändert werden. Im Fall einer externen Begriffsegmentiererbibliothek, wird die Eintrittspunktfunktion auch für die syntaktische Analyse von Abfragen in den Textindizes verwendet, bei denen der Begriffsegmentierer verwendet wird.

a_server_context-Struktur

Mehrere Callback-Funktionen werden vom Datenbankserver unterstützt und den externen Volltext-Bibliotheken über die a_server_context-Struktur zur Ausführung folgender Aufgaben bereitgestellt:

- Fehlermeldungen
- Unterbrechen der Verarbeitung
- Protokollierung von Meldungen

Syntax

```
typedef struct a_server_context {
    void (SQL_CALLBACK *set_error)( a_server_context *this
                                   , a_sql_uint32      error_code
                                   , const char         *error_string
                                   , short              error_string_length
                                   );
    a_sql_uint32 (SQL_CALLBACK *get_is_cancelled)( a_server_context *this
                                                  );
    void (SQL_CALLBACK *log_message)( a_server_context *this
```

```

, const char    *message_string
, short         message_string_length
);

void *_context;
} a_server_context, *p_server_context;

```

Mitglieder

Mitgliedsname	Typ	Beschreibung
set_error	void	<p>Mit dieser Methode können externe Vorfilter und Begriffsegmentierer einen Fehler im Datenbankserver ablegen, indem sie einen Fehlercode und eine Fehlerzeichenfolge übergeben. Der Datenbankserver setzt den aktuellen Vorgang zurück und gibt den Fehlercode und die Zeichenfolge an den Benutzer in der folgenden Form zurück.</p> <pre>"Error from external library: -<error_code>: <error_string>"</pre> <p>error_code muss eine positive Ganzzahl größer als 17000 sein.</p> <p>error_string muss eine mit NULL abgeschlossene Zeichenfolge sein.</p> <p>str_len ist die Länge der error_string-Zeichenfolge in Byte.</p>
get_is_cancelled	a_sql_uint32	<p>Externe Vorfilter und Begriffsegmentierer müssen diese Methode regelmäßig aufrufen, um zu prüfen, ob der aktuelle Vorgang unterbrochen wurde. Diese Methode gibt 1 zurück, wenn der aktuelle Vorgang unterbrochen wurde, und 0, wenn er nicht unterbrochen wurde. Wenn 1 zurückgegeben wird, muss der Aufrufer die weitere Prozessverarbeitung unterbrechen und sofort zurückgeben.</p>
log_message	void	<p>Diese Methode ermöglicht, dass externe Vorfilter und Begriffsegmentierer Meldungen im Datenbankserverlog protokollieren.</p> <p>message muss eine mit NULL abgeschlossene Zeichenfolge sein.</p> <p>msg_len ist die Länge der Meldung in Byte.</p>
_context	void	<p>Wird nur intern verwendet. Ein Zeiger auf den Datenbankserverkontext.</p>

Bemerkungen

Die a_server_context-Struktur ist in der Header-Datei *exttxtcmn.h* im Unterverzeichnis *SDK\Include* Ihres SQL Anywhere-Installationsverzeichnis definiert.

Die externe Bibliothek sollte daher nicht mit einem Betriebssystem-Synchronisationsbasiselement verwendet werden, wenn die Methoden aufgerufen werden, die von der `a_server_context`-Struktur bereitgestellt werden.

a_init_pre_filter-Struktur

Struktur zur Parameterübergabe an einen externen Vorfilter. Diese Struktur wird als Parameter an die Eintrittspunktfunktion des Vorfilters übergeben.

Syntax

```
typedef struct a_init_pre_filter {
    a_text_source      *in_text_source;    /* IN */
    a_text_source      *out_text_source;   /* OUT */
    const char          *desired_charset;   /* IN */
    char                *actual_charset;    /* OUT */
    short               is_binary;          /* IN */
} a_init_pre_filter;
```

Mitglieder

Name	Typ	Beschreibung
in_text_source	a_text_source *	Der Zeiger auf den Producer des zu erstellenden externen Vorfilters (a_text_source-Objekt). Angegeben vom Aufrufer der Eintrittspunktfunktion des Vorfilters.
out_text_source	a_text_source *	Der Zeiger auf den externen Vorfilter (a_text_source-Objekt), festgelegt durch die Eintrittspunktfunktion des Vorfilters.
desired_charset	const char *	Der vom Aufrufer der Eintrittspunktfunktion erwartete Zeichensatz für Ausgabedaten des Vorfilters. Wenn der is_binary-Parameter 0 ist, gilt dieser Zeichensatz auch für die Eingabedaten für den Vorfilter, es sei denn, dies wird auf andere Art und Weise ausgehandelt.
actual_charset	char *	Der Zeichensatz , in dem die externe Vorfilterbibliothek ihre Ausgabe erzeugt (festgelegt durch die externe Bibliothek als Teil der Aushandlung). Wenn der is_binary-Parameter 0 ist, ist dies auch der Zeichensatz, der für die Eingabe in den Vorfilter benutzt wird. Wenn möglich, sollte die Bibliothek Eingabe- und Ausgabedaten in dem Zeichensatz verarbeiten, der durch desired_charset festgelegt wurde.
is_binary	short	Bestimmt, ob die Eingabedaten in binärer Form (1) oder in einem durch desired_charset (0) bestimmten Zeichensatz vorliegen. Wenn die Daten in binärer Form vorliegen, fügt der Datenbankserver in der Pipeline keine Zeichensatzkonvertierung vor dem Vorfilter ein.

Bemerkungen

Die `a_init_pre_filter`-Struktur ist in der Header-Datei `extpfapiv1.h` im Unterverzeichnis `SDK\Include` Ihres SQL Anywhere-Installationsverzeichnisses definiert.

Siehe auch

- „`a_text_source`-Schnittstelle“ auf Seite 459
- „`a_word_source`-Schnittstelle“ auf Seite 463
- „Eintrittspunktfunktion für Vorfilter“ auf Seite 468

a_text_source-Schnittstelle

Die Schnittstelle, die eine externe Vorfilterbibliothek implementieren muss, um Dokumente zum Füllen oder Aktualisieren des Volltextindex vorzufiltern.

Syntax

```
typedef struct a_text_source {
    a_sql_uint32 ( SQL_CALLBACK *begin_document )( a_text_source *This );
    a_sql_uint32 ( SQL_CALLBACK *get_next_piece )(
        a_text_source *This
        , unsigned char ** buffer
        , a_sql_uint32* len );

    a_sql_uint32 ( SQL_CALLBACK *end_document )( a_text_source *This);
    a_sql_uint64 ( SQL_CALLBACK *get_document_size )( a_text_source *This );
    a_sql_uint32 ( SQL_CALLBACK *fini_all )( a_text_source *This );
    a_server_context *_context;

    // Only one of the following two members can have a valid pointer in a
    // given implementation.
    // These members point to the current module's producer
    a_text_source *_my_text_producer;
    a_word_source *_my_word_producer;

    // Following members have been reserved for
    // future use ONLY
    a_text_source *_my_text_consumer;
    a_word_source *_my_word_consumer;
} a_text_source, *p_text_source;
```

Mitglieder

Mitglied	Typ	Beschreibung
<code>begin_document</code>	<code>a_sql_uint32</code>	Führt die erforderlichen Konfigurationsschritte für die Verarbeitung eines Dokuments durch. Diese Methode gibt bei Erfolg 0 und bei Auftreten eines Fehlers, oder wenn keine weiteren Dokumente zur Verfügung stehen, 1 zurück.

Mitglied	Typ	Beschreibung
get_next_piece	a_sql_uint32	<p>Gibt ein Fragment des gefilterten Eingabe-Bytestroms und die Länge des Fragments zurück. Diese Methode wird mehrfach für ein bestimmtes Dokument aufgerufen und sollte bei jedem Aufruf den jeweils nächsten Abschnitt des Dokuments zurückgeben, bis alle Eingabedaten für ein Dokument verarbeitet wurden oder ein Fehler auftritt.</p> <p>buffer ist der OUT-Parameter, der durch den Vorfilter gefüllt werden soll, um auf die erzeugten Daten zu zeigen. Der Speicher wird durch den Vorfilter verwaltet.</p> <p>len ist der OUT-Parameter, der die Länge der erzeugten Daten angibt.</p>
end_document	a_sql_uint32	Markiert den Abschluss des Filtervorgangs für das jeweilige Dokument und führt gegebenenfalls eine dokumentspezifische Bereinigung durch.
get_document_size	a_sql_uint64	Gibt die Gesamtlänge des Dokuments (in Byte) zurück, wie es durch den Vorfilter erzeugt wurde. Das Objekt a_text_source muss eine immer aktuelle Zählung der Gesamtzahl der von ihm bisher für das aktuelle Dokument erzeugten Byte bereithalten.
fini_all	a_sql_uint32	Wird durch den Datenbankserver nach der Verarbeitung aller Dokumente und unmittelbar vor dem Abschluss der Pipeline aufgerufen. fini_all führt die abschließenden Bereinigungsschritte durch.
_context	a_server_context *	Verwenden Sie dieses Element zum Speichern des Datenbankserverkontexts, der der Eintrittspunktfunktion innerhalb der a_init_pre_filter-Struktur bereitgestellt wird. Das Vorfiltermodul verwendet diesen Kontext zur Einrichtung einer direkten Kommunikation mit dem Datenbankserver.
_my_text_producer	a_text_source *	Verwenden Sie dieses Element zum Speichern des Zeigers auf den a_text_source-Producer des Vorfilters, der der Eintrittspunktfunktion innerhalb der a_init_pre_filter-Struktur bereitgestellt wird. Dieser Zeiger kann, wenn eine Zeichensatzkonvertierung erforderlich ist, durch den Datenbankserver ersetzt werden, nachdem die Eintrittspunktfunktion ausgeführt wurde. Aus diesem Grund kann nur dieser Zeiger auf den Textproducer vom Vorfilter verwendet werden.
_my_word_producer	a_word_source *	Ist für eine spätere Verwendung reserviert und sollte mit dem Wert NULL initialisiert werden.

Mitglied	Typ	Beschreibung
_my_text_consumer	a_text_source *	Ist für eine spätere Verwendung reserviert und sollte mit dem Wert NULL initialisiert werden.
_my_word_consumer	a_word_source *	Ist für eine spätere Verwendung reserviert und sollte mit dem Wert NULL initialisiert werden.

Bemerkungen

Die a_text_source-Schnittstelle ist für Datenstrom-basierte Daten vorgesehen. Die Daten werden vom Producer der Reihe nach abgerufen; jedes Byte kommt nur einmal vor.

Die a_text_source-Schnittstelle ist in der Header-Datei *extpfapiv1.h* im Unterverzeichnis *SDK\Include* des SQL Anywhere-Installationsverzeichnis definiert.

Die externe Bibliothek sollte daher keine Betriebssystem-Synchronisationsbasiselemente über Funktionsaufrufe enthalten.

Siehe auch

- „a_server_context-Struktur“ auf Seite 456
- „a_init_pre_filter-Struktur“ auf Seite 458
- „Eintrittspunktfunktion für Vorfilter“ auf Seite 468

a_init_term_breaker-Struktur

Struktur für die Aushandlung der Eingabe- und Ausgabeanforderungen für Instanzen eines externen Begriffsegmentierers. Diese Struktur wird als Parameter an die Eintrittspunktfunktion des externen Begriffsegmentierers übergeben.

Syntax

```
typedef struct a_init_term_breaker
{
    a_text_source      *in_text_source;
    const char         *desired_charset;
    a_word_source      *out_word_source;
    char               *actual_charset;
    a_term_breaker_for term_breaker_for;
} a_init_term_breaker, *p_init_term_breaker;
```

Mitglieder

Mitglied	Typ	Beschreibung
in_text_source	a_text_source *	Der Zeiger auf den Producer des zu erstellenden externen Begriffsegmentierers (a_text_source-Objekt).
out_word_source	a_word_source *	Der Zeiger auf den von der Eintrittspunktfunktion angegebenen externen Begriffsegmentierer (a_word_source-Objekt).

Mitglied	Typ	Beschreibung
desired_charset	const char *	Der vom Aufrufer der Eintrittspunktfunktion erwartete Zeichensatz für Ausgabedaten des Begriffsegmentierers. Wenn der is_binary-Parameter 0 ist, gilt dieser Zeichensatz auch für die Eingabedaten für den Begriffsegmentierer, es sei denn, dies wird auf andere Art und Weise ausgehandelt.
actual_charset	char *	Der Zeichensatz, in dem die externe Begriffsegmentiererbibliothek ihre Ausgabe erzeugt (festgelegt durch die externe Bibliothek als Teil der Aushandlung). Wenn der is_binary-Parameter 0 ist, ist dies auch der Zeichensatz, der für die Eingabe in den Begriffsegmentierer benutzt wird. Wenn möglich, sollte die Bibliothek Eingabe- und Ausgabedaten in dem Zeichensatz verarbeiten, der durch desired_charset festgelegt wurde.
term_breaker_for	a_term_breaker_for	<p>Der Zweck der Initialisierung des Begriffsegmentierers:</p> <ul style="list-style-type: none"> • TERM_BREAKER_FOR_LOAD Wird für Erstell-, Einfüge-, Aktualisierungs- und Löschvorgänge am Textindex verwendet. Die Eingabe kann vorgefiltert werden, wenn ein Vorfilter angegeben ist. • TERM_BREAKER_FOR_QUERY Wird für die syntaktische Analyse von Abfrageelementen, Stoppliste und Eingabe für die Systemprozedur sa_char_term verwendet. Im Fall von TERM_BREAKER_FOR_QUERY wird nicht vorgefiltert; auch dann nicht, wenn eine externe Vorfilterbibliothek für den Textindex angegeben ist.

Bemerkungen

Die a_init_term_breaker-Struktur ist in der Header-Datei *exttbapiv1.h* im Unterverzeichnis *SDK\Include* Ihres SQL Anywhere-Installationsverzeichnis definiert.

Siehe auch

- „sa_char_terms-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „a_term_breaker_for-Enumeration“ auf Seite 462
- „Eintrittspunktfunktion für Begriffsegmentierer“ auf Seite 469
- „a_text_source-Schnittstelle“ auf Seite 459
- „a_word_source-Schnittstelle“ auf Seite 463

a_term_breaker_for-Enumeration

Mit dieser Option können Sie angeben, ob die Pipeline für die Verwendung während der Aktualisierung oder Abfrage des Textindex erstellt wird.

Parameter

- **TERM_BREAKER_FOR_LOAD** Wird für Erstell-, Einfüge-, Aktualisierungs- und Löschvorgänge am Textindex verwendet.
- **TERM_BREAKER_FOR_QUERY** Wird für die syntaktische Analyse von Abfrageelementen, Stoppliste und Eingabe für die Systemprozedur `sa_char_term` verwendet. Im Fall von `TERM_BREAKER_FOR_QUERY` wird nicht vorgefiltert; auch dann nicht, wenn eine externe Vorfilterbibliothek für den Textindex angegeben ist.

Bemerkungen

Der Datenbankserver stellt den Wert für `a_init_term_breaker::term_breaker_for` beim Initialisieren des externen Begriffsegmentierers ein.

```
typedef enum a_term_breaker_for {
    TERM_BREAKER_FOR_LOAD = 0,
    TERM_BREAKER_FOR_QUERY
} a_term_breaker_for;
```

Die `a_term_breaker_for`-Enumeration ist in der Header-Datei `exttbapiv1.h` im Unterverzeichnis *SDK* `\Include` Ihres SQL Anywhere-Installationsverzeichnisses definiert.

Siehe auch

- „`a_init_term_breaker`-Struktur“ auf Seite 461
- „`sa_char_terms`-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

a_word_source-Schnittstelle

Die Schnittstelle, die eine externe Begriffsegmentiererbibliothek implementieren muss, um Vorgänge zum Segmentieren in Begriffe für Textindizes durchführen zu können.

Syntax

```
typedef struct a_word_source {
    a_sql_uint32 ( SQL_CALLBACK *begin_document )(
        a_word_source *This
        , a_sql_uint32 has_prefix );

    a_sql_uint32 ( SQL_CALLBACK *get_words )(
        a_word_source *This
        , a_term** words
        , a_sql_uint32 *num_words );

    a_sql_uint32 ( SQL_CALLBACK *end_document )(
        a_word_source *This );

    a_sql_uint32 ( SQL_CALLBACK *fini_all )(
        a_word_source *This );

    a_server_context    *_context;

    a_text_source        *_my_text_producer;
    a_word_source        *_my_word_producer;

    a_text_source        *_my_text_consumer;
    a_word_source        *_my_word_consumer;
} a_word_source, *p_word_source;
```

Mitglieder

Mitglied	Typ	Beschreibung
begin_document	a_sql_uint32	<p>Führt die erforderlichen Konfigurationsschritte für die Verarbeitung eines Dokuments durch. Der Parameter <code>has_prefix</code> ist auf 1 (nicht wahr) oder auf TRUE gesetzt, wenn das in Token übersetzte Dokument ein Präfix-Abfragebegriff ist. Wenn <code>has_prefix</code> auf TRUE gesetzt ist, muss der Begriffsegmentierer mindestens einen Begriff (möglicherweise leer) zurückgeben.</p> <p><code>has_prefix</code> kann nur die Werte 1 (nicht wahr) oder TRUE haben, wenn der Zweck der Pipeline-Initialisierung <code>TERM_BREAKER_FOR_QUERY</code> ist.</p> <p>Das Ergebnis einer Präfix-Tokenisierung wird als eine Phrase behandelt, wobei der letzte Begriff der Phrase die tatsächliche Präfix-Zeichenfolge ist.</p>
get_words	a_sql_uint32	<p>Gibt einen Zeiger auf ein Array von <code>a_term</code>-Strukturen zurück. Diese Methode wird so lange in einer Schleife für ein bestimmtes Dokument aufgerufen, bis alle Inhalte des Dokuments in Begriffe segmentiert worden sind.</p> <p>Der Datenbankserver erwartet, dass zwei unmittelbar aufeinander folgende Begriffe in einem Dokument Positionen haben, die sich um 1 unterscheiden. Wenn der Begriffsegmentierer seine eigene Stopplistenverarbeitung ausführt, ist es möglich, dass die Differenz zwischen zwei aufeinander folgenden Begriffen größer als 1 ist; dies ist zu erwarten und zulässig. In anderen Fällen jedoch, bei denen Zahlen nicht aufeinander folgen und die Positionen sich um 1 unterscheiden, können die willkürlichen Positionen sich darauf auswirken, wie Volltextabfragen ausgeführt werden, und können zu unerwarteten Ergebnissen für nachfolgende Volltextabfragen führen.</p>
end_document	a_sql_uint32	Markiert den Abschluss der Verarbeitung des Dokuments durch die Pipeline und führt eine dokumentspezifische Bereinigung durch.
fini_all	a_sql_uint32	Wird durch den Datenbankserver nach der Verarbeitung aller Dokumente und unmittelbar vor dem Abschluss der Pipeline aufgerufen. <code>fini_all</code> führt die abschließenden Bereinigungsschritte durch.

Mitglied	Typ	Beschreibung
_context	a_server_context *	Der der Eintrittspunktfunktion innerhalb der a_init_term_breaker-Struktur bereitgestellte Datenbankserverkontext. Das Begriffsegmentierermodul verwendet diesen Kontext zur Einrichtung einer direkten Kommunikation mit dem Datenbankserver.
_my_text_producer	a_text_source *	Ein Zeiger auf den a_text_source-Producer von dem Begriffsegmentierer, der der Eintrittspunktfunktion innerhalb der a_init_term_breaker-Struktur bereitgestellt wird. Dieser Zeiger kann, wenn eine Zeichensatzkonvertierung erforderlich ist, durch den Datenbankserver ersetzt werden, nachdem die Eintrittspunktfunktion ausgeführt wurde. Aus diesem Grund kann nur dieser Zeiger auf den Textpproducer vom Begriffsegmentierer verwendet werden.
_my_word_producer	a_word_source *	Ist für eine spätere Verwendung reserviert und sollte mit dem Wert NULL initialisiert werden.
_my_text_consumer	a_text_source *	Ist für eine spätere Verwendung reserviert und sollte mit dem Wert NULL initialisiert werden.
_my_word_consumer	a_word_source *	Ist für eine spätere Verwendung reserviert und sollte mit dem Wert NULL initialisiert werden.

Bemerkungen

Die a_word_source-Schnittstelle ist in der Header-Datei *exttbapiv1.h* im Unterverzeichnis *SDK\Include* des SQL Anywhere-Installationsverzeichnis definiert.

Die externe Bibliothek sollte daher keine Betriebssystem-Synchronisationsbasiselemente über Funktionsaufrufe enthalten.

Siehe auch

- „a_server_context-Struktur“ auf Seite 456
- „a_term-Struktur“ auf Seite 465
- „a_init_term_breaker-Struktur“ auf Seite 461
- „a_text_source-Schnittstelle“ auf Seite 459
- „Eintrittspunktfunktion für Begriffsegmentierer“ auf Seite 469

a_term-Struktur

Die a_term-Struktur speichert einen Begriff, seine Länge und seine Position.

Syntax

```
typedef struct a_term
{
```

```
    unsigned char  *word;
    a_sql_uint32   len;
    a_sql_uint32   ch_len;
    a_sql_uint64   pos;
} a_term, *p_term;
```

Mitglieder

Mitglied	Typ	Beschreibung
term	unsigned char *	Der zu indizierende Begriff.
len	a_sql_uint32	Die Länge des Begriffs in Byte.
ch_len	a_sql_uint32	Die Länge des Begriffs in Zeichen.
pos	a_sql_uint64	Position des Begriffs innerhalb des Dokumentes. Der Datenbankserver erwartet, dass zwei unmittelbar aufeinander folgende Begriffe in einem Dokument Positionen haben, die sich um 1 unterscheiden. Wenn der Begriffsegmentierer seine eigene Stopp-listenverarbeitung ausführt, ist es möglich, dass die Differenz zwischen zwei aufeinander folgenden Begriffen größer als 1 ist; dies ist zu erwarten und zulässig. In anderen Fällen jedoch, bei denen Zahlen nicht aufeinander folgen und die Positionen sich um 1 unterscheiden, können die willkürlichen Positionen sich darauf auswirken, wie Volltextabfragen ausgeführt werden, und können zu unerwarteten Ergebnissen für nachfolgende Volltextabfragen führen.

Bemerkungen

Jede a_term-Struktur steht für einen Begriff, ergänzt durch seine Länge in Byte, seine Länge in Zeichen und seine Position im Dokument.

Ein Zeiger auf ein Array mit a_term-Elementen wird von der als Teil der a_word_source-Schnittstelle implementierten get_words-Methode im OUT-Parameter zurückgegeben.

Die a_term-Struktur ist in der Header-Datei *exttbapiv1.h* im Unterverzeichnis *SDK\Include* Ihres SQL Anywhere-Installationsverzeichnis definiert.

extpf_use_new_api-Eintrittspunktfunktion (Vorfilter)

Meldet dem Datenbankserver die in der externen Vorfilterbibliothek implementierte Schnittstellenversion. Derzeit werden nur Schnittstellen der Version 1 unterstützt.

Diese Funktion ist für eine externe Vorfilterbibliothek erforderlich.

Syntax

```
extern "C" a_sql_uint32 ( extpf_use_new_api )( void );
```

Rückgabe

Die Funktion gibt eine vorzeichenlose 32-Bit-Ganzzahl zurück. Der zurückgegebene Wert muss die Versionsnummer der Schnittstelle EXTPF_V1_API gemäß der Definition in *extpfapiv1.h* sein.

Bemerkungen

Im Folgenden wird eine typische Implementierung dieser Funktion gezeigt:

```
extern "C" a_sql_uint32 ( extpf_use_new_api )( void )
{
    return EXTPF_V1_API;
}
```

exttb_use_new_api-Eintrittspunktfunktion (Begriffsegmentierer)

Stellt Informationen zu der in der externen Begriffsegmentiererbibliothek implementierten Schnittstellenversion bereit. Derzeit werden nur Schnittstellen der Version 1 unterstützt.

Diese Funktion ist für eine externe Begriffsegmentiererbibliothek erforderlich.

Syntax

```
extern "C" a_sql_uint32 (exttb_use_new_api )( void );
```

Rückgabe

Die Funktion gibt eine vorzeichenlose 32-Bit-Ganzzahl zurück. Der zurückgegebene Wert muss die Versionsnummer der Schnittstelle EXTTB_V1_API gemäß der Definition in *exttbapiv1.h* sein.

Bemerkungen

Im Folgenden wird eine typische Implementierung dieser Funktion gezeigt:

```
extern "C" a_sql_uint32 ( exttb_use_new_api )( void )
{
    return EXTTB_V1_API;
}
```

Globale Eintrittspunktfunktion extfn_post_load_library

Wenn diese Funktion implementiert ist und in der externen Bibliothek bereitgestellt wird, wird sie vom Datenbankserver ausgeführt, nachdem die externe Bibliothek geladen und die Versionsprüfung durchgeführt wurde, und bevor eine andere der in der externen Bibliothek definierten Funktionen aufgerufen wird.

Diese Funktion ist nur dann erforderlich, wenn es eine Bibliothek-spezifische Anforderung gibt, ein sich über die gesamte Bibliothek erstreckendes Setup auszuführen, bevor eine der Funktionen in der Bibliothek aufgerufen wird.

Syntax

```
extern "C" void ( extfn_post_load_library )( void );
```

Bemerkungen

Sowohl externe Begriffsegmentierer- als auch externe Vorfilterbibliotheken können diese Funktion implementieren.

Siehe auch

- „extfn_post_load_library-Methode“ [[SQL Anywhere Server - Programmierung](#)]

Globale Eintrittspunktfunktion extfn_pre_unload_library

Wenn diese Funktion implementiert ist und in der externen Bibliothek bereitgestellt wird, wird sie vom Datenbankserver unmittelbar vor dem Entladen der externen Bibliothek ausgeführt.

Diese Funktion ist nur dann erforderlich, wenn es eine Bibliothek-spezifische Anforderung gibt, eine sich über die gesamte Bibliothek erstreckende Bereinigung auszuführen, bevor die Bibliothek entladen wird.

Syntax

```
extern "C" void ( extfn_pre_unload_library )( void );
```

Bemerkungen

Sowohl externe Begriffsegmentierer- als auch externe Vorfilterbibliotheken können diese Funktion implementieren.

Siehe auch

- „extfn_pre_unload_library-Methode“ [[SQL Anywhere Server - Programmierung](#)]

Eintrittspunktfunktion für Vorfilter

Eintrittspunktfunktion, die eine Instanz eines externen Vorfilters initialisiert und den Zeichensatz der Daten aushandelt.

Syntax

```
extern "C" a_sql_uint32 ( SQL_CALLBACK *entry-point-function )( a_init_pre_filter *data );
```

Rückgabe

1 bei Fehler und 0 bei erfolgreicher Ausführung

Parameter

- **entry-point-function** Der Name der Eintrittspunktfunktion für den Vorfilter.
- **data** Ein Zeiger auf eine a_init_pre_filter-Struktur.

Bemerkungen

Diese Funktion muss in der externen Vorfilterbibliothek implementiert werden und muss eintrittsinvariant sein, da sie für mehrere Threads gleichzeitig ausgeführt werden kann.

Der Aufrufer der Funktion (Datenbankserver) stellt einen Zeiger auf ein `a_text_source`-Objekt bereit, das als Produzent für den Vorfilter fungiert. Der Aufrufer stellt auch den Zeichensatz der Eingabe bereit.

Diese Funktion stellt einen Zeiger auf den externen Vorfilter bereit (`a_text_source`-Struktur). Darüber hinaus handelt sie den Zeichensatz der Eingabedaten (wenn diese nicht binär sind) und der Ausgabedaten durch Ändern des `actual_charset`-Felds aus, falls erforderlich.

Wenn `desired_charset` und `actual_charset` nicht identisch sind, führt der Datenbankserver eine Zeichensatzkonvertierung der Eingabedaten durch, es sei denn, das Feld `data->is_binary` hat den Wert 1. Wenn `is_binary` gleich 0 ist, liegen die Eingabedaten im durch `actual_charset` festgelegten Zeichensatz vor.

Erforderliche Zeichensatzkonvertierungen können zu einer Verminderung der Performance führen.

Diese Eintrittspunktfunktion wird vom Benutzer durch Aufruf von `ALTER TEXT CONFIGURATION... PREFILTER EXTERNAL NAME` angegeben.

Siehe auch

- „`a_init_pre_filter`-Struktur“ auf Seite 458
- „`a_init_pre_filter`-Struktur“ auf Seite 458
- „`a_text_source`-Schnittstelle“ auf Seite 459
- `PREFILTER EXTERNAL NAME`-Klausel, `ALTER TEXT CONFIGURATION`-Anweisung [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Eintrittspunktfunktion für Begriffsegmentierer

Eintrittspunktfunktion, die eine Instanz eines externen Begriffsegmentierers initialisiert und den Zeichensatz der Daten aushandelt.

Syntax

```
extern "C" a_sql_uint32 ( SQL_CALLBACK *entry-point-function )( a_init_term_breaker *data );
```

Rückgabe

1 bei Fehler und 0 bei erfolgreicher Ausführung

Parameter

- **entry-point-function** Der Name der Eintrittspunktfunktion für den Begriffsegmentierer.
- **data** Ein Zeiger auf eine `a_init_term_breaker`-Struktur.

Bemerkungen

Diese Funktion muss in der externen Begriffsegmentiererbibliothek implementiert werden und muss eintrittsinvariant sein, da sie für mehrere Threads gleichzeitig ausgeführt werden kann.

Der Aufrufer der Funktion stellt einen Zeiger auf ein `a_text_source`-Objekt bereit, das als Produzent für den Begriffsegmentierer fungiert. Der Aufrufer sollte auch den Zeichensatz der Eingabe bereitstellen.

Diese Funktion stellt dem Aufrufer einen Zeiger auf einen externen Begriffsegmentierer (a_word_source-Struktur) und den unterstützten Zeichensatz bereit.

Wenn desired_charset und actual_charset nicht identisch sind, konvertiert der Datenbankserver die Begriffsegmentierer-Eingabe in den durch actual_charset angegebenen Zeichensatz.

Erforderliche Zeichensatzkonvertierungen können zu einer Verminderung der Performance führen.

Siehe auch

- „a_word_source-Schnittstelle“ auf Seite 463
- „a_text_source-Schnittstelle“ auf Seite 459
- „a_init_term_breaker-Struktur“ auf Seite 461

Abfrageergebnisse zusammenfassen, gruppieren und sortieren

Aggregatfunktionen, die Abfrageergebnisse zusammenfassen

Aggregatfunktionen zeigen Zusammenfassungen der Werte in angegebenen Spalten. Sie können auch die GROUP BY-Klausel, die HAVING-Klausel und die ORDER BY-Klausel verwenden, um die Ergebnisse von Abfragen mit Aggregatfunktionen zu gruppieren und zu sortieren sowie den UNION-Operator, um die Ergebnisse von Abfragen zu kombinieren.

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
```

Sie können Aggregatfunktionen auf alle Zeilen einer Tabelle, auf eine in einer WHERE-Klausel definierte Teilmenge oder auf eine oder mehrere Gruppen von Zeilen in einer Tabelle anwenden. Von jeder Zeilengruppe, auf die die Aggregatfunktion angewendet wird, erstellt SQL Anywhere einen einzelnen Wert.

Unter Anderem werden folgende Aggregatfunktionen unterstützt:

- **AVG(expression)** Der Durchschnitt des angegebenen Ausdrucks aus den zurückgegebenen Zeilen

- **COUNT(*expression*)** Die Anzahl von Zeilen in der gelieferten Gruppe, in denen der Ausdruck nicht NULL ist
- **COUNT(*)** Die Anzahl von Zeilen in jeder Gruppe
- **LIST(*string-expr*)** Eine Zeichenfolge, die eine durch Kommas getrennte Liste aller Werte für *string-expr* in jeder Zeilengruppe enthält
- **MAX(*expression*)** Der Höchstwert des Ausdrucks in allen zurückgegebenen Zeilen
- **MIN(*expression*)** Der Mindestwert des Ausdrucks von allen zurückgegebenen Zeilen
- **STDDEV(*expression*)** Die Standard-Abweichung des Ausdrucks von allen zurückgegebenen Zeilen
- **SUM(*expression*)** Die Summe des Ausdrucks von allen zurückgegebenen Zeilen
- **VARIANCE(*expression*)** Die Varianz des Ausdrucks von allen zurückgegebenen Zeilen

Sie können das optionale Schlüsselwort DISTINCT gemeinsam mit AVG, SUM, LIST und COUNT verwenden, um Doppelwerte zu eliminieren, bevor die Aggregatfunktion angewendet wird.

Der Ausdruck, auf den sich die Syntaxanweisung bezieht, ist in der Regel ein Spaltenname. Es kann sich aber auch um einen allgemeineren Ausdruck handeln.

Beispielsweise können Sie mit diesem Ausdruck herausfinden, wie hoch der Durchschnittspreis aller Artikel sein würde, wenn zu jedem Preis ein Dollar addiert würde:

```
SELECT AVG ( UnitPrice + 1 )  
FROM Products;
```

Siehe auch

- „Aggregatfunktionen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Beispiel

Um die Lohnsumme aus den Jahresgehältern der Mitarbeiter in der Tabelle "Employees" zu erstellen, gehen Sie folgendermaßen vor:

```
SELECT SUM( Salary )  
FROM Employees;
```

Um Aggregatfunktionen anzuwenden, müssen Sie den Funktionsnamen eingeben, gefolgt von einem Ausdruck, für den diese Werte gelten. Der Ausdruck, in diesem Beispiel die Spalte Salary, ist das Argument der Funktion und muss in Klammern angegeben werden.

Wo Sie Aggregatfunktionen verwenden können

Aggregatfunktionen können in einer SELECT-Liste oder in der HAVING-Klausel eines Blocks mit gruppierten Abfragen angegeben werden.

Sie können Aggregatfunktionen nicht in einer WHERE-Klausel oder in einer JOIN-Bedingung verwenden. Ein SELECT-Anweisungsblock mit Aggregatfunktionen in ihrer SELECT-Liste enthält jedoch häufig eine WHERE-Klausel, mit der die Zeilen beschränkt werden, auf die die Aggregatfunktion angewendet wird.

Wenn eine Aggregatfunktion in einem SELECT-Abfrageblock verwendet wird, der keine GROUP BY-Klausel enthält, wird ungeachtet der Tatsache, ob sie für alle Zeilen in einer Tabelle oder für eine durch eine WHERE-Klausel definierte Teilgruppe ausgeführt wird, ein einzelner Wert produziert.

Sie können mehr als eine Aggregatfunktion in derselben SELECT-Liste verwenden und mehr als ein Aggregat in einem einzelnen SELECT-Anweisungsblock erzeugen.

Siehe auch

- „Die HAVING-Klausel: Datengruppen auswählen“ auf Seite 480
- „GROUP BY-Klausel“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Aggregatfunktionen und Datentypen

Manche Aggregatfunktionen haben nur für bestimmte Arten von Daten Bedeutung. Sie können beispielsweise SUM und AVG nur mit numerischen Spalten verwenden.

Allerdings können Sie MIN benutzen, um in einer Zeichenspalte den niedrigsten Wert - denjenigen, der im Alphabet vorne liegt - zu finden:

```
SELECT MIN( Surname )  
FROM Contacts;
```

COUNT(*)

COUNT(*) gibt die Anzahl der Zeilen in der angegebenen Tabelle zurück, ohne Duplikate zu eliminieren. Die Funktion zählt jede Zeile getrennt und schließt auch Zeilen ein, die NULL enthalten. Diese Funktion benötigt keinen Ausdruck als Argument, da sie definitionsgemäß keine Informationen über eine bestimmte Spalte benutzt.

Die folgende Anweisung gibt die Gesamtzahl der Mitarbeiter in der Employees-Tabelle zurück:

```
SELECT COUNT( * )  
FROM Employees;
```

Wie andere Aggregatfunktionen können Sie auch COUNT(*) mit anderen Aggregatfunktionen in der SELECT-Liste, mit WHERE-Klauseln usw. kombinieren. Beispiel:

```
SELECT COUNT( * ), AVG( UnitPrice )  
FROM Products  
WHERE UnitPrice > 10;
```

COUNT()	AVG(Products.UnitPrice)
5	18.2

Aggregatfunktionen mit DISTINCT

Das Schlüsselwort DISTINCT ist bei SUM, AVG und COUNT optional. Wenn Sie DISTINCT verwenden, werden Duplikatwerte eliminiert, bevor die Summe, der Durchschnitt oder die Anzahl berechnet werden. Beispiel: Um die Anzahl verschiedener Städte, in denen Kontaktpersonen leben, zu finden, führen Sie die folgende Anweisung aus:

```
SELECT COUNT( DISTINCT City )
FROM Contacts;
```

COUNT(DISTINCT Contacts.City)
16

Sie können in einer Abfrage mehr als eine Aggregatfunktion mit DISTINCT verwenden. Jedes DISTINCT wird unabhängig ausgewertet. Beispiel:

```
SELECT COUNT( DISTINCT GivenName ) "first names",
COUNT( DISTINCT Surname ) "last names"
FROM Contacts;
```

first names	last names
48	60

Aggregatfunktionen und NULL

Alle Nullwerte in der Spalte, mit der die Aggregatfunktion arbeitet, werden für die Funktion ignoriert, mit Ausnahme von COUNT(*), diese Funktion bezieht sie ein. Wenn alle Werte in einer Spalte NULL sind, gibt COUNT(Spaltenname) 0 zurück.

Wenn die Bedingungen, die in der WHERE-Klausel angegeben wurden, auf keine Zeile zutreffen, gibt COUNT den Wert 0 zurück. Die anderen Funktionen geben NULL zurück. Ein paar Beispiele:

```
SELECT COUNT( DISTINCT Name )
FROM Products
WHERE UnitPrice > 50;
```

COUNT(DISTINCT Name)
0

```
SELECT AVG( UnitPrice )
FROM Products
WHERE UnitPrice > 50;
```

AVG(Products.UnitPrice)
(NULL)

Die GROUP BY-Klausel: Abfrageergebnisse in Gruppen organisieren

Die GROUP BY-Klausel teilt die Ausgabe einer Tabelle in Gruppen. Sie können Zeilen anhand eines oder mehrerer Spaltennamen oder anhand der Ergebnisse von berechneten Spalten gruppieren.

Reihenfolge von Klauseln

Wenn sowohl eine WHERE-Klausel als auch eine GROUP BY-Klausel vorhanden sind, muss die WHERE-Klausel vor der GROUP BY-Klausel stehen. Eine GROUP BY-Klausel muss, falls vorhanden, immer vor einer HAVING-Klausel stehen. Wenn eine HAVING-Klausel angegeben wird, aber keine GROUP BY-Klausel, wird eine GROUP BY ()-Klausel angenommen.

Sowohl HAVING- als auch WHERE-Klauseln können in einer einzelnen Abfrage verwendet werden. Bedingungen in der HAVING-Klausel schränken die Ergebniszeilen erst logisch ein, nachdem die Gruppen erstellt wurden. Die Kriterien in der WHERE-Klausel werden logisch berücksichtigt, bevor die Gruppen erstellt werden, und sparen somit Zeit.

Siehe auch

- „Aggregatfunktionen in Abfragen“ auf Seite 325

Wie Abfragen mit GROUP BY ausgeführt werden

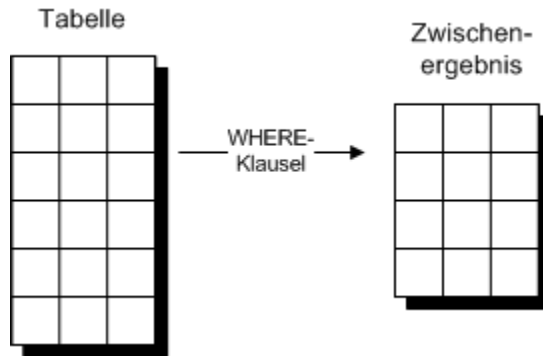
Dieser Abschnitt verwendet die Unterklausel ROLLUP der Klausel GROUP BY in der Erklärung und im Beispiel.

Gehen wir von einer Abfrage in einer einzelnen Tabelle der folgenden Form aus:

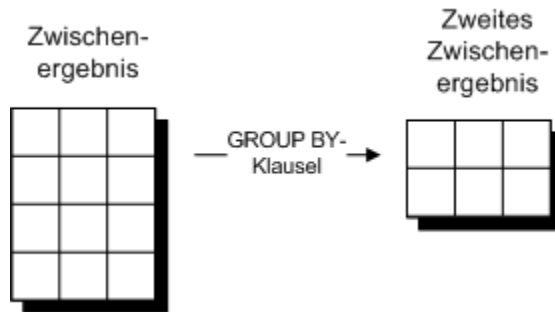
```
SELECT select-list
FROM table
WHERE where-search-condition
GROUP BY [ group-by-expression | ROLLUP (group-by-expression) ]
HAVING having-search-condition
```

Die Abfrage wird auf folgende Weise ausgeführt:

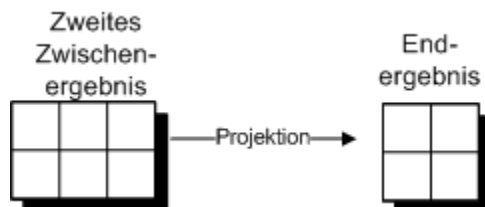
1. **WHERE-Klausel anwenden** Damit wird ein Zwischenergebnis erstellt, das nur einige Zeilen aus der Tabelle enthält.



2. **Ergebnis in Gruppen teilen** Dieser Vorgang erstellt ein Zwischenergebnis mit einer Zeile für jede Gruppe gemäß den Anweisungen der GROUP BY-Klausel. Jede generierte Zeile enthält den GROUP BY-Ausdruck *group-by-expression* für jede Gruppe und die berechneten Aggregatfunktionen in der Auswahlliste *select-list* und der HAVING-Suchbedingung *having-search-condition*.



3. **Einen ROLLUP-Vorgang anwenden** Zwischensummen-Zeilen, die als Teil eines ROLLUP-Vorgangs berechnet werden, werden der Ergebnismenge hinzugefügt.
4. **HAVING-Klausel anwenden** Zeilen aus diesem zweiten Zwischenergebnis, die die Kriterien der HAVING-Klausel nicht erfüllen, werden an dieser Stelle entfernt.
5. **Ergebnisse für die Anzeige zusammenstellen (Projektion)** Bei diesem Vorgang werden aus Schritt 3 nur jene Spalten entnommen, die in der Ergebnismenge der Abfrage angezeigt werden müssen - also nur die Spalten, die den Ausdrücken aus der *select-list* entsprechen.



Dieser Vorgang stellt an Abfragen mit einer GROUP BY-Klausel einige Anforderungen:

- Die WHERE-Klausel wird zuerst aufgelöst. Daher werden Aggregatfunktionen nur über diese Zeilen aufgelöst, die den Bedingungen der WHERE-Klausel entsprechen.
- Die endgültige Ergebnismenge wird aus dem zweiten Zwischenergebnis erstellt, das die aufgeteilten Zeilen enthält. Das zweite Zwischenergebnis enthält Zeilen, die dem *group-by-expression* entsprechen. Wenn daher ein Ausdruck, der keine Aggregatfunktion ist, in der *select-list* vorkommt, dann muss er auch im *group-by-expression* vorkommen. Während der Projektionsphase kann keine Funktionsauflösung erfolgen.
- Ein Ausdruck kann in den *group-by-expression*, aber nicht in die *select-list* einbezogen sein. Er wird im Ergebnis verworfen.

Siehe auch

- „[ROLLUP und CUBE als Abkürzung für GROUPING SETS](#)“ auf Seite 558

GROUP BY mit mehreren Spalten

Sie können mehr als einen Ausdruck in der GROUP BY-Klausel auflisten, d.h. Sie können eine Tabelle mit jeder beliebigen Kombination von Ausdrücken gruppieren.

So listen Sie den Durchschnittspreis zuerst nach Namen und dann nach Größe gruppiert auf:

```
SELECT Name, Size, AVG( UnitPrice )
FROM Products
GROUP BY Name, Size;
```

Name	Size	AVG(Products.UnitPrice)
Baseball Cap	One size fits all	9.5
Sweatshirt	Large	24
Tee Shirt	Large	14
Tee Shirt	One size fits all	14
...

WHERE-Klausel und GROUP BY

Sie können eine WHERE-Klausel in einer Anweisung mit GROUP BY verwenden. Die WHERE-Klausel wird vor der GROUP BY-Klausel berücksichtigt. Zeilen, die die Bedingungen in der WHERE-Klausel nicht erfüllen, werden eliminiert, bevor eine Gruppierung erfolgt. Ein Beispiel:

```
SELECT Name, AVG( UnitPrice )
FROM Products
WHERE ID > 400
GROUP BY Name;
```

Nur die Zeilen mit ID-Werten von mehr als 400 werden in die Gruppen einbezogen, die zur Erstellung der Abfrageergebnisse benutzt werden.

Beispiel

Folgende Abfrage zeigt den Einsatz der Klauseln WHERE, GROUP BY und HAVING in einer Abfrage:

```
SELECT Name, SUM( Quantity )
FROM Products
WHERE Name LIKE '%shirt%'
GROUP BY Name
HAVING SUM( Quantity ) > 100;
```

Name	SUM(Products.Quantity)
Tee Shirt	157

In diesem Beispiel gilt:

- Die WHERE-Klausel enthält nur Zeilen, die einen Namen mit dem Wortbestandteil *shirt* enthalten (Tee Shirt, Sweatshirt).
- Die GROUP BY-Klausel sammelt die Zeilen mit einem gemeinsamen Namen.
- Die Aggregatfunktion SUM berechnet die Gesamtmenge der für jede Gruppe verfügbaren Artikel.
- Die HAVING-Klausel schließt aus den Endergebnissen die Gruppe aus, deren Lagersummen 100 nicht überschreiten.

GROUP BY mit Aggregatfunktionen

Eine GROUP BY-Klausel erscheint fast immer in Anweisungen, die Aggregatfunktionen enthalten. In diesem Fall produziert das Aggregat einen Wert für jede Gruppe. Diese Werte werden **Vektoraggregat** genannt. (Ein **Skalaraggregat** ist ein einzelner Wert, der von einer Aggregatfunktion ohne GROUP BY-Klausel produziert wurde.)

Beispiel

Folgende Abfrage listet den Durchschnittspreis der Produkttypen auf:

```
SELECT Name, AVG( UnitPrice ) AS Price
FROM Products
GROUP BY Name;
```

Name	Price
Tee Shirt	12.333333333
Baseball Cap	9.5
Visor	7

Name	Price
Sweatshirt	24
...	...

Die Vektoraggregate, die von SELECT-Anweisungen mit Aggregaten und einer GROUP BY-Klausel produziert werden, erscheinen als Spalten in jeder Zeile der Ergebnisse. Im Gegensatz dazu erscheinen die Skalaraggregate, die von Abfragen mit Aggregaten und ohne GROUP BY produziert werden, ebenfalls als Spalten, aber nur mit einer Zeile. Beispiel:

```
SELECT AVG( UnitPrice )
FROM Products;
```

AVG(Products.UnitPrice)
13.3

GROUP BY und der Standard SQL/2008

Der SQL/2008-Standard ist in seiner Syntax erheblich restriktiver als der von SQL Anywhere unterstützte. Im SQL/2008-Standard erfordert GROUP BY folgende Bedingungen:

- Jedes in einer GROUP BY-Klausel angegebene *group-by-term* muss eine *Spaltenreferenz* sein, d.h. eine Referenz auf eine Spalte aus einer Tabelle, die in der FROM-Klausel der Abfrage referenziert wird. Diese Ausdrücke werden **gruppierten Spalten** genannt.
- Ein Ausdruck in einer SELECT-Liste, HAVING-Klausel oder ORDER BY-Klausel, der keine Aggregatfunktion ist, muss eine gruppierte Spalte sein oder darf nur gruppierte Spalten referenzieren. Wenn jedoch die optionale SQL/2008-Sprachfunktion T301, "Funktionale Abhängigkeiten", unterstützt wird, kann so eine Referenz Spalten aus der FROM-Klausel der Abfrage referenzieren, die funktionell durch gruppierte Spalten festgelegt werden.

In einer GROUP BY-Klausel in SQL Anywhere kann *group-by-term* ein beliebiger Ausdruck sein, der Spaltenreferenzen, literale Konstanten, Variablen oder Hostvariablen sowie skalare und benutzerdefinierte Funktionen betrifft. Diese Abfrage beispielsweise partitioniert die Tabelle "Employee" in drei Gruppen basierend auf die Spalte "Salary" und erzeugt eine Zeile pro Gruppe:

```
SELECT COUNT() FROM Employees
GROUP BY (
    IF SALARY < 25000
    THEN 'low range'
    ELSE IF Salary < 50000
    THEN 'mid range'
    ELSE 'high range'
    ENDIF
ENDIF);
```

Um den Partitionswert in das Abfrageergebnis aufzunehmen, müssen Sie ein *group-by-term* der SELECT-Liste der Abfrage hinzufügen. Um syntaktisch gültig zu sein, stellt SQL Anywhere sicher, dass die Syntax des SELECT-Listenelements und von *group-by-term* identisch sind. Allerdings scheitern möglicherweise

syntaktisch umfassende SQL-Konstruktionen bei dieser Analyse; auch werden Ausdrücke, die Unterabfragen betreffen, nie als gleichwertig verglichen.

Im unten stehenden Beispiel erkennt SQL Anywhere, dass die zwei IF-Ausdrücke identisch sind, und ermittelt das Ergebnis ohne Fehler:

```
SELECT (IF SALARY < 25000 THEN 'low range' ELSE IF Salary < 50000 THEN 'mid range' ELSE 'high range' ENDIF ENDIF), COUNT()  
FROM Employees  
GROUP BY (IF SALARY < 25000 THEN 'low range' ELSE IF Salary < 50000 THEN 'mid range' ELSE 'high range' ENDIF ENDIF);
```

Jedoch enthält diese Abfrage eine Unterabfrage in der GROUP BY-Klausel und gibt einen Fehler zurück:

```
SELECT (Select State from Employees e WHERE e.EmployeeID = e2.EmployeeID),  
COUNT()  
FROM Employees e2  
GROUP BY (Select State from Employees e WHERE EmployeeID = e2.EmployeeID)
```

Eine besserer Ansatz ist es, dem SELECT-Listenausdruck einen Alias zuzuweisen und diesen in der GROUP BY-Klausel zu referenzieren. Die Verwendung eines Alias ermöglicht es der SELECT-Liste und der GROUP BY-Klausel, korrelierte Unterabfragen zu enthalten. Bei auf diese Weise verwendeten SELECT-Listen-Aliasnamen handelt es sich um eine Erweiterung des Herstellers:

```
SELECT (  
  IF SALARY < 25000  
  THEN 'low range'  
  ELSE IF Salary < 50000  
  THEN 'mid range'  
  ELSE 'high range'  
  ENDIF  
ENDIF) AS Salary_Range,  
COUNT() FROM Employees GROUP BY Salary_Range;
```

Auch wenn SQL Anywhere nicht alle Facetten der SQL/2008-Sprachfunktion T301 (Funktionale Abhängigkeiten) unterstützt, so bietet SQL Anywhere doch etwas Unterstützung für abgeleitete Werte, die auf GROUP BY-Begriffen basieren. SQL Anywhere unterstützt SELECT-Listenausdrücke, die GROUP BY-Begriffe, literale Konstanten und (Host-)Variablen mit oder ohne Skalarfunktionen referenzieren, die diese Werte ändern können. Die folgende Abfrage beispielsweise listet die Anzahl von Mitarbeitern anhand der City-/State-Kombination auf:

```
SELECT City || ' ' || State, SUBSTRING(City,1,3), COUNT()  
FROM Employees  
GROUP BY City, State
```

Siehe auch

- „GROUP BY-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Fehlerbehandlung: Aggregatfunktionen und äußere Referenzen“ [[SQL Anywhere 16 - Änderungen und Upgrades](#)]

Die HAVING-Klausel: Datengruppen auswählen

Die HAVING-Klausel schränkt die Zeilen ein, die von einer Abfrage zurückgegeben werden. Sie setzt Bedingungen für die GROUP BY-Klausel auf ähnliche Art wie WHERE Bedingungen für die SELECT-Klausel setzt.

Die Suchbedingungen der HAVING-Klausel sind mit den WHERE-Suchbedingungen identisch, nur dass die WHERE-Suchbedingungen keine Aggregate enthalten dürfen. Die folgende Verwendung beispielsweise ist zulässig:

```
HAVING AVG( UnitPrice ) > 20
```

Die folgende Verwendung ist nicht zulässig:

```
WHERE AVG( UnitPrice ) > 20
```

HAVING mit Aggregatfunktionen verwenden

Die nachfolgende Anweisung ist ein Beispiel für den einfachen Einsatz der HAVING-Klausel mit einer Aggregatfunktion.

Um die Produkte aufzulisten, die in verschiedenen Größen oder Farben verfügbar sind, benötigen Sie eine Abfrage, die die Zeilen in der Tabelle "Products" nach Namen gruppiert und die Gruppen eliminiert, die nur ein bestimmtes Produkt enthalten:

```
SELECT Name
FROM Products
GROUP BY Name
HAVING COUNT( * ) > 1;
```

Name
Tee Shirt
Baseball Cap
Visor
Sweatshirt

HAVING ohne Aggregatfunktionen verwenden

Die HAVING-Klausel kann auch ohne Aggregatfunktionen verwendet werden.

Mit der nachstehenden Abfrage werden die Produkte gruppiert und das Ergebnis wird nur auf jene Gruppen eingeschränkt, deren Name mit B beginnt.

```
SELECT Name
FROM Products
GROUP BY Name
HAVING Name LIKE 'B%';
```

Name
Baseball Cap

Mehr als eine Bedingung in der HAVING-Klausel

In die HAVING-Klausel können auch mehrere Suchbedingungen eingebaut werden. Sie werden mit den Operatoren AND, OR oder NOT kombiniert, wie im nachstehenden Beispiel gezeigt wird.

Um die Produkte aufzulisten, die in verschiedenen Größen oder Farben verfügbar sind und für die eine Version mehr als 10 Dollar kostet, benötigen Sie eine Abfrage, die die Zeile in der Tabelle "Products" nach Namen gruppiert, jedoch die Gruppen eliminiert, die nur ein bestimmtes Produkt enthalten sowie die Gruppen, deren Maximalpreis unter 10 Dollar liegt.

```
SELECT Name
FROM Products
GROUP BY Name
HAVING COUNT( * ) > 1
AND MAX( UnitPrice ) > 10;
```

Name
Tee Shirt
Sweatshirt

Siehe auch

- „Wo Sie Aggregatfunktionen verwenden können“ auf Seite 471

Die ORDER BY-Klausel: Abfrageergebnisse sortieren

Die ORDER BY-Klausel ermöglicht die Sortierung von Abfrageergebnissen nach einer oder mehreren Spalten. Jede Sortierung kann aufsteigend (ASC) oder absteigend (DESC) erfolgen. Wenn nichts angegeben ist, wird ASC angenommen.

Ein einfaches Beispiel

Die folgende Abfrage gibt die Ergebnisse nach Namen sortiert zurück:

```
SELECT ID, Name
FROM Products
ORDER BY Name;
```

ID	Name
400	Baseball Cap
401	Baseball Cap

ID	Name
700	Shorts
600	Sweatshirt
...	...

Mit mehr als einer Spalte sortieren

Wenn Sie mehr als eine Spalte in der ORDER BY-Klausel nennen, werden die Sortierungen verschachtelt.

Mit der folgenden Anweisung werden die Hemden in der Products-Tabelle erst in aufsteigender Reihenfolge nach Namen und dann innerhalb jedes Namens in absteigender Reihenfolge nach Menge sortiert.

```
SELECT ID, Name, Quantity
FROM Products
WHERE Name like '%shirt%'
ORDER BY Name, Quantity DESC;
```

ID	Name	Quantity
600	Sweatshirt	39
601	Sweatshirt	32
302	Tee Shirt	75
301	Tee Shirt	54
...

Spaltenposition verwenden

Sie können anstelle des Spaltennamens die Positionsnummer einer Spalte in einer SELECT-Liste verwenden. Spaltennamen und SELECT-Listennummern können gemischt werden. Beide nachfolgenden Anweisungen produzieren dieselben Ergebnisse wie die vorhergehenden.

```
SELECT ID, Name, Quantity
FROM Products
WHERE Name like '%shirt%'
ORDER BY 2, 3 DESC;
SELECT ID, Name, Quantity
FROM Products
WHERE Name like '%shirt%'
ORDER BY 2, Quantity DESC
```

Die meisten Versionen von SQL verlangen, dass ORDER BY-Elemente in der SELECT-Liste erscheinen, aber SQL Anywhere hat keine solche Einschränkung. Bei der folgenden Abfrage werden die Ergebnisse nach der Menge geordnet, obwohl die Mengenspalte nicht in der SELECT-Liste erscheint:

```
SELECT ID, Name
FROM Products
WHERE Name like '%shirt%'
ORDER BY 2, Quantity DESC;
```

ORDER BY und NULL

Mit ORDER BY wird NULL vor allen anderen Werten in aufsteigender Reihenfolge sortiert.

ORDER BY und Berücksichtigung von Groß- und Kleinschreibung

Die Auswirkungen der ORDER BY-Klausel auf eine Mischung von Groß- und Kleinschreibung hängen von der Kollation der Datenbank und der Einstellung für die Berücksichtigung von Groß- und Kleinschreibung bei der Erstellung der Datenbank ab.

Zeilenbeschränkungsklauseln in SELECT-, UPDATE- und DELETE-Abfrageblöcken

Mithilfe der Klauseln FIRST, TOP und LIMIT können Sie eine Teilmenge der Zeilen zurückgeben, aktualisieren oder löschen, die die WHERE-Klausel erfüllen. Die Klauseln FIRST, TOP und LIMIT können in jedem SELECT-Abfrageblock verwendet werden, der eine ORDER BY-Klausel enthält. Die Klauseln FIRST und TOP können auch in DELETE- und UPDATE-Abfrageblöcken verwendet werden.

Die Klauseln FIRST, TOP und LIMIT sind Zeilenbeschränkungsklauseln und haben die folgende Syntax:

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*)

Für eine SELECT-Klausel kann nur eine Zeilenbeschränkungsklausel angegeben werden. Wenn diese Klauseln angegeben werden, ist auch eine ORDER BY-Klausel erforderlich, um die Zeilen sinnvoll zu ordnen.

- **row-limitation-option-1** Dieser Klauseltyp kann mit SELECT-, UPDATE- oder DELETE-Abfrageblöcken verwendet werden. Die Argumente TOP und START AT können einfache arithmetische Ausdrücke über Hostvariablen, Ganzzahl-Konstanten oder Ganzzahl-Variablen 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.

Der Ausdruck `limit-expression + startat-expression -1` muss mit einem Wert von weniger als $9223372036854775807 = 2^{64}-1$ ausgewertet werden. 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** Dieser Klauseltyp kann nur in SELECT-Abfrageblöcken verwendet werden. Die Argumente LIMIT und OFFSET können einfache arithmetische Ausdrücke über Hostvariablen, Ganzzahl-Konstanten oder Ganzzahl-Variablen 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. Der Ausdruck `limit-expression + offset-expression` muss mit einem Wert von weniger als $9223372036854775807 = 2^{64}-1$ ausgewertet werden.

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.

Beispiele

Mit der folgenden Abfrage werden Informationen über den Mitarbeiter zurückgegeben, der nach dem Alphabet zuerst kommt, wenn nach Nachnamen sortiert wird:

```
SELECT FIRST *  
FROM Employees  
ORDER BY Surname;
```

Die folgenden Abfragen geben die ersten fünf Mitarbeiter zurück, wenn ihre Namen nach Nachnamen sortiert werden:

```
SELECT TOP 5 *  
FROM Employees  
ORDER BY Surname;
```

```
SELECT *  
FROM Employees  
ORDER BY Surname  
LIMIT 5;
```

Wenn Sie TOP verwenden, können Sie auch START AT verwenden, um einen Offset zu liefern. Die folgenden Anweisungen geben den fünften und sechsten Mitarbeiter zurück, wenn in absteigender Reihenfolge nach Nachnamen sortiert wird:

```
SELECT TOP 2 START AT 5 *  
FROM Employees  
ORDER BY Surname DESC;  
  
SELECT *
```

```
FROM Employees
ORDER BY Surname DESC
LIMIT 2 OFFSET 4;
```

```
SELECT *
FROM Employees
ORDER BY Surname DESC
LIMIT 4,2;
```

FIRST und TOP sollten nur in Verbindung mit einer ORDER BY-Klausel verwendet werden, um konsistente Ergebnisse zu gewährleisten. Die Verwendung von FIRST oder TOP ohne ORDER BY erzeugt eine Syntaxwarnung und kann unvorhersehbare Ergebnisse liefern.

```
CREATE OR REPLACE VARIABLE atop INT = 10;
```

Die folgenden Abfragen geben die ersten fünf Mitarbeiter zurück, wenn ihre Namen nach Nachnamen sortiert werden:

```
SELECT TOP (atop -5) *
FROM Employees
ORDER BY Surname;
```

```
SELECT *
FROM Employees
ORDER BY Surname
LIMIT (atop-5);
```

Die folgenden Anweisungen geben den fünften und sechsten Mitarbeiter zurück, wenn in absteigender Reihenfolge nach Nachnamen sortiert wird:

```
SELECT TOP (atop - 8) START AT (atop -2 -3) *
FROM Employees
ORDER BY Surname DESC;
```

```
SELECT *
FROM Employees
ORDER BY Surname DESC
LIMIT (atop - 8) OFFSET (atop -2 -3 -1);
```

Siehe auch

- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „UPDATE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DELETE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „reserved_keywords-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

ORDER BY und GROUP BY

Sie können eine ORDER BY-Klausel verwenden, um die Ergebnisse einer GROUP BY-Abfrage in bestimmter Weise zu sortieren.

Beispiel

Folgende Abfrage sucht den Durchschnittspreis jedes Produkts und sortiert die Ergebnisse nach dem Durchschnittspreis:

```
SELECT Name, AVG( UnitPrice )
FROM Products
```

```
GROUP BY Name  
ORDER BY AVG( UnitPrice );
```

Name	AVG(Products.UnitPrice)
Visor	7
Baseball Cap	9.5
Tee Shirt	12.33333333
Shorts	15
...	...

Mengenoperationen für Abfrageergebnisse mit UNION, INTERSECT und EXCEPT

Die in diesem Abschnitt beschriebenen Operatoren führen Mengenoperationen mit den Ergebnissen von zwei oder mehr Abfragen aus. Auch wenn viele dieser Operationen ausgeführt werden können, indem Operationen in der WHERE- oder HAVING-Klausel verwendet werden, gibt es einige Operationen, bei denen ein Ausführen ohne Verwendung dieser Set-basierten Operatoren sehr schwierig ist. Beispiel:

- Wenn Daten nicht normalisiert sind, können Sie scheinbar unvereinbare Informationen in eine einzige Ergebnismenge zusammenführen, obwohl die Tabellen nicht verknüpft sind.
- NULL wird von Mengenoperatoren anders als in der WHERE-Klausel oder HAVING-Klausel behandelt. In der WHERE-Klausel oder HAVING-Klausel werden zwei NULL-enthaltende Zeilen mit identischen Nicht-NULL-Einträgen als nicht identisch angesehen, da die zwei NULL-Werte nicht als identisch definiert sind. Die Mengenoperatoren sehen zwei solche Zeilen als gleich an.

Siehe auch

- „Mengenoperatoren und NULL“ auf Seite 489
- „EXCEPT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „INTERSECT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „UNION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

UNION-Klausel: Kombinieren von Ergebnismengen

Mit dem UNION-Operator werden die Ergebnisse von zwei oder mehr Abfragen in einer einzigen Ergebnismenge kombiniert.

Standardmäßig entfernt der UNION-Operator Duplikatzellen aus der Ergebnismenge. Wenn Sie die Option ALL verwenden, werden Duplikate nicht entfernt. Die Spalten in der endgültigen Ergebnismenge haben dieselben Namen wie die Spalten in der ersten Ergebnismenge. Jede beliebige Anzahl von UNION-Operatoren kann verwendet werden.

Standardmäßig wird eine Anweisung, die mehrere UNION-Operatoren enthält, von links nach rechts aufgelöst. Klammern können benutzt werden, um die Reihenfolge der Auflösung festzulegen.

Die beiden folgenden Ausdrücke beispielsweise sind nicht gleichwertig, weil Duplikatzeilen auf unterschiedliche Weise aus der Ergebnismenge entfernt werden:

```
x UNION ALL ( y UNION z )  
  
(x UNION ALL y) UNION z
```

Im ersten Ausdruck werden Duplikate in der Vereinigung zwischen y und z eliminiert. In der Vereinigung zwischen dieser Menge und x werden Duplikate nicht eliminiert. Im zweiten Ausdruck werden Duplikate in der Vereinigung zwischen x und y eingeschlossen, aber in der darauf folgenden Vereinigung mit z eliminiert.

EXCEPT und INTERSECT

Die EXCEPT-Klausel listet die Unterschiede zwischen zwei Ergebnismengen auf. Die folgende allgemeine Konstruktion listet alle Zeilen auf, die in der Ergebnismenge von query-1, aber nicht in der Ergebnismenge von query-2 vorkommen.

```
query-1  
EXCEPT  
query-2
```

Die INTERSECT-Klausel listet die Zeilen auf, die jeweils in beiden Ergebnismengen vorkommen. Die folgende allgemeine Konstruktion listet alle Zeilen auf, die in der Ergebnismenge von sowohl query-1 als auch query-2 vorkommen.

```
query-1  
INTERSECT  
query-2
```

Wie die UNION-Klausel berücksichtigen sowohl EXCEPT als auch INTERSECT den ALL-Modifizierer, der verhindert, dass doppelte Zeilen aus der Ergebnismenge eliminiert werden.

Siehe auch

- „EXCEPT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „INTERSECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Regeln für Mengenoperationen

Die folgenden Regeln gelten für UNION-, EXCEPT- und INTERSECT-Anweisungen:

- **Vorrang** Die Operatoren UNION und EXCEPT haben die gleiche Priorität und werden beide von links nach rechts ausgewertet. Der INTERSECT-Operator hat eine höhere Priorität als die Operatoren UNION und EXCEPT und wird ebenfalls von links nach rechts ausgewertet, wenn mehr als eine INTERSECT verwendet wird.

- **Gleiche Anzahl von Elementen in den SELECT-Listen** Alle SELECT-Listen in den Abfragen müssen dieselbe Anzahl von Ausdrücken (z.B. von Spaltennamen, arithmetischen Ausdrücken und Aggregatfunktionen) aufweisen. Die folgende Anweisung ist ungültig, weil die erste SELECT-Liste länger ist als die zweite:

```
SELECT store_id, city, state
FROM stores
UNION
SELECT store_id, city
FROM stores_east;
```

- **Datentypen müssen übereinstimmen** Einander entsprechende Ausdrücke in den Auswahllisten müssen denselben Datentyp haben, oder eine implizite Datenkonvertierung muss zwischen den beiden Datentypen möglich sein bzw. eine explizite Konvertierung eingeplant werden.

Beispiel: Ein UNION-, INTERSECT- oder EXCEPT-Vorgang ist nicht möglich zwischen einer Spalte des Datentyps CHAR und einer Spalte des Datentyps INT, wenn nicht eine explizite Konvertierung eingebaut wird. Allerdings ist eine Mengenoperation möglich zwischen einer Spalte des Datentyps MONEY und des Datentyps INT.

- **Spaltenreihenfolge** Sie müssen einander entsprechende Ausdrücke in die einzelnen Abfragen einer Mengenoperation in derselben Reihenfolge eingeben, da die Mengenoperatoren eins-zu-eins in der Reihenfolge vergleichen, die in den einzelnen Abfragen in den SELECT-Klauseln angegeben wird.
- **Mehrere Mengenoperationen** Sie können mehrere Mengenoperationen zusammenbinden, wie dies im folgenden Beispiel der Fall ist:

```
SELECT City AS Cities
FROM Contacts
UNION
SELECT City
FROM Customers
UNION
SELECT City
FROM Employees;
```

Bei UNION-Anweisungen ist die Reihenfolge der Abfragen ohne Bedeutung. Bei INTERSECT ist die Reihenfolge von Bedeutung, wenn es zwei oder mehr Abfragen gibt. Bei EXCEPT ist die Reihenfolge immer von Bedeutung.

- **Spaltentitel** Die Spaltennamen in der Tabelle, die aus einem UNION-Vorgang hervorgehen, werden der ersten individuellen Abfrage in der Anweisung entnommen. Definieren Sie einen neuen Spaltentitel für die Ergebnismenge in der SELECT-Liste der ersten Abfrage, wie im folgenden Beispiel gezeigt wird:

```
SELECT City AS Cities
FROM Contacts
UNION
SELECT City
FROM Customers;
```

In der folgenden Abfrage bleibt der Spaltentitel "City" erhalten, da er in der ersten Abfrage der UNION-Klausel definiert wurde.

```
SELECT City
FROM Contacts
UNION
SELECT City AS Cities
FROM Customers;
```

Oder Sie verwenden die WITH-Klausel, um die Spaltennamen festzulegen. Beispiel:

```
WITH V( Cities )
AS ( SELECT City
FROM Contacts
UNION
SELECT City
FROM Customers )
SELECT * FROM V;
```

- **Sortierung der Ergebnisse** Mit der WITH-Klausel der SELECT-Anweisung können Sie die Reihenfolge der Spaltennamen in der SELECT-Liste bestimmen. Beispiel:

```
WITH V( CityName )
AS ( SELECT City AS Cities
FROM Contacts
UNION
SELECT City
FROM Customers )
SELECT * FROM V
ORDER BY CityName;
```

Sie können auch eine einzige ORDER BY-Klausel am Ende der Liste der Abfragen verwenden, müssen aber Ganzzahlen anstelle von Spaltennamen benutzen, wie im folgenden Beispiel gezeigt wird:

```
SELECT City AS Cities
FROM Contacts
UNION
SELECT City
FROM Customers
ORDER BY 1;
```

Mengenoperatoren und NULL

NULL wird von den Mengenoperatoren UNION, EXCEPT und INTERSECT anders behandelt, als dies in Suchbedingungen der Fall ist. Der Unterschied ist einer der Hauptgründe dafür, Mengenoperatoren zu verwenden.

Wenn Zeilen verglichen werden, behandeln Mengenoperatoren NULL gleichwertig. Wenn dagegen NULL mit NULL in einer Suchbedingung verglichen wird, ist das Ergebnis unbekannt (nicht TRUE).

Eine Konsequenz dieses Unterschieds liegt darin, dass die Zeilenanzahl in der Ergebnismenge für `query-1 EXCEPT ALL query-2` immer die Differenz der Zeilenanzahl in den Ergebnismengen der einzelnen Abfragen ist.

Nehmen wir zum Beispiel zwei Tabellen T1 und T2, jeweils mit den folgenden Spalten:

```
col1 INT,
col2 CHAR(1)
```

Tabellen und Daten werden wie folgt eingerichtet:

```
CREATE TABLE T1 (col1 INT, col2 CHAR(1));
CREATE TABLE T2 (col1 INT, col2 CHAR(1));
INSERT INTO T1 (col1, col2) VALUES(1, 'a');
INSERT INTO T1 (col1, col2) VALUES(2, 'b');
INSERT INTO T1 (col1) VALUES(3);
INSERT INTO T1 (col1) VALUES(3);
INSERT INTO T1 (col1) VALUES(4);
INSERT INTO T1 (col1) VALUES(4);
INSERT INTO T2 (col1, col2) VALUES(1, 'a');
INSERT INTO T2 (col1, col2) VALUES(2, 'x');
INSERT INTO T2 (col1) VALUES(3);
```

Die Daten in den Tabellen sind Folgende:

- Tabelle T1.

col1	col2
1	a
2	b
3	(NULL)
3	(NULL)
4	(NULL)
4	(NULL)

- Tabelle T2.

col1	col2
1	a
2	x
3	(NULL)

Die folgende Abfrage sucht nach Zeilen in T1, die auch in T2 vorkommen:

```
SELECT T1.col1, T1.col2
FROM T1 JOIN T2
ON T1.col1 = T2.col1
AND T1.col2 = T2.col2;
```

T1.col1	T1.col2
1	a

Die Zeile (3, NULL) kommt nicht in der Ergebnismenge vor, da der Vergleich zwischen NULL und NULL nicht TRUE ist. Wenn dagegen das Problem mit dem INTERSECT-Operator behandelt wird, wird eine Zeile mit NULL einbezogen:

```
SELECT col1, col2
FROM T1
INTERSECT
SELECT col1, col2
FROM T2;
```

col1	col2
1	a
3	(NULL)

Die folgende Abfrage verwendet Suchbedingungen, um Zeilen aufzulisten, die in T1, aber nicht in T2 vorkommen:

```
SELECT col1, col2
FROM T1
WHERE col1 NOT IN (
SELECT col1
FROM T2
WHERE T1.col2 = T2.col2 )
OR col2 NOT IN (
SELECT col2
FROM T2
WHERE T1.col1 = T2.col1 );
```

col1	col2
2	b
3	(NULL)
4	(NULL)
3	(NULL)
4	(NULL)

Die NULL-enthaltenden Zeilen von T1 werden durch den Vergleich nicht ausgeschlossen. Wenn dagegen das Problem unter Verwendung von EXCEPT ALL behandelt wird, werden NULL-enthaltende Zeilen ausgeschlossen, die in beiden Tabellen vorkommen. In diesem Fall wird die Zeile (3, NULL) in T2 als gleichwertig mit der Zeile (3, NULL) in T1 identifiziert.

```
SELECT col1, col2
FROM T1
EXCEPT ALL
SELECT col1, col2
FROM T2;
```

col1	col2
2	b
3	(NULL)
4	(NULL)
4	(NULL)

Der EXCEPT-Operator ist sogar noch restriktiver. Er eliminiert beide Zeilen (3, NULL) von T1 und schließt eine der Zeilen (4, NULL) als mehrfach vorhanden aus.

```
SELECT col1, col2
FROM T1
EXCEPT
SELECT col1, col2
FROM T2;
```

col1	col2
2	b
4	(NULL)

Joins: Daten aus mehreren Tabellen abrufen

Wenn Sie eine Datenbank erstellen, standardisieren Sie die Daten, indem Sie Informationen, die zu verschiedenen Objekten gehören, in verschiedenen Tabellen ablegen, und nicht in einer einzigen Tabelle mit vielen redundanten Einträgen. Um zusammengehörige Daten aus mehreren Tabellen abzurufen, führen Sie daher einen Join-Vorgang mit dem SQL-Operator JOIN durch. Ein Join-Vorgang erstellt eine größere Tabelle auf der Basis der Informationen, die aus zwei oder mehr Tabellen (oder Ansichten) stammen. Durch die Verwendung unterschiedlicher Joins können Sie eine Vielzahl solcher virtueller Tabellen erstellen, die jeweils für eine bestimmte Aufgabe geeignet sind.

Tabellenliste anzeigen

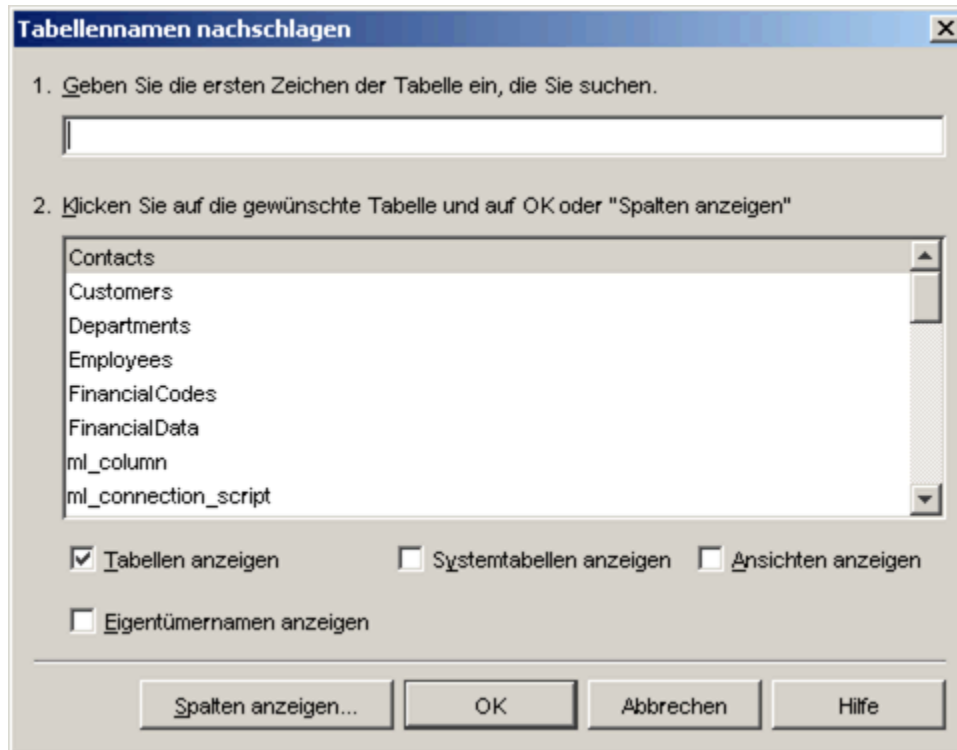
Sie können alle Tabellen sowie deren Spalten für die Datenbank anzeigen, mit der Sie über Interactive SQL verbunden sind.

Voraussetzungen

Sie müssen mit der Datenbank verbunden sein.

Aufgabe

1. In Interactive SQL drücken Sie die F7-Taste, um eine Liste der Tabellen in der Datenbank anzuzeigen, mit der Sie verbunden sind.



2. Wählen Sie eine Tabelle aus und klicken Sie auf **Spalten anzeigen**.
3. Drücken Sie die ESC-Taste, um zur Tabellenliste zurückzukehren. Drücken Sie nochmals die ESC-Taste, um zum Fensterausschnitt **SQL-Anweisungen** zurückzukehren. Drücken Sie die Eingabetaste, um den ausgewählten Tabellen- oder Spaltennamen in den Fensterausschnitt **SQL-Anweisungen** an der aktuellen Cursorposition zu kopieren.
4. Drücken Sie ESC, um die Liste zu verlassen.

Weitere Hinweise zu den Tabellen in der SQL Anywhere-Beispieldatenbank finden Sie unter „Praktische Einführung: Verbindung mit der Beispieldatenbank herstellen“ [[SQL Anywhere Server - Datenbankadministration](#)].

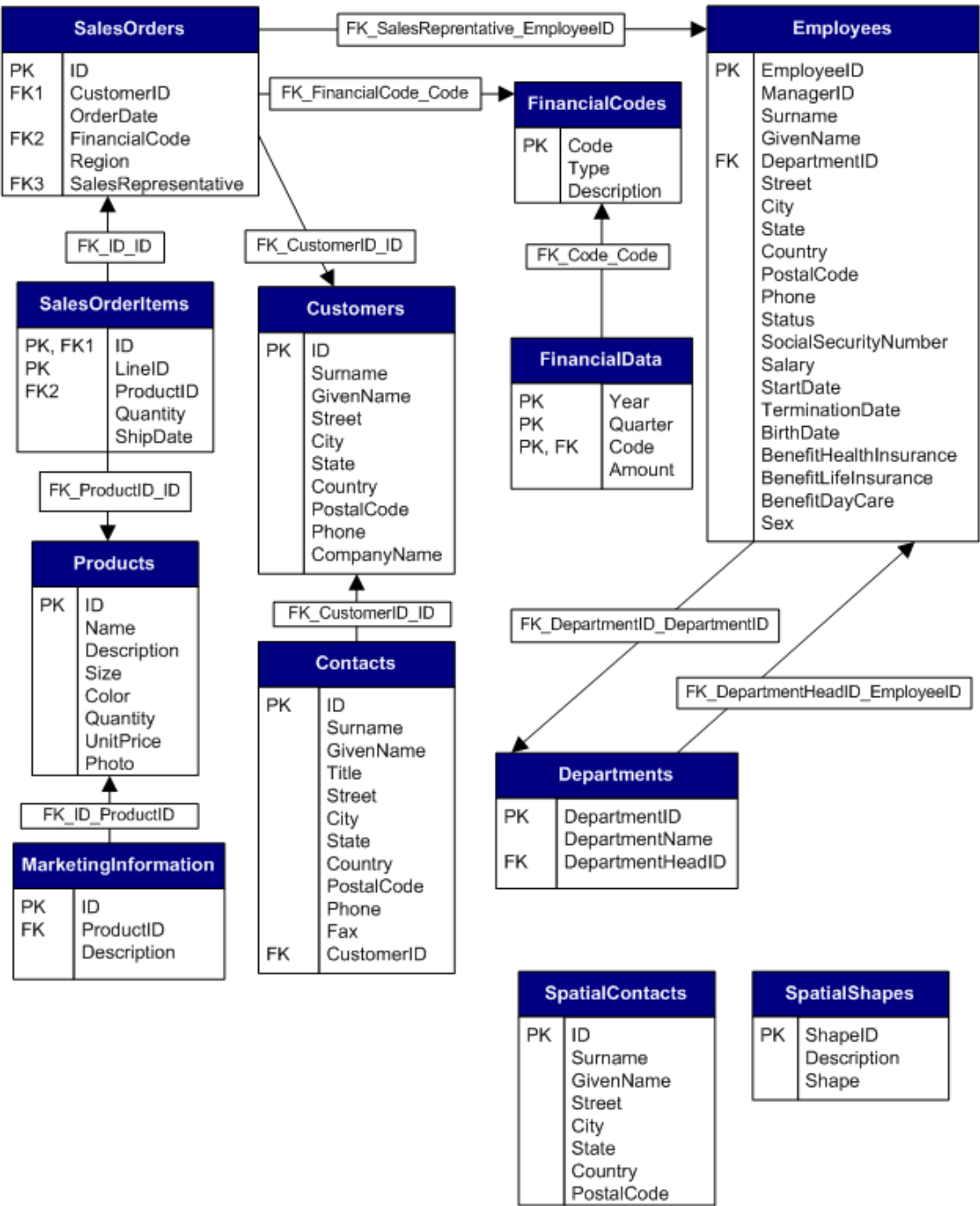
Ergebnisse

Eine Liste aller Tabellen der Datenbank, mit der Sie verbunden sind, wird angezeigt. Sie haben die Möglichkeit, die Spalten jeder einzelnen Tabelle anzuzeigen.

Schema der Beispieldatenbank

In der folgenden Darstellung wird die SQL Anywhere-Beispieldatenbank mit den Namen der Fremdschlüssel angezeigt, die die Tabellen zueinander in Beziehung setzen. Diese Fremdschlüssel-Rollenamen sind für einige erweiterte Joins erforderlich.

Weitere Hinweise zu Rollennamen finden Sie unter „Schlüssel-Joins bei mehrfachen Fremdschlüsselbeziehungen“ auf Seite 525 .



Funktionsweise von Joins

Ein **Join** ist ein Vorgang, der die Zeilen in Tabellen kombiniert, indem er die Werte in bestimmten Spalten vergleicht. Dieser Abschnitt enthält einen Überblick über die Join-Syntax von SQL Anywhere.

Eine relationale Datenbank speichert Informationen über unterschiedliche Arten von Objekten in unterschiedlichen Tabellen. Zum Beispiel befinden sich Informationen, die zu bestimmten Mitarbeitern gehören, in einer Tabelle, Informationen zu Abteilungen hingegen in einer anderen. Die Tabelle "Employees" enthält Daten wie z.B. Namen und Adressen von Mitarbeitern. Die Tabelle "Departments" enthält Informationen über eine Abteilung, wie z.B. den Namen der Abteilung und wer der Abteilungsleiter ist.

Die meisten Fragen können nur mithilfe einer Kombination der Informationen aus verschiedenen Tabellen beantwortet werden. Um beispielsweise die Frage zu beantworten: "Wer leitet die Verkaufsabteilung?", verwenden Sie die Departments-Tabelle, um den korrekten Mitarbeiter zu identifizieren, und suchen dann den Mitarbeiternamen in der Employees-Tabelle.

Joins bieten eine Möglichkeit, solche Fragen zu bearbeiten, indem eine neue virtuelle Tabelle erstellt wird, die Informationen aus mehreren Tabellen enthält. Beispiel: Sie können eine Liste der Abteilungsleiter erstellen, indem Sie die Informationen aus der Tabelle "Employees" und der Tabelle "Departments" kombinieren. Sie geben mit der Klausel FROM an, welche Tabellen die benötigten Daten enthalten.

Damit der Join sinnvoll ist, müssen Sie die richtigen Spalten jeder Tabelle kombinieren. Um Abteilungsleiter aufzulisten, muss jede Zeile der kombinierten Tabelle den Namen einer Abteilung und den Namen des Mitarbeiters enthalten, der die Abteilung leitet. Wie die Spalten in der kombinierten Tabelle aufeinander abgestimmt werden, steuern Sie entweder durch die Angabe eines bestimmten Typs für einen Join-Vorgang oder durch die Klausel ON.

Siehe auch

- „FROM-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Join-Bedingungen

Tabellen können mit **Join-Bedingungen** kombiniert werden. Eine Join-Bedingung ist eine Suchbedingung. Sie wählt eine Teilmenge von Zeilen aus den verknüpften Tabellen aus, basierend auf den Beziehungen zwischen den Werten in den Spalten. Folgende Abfrage ruft z.B. Daten aus den Tabellen "Products" und "SalesOrderItems" ab.

```
SELECT *
FROM Products JOIN SalesOrderItems
ON Products.ID = SalesOrderItems.ProductID;
```

Die Join-Bedingung in dieser Abfrage lautet wie folgt:

```
Products.ID = SalesOrderItems.ProductID
```

Diese Join-Bedingung bedeutet, dass Zeilen in der Ergebnismenge nur kombiniert werden können, wenn sie in beiden Tabellen dieselbe Produkt-ID haben.

Join-Bedingungen können explizit sein oder generiert werden. Eine **explizite Join-Bedingung** ist eine Join-Bedingung, die sich in einer ON- oder WHERE-Klausel befindet. Folgende Abfrage verwendet eine ON-Klausel: Sie produziert ein Kreuzprodukt der beiden Tabellen (alle Zeilenkombinationen), wobei jedoch Zeilen ausgeschlossen werden, deren ID-Nummern nicht übereinstimmen. Das Ergebnis ist eine Liste mit Kunden und Details ihrer Bestellungen.

```
SELECT *  
FROM Customers  
JOIN SalesOrders  
ON SalesOrders.CustomerID = Customers.ID;
```

Eine **generierte Join-Bedingung** ist eine Join-Bedingung, die automatisch erstellt wird, wenn Sie KEY JOIN oder NATURAL JOIN festlegen. Bei Schlüssel-Joins basiert die generierte Join-Bedingung auf den Beziehungen der Fremdschlüssel zwischen den Tabellen. Im Fall natürlicher Joins basiert die generierte Join-Bedingung auf Spalten, die denselben Namen haben.

Tipp

Die Syntax des Schlüssel-Joins und des natürlichen Joins sind Abkürzungen: Sie erhalten identische Ergebnisse, wenn Sie das Schlüsselwort JOIN ohne KEY oder NATURAL verwenden und dann dieselbe Join-Bedingung in einer ON-Klausel angeben.

Wenn Sie eine ON-Klausel mit einem Schlüssel-Join oder einem natürlichen Join verwenden, ist die Join-Bedingung die **Verbindung** der expliziten Join-Bedingung mit der generierten Join-Bedingung. Die Join-Bedingungen werden mit dem Schlüsselwort AND verknüpft.

Verknüpfte Tabellen

SQL Anywhere unterstützt folgende Klassen von verknüpften Tabellen:

- **CROSS JOIN** Dieser Join-Typ von zwei Tabellen produziert alle möglichen Zeilenkombinationen aus den beiden Tabellen. Die Größe der Ergebnismenge ist die Anzahl der Zeilen in der ersten Tabelle multipliziert mit der Anzahl der Zeilen in der zweiten Tabelle. Ein Cross-Join wird auch als Kreuzprodukt oder kartesisches Produkt bezeichnet. Es ist nicht möglich, eine ON-Klausel mit einem CROSS JOIN zu verwenden.
- **KEY JOIN** Diese Art von Join-Bedingung verwendet Fremdschlüssel-Beziehungen zwischen den Tabellen. Der Schlüssel-Join ist die Standardeinstellung, wenn das JOIN-Schlüsselwort ohne Angabe eines Join-Typs (wie z.B. INNER, OUTER usw.) verwendet wird und keine ON-Klausel enthalten ist.
- **NATURAL JOIN** Dieser Join wird automatisch basierend auf Spalten erzeugt, die denselben Namen haben.
- **Join mit ON-Klausel** Dieser Join-Typ ist das Ergebnis von expliziten Spezifikation der Join-Bedingung in einer ON-Klausel. Bei der Verwendung mit einem Schlüssel-Join oder einem natürlichen Join enthält die Join-Bedingung sowohl die generierte Join-Bedingung als auch die explizite Join-Bedingung. Wenn das Schlüsselwort JOIN ohne die Schlüsselwörter KEY oder NATURAL verwendet wird, gibt es keine generierte Join-Bedingung.

Inner- und Outer-Joins

Schlüssel-Joins, natürliche Joins und Joins mit einer ON-Klausel können durch die Angabe von INNER, LEFT OUTER, RIGHT OUTER oder FULL OUTER qualifiziert werden. Die Standardeinstellung ist INNER. Wenn Sie die Schlüsselwörter LEFT, RIGHT oder FULL verwenden, ist das Schlüsselwort OUTER optional.

In einem Inner-Join erfüllt jede Zeile im Ergebnis die Join-Bedingung.

In einem Links- oder Rechts-Outer-Join werden alle Zeilen für eine der Tabellen beibehalten, und für die andere Tabelle werden für die Zeilen, die die Join-Bedingung nicht erfüllen, NULL zurückgegeben. In einem Rechts-Outer-Join wird z.B. die rechte Seite beibehalten und die linke Seite liefert NULL.

In einem vollständigen Outer-Join werden alle Zeilen für beide Tabellen beibehalten, und Zeilen, die die Join-Bedingung nicht erfüllen, liefern NULL.

Siehe auch

- „Explizite Join-Bedingungen (ON-Klausel)“ auf Seite 499

Joins zwischen zwei Tabellen

Betrachten Sie folgende Abfrage, um zu verstehen, wie ein einfacher Inner-Join ausgewertet wird. Sie beantwortet die Frage: Welche Produktgrößen wurden in der Menge bestellt, die der Lagermenge entspricht?

```
SELECT DISTINCT Name, Size,
               SalesOrderItems.Quantity
FROM Products JOIN SalesOrderItems
ON Products.ID = SalesOrderItems.ProductID
   AND Products.Quantity = SalesOrderItems.Quantity;
```

Name	Size	Quantity
Baseball Cap	One size fits all	12
Visor	One size fits all	36

Sie können die Abfrage wie folgt interpretieren. Es handelt sich hierbei um eine konzeptuelle Erklärung der Verarbeitung dieser Abfrage, um die Semantik einer Abfrage mit einem Join zu erklären. Die Erklärung stellt nicht dar, wie SQL Anywhere die Ergebnismenge tatsächlich ermittelt.

- Es wird ein Kreuzprodukt der Tabellen "Products" und "SalesOrderItems" erstellt. Ein Kreuzprodukt enthält alle Zeilenkombinationen der beiden Tabellen.
- Es werden alle Zeilen ausgeschlossen, bei denen die Produkt-IDs nicht identisch sind (mithilfe der Join-Bedingung `Products.ID = SalesOrderItems.ProductID`).
- Es werden alle Zeilen ausgeschlossen, bei denen die Menge nicht identisch ist (mithilfe der Join-Bedingung `Products.Quantity = SalesOrderItems.Quantity`).

- Es wird eine Ergebnistabelle mit drei Spalten erstellt: "Products.Name", "Products.Size" und "SalesOrderItems.Quantity".
- Es werden alle Duplikatzellen ausgeschlossen (mithilfe des Schlüsselworts DISTINCT).

Siehe auch

- [„Outer-Joins“ auf Seite 505](#)

Joins zwischen mehr als zwei Tabellen

In SQL Anywhere gibt es keine feste Obergrenze für die Anzahl der Tabellen, die verknüpft werden können.

Wenn Sie mehr als zwei Tabellen verknüpfen, ist die Verwendung von Klammern optional. Wenn Sie keine Klammern verwenden, wertet SQL Anywhere die Anweisung von links nach rechts aus. Daher ist A JOIN B JOIN C äquivalent mit (A JOIN B) JOIN C. Auch die beiden folgenden SELECT-Anweisungen sind gleichwertig:

```
SELECT *  
FROM A JOIN B JOIN C JOIN D;
```

```
SELECT *  
FROM ( ( A JOIN B ) JOIN C ) JOIN D;
```

Wenn mehr als zwei Tabellen verknüpft werden, umfasst der Join Tabellenausdrücke. Im Beispiel A JOIN B JOIN C wird der Tabellenausdruck A JOIN B mit C verknüpft. Das bedeutet, konzeptuell gesehen, dass A und B verknüpft werden und dann das Ergebnis mit C verknüpft wird.

Die Reihenfolge von Joins ist wichtig, wenn der Tabellenausdruck Outer-Joins enthält. A JOIN B LEFT OUTER JOIN C wird beispielsweise als (A JOIN B) LEFT OUTER JOIN C interpretiert. Der Tabellenausdruck A JOIN B wird mit C verknüpft. Der Tabellenausdruck A JOIN B wird beibehalten und Tabelle C liefert auch Nullwerte.

Siehe auch

- [„Outer-Joins“ auf Seite 505](#)
- [„Schlüssel-Joins von Tabellenausdrücken“ auf Seite 528](#)
- [„Natürliche Joins von Tabellenausdrücken“ auf Seite 522](#)

Join-kompatible Datentypen

Wenn Sie zwei Tabellen durch einen Join verbinden, müssen die verglichenen Spalten denselben oder einen kompatiblen Datentyp haben.

Siehe auch

- [„Datentypvergleiche“ \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)

Joins in DELETE-, UPDATE- und INSERT-Anweisungen

Sie können Joins in DELETE-, UPDATE-, INSERT- und SELECT-Anweisungen verwenden. Sie können einige Cursor, die Joins enthalten, aktualisieren, wenn die `ansi_update_constraints`-Option auf "Off" gesetzt ist. Dies ist die Standardeinstellung für Datenbanken, die vor Version 7 von SQL Anywhere erstellt wurden. Bei Datenbanken, die mit Version 7 oder später erstellt wurden, ist der Standardwert Cursor.

Siehe auch

- „ansi_update_constraints-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Nicht-ANSI-Joins

SQL Anywhere unterstützt ISO/ANSI-Standards für Joins. Des weiteren werden folgende Nicht-Standard-Joins unterstützt:

- „Transact-SQL-Outer-Joins (*= oder =*)“
- „Doppelte Korrelationsnamen in Joins (Stern-Joins)“
- „Schlüssel-Joins“

Verwenden Sie die Funktion `REWRITE`, um die ANSI-Entsprechung eines Nicht-ANSI-Joins anzuzeigen.

Siehe auch

- „REWRITE-Funktion [Verschiedene]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Explizite Join-Bedingungen (ON-Klausel)

Anstelle oder zusammen mit einem Schlüssel-Join oder einem natürlichen Join können Sie einen Join festlegen, der eine explizite Join-Bedingung verwendet. Sie können eine Join-Bedingung angeben, indem Sie eine ON-Klausel unmittelbar nach dem Join verwenden. Die Join-Bedingung bezieht sich immer auf den Join, der unmittelbar vorausgeht. Mit der ON-Klausel wird eine Einschränkung auf die Zeilen in einem Join angewendet, etwa so, wie mit der WHERE-Klausel Einschränkungen auf die Zeilen in einer Abfrage angewendet werden.

Die ON-Klausel ermöglicht es Ihnen, nützlichere Joins zu konstruieren als `CROSS JOIN`. Sie können die ON-Klausel z.B. auf einen Join der Tabellen "SalesOrders" und "Employees" anwenden, um nur die Zeilen abzurufen, für die der Verkäufer (SalesRepresentative) in der Tabelle "SalesOrders" derselbe ist wie in jeder Zeile des Ergebnisses in der Tabelle "Employees". Auf diese Weise enthält jede Zeile dann Informationen über einen Auftrag und den dafür verantwortlichen Verkäufer.

In folgender Abfrage dient beispielsweise die erste ON-Klausel dazu, "SalesOrders" mit "Customers" zu verknüpfen. Die zweite ON-Klausel dient dazu, den Tabellenausdruck (SalesOrders JOIN Customers) mit der Basistabelle "SalesOrderItems" zu verknüpfen.

```
SELECT *
FROM SalesOrders JOIN Customers
    ON SalesOrders.CustomerID = Customers.ID
```

```
JOIN SalesOrderItems
ON SalesOrderItems.ID = SalesOrders.ID;
```

Tabellenreferenzen in ON-Klauseln

Die Tabellen, die in einer ON-Klausel referenziert werden, müssen Teil des Joins sein, den die ON-Klausel ändert. Folgende Anweisung ist z.B. ungültig:

```
FROM ( A KEY JOIN B ) JOIN ( C JOIN D ON A.x = C.x )
```

Das Problem ist hier, dass die Join-Bedingung $A.x = C.x$ Tabelle A referenziert, die nicht Teil des Joins ist, den die Bedingung modifiziert (in diesem Fall $C \text{ JOIN } D$).

Seit ANSI/ISO-Standard SQL99 und SQL Anywhere 7.0 gibt es jedoch eine Ausnahme von dieser Regel: Wenn Sie Kommata zwischen Tabellenausdrücken verwenden, kann eine ON-Bedingung eines Joins eine Tabelle referenzieren, die ihr in der FROM-Klausel syntaktisch vorausgeht. Folgende Anweisung ist daher gültig:

```
FROM ( A KEY JOIN B ) , ( C JOIN D ON A.x = C.x )
```

Siehe auch

- „Schlüssel-Joins“ auf Seite 523
- „Kommas“ auf Seite 503

Beispiel

Im folgenden Beispiel werden die Tabellen "SalesOrders" und "Employees" durch einen Join verknüpft. Jede Zeile im Ergebnis reflektiert die Zeilen in der Tabelle "SalesOrders", bei denen der Wert der Spalte "SalesRepresentative" mit dem Wert der Spalte "EmployeeID" in der Tabelle "Employees" übereinstimmt.

```
SELECT Employees.Surname, SalesOrders.ID, SalesOrders.OrderDate
FROM SalesOrders
JOIN Employees
ON SalesOrders.SalesRepresentative = Employees.EmployeeID;
```

Surname	ID	OrderDate
Chin	2008	4/2/2001
Chin	2020	3/4/2001
Chin	2032	7/5/2001
Chin	2044	7/15/2000
Chin	2056	4/15/2001
...

Es folgen einige Hinweise zu diesem Beispiel:

- Die Ergebnisse dieser Abfrage enthalten nur 648 Zeilen (eine für jede Zeile in der Tabelle "SalesOrders"). Von den 48.600 Zeilen in dem Kreuzprodukt enthalten nur 648 in den beiden Tabellen dieselbe Mitarbeiternummer.
- Die Reihenfolge der Ergebnisse ist ohne Bedeutung. Sie können eine ORDER BY-Klausel hinzufügen, um der Abfrage eine bestimmte Reihenfolge aufzuzwingen.
- Die ON-Klausel schließt Spalten mit ein, die nicht in der endgültigen Ergebnismenge enthalten sind.

Generierte Joins und die ON-Klausel

Schlüssel-Joins sind die Standardeinstellung, wenn das Schlüsselwort JOIN verwendet und kein Join-Typ angegeben wird—außer Sie verwenden eine ON-Klausel. Wenn Sie eine ON-Klausel mit einem nicht spezifizierten JOIN verwenden, ist der Schlüssel-Join nicht die Standardeinstellung, und es wird keine generierte Join-Bedingung angewendet.

Folgendes ist z.B. ein Schlüssel-Join, da der Schlüssel-Join die Standardeinstellung ist, wenn das Schlüsselwort JOIN verwendet wird und keine ON-Klausel vorhanden ist:

```
SELECT *
FROM A JOIN B;
```

Folgendes ist ein Join zwischen Tabelle A und Tabelle B mit der Join-Bedingung $A.x = B.y$. Das ist kein Schlüssel-Join.

```
SELECT *
FROM A JOIN B ON A.x = B.y;
```

Wenn Sie einen KEY JOIN oder einen NATURAL JOIN angeben und eine ON-Klausel verwenden, ist die sich ergebende Join-Bedingung die Verbindung der generierten Join-Bedingung und der expliziten Join-Bedingung(en). Folgende Anweisung enthält z.B. zwei Join-Bedingungen: Eine wird aufgrund des Schlüssel-Joins generiert, und eine zweite ist explizit in der ON-Klausel angegeben.

```
SELECT *
FROM A KEY JOIN B ON A.x = B.y;
```

Wenn die vom Schlüssel-Join generierte Join-Bedingung $A.w = B.z$ ist, ist die folgende Anweisung gleichwertig:

```
SELECT *
FROM A JOIN B
  ON A.x = B.y
 AND A.w = B.z;
```

Siehe auch

- „Schlüssel-Joins“ auf Seite 523

Typen expliziter Join-Bedingungen

Die meisten Join-Bedingungen basieren auf Gleichheit. Sie werden als **Equi-Joins** bezeichnet. Beispiel:

```
SELECT *  
FROM Departments JOIN Employees  
ON Departments.DepartmentID = Employees.DepartmentID;
```

Sie brauchen jedoch in einer Join-Bedingung nicht mit Gleichheit (=) zu arbeiten. Sie können eine Suchbedingung verwenden, wie z.B. Bedingungen, die LIKE, SOUNDEx, BETWEEN, > (größer als) und != (ungleich) enthalten.

Beispiel

Folgendes Beispiel beantwortet die Frage: Für welche Produkte hat jemand mehr als den Lagerbestand bestellt?

```
SELECT DISTINCT Products.Name  
FROM Products JOIN SalesOrderItems  
ON Products.ID = SalesOrderItems.ProductID  
AND SalesOrderItems.Quantity > Products.Quantity;
```

Siehe auch

- „Suchbedingungen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

WHERE-Klauseln in Join-Bedingungen

Außer bei der Verwendung von Outer-Joins können Sie Join-Bedingungen in der WHERE-Klausel anstatt in der ON-Klausel angeben. Sie müssen jedoch beachten, dass semantische Unterschiede zwischen den beiden bestehen können, wenn die Abfrage Outer-Joins enthält.

Die ON-Klausel ist Teil der FROM-Klausel und wird daher vor der WHERE-Klausel verarbeitet. Dies führt nur bei Outer-Joins zu einem unterschiedlichen Ergebnis, bei denen die WHERE-Klausel den Join in einen Inner-Join umwandeln kann.

Wenn Sie entscheiden, Join-Bedingungen in eine ON- oder eine WHERE-Klausel einzubeziehen, sollten Sie folgende Regeln beachten:

- Wenn Sie einen Outer-Join verwenden und eine Join-Bedingung in die WHERE-Klausel einbeziehen, wird der Outer-Join möglicherweise in einen Inner-Join umgewandelt.
- Bedingungen in einer ON-Klausel können nur Tabellen referenzieren, die in den Tabellenausdrücken enthalten sind, die von dem zugehörigen JOIN verknüpft werden. Bedingungen in einer WHERE-Klausel können beliebige Tabellen referenzieren, selbst wenn sie nicht Teil des Joins sind.
- Es ist nicht möglich, eine ON-Klausel mit den Schlüsselwörtern CROSS JOIN zu verwenden. Sie können jedoch immer eine WHERE-Klausel verwenden.
- Wenn Join-Bedingungen in einer ON-Klausel enthalten sind, ist der Schlüssel-Join nicht die Standardeinstellung. Der Schlüssel-Join kann jedoch die Standardeinstellung sein, wenn Join-Bedingungen in einer WHERE-Klausel untergebracht werden.

In den Beispielen in dieser Dokumentation werden Join-Bedingungen in einer ON-Klausel verwendet. In Beispielen mit Outer-Joins ist dies erforderlich. In anderen Fällen wird so vorgegangen, um darauf hinzuweisen, dass es sich um Join-Bedingungen handelt und nicht um allgemeine Suchbedingungen.

Siehe auch

- „Outer-Joins und Join-Bedingungen“ auf Seite 506
- „Schlüssel-Joins“ auf Seite 523

Cross-Joins

Ein Cross-Join von zwei Tabellen produziert alle möglichen Zeilenkombinationen aus den beiden Tabellen. Ein Cross-Join wird auch als Kreuzprodukt oder kartesisches Produkt bezeichnet.

Jede Zeile der ersten Tabelle erscheint einmal mit jeder Zeile der zweiten Tabelle. Daher ist die Anzahl der Zeilen in der Ergebnismenge das Produkt der Anzahl der Zeilen in der ersten Tabelle und der Anzahl der Zeilen in der zweiten Tabelle, abzüglich der Zeilen, die aufgrund von Einschränkungen einer WHERE-Klausel weggelassen werden.

Es ist nicht möglich, eine ON-Klausel mit Cross-Joins zu verwenden. Sie können jedoch Einschränkungen in einer WHERE-Klausel festlegen.

Innere und äußere Modifizierer sind für Cross-Joins nicht anwendbar

Außer bei zusätzlichen Einschränkungen in der WHERE-Klausel erscheinen immer alle Zeilen beider Tabellen im Ergebnis eines Cross-Joins. Die Schlüsselwörter INNER, LEFT OUTER und RIGHT OUTER sind daher auf Cross-Joins nicht anwendbar.

Folgende Anweisung verknüpft z.B. zwei Tabellen:

```
SELECT *
FROM A CROSS JOIN B;
```

Die Ergebnismenge dieser Abfrage umfasst alle Spalten in A und alle Spalten in B. In der Ergebnismenge befindet sich jeweils eine Zeile für jede Kombination einer Zeile in A und einer Zeile in B. Wenn A n Zeilen und B m Zeilen enthält, gibt die Abfrage $n \times m$ Zeilen zurück.

Kommas

Ein Komma hat die Funktion eines Join-Operators, ist jedoch keiner. Ein Komma erstellt ein Kreuzprodukt, genau wie das Schlüsselwort CROSS JOIN. Join-Schlüsselwörter erstellen jedoch Tabellenausdrücke, und Kommas erstellen Listen von Tabellenausdrücken.

Im folgenden einfachen Inner-Join zweier Tabellen sind ein Komma und die Schlüsselwörter CROSS JOIN gleichwertig:

```
SELECT *
FROM A, B, C
WHERE A.x = B.y;

SELECT *
FROM A CROSS JOIN B CROSS JOIN C
WHERE A.x = B.y;
```

Im Allgemeinen können Sie ein Komma anstelle der Schlüsselwörter CROSS JOIN verwenden. Die Kommasyntax ist gleichwertig zur Cross-Join-Syntax, außer bei generierten Join-Bedingungen in Tabellenausdrücken, die Kommas verwenden.

In der Syntax von Stern-Joins werden Kommas auf besondere Weise eingesetzt.

Siehe auch

- „Schlüssel-Joins von Tabellenausdrücken“ auf Seite 528
- „Doppelte Korrelationsnamen in Joins (Stern-Joins)“ auf Seite 513

Inner- und Outer-Joins

Mit den Schlüsselwörtern INNER, LEFT OUTER, RIGHT OUTER und FULL OUTER können Sie Schlüssel-Joins, natürliche Joins und Joins mit einer ON-Klausel ändern. Die Standardeinstellung ist INNER. Diese Modifizierer gelten nicht für Cross-Joins.

Inner-Joins

Standardmäßig sind Joins **Inner-Joins**. Zeilen werden in die Ergebnismenge einbezogen, wenn sie die Join-Bedingung erfüllen.

Beispiel

Beispiel: Jede Zeile der Ergebnismenge folgender Abfrage enthält die Informationen einer Customer-Zeile und einer SalesOrder-Zeile und erfüllt damit die Schlüssel-Join-Bedingung. Wenn ein bestimmter Kunde keine Aufträge erteilt hat, ist die Bedingung nicht erfüllt, und die Ergebnismenge enthält die Zeile für den betreffenden Kunden nicht.

```
SELECT GivenName, Surname, OrderDate
FROM Customers KEY INNER JOIN SalesOrders
ORDER BY OrderDate;
```

GivenName	Surname	OrderDate
Hardy	Mums	2000-01-02
Aram	Najarian	2000-01-03
Tommie	Wooten	2000-01-03
Alfredo	Margolis	2000-01-06
...

Da Inner-Joins und Schlüssel-Joins die Standardwerte sind, erhalten Sie dieselben Ergebnisse wie oben, wenn Sie die FROM-Klausel folgendermaßen verwenden:

```
SELECT GivenName, Surname, OrderDate
FROM Customers JOIN SalesOrders
ORDER BY OrderDate;
```

Outer-Joins

Normalerweise erstellen Sie Joins, die Zeilen nur zurückgeben, wenn sie Join-Bedingungen erfüllen. Diese werden als Inner-Joins bezeichnet und bilden die Standard-Joins bei Abfragen. Manchmal möchten Sie aber vielleicht alle Zeilen in einer Tabelle behalten. Dies können Sie mithilfe eines **Outer-Joins** erreichen.

In einem Left- oder Right-**Outer-Join** zweier Tabellen werden alle Zeilen einer der Tabellen beibehalten, und für Zeilen der anderen Tabelle, die die Join-Bedingung nicht erfüllen, wird NULL zurückgegeben. In einem **Left-Outer-Join** wird jede Zeile in der linken Tabelle beibehalten und bei einem **Right-Outer-Join** jede Zeile in der rechten Tabelle. In einem **Full-Outer-Join** werden alle Zeilen aus beiden Tabellen beibehalten und beide Tabellen liefern Nullwerte.

Die Tabellenausdrücke auf einer Seite eines Left- oder Right-Outer-Joins werden als **bewahrt** und **Nullwert-liefernd** bezeichnet. In einem Left-Outer-Join wird der Ausdruck der linken Tabelle beibehalten und der Ausdruck der rechten Tabelle liefert einen Nullwert. In einem Full-Outer-Join werden die Ausdrücke sowohl der linken als auch der rechten Tabelle beibehalten und beide liefern Nullwerte.

Beispiel

Die folgende Anweisung umfasst alle Kunden. Wenn ein bestimmter Kunde keine Bestellung aufgegeben hat, enthält jede Spalte im Ergebnis, die sich auf die Bestelldaten bezieht, NULL.

```
SELECT Surname, OrderDate, City
FROM Customers LEFT OUTER JOIN SalesOrders
    ON Customers.ID = SalesOrders.CustomerID
WHERE Customers.State = 'NY'
ORDER BY OrderDate;
```

Surname	OrderDate	City
Thompson	(NULL)	Bancroft
Reiser	2000-01-22	Rockwood
Clarke	2000-01-27	Rockwood
Mentary	2000-01-30	Rockland
...

Sie können den Outer-Join in dieser Anweisung wie folgt interpretieren. Es handelt sich hierbei um eine konzeptuelle Erklärung, die nicht darstellt, wie SQL Anywhere die Ergebnismenge tatsächlich ermittelt.

- Für jede von einem Kunden aufgegebenen Bestellung wird eine Zeile zurückgegeben. Wenn der Kunde zwei oder mehr Bestellungen aufgegeben hat, werden mehrere Zeilen zurückgegeben, da für jede Bestellung eine Zeile zurückgegeben wird. Dies ist dasselbe Ergebnis wie bei einem Inner-Join. Die ON-Bedingung wird verwendet, um Kunden- und Bestellszeilen in Übereinstimmung zu bringen. Die WHERE-Klausel wird für diesen Schritt nicht verwendet.
- Für jeden Kunden, der keine Bestellungen aufgegeben hat, wird eine Zeile einbezogen. Damit wird sichergestellt, dass jede Zeile aus der Tabelle "Customers" enthalten ist. Für alle diese Zeilen werden

die Spalten aus "SalesOrders" mit NULL gefüllt. Diese Zeilen werden hinzugefügt, weil das Schlüsselwort OUTER verwendet wird, sie wären in einem Inner-Join nicht enthalten gewesen. Weder die ON-Bedingung noch die WHERE-Klausel wird für diesen Schritt verwendet.

- Mithilfe der WHERE-Klausel werden alle Zeilen von Kunden ausgeschlossen, die nicht in New York leben.

Siehe auch

- „[Transact-SQL-Outer-Joins \(*= oder =*\)](#)“ auf Seite 509
- „[Schlüssel-Joins](#)“ auf Seite 523

Outer-Joins und Join-Bedingungen

Ein häufiger Fehler bei der Verwendung von Outer-Joins ist die Positionierung der Join-Bedingung. Wenn Sie Einschränkungen für die Nullwert-liefernde Tabelle in einer WHERE-Klausel festlegen, ist der Join in den meisten Fällen gleichwertig einem Inner-Join.

Das liegt daran, dass die meisten Suchbedingungen nicht als TRUE ausgewertet werden können, wenn eine ihrer Eingaben NULL ist. Die Einschränkungen der WHERE-Klausel für die Nullwert-liefernde Tabelle vergleicht Werte mit NULL, was zum Ausschluss der betreffenden Zeile aus der Ergebnismenge führt. Die Zeilen in der beibehaltenen Tabelle werden nicht beibehalten, wodurch der Join ein Inner-Join wird.

Eine Ausnahme bilden Vergleiche, die als TRUE ausgewertet werden, wenn eine der Eingaben NULL ist. Dazu zählen IS NULL, IS UNKNOWN, IS FALSE, IS NOT TRUE und Ausdrücke, die ISNULL oder COALESCE enthalten.

Beispiel

Die nachstehende Anweisung berechnet z.B. einen Links-Outer-Join:

```
SELECT *
FROM Customers KEY LEFT OUTER JOIN SalesOrders
ON SalesOrders.OrderDate < '2000-01-03';
```

Folgende Anweisung erstellt dagegen einen Inner-Join:

```
SELECT Surname, OrderDate
FROM Customers KEY LEFT OUTER JOIN SalesOrders
WHERE SalesOrders.OrderDate < '2000-01-03';
```

Die erste dieser beiden Anweisungen kann man sich wie folgt vorstellen: Zuerst wird in einem Links-Outer-Join die Tabelle "Customers" mit der Tabelle "SalesOrders" verknüpft. Die Ergebnismenge umfasst alle Zeilen in der Tabelle "Customers". Für diejenigen Kunden, die keine Bestellung vor dem 3. Januar 2000 aufgegeben haben, werden die Felder für die Bestellung mit NULL gefüllt.

In der zweiten Anweisung wird in einem Links-Outer-Join "Customers" mit "SalesOrders" verknüpft. Die Ergebnismenge umfasst alle Zeilen in der Tabelle "Customers". Für diejenigen Kunden, die keine Bestellung aufgegeben haben, werden die Felder für die Bestellung mit NULL gefüllt. Anschließend wird die WHERE-Bedingung angewendet, indem nur die Zeilen ausgewählt werden, in denen der Kunde seit dem 3. Januar 2000 einen Auftrag erteilt hat. Bei Kunden, die keine Aufträge erteilt haben, sind diese

Werte NULL. Das Ergebnis eines Wertevergleichs mit NULL ist immer UNKNOWN. Diese Zeilen werden daher eliminiert, und die Anweisung wird zu einem Inner-Join reduziert.

Siehe auch

- „Suchbedingungen“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Komplexe Outer-Joins

Die Reihenfolge von Joins ist wichtig, wenn der Tabellenausdruck Outer-Joins enthält. `A JOIN B LEFT OUTER JOIN C` wird beispielsweise als `(A JOIN B) LEFT OUTER JOIN C` interpretiert. Der Tabellenausdruck `(A JOIN B)` wird mit `C` verknüpft. Der Tabellenausdruck `(A JOIN B)` wird beibehalten und Tabelle `C` liefert auch Nullwerte.

In folgender Anweisung werden die Tabellen `A`, `B` und `C` verknüpft:

```
SELECT *
FROM A LEFT OUTER JOIN B RIGHT OUTER JOIN C;
```

Um diese Anweisung zu verstehen, muss berücksichtigt werden, dass SQL Anywhere Anweisungen von links nach rechts auswertet und Klammern hinzufügt. Das ergibt folgende Anweisung:

```
SELECT *
FROM (A LEFT OUTER JOIN B) RIGHT OUTER JOIN C;
```

Anschließend sollten Sie den Rechts-Outer-Join in einen Links-Outer-Join umwandeln, sodass beide Joins vom selben Typ sind. Hierzu drehen Sie einfach die Position der Tabellen im Rechts-Outer-Join um, wodurch Sie folgendes Ergebnis erhalten:

```
SELECT *
FROM C LEFT OUTER JOIN (A LEFT OUTER JOIN B);
```

`A` ist die beibehaltene Tabelle und `B` ist die Nullwert-liefernde Tabelle für den verschachtelten Outer-Join. `C` ist die beibehaltene Tabelle für den ersten Outer-Join.

Sie können diesen Join folgendermaßen interpretieren:

- `A` wird mit `B` verknüpft, wobei alle Zeilen in `A` beibehalten werden.
- Anschließend wird `C` mit den Ergebnissen des Joins von `A` und `B` verknüpft, wobei alle Zeilen in `C` beibehalten werden.

Der Join enthält keine ON-Klausel und ist daher standardmäßig ein Schlüssel-Join.

Außerdem darf die Join-Bedingung für einen Outer-Join nur Tabellen enthalten, die zuvor in der FROM-Klausel referenziert wurden. Diese Einschränkung entspricht dem ANSI/ISO-Standard und wird erzwungen, um Mehrdeutigkeiten zu vermeiden. Folgende Anweisungen sind z.B. syntaktisch falsch, da `C` in der Join-Bedingung referenziert wird, bevor die Tabelle selbst referenziert wird.

```
SELECT *
FROM (A LEFT OUTER JOIN B ON B.x = C.x) JOIN C;
```

```
SELECT *
FROM A LEFT OUTER JOIN B ON A.x = C.x, C;
```

Siehe auch

- „Schlüssel-Joins von Tabellenausdrücken ohne Kommas“ auf Seite 528

Outer-Joins von Ansichten und abgeleiteten Tabellen

Outer-Joins können auch für Ansichten und abgeleitete Tabellen verwendet werden.

Anweisung:

```
SELECT *  
FROM V LEFT OUTER JOIN A ON (V.x = A.x);
```

Diese Anweisung kann wie folgt interpretiert werden:

- Ansicht V wird ermittelt.
- Alle Zeilen aus der ermittelten Ansicht V werden mit A verknüpft, indem alle Zeilen aus V mithilfe der Join-Bedingung $V.x = A.x$ beibehalten werden.

Beispiel

Folgendes Beispiel definiert eine Ansicht V, die die IDs der Mitarbeiterinnen zurückgibt, die mehr als \$60.000 verdienen sowie die zugehörigen Abteilungsnamen.

```
CREATE VIEW V AS  
SELECT Employees.EmployeeID, DepartmentName  
FROM Employees JOIN Departments  
ON Employees.DepartmentID = Departments.DepartmentID  
WHERE Sex = 'F' and Salary > 60000;
```

Anschließend wird diese Ansicht verwendet, um eine Liste der Abteilungen und der Regionen hinzuzufügen, in denen die MitarbeiterInnen arbeiten. Die Ansicht V wird beibehalten und "SalesOrders" liefert NULL.

```
SELECT DISTINCT V.EmployeeID, Region, V.DepartmentName  
FROM V LEFT OUTER JOIN SalesOrders  
ON V.EmployeeID = SalesOrders.SalesRepresentative;
```

EmployeeID	Region	DepartmentName
243	(NULL)	R & D
316	(NULL)	R & D
529	(NULL)	R & D
902	Eastern	Sales
...

Transact-SQL-Outer-Joins (*= oder =*)

Hinweis

Die Unterstützung für die Transact-SQL-Outer-Join-Operatoren *= und =* wird nicht weiterentwickelt und in einer zukünftigen Version entfernt werden.

In Übereinstimmung mit ANSI/ISO SQL-Standards unterstützt SQL Anywhere die Schlüsselwörter LEFT OUTER, RIGHT OUTER und FULL OUTER. Aus Gründen der Kompatibilität mit Adaptive Server Enterprise vor Version 12 unterstützt SQL Anywhere auch die Transact-SQL-Gegenstücke dieser Schlüsselwörter, also *= und =*, falls die tsq_outer_joins-Datenbankoption auf "On" gesetzt ist.

Es gibt jedoch Einschränkungen und potenzielle Probleme mit der Semantik von Transact-SQL. Eine ausführliche Beschreibung von Transact-SQL Outer-Joins finden Sie im Whitepaper *Semantics and Compatibility of Transact-SQL Outer Joins* auf <http://www.sybase.com/detail?id=1017447>.

Im Transact-SQL-Dialekt erstellen Sie Outer-Joins, indem Sie eine kommasetrennte Liste von Tabellen in der FROM-Klausel bereitstellen und die Sonderoperatoren *= oder =* in der WHERE-Klausel verwenden. In Adaptive Server Enterprise vor Version 12 muss die Bedingung in der WHERE-Klausel enthalten sein ("ON" wurde nicht unterstützt).

Vorsicht

Wenn Sie Outer-Joins erstellen, darf die Syntax *= nicht mit der Syntax der ON-Klausel gemischt werden. Diese Einschränkung gilt auch für Ansichten, die in der Abfrage referenziert werden.

Siehe auch

- „tsq_outer_joins-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Beispiel

Der folgende Links-Outer-Join listet alle Kunden auf und findet ihre Bestelldaten (wenn vorhanden):

```
SELECT GivenName, Surname, OrderDate
FROM Customers, SalesOrders
WHERE Customers.ID *= SalesOrders.CustomerID
ORDER BY OrderDate;
```

Diese Anweisung ist gleichwertig zur folgenden Anweisung, in der die ANSI/ISO-Syntax verwendet wird:

```
SELECT GivenName, Surname, OrderDate
FROM Customers LEFT OUTER JOIN SalesOrders
ON Customers.ID = SalesOrders.CustomerID
ORDER BY OrderDate;
```

Einschränkungen bei Transact-SQL-Outer-Joins

Hinweis

Die Unterstützung für die Transact-SQL-Outer-Join-Operatoren *= und =* wird nicht weiterentwickelt und in einer zukünftigen Version entfernt werden.

Für Outer-Joins gelten einige Einschränkungen in Transact-SQL:

- Wenn Sie einen Outer-Join und eine Bedingung für eine Spalte aus der Nullwert-liefernden Tabelle des Outer-Joins eingeben, werden unter Umständen unerwartete Ergebnisse geliefert. Die Bedingung in der Abfrage schließt keine Zeilen aus der Ergebnismenge aus, sondern hat Auswirkungen auf die Werte, die in den Zeilen der Ergebnismenge erscheinen. Für Zeilen, die der Bedingung nicht entsprechen, erscheint NULL in den Spalten der Nullwert-liefernden Tabelle.
- Sie können die Syntax für ANSI/ISO SQL und die Transact-SQL-Syntax nicht in einer einzelnen Abfrage mischen. Wenn eine Ansicht mit einem Dialekt für einen Outer-Join verwendet wird, muss für alle Outer-Join-Abfragen für diese Ansicht derselbe Dialekt verwendet werden.
- Eine Nullwert-liefernde Tabelle kann nicht sowohl in einer Outer-Join-Klausel für Transact-SQL und in einer normalen Join-Klausel oder in zwei Outer-Joins verwendet werden. Folgende WHERE-Klausel ist beispielsweise nicht zulässig, da Tabelle S gegen diese Einschränkung verstößt:

```
WHERE R.x *= S.x  
AND S.y = T.y
```

Wenn Sie Ihre Abfrage nicht neu schreiben können, um die Verwendung einer Tabelle in einem Outer-Join und einem normalen Join zu vermeiden, müssen Sie Ihre Anforderung in zwei getrennte Abfragen aufteilen oder lediglich die ANSI/ISO SQL-Syntax verwenden.

- Sie können keine Unterabfrage verwenden, die eine Join-Bedingung mit einer Nullwert-liefernden Tabelle eines Outer-Joins enthält. Folgende WHERE-Klausel ist beispielsweise nicht zulässig:

```
WHERE R.x *= S.y  
AND EXISTS ( SELECT *  
              FROM T  
              WHERE T.x = S.x )
```

Ansichten und Transact-SQL-Outer-Joins

Wenn Sie eine Ansicht mit einem Outer-Join definieren und dann die Ansicht mit einer Qualifikation für eine Spalte aus der Nullwert-liefernden Tabelle des Outer-Joins abfragen, kann es zu unerwarteten Ergebnissen kommen. Die Abfrage gibt alle Zeilen aus der Nullwert-liefernden Tabelle zurück. Zeilen, die den Qualifikationen nicht entsprechen, zeigen NULL in den entsprechenden Spalten dieser Zeilen.

Die folgenden Regeln bestimmen, welche Typen von Aktualisierungen Sie für Spalten durch Ansichten vornehmen können, die Outer-Joins enthalten:

- INSERT- und DELETE-Anweisungen sind für Outer-Join-Ansichten nicht erlaubt.
- UPDATE-Anweisungen sind für Outer-Joins erlaubt. Falls die Ansicht mit der Option WITH CHECK definiert wurde, schlägt die Aktualisierung fehl, wenn eine der betroffenen Spalten in der WHERE-Klausel in einem Ausdruck erscheint, der Spalten aus mehr als einer Tabelle enthält.

Wie NULL Transact-SQL-Joins beeinflusst

NULL-Werte in verknüpften Tabellen oder Ansichten passen in einem Transact-SQL-Outer-Join nie zusammen. Der Vergleich von NULL mit NULL ergibt FALSE.

Spezielle Joins

Dieser Abschnitt beschreibt, wie Sie verschiedene spezielle Joins erstellen, wie etwa Selbst-Joins, Stern-Joins und Joins, die abgeleitete Tabellen verwenden.

Selbst-Joins

In einem **Selbst-Join** wird eine Tabelle mit sich selbst verknüpft, indem sie mit einem unterschiedlichen Korrelationsnamen referenziert wird.

Beispiel 1

Der folgende Selbst-Join produziert eine Liste von Mitarbeiterpaaren. Jeder Mitarbeitername erscheint in Kombination mit jedem anderen Mitarbeiternamen.

```
SELECT a.GivenName, a.Surname,
       b.GivenName, b.Surname
FROM Employees AS a CROSS JOIN Employees AS b;
```

GivenName	Surname	GivenName	Surname
Fran	Whitney	Fran	Whitney
Fran	Whitney	Matthew	Cobb
Fran	Whitney	Philip	Chin
Fran	Whitney	Julie	Jordan
...

Da die Tabelle "Employees" 75 Zeilen hat, enthält dieser Join $75 \times 75 = 5625$ Zeilen. Allerdings enthält er auch Zeilen, in denen der Mitarbeiter mit sich selbst aufgelistet wird. Beispielsweise enthält er folgende Zeile:

GivenName	Surname	GivenName	Surname
Fran	Whitney	Fran	Whitney

Um Zeilen auszuschließen, die einen Namen doppelt enthalten, fügen Sie in einer Join-Bedingung hinzu, dass die Mitarbeiter-IDs nicht übereinstimmen dürfen.

```
SELECT a.GivenName, a.Surname,
       b.GivenName, b.Surname
FROM Employees AS a CROSS JOIN Employees AS b
WHERE a.EmployeeID != b.EmployeeID;
```

Ohne diese Duplikatzellen enthält der Join $75 \times 74 = 5550$ Zeilen.

Dieser neue Join enthält Zeilen, die jeden Mitarbeiter mit allen anderen Mitarbeitern auflistet, wobei aber jedes Namenspaar in zwei Kombinationen erscheint. So werden im Ergebnis des oben genannten Joins die beiden folgenden Zeilen aufgelistet:

GivenName	Surname	GivenName	Surname
Matthew	Cobb	Fran	Whitney
Fran	Whitney	Matthew	Cobb

Wenn die Reihenfolge der Namen nicht wichtig ist, können Sie eine Liste der $(75 \times 74) / 2 = 2775$ eindeutigen Paare erstellen.

```
SELECT a.GivenName, a.Surname,
       b.GivenName, b.Surname
FROM Employees AS a CROSS JOIN Employees AS b
WHERE a.EmployeeID < b.EmployeeID;
```

Diese Anweisung entfernt doppelte Zeilen, indem nur jene Zeilen ausgewählt werden, in denen die "EmployeeID" des Mitarbeiters a niedriger ist als die des Mitarbeiters b.

Beispiel 2

Folgender Selbst-Join verwendet die Korrelationsnamen "report" und "manager", um die beiden Instanzen der Tabelle "Employees" zu unterscheiden, sodass eine Liste der Mitarbeiter und ihrer Abteilungsleiter erstellt werden kann.

```
SELECT report.GivenName, report.Surname,
       manager.GivenName, manager.Surname
FROM Employees AS report JOIN Employees AS manager
ON (report.ManagerID = manager.EmployeeID)
ORDER BY report.Surname, report.GivenName;
```

Diese Anweisung liefert das Ergebnis, das nachstehend auszugsweise dargestellt wird. Die Mitarbeiternamen erscheinen in den beiden linken Spalten, die Namen der jeweiligen Abteilungsleiter rechts.

GivenName	Surname	GivenName	Surname
Alex	Ahmed	Scott	Evans
Joseph	Barker	Jose	Martinez
Irene	Barletta	Scott	Evans
Jeannette	Bertrand	Jose	Martinez
...

Beispiel 3

Der folgende Selbst-Join erzeugt eine Liste aller Manager, denen zwei Ebenen von Mitarbeitern unterstellt sind, mit der Anzahl der betreffenden Mitarbeiter auf der zweiten Ebene.

```
SELECT higher.managerID, count(*) second_level_reports
FROM employees lower JOIN employees higher
      ON ( lower.managerID = higher.employeeID )
GROUP BY higher.managerID
ORDER BY higher.managerID DESC;
```

Das Ergebnis der oben stehenden Abfrage enthält die folgenden Zeilen:

ManagerID	second_level_reports
1293	30
902	23
501	22

Doppelte Korrelationsnamen in Joins (Stern-Joins)

Der Grund für die Verwendung doppelter Tabellennamen ist die Erstellung eines **Stern-Joins**. In einem Stern-Join wird eine Tabelle oder Ansicht mit mehreren anderen verknüpft.

Um einen Stern-Join zu erstellen, verwenden Sie denselben Tabellen-, Ansichts- oder Korrelationsnamen in der FROM-Klausel mehrfach. Dies ist eine Erweiterung zum ANSI/ISO SQL-Standard. Die Möglichkeit, Duplikatnamen zu verwenden, bietet keine zusätzliche Funktionalität, doch es wird einfacher, bestimmte Abfragen zu erstellen.

Die Duplikatnamen müssen in verschiedenen Joins enthalten sein, um sinnvoll zu sein. Wenn ein Tabellen- oder Ansichtsname in einem Join doppelt vorkommt, wird die zweite Instanz ignoriert. Beispiel: FROM A, A und FROM A CROSS JOIN A werden beide als FROM A interpretiert.

Folgendes Beispiel, in dem A, B und C Tabellen sind, ist in SQL Anywhere gültig. In diesem Beispiel wird dieselbe Instanz von Tabelle A sowohl mit B als auch mit C verknüpft. Ein Komma ist erforderlich, um die Joins in einem Stern-Join zu trennen. Die Verwendung eines Kommas in einem Stern-Join ist spezifisch für die Syntax von Stern-Joins.

```
SELECT *
FROM A LEFT OUTER JOIN B ON A.x = B.x,
      A LEFT OUTER JOIN C ON A.y = C.y;
```

Das nächste Beispiel ist gleichwertig.

```
SELECT *
FROM A LEFT OUTER JOIN B ON A.x = B.x,
      C RIGHT OUTER JOIN A ON A.y = C.y;
```

Beide Beispiele sind gleichwertig zur folgenden ANSI/ISO-Standardsyntax: (Die Klammern sind optional.)

```
SELECT *
FROM (A LEFT OUTER JOIN B ON A.x = B.x)
LEFT OUTER JOIN C ON A.y = C.y;
```

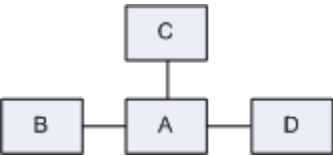
Im nächsten Beispiel wird Tabelle mit drei Tabellen verknüpft: A, B und C.

```
SELECT *
FROM A JOIN B ON A.x = B.x,
     A JOIN C ON A.y = C.y,
     A JOIN D ON A.w = D.w;
```

Dies ist gleichwertig zur folgenden ANSI/ISO-Standardsyntax: (Die Klammern sind optional.)

```
SELECT *
FROM ( (A JOIN B ON A.x = B.x)
      JOIN C ON A.y = C.y)
      JOIN D ON A.w = D.w;
```

Bei komplexen Joins kann es hilfreich sein, ein Diagramm zu zeichnen. Das obige Beispiel kann mit folgendem Diagramm beschrieben werden, das illustriert, dass die Tabellen B, C und D über Tabelle A verknüpft sind:



Hinweis
Sie können doppelte Tabellennamen nur dann verwenden, wenn die Option "extended_join_syntax" auf "On" gesetzt ist (Standardeinstellung).
-

Beispiel 1

Erstellen Sie eine Liste mit den Namen der Kunden, die Bestellungen bei Rollin Overbey aufgegeben haben. Beachten Sie, dass eine der Tabellen in der FROM-Klausel, die Tabelle "Employees", keine Spalten zu den Ergebnissen beiträgt. Darüber hinaus erscheinen die durch den Join verbundenen Spalten —wie "Customer.ID" oder "Employees.EmployeeID"—nicht in den Ergebnissen. Trotzdem ist dieser Join nur durch die Verwendung der Tabelle "Employees" in der FROM-Klausel möglich.

```
SELECT Customers.GivenName, Customers.Surname,
       SalesOrders.OrderDate
FROM   SalesOrders KEY JOIN Customers,
       SalesOrders KEY JOIN Employees
WHERE  Employees.GivenName = 'Rollin'
       AND Employees.Surname = 'Overbey'
ORDER BY SalesOrders.OrderDate;
```

GivenName	Surname	OrderDate
Tommie	Wooten	2000-01-03
Michael	Agliori	2000-01-08
Salton	Pepper	2000-01-17
Tommie	Wooten	2000-01-23

GivenName	Surname	OrderDate
...

Im Folgenden sehen Sie die gleichwertige Anweisung in der ANSI/ISO-Standardsyntax:

```
SELECT Customers.GivenName, Customers.Surname,
       SalesOrders.OrderDate
FROM SalesOrders JOIN Customers
  ON SalesOrders.CustomerID =
     Customers.ID
JOIN Employees
  ON SalesOrders.SalesRepresentative =
     Employees.EmployeeID
WHERE Employees.GivenName = 'Rollin'
      AND Employees.Surname = 'Overbey'
ORDER BY SalesOrders.OrderDate;
```

Beispiel 2

Dieses Beispiel beantwortet folgende Frage: Wie viele Produkte haben die einzelnen Kunden bestellt, und wer ist der Abteilungsleiter des Verkäufers, der die Bestellung aufgenommen hat?

Um die Frage zu beantworten, listen Sie zunächst die Informationen auf, die Sie abfragen müssen. In diesem Fall handelt es sich um Produkt, Menge, Kundenname und Abteilungsleitername. Danach listen Sie die Tabellen auf, die diese Informationen enthalten. Hierbei handelt es sich um die Tabellen "Products", "SalesOrderItems", "Customers" und "Employees". Wenn Sie sich die Struktur der SQL Anywhere-Beispieldatenbank ansehen, stellen Sie fest, dass diese Tabellen über die Tabelle "SalesOrders" verknüpft sind. Sie können einen Stern-Join für die Tabelle "SalesOrders" erstellen, um die Informationen aus den anderen Tabellen abzufragen.

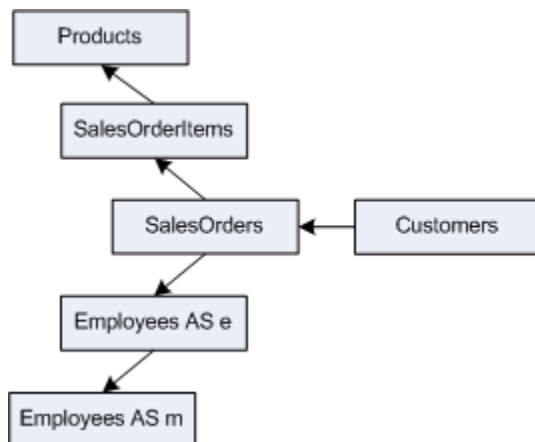
Außerdem müssen Sie einen Selbst-Join erstellen, um den Namen des Abteilungsleiters zu erhalten, da die Tabelle "Employees" die ID-Nummern für Abteilungsleiter und die Namen aller Mitarbeiter enthält, nicht aber eine Spalte, die nur die Namen der Abteilungsleiter auflistet.

Die folgende Anweisung erstellt einen Stern-Join um die Tabelle "SalesOrders". Alle Joins sind Outer-Joins, sodass die Ergebnismenge alle Kunden umfasst. Einige Kunden haben keine Bestellungen aufgegeben, daher sind die anderen Werte für diese Kunden NULL. Die Ergebnismenge enthält Spalten für Kunden, Produkte, Bestellmengen und die Namen der Abteilungsleiter der Verkäufer.

```
SELECT Customers.GivenName, Products.Name,
       SUM(SalesOrderItems.Quantity), m.GivenName
FROM SalesOrders
  KEY RIGHT OUTER JOIN Customers,
SalesOrders
  KEY LEFT OUTER JOIN SalesOrderItems
  KEY LEFT OUTER JOIN Products,
SalesOrders
  KEY LEFT OUTER JOIN Employees AS e
  LEFT OUTER JOIN Employees AS m
  ON (e.ManagerID = m.EmployeeID)
WHERE Customers.State = 'CA'
GROUP BY Customers.GivenName, Products.Name, m.GivenName
ORDER BY SUM(SalesOrderItems.Quantity) DESC,
       Customers.GivenName;
```

GivenName	Name	SUM(SalesOrderItems.Quantity)	GivenName
Sheng	Baseball Cap	240	Moir
Laura	Tee Shirt	192	Moir
Moe	Tee Shirt	192	Moir
Leilani	Sweatshirt	132	Moir
...

Es folgt ein Diagramm der Tabellen in diesem Stern-Join. Die Pfeile zeigen die Richtung (links oder rechts) der Outer-Joins an. Wie Sie sehen, wird die vollständige Liste der Kunden in allen Joins beibehalten.



Folgende ANSI/ISO-Standardsyntax ist gleichwertig zum Stern-Join in Beispiel 2.

```

SELECT Customers.GivenName, Products.Name,
       SUM(SalesOrderItems.Quantity), m.GivenName
FROM SalesOrders LEFT OUTER JOIN SalesOrderItems
  ON SalesOrders.ID = SalesOrderItems.ID
  LEFT OUTER JOIN Products
    ON SalesOrderItems.ProductID = Products.ID
  LEFT OUTER JOIN Employees as e
    ON SalesOrders.SalesRepresentative = e.EmployeeID
  LEFT OUTER JOIN Employees as m
    ON e.ManagerID = m.EmployeeID
  RIGHT OUTER JOIN Customers
    ON SalesOrders.CustomerID = Customers.ID
WHERE Customers.State = 'CA'
GROUP BY Customers.GivenName, Products.Name, m.GivenName
ORDER BY SUM(SalesOrderItems.Quantity) DESC,
       Customers.GivenName;
  
```

Siehe auch

- „Schema der Beispieldatenbank“ auf Seite 493
- „extended_join_syntax-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Selbst-Joins“ auf Seite 511

Joins, die abgeleitete Tabellen verwenden

Mit abgeleiteten Tabellen können Sie Abfragen innerhalb einer FROM-Klausel verschachteln. Mit abgeleiteten Tabellen können Sie Gruppen gruppieren oder einen Join für eine Gruppe aufbauen, ohne eine separate Ansicht oder Tabelle erstellen und sie verknüpfen zu müssen.

Im folgenden Beispiel erstellt die innere SELECT-Anweisung (in Klammern gesetzt) eine abgeleitete Tabelle, die nach "customer ID"-Werten gruppiert ist. Die äußere SELECT-Anweisung weist dieser Tabelle den Korrelationsnamen "sales_order_counts" zu und verbindet sie über eine Join-Bedingung mit der Tabelle "Customers".

```
SELECT Surname, GivenName, number_of_orders
FROM Customers JOIN
  ( SELECT CustomerID, COUNT(*)
    FROM SalesOrders
    GROUP BY CustomerID )
  AS sales_order_counts ( CustomerID, number_of_orders )
ON ( Customers.ID = sales_order_counts.CustomerID )
WHERE number_of_orders > 3;
```

Das Ergebnis ist eine Tabelle der Namen jener Kunden, die mehr als drei Bestellungen aufgegeben haben sowie der Anzahl der aufgegebenen Bestellungen.

Siehe auch

- „Schlüssel-Joins von Ansichten und abgeleiteten Tabellen“ auf Seite 533
- „Natürliche Joins von Ansichten und abgeleiteten Tabellen“ auf Seite 523
- „Outer-Joins von Ansichten und abgeleiteten Tabellen“ auf Seite 508

Joins, die sich aus APPLY-Ausdrücken ergeben

Ein APPLY-Ausdruck bietet eine einfache Möglichkeit, Joins anzugeben, bei denen die rechte Seite von der linken Seite abhängig ist. Verwenden Sie zum Beispiel einen APPLY-Ausdruck, um eine Prozedur oder eine abgeleitete Tabelle für jede Zeile in einem Tabellenausdruck auszuwerten. APPLY-Ausdrücke werden in die FROM-Klausel einer SELECT-Anweisung platziert und erlauben keine Verwendung einer ON-Klausel.

APPLY verbindet Zeilen von mehreren Quellen, ähnlich wie JOIN, nur dass Sie bei APPLY keine ON-Bedingung angeben können. Der Hauptunterschied zwischen APPLY und JOIN besteht darin, dass sich die rechte Seite von APPLY abhängig von der aktuellen Zeile auf der linken Seite ändern kann. Für beide Zeilen auf der linken Seite wird die rechte Seite neu berechnet, und die resultierenden Zeilen werden mit der Zeile links verknüpft. Falls eine Zeile auf der linken Seite mehr als eine Zeile rechts zurückgibt, kommt die linke Seite in den Ergebnissen so oft vor, wie es von rechts zurückgegebene Zeilen gibt.

Es gibt zwei APPLY-Typen, die Sie angeben können: CROSS APPLY und OUTER APPLY. CROSS APPLY gibt nur Zeilen auf der linken Seite zurück, die Ergebnisse auf der rechten Seite produzieren. OUTER APPLY gibt alle Zeilen zurück, die CROSS APPLY zurückgibt, sowie alle Zeilen auf der linken Seite, für die die rechte Seite keine Zeilen zurückgibt (indem NULL-Werte für die rechte Seite geliefert werden).

Die Syntax eines APPLY-Ausdrucks lautet folgendermaßen:

```
table-expression { CROSS | OUTER } APPLY table-expression
```

Beispiel

Das folgende Beispiel erstellt eine Prozedur, EmployeesWithHighSalary, die als Eingabe eine Abteilungs-ID nimmt und die Namen aller Mitarbeiter in dieser Abteilung zurückgibt, deren Gehalt größer als US \$80.000 ist.

```
CREATE PROCEDURE EmployeesWithHighSalary( IN dept INTEGER )
RESULT ( Name LONG VARCHAR )
BEGIN
  SELECT E.GivenName || ' ' || E.Surname
  FROM Employees E
  WHERE E.DepartmentID = dept AND E.Salary > 80000;
END;
```

Die folgende Abfrage verwendet OUTER APPLY, um die Departments-Tabelle mit den Ergebnissen der EmployeesWithHighSalary-Prozedur zu verknüpfen und die Namen aller Mitarbeiter in den einzelnen Abteilungen zurückzugeben, deren Gehalt größer als US\$80.000 ist. Die Abfrage gibt Zeilen mit NULL auf der rechten Seite zurück, was angibt, dass es in den entsprechenden Abteilungen keine Mitarbeiter gibt, deren Gehalt größer als US\$80.000 ist.

```
SELECT D.DepartmentName, HS.Name
FROM Departments D
OUTER APPLY EmployeesWithHighSalary( D.DepartmentID ) AS HS;
```

DepartmentName	Name
R & D	Kim Lull
R & D	David Scott
R & D	John Sheffield
Sales	Moiria Kelly
Finance	Mary Anne Shea
Marketing	NULL
Shipping	NULL

Die nächste Abfrage verwendet CROSS APPLY, um die Departments-Tabelle mit den Ergebnissen der EmployeesWithHighSalary-Prozedur zu verknüpfen. Zeilen mit NULL auf der rechten Seite werden nicht aufgenommen.


```
SELECT D.DepartmentName, HS.Name
FROM Departments D
CROSS APPLY EmployeesWithHighSalary( D.DepartmentID ) AS HS;
```

DepartmentName	Name
R & D	Kim Lull
R & D	David Scott
R & D	John Sheffield
Sales	Moiria Kelly
Finance	Mary Anne Shea

Die nächste Abfrage gibt dieselben Ergebnisse wie die vorherige Abfrage zurück, verwendet aber eine abgeleitete Tabelle als rechte Seite von CROSS APPLY.

```
SELECT D.DepartmentName, HS.Name
FROM Departments D
CROSS APPLY (
    SELECT E.GivenName || ' ' || E.Surname
    FROM Employees E
    WHERE E.DepartmentID = D.DepartmentID AND E.Salary > 80000
) HS( Name );
```

Siehe auch

- „FROM-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Schlüssel-Joins“ auf Seite 523
- „Cross-Joins“ auf Seite 503
- „Inner- und Outer-Joins“ auf Seite 504

Natürliche Joins

Wenn Sie einen natürlichen Join festlegen, generiert SQL Anywhere eine Join-Bedingung, basierend auf Spalten mit demselben Namen. Damit dies in einem natürlichen Join von Basistabellen funktioniert, muss mindestens ein Spaltenpaar mit demselben Namen vorhanden sein, wobei aus jeder Tabelle jeweils eine Spalte stammt. Wenn kein gemeinsamer Spaltenname vorhanden ist, wird ein Fehler ausgegeben.

Tabelle A und Tabelle B haben einen gemeinsamen Spaltennamen, nämlich x:

```
SELECT *
FROM A NATURAL JOIN B;
```

Diese Anweisung entspricht folgender Anweisung:

```
SELECT *
FROM A JOIN B
ON A.x = B.x;
```

Wenn Tabelle A und Tabelle B zwei gemeinsame Spaltennamen enthalten, die a und b heißen, ist A NATURAL JOIN B gleichwertig zur folgenden Anweisung:

```
A JOIN B
ON A.a = B.a
AND A.b = B.b;
```

Beispiel 1

Sie können z.B. die Tabellen "Employees" und "Departments" mit einem natürlichen Join verbinden, da sie einen Spaltennamen gemeinsam haben, nämlich die Spalte "DepartmentID".

```
SELECT GivenName, Surname, DepartmentName
FROM Employees NATURAL JOIN Departments
ORDER BY DepartmentName, Surname, GivenName;
```

GivenName	Surname	DepartmentName
Janet	Bigelow	Finance
Kristen	Coe	Finance
James	Coleman	Finance
Jo Ann	Davidson	Finance
...

Folgende Anweisung ist gleichwertig. Sie gibt explizit die Join-Bedingung an, die im vorangehenden Beispiel generiert wurde.

```
SELECT GivenName, Surname, DepartmentName
FROM Employees JOIN Departments
ON (Employees.DepartmentID = Departments.DepartmentID)
ORDER BY DepartmentName, Surname, GivenName;
```

Beispiel 2

Führen Sie in Interactive SQL folgende Abfrage aus:

```
SELECT Surname, DepartmentName
FROM Employees NATURAL JOIN Departments;
```

Surname	DepartmentName
Whitney	R & D
Cobb	R & D
Breault	R & D
Shishov	R & D
Driscoll	R & D
...	...

SQL Anywhere prüft die beiden Tabellen und stellt fest, dass der einzige gemeinsame Spaltenname "DepartmentID" lautet. Die nachfolgende ON-Klausel wird intern generiert und zum Durchführen des Joins benutzt:

```
FROM Employees JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID
```

NATURAL JOIN ist lediglich eine Kurzform für die Eingabe der ON-Klausel. Die beiden Abfragen sind identisch.

Fehler bei der Verwendung von natürlichen Joins

Der NATURAL JOIN-Operator kann Probleme verursachen, indem er Spalten vergleicht, die Sie gar nicht vergleichen wollen. Folgende Abfrage generiert zum Beispiel nicht die gewünschten Ergebnisse:

```
SELECT *
FROM SalesOrders NATURAL JOIN Customers;
```

Das Ergebnis dieser Abfrage enthält keine Zeilen. SQL Anywhere erzeugt intern die folgende ON-Klausel:

```
FROM SalesOrders JOIN Customers
ON SalesOrders.ID = Customers.ID
```

Die Spalte "ID" in der Tabelle "SalesOrders" ist eine Kennung für die Bestellung (order). Die Spalte "ID" in der Tabelle "Customers" ist eine Kennung für den Kunden (customer). Für keine der Kennungen wurde eine Übereinstimmung gefunden. Selbst wenn eine Übereinstimmung gefunden würde, wäre sie bedeutungslos.

Natürliche Joins mit ON-Klausel

Wenn Sie einen NATURAL JOIN und eine Join-Bedingung in einer ON-Klausel verwenden, ist das Ergebnis die Verbindung der beiden Join-Bedingungen.

Die folgenden beiden Abfragen sind beispielsweise gleichwertig. In der ersten Abfrage generiert SQL Anywhere die Join-Bedingung `Employees.DepartmentID = Departments.DepartmentID`. Die Abfrage enthält auch eine explizite Join-Bedingung.

```
SELECT GivenName, Surname, DepartmentName
FROM Employees NATURAL JOIN Departments
ON Employees.ManagerID = Departments.DepartmentHeadID;
```

Die nächste Abfrage ist gleichwertig. Die natürliche Join-Bedingung, die in der vorherigen Abfrage erstellt wurde, wird in der ON-Klausel angegeben.

```
SELECT GivenName, Surname, DepartmentName
FROM Employees JOIN Departments
ON Employees.ManagerID = Departments.DepartmentHeadID
AND Employees.DepartmentID = Departments.DepartmentID;
```

Natürliche Joins von Tabellenausdrücken

Wenn auf mindestens einer Seite eines natürlichen Joins ein Ausdruck enthalten ist, der sich auf mehrere Tabellen bezieht, generiert SQL Anywhere eine Join-Bedingung, indem die Spaltengruppe auf den beiden Seiten des Join-Operators verglichen und nach Spalten gesucht wird, die denselben Namen haben.

Beispiel:

```
SELECT *  
FROM ( A JOIN B ) NATURAL JOIN ( C JOIN D );
```

Diese Anweisung enthält zwei Tabellenausdrücke. Die Spaltennamen im Tabellenausdruck `A JOIN B` werden mit den Spaltennamen im Tabellenausdruck `C JOIN D` verglichen, und für jedes unzweideutige Paar übereinstimmender Spaltennamen wird eine Join-Bedingung erstellt. Ein **unzweideutiges Paar übereinstimmender Spalten** bedeutet, dass der Spaltenname in beiden Tabellenausdrücken, nicht aber zwei Mal in demselben Tabellenausdruck vorkommt.

Wenn ein Paar zweideutiger Spaltennamen vorhanden ist, wird ein Fehler ausgegeben. Ein Spaltenname kann in einem Tabellenausdruck jedoch zwei Mal enthalten sein, sofern er nicht mit dem Namen einer Spalte aus dem anderen Tabellenausdruck übereinstimmt.

Natürliche Joins von Listen

Wenn auf mindestens einer Seite eines natürlichen Joins eine Liste von Tabellenausdrücken vorhanden ist, wird für jeden Tabellenausdruck in der Liste eine eigene Join-Bedingung generiert.

Sie haben z.B. folgende Tabellen:

- Tabelle A enthält die Spalten a, b und c.
- Tabelle B enthält die Spalten a und d.
- Tabelle C enthält die Spalten d und c.

In diesem Fall bewirkt der Join `(A , B) NATURAL JOIN C`, dass SQL Anywhere zwei Join-Bedingungen generiert:

```
ON A.c = C.c  
AND B.d = C.d
```

Wenn kein gemeinsamer Spaltenname für A-C oder B-C vorhanden ist, wird ein Fehler ausgegeben.

Wenn Tabelle C die Spalten a, d und c enthält, ist der Join `(A , B) NATURAL JOIN C` ungültig. Dies liegt daran, dass Spalte a in allen drei Tabellen vorkommt und der Join daher zweideutig ist.

Beispiel

Folgendes Beispiel beantwortet die Frage: Für jeden Verkauf werden Informationen darüber bereitgestellt, was von wem verkauft wurde.

```
SELECT *  
FROM ( Employees KEY JOIN SalesOrders )  
    NATURAL JOIN ( SalesOrderItems KEY JOIN Products );
```

Das ist äquivalent zu Folgendem:

```
SELECT *
FROM ( Employees KEY JOIN SalesOrders )
     JOIN ( SalesOrderItems KEY JOIN Products )
     ON SalesOrders.ID = SalesOrderItems.ID;
```

Natürliche Joins von Ansichten und abgeleiteten Tabellen

Eine Erweiterung des ANSI/ISO SQL-Standards ist, dass Sie auf beiden Seiten eines natürlichen Joins Ansichten und abgeleitete Tabellen verwenden können. Sie haben z.B. folgende Anweisung:

```
SELECT *
FROM View1 NATURAL JOIN View2;
```

Die Spalten in Ansicht 1 werden mit den Spalten in Ansicht 2 verglichen. Wenn z.B. in beiden Ansichten eine Spalte namens "EmployeeID" gefunden wird und keine anderen Spalten mit identischen Namen vorhanden sind, wird die Join-Bedingung (`View1.EmployeeID = View2.EmployeeID`) generiert.

Beispiel

Folgendes Beispiel illustriert, dass eine Ansicht in einem natürlichen Join nicht nur Spalten, sondern auch Ausdrücke enthalten kann, die auf dieselbe Weise behandelt werden. Erstellen Sie zunächst die Ansicht V mit der Spalte x:

```
CREATE VIEW V(x) AS
SELECT R.y + 1
FROM R;
```

Danach erstellen Sie einen natürlichen Join der Ansicht zu einer abgeleiteten Tabelle. Die abgeleitete Tabelle hat den Korrelationsnamen T mit einer Spalte namens x.

```
SELECT *
FROM V NATURAL JOIN (SELECT P.y FROM P) as T(x);
```

Dieser Join entspricht Folgendem:

```
SELECT *
FROM V JOIN (SELECT P.y FROM P) as T(x) ON (V.x = T.x);
```

Schlüssel-Joins

Viele allgemeine Joins verbinden zwei Tabellen bezüglich des Fremdschlüssels. Der häufigste Join beschränkt Fremdschlüsselwerte auf eine exakte Übereinstimmung mit den Primärschlüsselwerten. Der Operator `KEY JOIN` verknüpft zwei Tabellen basierend auf einer Fremdschlüsselbeziehung. Das heißt, SQL Anywhere generiert eine `ON`-Klausel, die die Primärschlüsselspalte einer Tabelle mit der Fremdspalte der anderen Tabelle gleichsetzt. Um einen Schlüssel-Join zu verwenden, muss eine Fremdschlüsselbeziehung zwischen den Tabellen bestehen. Anderenfalls wird ein Fehler ausgegeben.

Ein Schlüssel-Join kann als Kurzform für die `ON`-Klausel betrachtet werden. Die beiden Abfragen sind identisch. Sie können jedoch auch die `ON`-Klausel mit einem `KEY JOIN` benutzen. `KEY JOIN` ist die

Standardeinstellung, wenn Sie JOIN angeben, aber weder CROSS, NATURAL oder KEY festlegen noch eine ON-Klausel verwenden. Wenn Sie sich das Diagramm der SQL Anywhere-Beispieldatenbank ansehen, werden Sie feststellen, dass Fremdschlüssel durch Linien zwischen den Tabellen dargestellt werden. Überall dort, wo zwei Tabellen im Diagramm durch eine Linie verbunden sind, können Sie den Operator KEY JOIN verwenden.

Schlüssel-Join als Standardeinstellung

Der Schlüssel-Join ist in SQL Anywhere die Standardeinstellung, wenn alle folgenden Bedingungen zutreffen:

- Das Schlüsselwort JOIN wird verwendet.
- Die Schlüsselwörter CROSS, NATURAL und KEY werden nicht angegeben.
- Es ist keine ON-Klausel vorhanden.

Siehe auch

- [„Praktische Einführung: Verbindung mit der Beispieldatenbank herstellen“ \[SQL Anywhere Server - Datenbankadministration\]](#)

Beispiel

Folgende Abfrage verknüpft beispielsweise die Tabellen "Products" und "SalesOrderItems" basierend auf der Fremdschlüsselbeziehung in der Datenbank.

```
SELECT *  
FROM Products KEY JOIN SalesOrderItems;
```

Die nächste Abfrage ist gleichwertig. Hier fehlt das Schlüsselwort KEY, doch standardmäßig ist ein JOIN ohne eine ON-Klausel ein KEY JOIN, also ein Schlüssel-Join.

```
SELECT *  
FROM Products JOIN SalesOrderItems;
```

Die nächste Abfrage ist ebenfalls gleichwertig, da die in der ON-Klausel angegebene Join-Bedingung dieselbe ist wie die Join-Bedingung, die SQL Anywhere für diese Tabellen basierend auf ihrer Fremdschlüsselbeziehung in der SQL Anywhere-Beispieldatenbank generiert.

```
SELECT *  
FROM Products JOIN SalesOrderItems  
ON SalesOrderItems.ProductID = Products.ID;
```

Schlüssel-Joins mit ON-Klausel

Wenn Sie KEY JOIN und eine Join-Bedingung in einer ON-Klausel verwenden, ist das Ergebnis die Verbindung der beiden Join-Bedingungen. Beispiel:

```
SELECT *  
FROM A KEY JOIN B  
ON A.x = B.y;
```

Wenn die vom Schlüssel-Join von A und B generierte Join-Bedingung $A.w = B.z$ ist, ist folgende Abfrage gleichwertig:

```
SELECT *  
FROM A JOIN B  
ON A.x = B.y AND A.w = B.z;
```

Schlüssel-Joins bei mehrfachen Fremdschlüsselbeziehungen

Wenn SQL Anywhere versucht, eine Join-Bedingung basierend auf einer Fremdschlüsselbeziehung zu generieren, findet der Server möglicherweise mehr als eine Beziehung. In diesem Fall bestimmt SQL Anywhere, welche Fremdschlüsselbeziehung verwendet wird, indem der Rollename des Fremdschlüssels mit dem Korrelationsnamen der Primärschlüsseltabelle verglichen wird, die der Fremdschlüssel referenziert.

Die folgenden Abschnitte beschreiben, wie SQL Anywhere Join-Bedingungen für Schlüssel-Joins generiert.

Korrelationsname und Rollename

Ein **Korrelationsname** ist der Name einer Tabelle oder Ansicht, die in der FROM-Klausel der Abfrage verwendet wird — entweder unter ihrem ursprünglichen Namen oder unter einem Aliasnamen, der in der FROM-Klausel festgelegt wird.

Der **Rollename** ist der Name des Fremdschlüssels. Er muss für eine bestimmte Fremdtabelle (untergeordnete Tabelle) eindeutig sein.

Wenn Sie keinen Rollennamen für einen Fremdschlüssel festlegen, wird der Name wie folgt zugewiesen:

- Wenn es keinen Fremdschlüssel mit dem Namen der Primärtabelle gibt, wird der Primärtabellenname als Rollename zugeordnet.
- Wenn der Tabellename bereits von einem anderen Fremdschlüssel verwendet wird, ist der Rollename der Name der Primärtabelle, verbunden mit einer für die Fremdtabelle eindeutigen dreistelligen Zahl, die mit Nullen aufgefüllt wird.

Wenn Sie den Rollennamen eines Fremdschlüssels nicht kennen, können Sie ihn in Sybase Central suchen, indem Sie den Datenbankbehälter im linken Fensterausschnitt öffnen. Wählen Sie die Tabelle im linken Fensterausschnitt und klicken Sie auf die Registerkarte **Integritätsregeln** im rechten Fensterausschnitt. Eine Liste der Fremdschlüssel für diese Tabelle wird im rechten Fensterausschnitt angezeigt.

Join-Bedingungen generieren

SQL Anywhere sucht nach einem Fremdschlüssel, dessen Rollename dem Korrelationsnamen der Primärschlüsseltabelle entspricht:

- Wenn es genau einen Fremdschlüssel gibt, der den selben Namen wie eine Tabelle im Join trägt, verwendet SQL Anywhere ihn dazu, die Join-Bedingung zu generieren.
- Wenn mehrere Fremdschlüssel mit dem Namen einer Tabelle vorhanden sind, ist der Join zweideutig, und es wird ein Fehler ausgegeben.

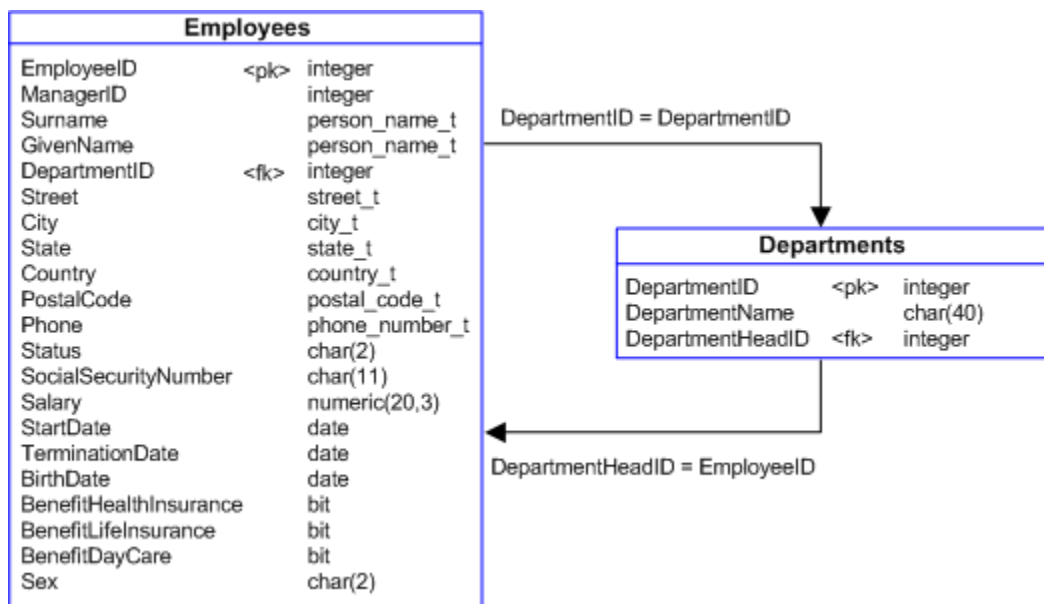
- Wenn kein Fremdschlüssel mit dem Namen einer Tabelle vorhanden ist, sucht SQL Anywhere nach einer Fremdschlüsselbeziehung, selbst wenn die Namen nicht übereinstimmen. Wenn es mehrere Fremdschlüsselbeziehungen gibt, ist der Join zweideutig, und es wird ein Fehler ausgegeben.

Siehe auch

- „Regeln, die die Verarbeitung von Schlüssel-Joins beschreiben“ auf Seite 535
- „Schema der Beispieldatenbank“ auf Seite 493

Beispiel 1

In der SQL Anywhere-Beispieldatenbank werden von den Tabellen "Employees" und "Departments" zwei Fremdschlüsselbeziehungen festgelegt: Der Fremdschlüssel "FK_DepartmentID_DepartmentID" in der Tabelle "Employees" referenziert die Tabelle "Departments", und der Fremdschlüssel "FK_DepartmentHeadID_EmployeeID" in der Tabelle "Departments" referenziert die Tabelle "Employees".



Folgende Abfrage ist zweideutig, da zwei Fremdschlüsselbeziehungen vorhanden sind und keine von ihnen denselben Rollennamen wie die Primärschlüsseltabelle hat. Diese Abfrage bewirkt daher den Syntaxfehler `SQL_E_AMBIGUOUS_JOIN (-147)`.

```
SELECT Employees.Surname, Departments.DepartmentName
FROM Employees KEY JOIN Departments;
```

Beispiel 2

Diese Abfrage ändert die Abfrage in Beispiel 1 durch die Angabe des Korrelationsnamens "FK_DepartmentID_DepartmentID" für die Tabelle "Departments". Der Fremdschlüssel "FK_DepartmentID_DepartmentID" hat nun denselben Namen wie die Tabelle, die er referenziert. Deshalb wird er verwendet, um die Join-Bedingung festzulegen. Das Ergebnis umfasst die Nachnamen aller Mitarbeiter und die Abteilungen, in denen sie arbeiten.


```
SELECT Employees.Surname,
       FK_DepartmentID_DepartmentID.DepartmentName
FROM Employees KEY JOIN Departments
     AS FK_DepartmentID_DepartmentID;
```

Folgende Abfrage ist gleichwertig. Es ist in diesem Beispiel nicht erforderlich, einen Aliasnamen für die Tabelle "Departments" zu erstellen. Dieselbe Join-Bedingung, die oben generiert wurde, wird in der ON-Klausel dieser Abfrage angegeben:

```
SELECT Employees.Surname, Departments.DepartmentName
FROM Employees JOIN Departments
     ON Departments.DepartmentID = Employees.DepartmentID;
```

Beispiel 3

Wenn beabsichtigt ist, alle Mitarbeiter aufzulisten, die Abteilungsleiter sind, muss der Fremdschlüssel "FK_DepartmentHeadID_EmployeeID" verwendet und Beispiel 1 wie folgt umgeschrieben werden. Diese Abfrage erfordert die Verwendung des Fremdschlüssels "FK_DepartmentHeadID_EmployeeID" durch die Angabe des Korrelationsnamens "FK_DepartmentHeadID_EmployeeID" für die Primärschlüsseltabelle "Employees".

```
SELECT FK_DepartmentHeadID_EmployeeID.Surname, Departments.DepartmentName
FROM Employees AS FK_DepartmentHeadID_EmployeeID
     KEY JOIN Departments;
```

Folgende Abfrage ist gleichwertig. Die Join-Bedingung, die oben generiert wurde, wird in der ON-Klausel dieser Abfrage angegeben:

```
SELECT Employees.Surname, Departments.DepartmentName
FROM Employees JOIN Departments
     ON Departments.DepartmentHeadID = Employees.EmployeeID;
```

Beispiel 4

Es ist kein Korrelationsname erforderlich, wenn der Name des Fremdschlüssels mit dem Namen der Primärschlüsseltabelle übereinstimmt. Es ist z.B. möglich, den Fremdschlüssel "Departments" für die Tabelle "Employees" festzulegen:

```
ALTER TABLE Employees
     ADD FOREIGN KEY Departments (DepartmentID)
     REFERENCES Departments (DepartmentID);
```

Diese Fremdschlüsselbeziehung ist nun die Standard-Join-Bedingung, wenn ein KEY JOIN für die beiden Tabellen festgelegt wird. Wenn der Fremdschlüssel "Departments" festgelegt wird, ist folgende Abfrage gleichwertig zu Beispiel 3:

```
SELECT Employees.Surname, Departments.DepartmentName
FROM Employees KEY JOIN Departments;
```

Hinweis

Wenn Sie dieses Beispiel in Interactive SQL ausprobieren, müssen Sie die Änderung der SQL Anywhere-Beispieldatenbank mit folgender Anweisung rückgängig machen:

```
ALTER TABLE Employees DROP FOREIGN KEY Departments;
```

Schlüssel-Joins von Tabellenausdrücken

SQL Anywhere generiert Join-Bedingungen für den Schlüssel-Join von Tabellenausdrücken, indem der Server die Fremdschlüsselbeziehung jedes Tabellenpaars in der Anweisung untersucht.

Folgendes Beispiel verknüpft vier Tabellenpaare:

```
SELECT *  
FROM (A NATURAL JOIN B) KEY JOIN (C NATURAL JOIN D);
```

Die Tabellenpaare sind A-C, A-D, B-C und B-D. SQL Anywhere überprüft die Beziehung zwischen jedem Paar und erstellt dann eine generierte Join-Bedingung für den gesamten Tabellenausdruck. Wie SQL Anywhere dabei vorgeht, hängt davon ab, ob die Tabellenausdrücke Kommas verwenden. Die generierten Join-Bedingungen in den beiden folgenden Beispielen sind daher unterschiedlich. A JOIN B ist ein Tabellenausdruck, der keine Kommas enthält, und (A, B) ist eine Tabellenausdrucksliste.

```
SELECT *  
FROM (A JOIN B) KEY JOIN C;
```

Diese Join-Bedingung unterscheidet sich von der Folgenden:

```
SELECT *  
FROM (A,B) KEY JOIN C;
```

Das Verhalten der beiden Join-Typen wird in den folgenden Abschnitten besprochen:

- „Schlüssel-Joins von Tabellenausdrücken ohne Kommas“
- „Schlüssel-Joins von Tabellenausdruckslisten“
- „Schlüssel-Joins bei mehrfachen Fremdschlüsselbeziehungen“ auf Seite 525

Schlüssel-Joins von Tabellenausdrücken ohne Kommas

Wenn beide Tabellenausdrücke keine Kommas enthalten, untersucht SQL Anywhere die Fremdschlüsselbeziehungen in den Tabellenpaaren in der Anweisung und generiert eine einzige Join-Bedingung.

Folgender Join hat z.B. zwei Tabellenpaare, A-C und B-C.

```
(A NATURAL JOIN B) KEY JOIN C
```

SQL Anywhere generiert eine einzige Join-Bedingung, um C mit (A NATURAL JOIN B) zu verknüpfen, indem der Server die Fremdschlüsselbeziehung in den Tabellenpaaren A-C und B-C untersucht. Der Server erstellt eine Join-Bedingung für die beiden Paare gemäß den Regeln zur Festlegung von Schlüssel-Joins, wenn mehrere Fremdschlüsselbeziehungen vorhanden sind:

- Zunächst wird in A-C und B-C nach einem einzigen Fremdschlüssel gesucht, dessen Rollename dem Korrelationsnamen einer der beiden Primärschlüsseltabellen entspricht, die er referenziert. Wenn genau ein Fremdschlüssel diese Bedingung erfüllt, wird dieser verwendet. Wenn mehrere Fremdschlüssel mit dem Rollennamen der Korrelation einer Tabelle vorhanden sind, ist der Join zweideutig, und es wird ein Fehler ausgegeben.

- Wenn kein Fremdschlüssel mit dem Namen der Korrelation einer Tabelle vorhanden ist, sucht SQL Anywhere nach einer beliebigen Fremdschlüsselbeziehung zwischen den Tabellen. Wenn es eine solche Beziehung gibt, wird sie verwendet. Wenn es mehrere gibt, ist der Join zweideutig, und es wird ein Fehler ausgegeben.
- Wenn keine Fremdschlüsselbeziehung vorhanden ist, wird ein Fehler ausgegeben.

Beispiel

Folgende Abfrage sucht alle Mitarbeiter, die Vertriebspersonen sind, und deren Abteilungen:

```
SELECT Employees.Surname,
       FK_DepartmentID_DepartmentID.DepartmentName
FROM ( Employees KEY JOIN Departments
      AS FK_DepartmentID_DepartmentID )
KEY JOIN SalesOrders;
```

Sie können die Abfrage wie folgt interpretieren:

- SQL Anywhere überprüft den Tabellenausdruck (Employees KEY JOIN Departments as FK_DepartmentID_DepartmentID) und generiert die Join-Bedingung Employees.DepartmentID = FK_DepartmentID_DepartmentID.DepartmentID basierend auf dem Fremdschlüssel "FK_DepartmentID_DepartmentID".
- SQL Anywhere prüft dann die Tabellenpaare "Employees/SalesOrders" und "Departments/SalesOrders". Zwischen den Tabellen "SalesOrders" und "Employees" und zwischen "SalesOrders" und "Departments" darf es nur einen Fremdschlüssel geben. Andernfalls ist der Join zweideutig. Wie sich herausstellt, gibt es genau eine Fremdschlüsselbeziehung zwischen den Tabellen "SalesOrders" und "Employees" (FK_SalesRepresentative_EmployeeID) und keine Fremdschlüsselbeziehung zwischen "SalesOrders" und "Departments". Daher lautet die generierte Join-Bedingung SalesOrders.EmployeeID = Employees.SalesRepresentative.

Die nachstehende Abfrage ist daher der vorherigen Abfrage gleichwertig:

```
SELECT Employees.Surname, Departments.DepartmentName
FROM ( Employees JOIN Departments
      ON ( Employees.DepartmentID = Departments.DepartmentID ) )
JOIN SalesOrders
  ON ( Employees.EmployeeID = SalesOrders.SalesRepresentative );
```

Schlüssel-Joins von Tabellenausdruckslisten

Um eine Join-Bedingung für den Schlüssel-Join von zwei Tabellenausdruckslisten zu erstellen, untersucht SQL Anywhere die Tabellenpaare aus der Anweisung und generiert für jedes Paar eine Join-Bedingung. Die endgültige Join-Bedingung ist schließlich die Verbindung der Join-Bedingungen für jedes Paar. Zwischen jedem Paar muss eine Fremdschlüsselbeziehung bestehen.

Folgendes Beispiel verknüpft die beiden Tabellenpaare A-C und B-C:

```
SELECT *
FROM ( A,B ) KEY JOIN C;
```

SQL Anywhere generiert eine Join-Bedingung, um C mit (A, B) zu verknüpfen, indem für jedes der beiden Paare A-C und B-C eine Join-Bedingung generiert wird. Dabei werden die Regeln für Schlüssel-Joins bei mehreren Fremdschlüsselbeziehungen befolgt:

- SQL Anywhere sucht für jedes Paar nach einem Fremdschlüssel, dessen Rollennamen dem Korrelationsnamen der Primärschlüsseltabelle entspricht. Wenn genau ein Fremdschlüssel diese Bedingung erfüllt, wird dieser verwendet. Wenn es mehrere gibt, ist der Join zweideutig, und es wird ein Fehler ausgegeben.
- Wenn für keines der Paare ein Fremdschlüssel mit dem Korrelationsnamen der Tabelle vorhanden ist, sucht SQL Anywhere nach einer beliebigen Fremdschlüsselbeziehung zwischen den Tabellen. Wenn es eine solche Beziehung gibt, wird sie verwendet. Wenn es mehrere gibt, ist der Join zweideutig, und es wird ein Fehler ausgegeben.
- Wenn für keines der Paare eine Fremdschlüsselbeziehung vorhanden ist, wird ein Fehler ausgegeben.
- Wenn SQL Anywhere für jedes Paar genau eine Join-Bedingung bestimmen kann, werden die Join-Bedingungen mit AND verknüpft.

Beispiel

Folgende Abfrage gibt die Namen aller Vertriebsangestellten zurück, die mindestens eine Bestellung aus einer bestimmten Region aufgenommen haben.

```
SELECT DISTINCT Employees.Surname,  
                FK_DepartmentID_DepartmentID.DepartmentName,  
                SalesOrders.Region  
FROM ( SalesOrders, Departments  
      AS FK_DepartmentID_DepartmentID )  
KEY JOIN Employees;
```

Surname	DepartmentName	Region
Chin	Sales	Eastern
Chin	Sales	Western
Chin	Sales	Central
...

Diese Abfrage arbeitet mit zwei Tabellenpaaren: "SalesOrders" und "Employees" sowie "Departments AS FK_DepartmentID_DepartmentID" und "Employees".

Für das Paar "SalesOrders" und "Employees" gibt es keinen Fremdschlüssel, der dem Rollennamen einer der Tabellen entspricht. Es gibt jedoch einen Fremdschlüssel (FK_SalesRepresentative_EmployeeID), der sich auf diese beiden Tabellen bezieht. Dies ist der einzige Fremdschlüssel, der sich auf die beiden Tabellen bezieht, und er wird daher in der generierten Join-Bedingung (Employees.EmployeeID = SalesOrders.SalesRepresentative) verwendet.

Für das Paar "Departments AS FK_DepartmentID_DepartmentID" und "Employees" gibt es einen Fremdschlüssel, der denselben Rollennamen hat wie die Primärschlüsseltabelle. Er heißt

"FK_DepartmentID_DepartmentID" und damit genauso wie der Korrelationsname der Tabelle "Departments" in der Abfrage. Es gibt keine anderen Fremdschlüssel mit dem Namen der Korrelation der Primärschlüsseltabelle, sodass "FK_DepartmentID_DepartmentID" verwendet wird, um die Join-Bedingung für das Tabellenpaar zu bilden. Die Join-Bedingung, die generiert wird, lautet (Employees.DepartmentID = FK_DepartmentID_DepartmentID.DepartmentID). Es ist ein weiterer Fremdschlüssel vorhanden, der die beiden Tabellen verbindet, doch da dieser einen anderen Namen hat als die beiden Tabellen, ist er nicht relevant.

Die endgültige Join-Bedingung fügt schließlich die für die beiden Tabellenpaare generierten Join-Bedingungen zusammen. Folgende Abfrage ist daher gleichwertig:

```
SELECT DISTINCT Employees.Surname,
    Departments.DepartmentName,
    SalesOrders.Region
FROM ( SalesOrders, Departments )
JOIN Employees
ON Employees.EmployeeID = SalesOrders.SalesRepresentative
AND Employees.DepartmentID = Departments.DepartmentID;
```

Siehe auch

- „Schlüssel-Joins bei mehrfachen Fremdschlüsselbeziehungen“ auf Seite 525

Schlüssel-Joins von Listen und Tabellenausdrücken ohne Kommas

Wenn Tabellenausdruckslisten über einen Schlüssel-Join mit Tabellenausdrücken verknüpft werden, die keine Kommas enthalten, generiert SQL Anywhere für jede Tabelle in der Tabellenausdrucksliste eine Join-Bedingung.

Folgende Anweisung ist z.B. der Schlüssel-Join einer Tabellenausdrucksliste und eines Tabellenausdrucks, der keine Kommas enthält. Dieses Beispiel generiert eine Join-Bedingung für Tabelle A mit dem Tabellenausdruck C NATURAL JOIN D, und für Tabelle B mit dem Tabellenausdruck C NATURAL JOIN D.

```
SELECT *
FROM (A,B) KEY JOIN (C NATURAL JOIN D);
```

(A,B) ist eine Liste von Tabellenausdrücken, und C NATURAL JOIN D ist ein Tabellenausdruck. SQL Anywhere muss daher zwei Join-Bedingungen generieren: Es wird eine Join-Bedingung für die Paare A-C und A-D generiert und eine zweite Join-Bedingung für die Paare B-C und B-D. Dabei werden die Regeln für Schlüssel-Joins bei mehreren Fremdschlüsselbeziehungen befolgt:

- SQL Anywhere sucht für jedes Tabellenpaar nach einem Fremdschlüssel, dessen Rollenname dem Korrelationsnamen einer der Primärschlüsseltabellen entspricht. Wenn genau ein Fremdschlüssel diese Bedingung erfüllt, wird dieser verwendet. Wenn es mehrere gibt, ist der Join zweideutig, und es wird ein Fehler ausgegeben.
- Wenn für keines der Tabellenpaare ein Fremdschlüssel mit dem Korrelationsnamen einer Tabelle vorhanden ist, sucht SQL Anywhere nach einer beliebigen Fremdschlüsselbeziehung zwischen den Tabellen. Wenn es genau eine solche Beziehung gibt, wird sie verwendet. Wenn es mehrere gibt, ist der Join zweideutig, und es wird ein Fehler ausgegeben.

- Wenn für keines der Paare eine Fremdschlüsselbeziehung vorhanden ist, wird ein Fehler ausgegeben.
- Wenn SQL Anywhere für jedes Paar genau eine Join-Bedingung feststellen kann, werden die Join-Bedingungen mit dem Schlüsselwort AND verknüpft.

Beispiel 1

Untersuchen Sie folgende Verknüpfung von fünf Tabellen:

```
((A,B) JOIN (C NATURAL JOIN D) ON A.x = D.y) KEY JOIN E
```

In diesem Fall generiert SQL Anywhere eine Join-Bedingung für den Schlüssel-Join zu E, indem eine Bedingung entweder zwischen (A,B) und E oder zwischen C NATURAL JOIN D und E generiert wird.

Wenn SQL Anywhere eine Join-Bedingung zwischen (A,B) und E generiert, muss der Server zwei Join-Bedingungen erstellen, nämlich eine für A-E und eine zweite für B-E. Dabei muss eine gültige Fremdschlüsselbeziehung innerhalb jedes Tabellenpaares gefunden werden.

Wenn SQL Anywhere eine Join-Bedingung zwischen C NATURAL JOIN D und E erstellt, wird nur eine Join-Bedingung erstellt und daher muss nur eine Fremdschlüsselbeziehung in den Paaren C-E und D-E gefunden werden.

Beispiel 2

Folgendes Beispiel enthält einen Schlüssel-Join eines Tabellenausdrucks und eine Liste von Tabellenausdrücken. Das Beispiel liefert die Namen und Abteilungen der Mitarbeiter, die Vertriebsmitarbeiter und gleichzeitig Abteilungsleiter sind.

```
SELECT DISTINCT Employees.Surname,  
               FK_DepartmentID_DepartmentID.DepartmentName  
FROM ( SalesOrders, Departments  
      AS FK_DepartmentID_DepartmentID )  
KEY JOIN ( Employees JOIN Departments AS d  
          ON Employees.EmployeeID = d.DepartmentHeadID );
```

SQL Anywhere generiert zwei Join-Bedingungen:

- Es gibt genau eine Fremdschlüssel-Beziehung zwischen den Tabellenpaaren "SalesOrders/Employees" und "SalesOrders/d": `SalesOrders.SalesRepresentative = Employees.EmployeeID`
- Es gibt genau eine Fremdschlüssel-Beziehung zwischen den Tabellenpaaren "FK_DepartmentID_DepartmentID/Employees" und "FK_DepartmentID_DepartmentID/d": `FK_DepartmentID_DepartmentID.DepartmentID = Employees.DepartmentID`.

Dieses Beispiel ist gleichwertig zu Folgendem: In der folgenden Version ist es nicht erforderlich, den Korrelationsnamen `Departments AS FK_DepartmentID_DepartmentID` zu erstellen, da dies nur nötig war, um zu klären, welcher der beiden Fremdschlüssel verwendet werden soll, um "Employees" und "Departments" zu verknüpfen.

```
SELECT DISTINCT Employees.Surname,  
               Departments.DepartmentName  
FROM ( SalesOrders, Departments )  
JOIN ( Employees JOIN Departments AS d
```

```

ON Employees.EmployeeID = d.DepartmentHeadID )
ON SalesOrders.SalesRepresentative = Employees.EmployeeID
AND Departments.DepartmentID = Employees.DepartmentID;

```

Siehe auch

- „Schlüssel-Joins von Tabellenausdruckslisten“ auf Seite 529

Schlüssel-Joins von Ansichten und abgeleiteten Tabellen

Wenn Sie eine Ansicht oder eine abgeleitete Tabelle in einen Schlüssel-Join einbeziehen, befolgt SQL Anywhere dieselbe grundlegende Prozedur wie bei Tabellen, bis auf die folgenden Unterschiede:

- Für jeden Schlüssel-Join untersucht SQL Anywhere die Tabellenpaare in der FROM-Klausel der Abfrage und der Ansicht und generiert eine Join-Bedingung für alle Paare, und zwar unabhängig davon, ob die FROM-Klausel in der Ansicht Kommas oder Join-Schlüsselwörter enthält.
- SQL Anywhere verknüpft die Tabellen basierend auf dem Fremdschlüssel, dessen Rollename dem Korrelationsnamen der Ansicht oder dem der abgeleiteten Tabelle entspricht.
- Wenn Sie eine Ansicht oder eine abgeleitete Tabelle in einen Schlüssel-Join einbeziehen, kann die Ansicht oder die Definition der abgeleiteten Tabelle keines der folgenden Elemente enthalten: UNION, INTERSECT, EXCEPT, ORDER BY, DISTINCT, GROUP BY, Aggregatfunktionen, Fensterfunktionen, TOP, FIRST, START AT oder FOR XML. Wenn sie eines dieser Elemente enthält, wird ein Fehler gemeldet. Außerdem kann die abgeleitete Tabelle nicht als rekursiver Tabellenausdruck definiert werden.

Eine abgeleitete Tabelle funktioniert fast genauso wie eine Ansicht. Der einzige Unterschied besteht darin, dass anstelle der Referenzierung einer vordefinierten Ansicht die Definition für die Tabelle in die Anweisung einbezogen wird.

Beispiel 1

Folgende Anweisung verwendet z.B. die Ansicht Ansicht 1.

```

SELECT *
FROM View1 KEY JOIN B;

```

Die Definition von Ansicht 1 kann eine der folgenden Anweisungen sein. Sie ergeben jeweils dieselbe Join-Bedingung für B. (Die Ergebnismengen sind unterschiedlich, doch die Join-Bedingung ist jeweils gleich.)

```

SELECT *
FROM C CROSS JOIN D;

```

```

SELECT *
FROM C,D;

```

```

SELECT *
FROM C JOIN D ON (C.x = D.y);

```

Um eine Join-Bedingung für den Schlüssel-Join von Ansicht1 zu generieren, untersucht SQL Anywhere in jedem Fall die Tabellenpaare C-B und D-B. Es wird eine einzige Join-Bedingung erzeugt. Die Join-Bedingung wird basierend auf den Regeln für mehrfache Fremdschlüsselbeziehungen generiert, wobei

jedoch nach einem Fremdschlüssel gesucht wird, der denselben Namen hat wie die Korrelation der Ansicht (und nicht wie eine in der Ansicht referenzierte Tabelle).

Mit jeder der obigen Ansichtsdefinitionen können Sie die Verarbeitung von `View1 KEY JOIN B` wie folgt interpretieren:

SQL Anywhere generiert eine einzige Join-Bedingung, wobei die Tabellenpaare C-B und D-B berücksichtigt werden. Die Join-Bedingung wird gemäß den Regeln zur Festlegung der Schlüssel-Joins bei mehreren Fremdschlüsselbeziehungen erzeugt:

- Zunächst wird in C-B und D-B nach einem einzigen Fremdschlüssel gesucht, dessen Rollename dem Korrelationsnamen der Ansicht entspricht. Wenn genau ein Fremdschlüssel diese Bedingung erfüllt, wird dieser verwendet. Wenn mehrere Fremdschlüssel mit dem Rollennamen der Korrelation der Ansicht vorhanden sind, ist der Join zweideutig, und es wird ein Fehler ausgegeben.
- Wenn kein Fremdschlüssel mit dem Namen der Korrelation einer Ansicht vorhanden ist, sucht SQL Anywhere nach einer beliebigen Fremdschlüsselbeziehung zwischen den Tabellen. Wenn es eine solche Beziehung gibt, wird sie verwendet. Wenn es mehrere gibt, ist der Join zweideutig, und es wird ein Fehler ausgegeben.
- Wenn keine Fremdschlüsselbeziehung vorhanden ist, wird ein Fehler ausgegeben.

Angenommen, die generierte Join-Bedingung lautet: `B.y = D.z`. Es ist nun möglich, den ursprünglichen Join zu erweitern. Die folgenden Anweisungen sind beispielsweise gleichwertig:

```
SELECT *  
FROM View1 KEY JOIN B;  
  
SELECT *  
FROM View1 JOIN B ON B.y = View1.z;
```

Beispiel 2

Folgende Ansicht enthält alle Mitarbeiterdaten über die Leiter der einzelnen Abteilungen:

```
CREATE VIEW V AS  
SELECT Departments.DepartmentName, Employees.*  
FROM Employees JOIN Departments  
    ON Employees.EmployeeID = Departments.DepartmentHeadID;
```

Folgende Abfrage verknüpft die Ansicht mit einem Tabellenausdruck:

```
SELECT *  
FROM V KEY JOIN ( SalesOrders,  
    Departments FK_DepartmentID_DepartmentID );
```

Die nachstehende Abfrage ist der vorherigen Abfrage gleichwertig:

```
SELECT *  
FROM V JOIN ( SalesOrders,  
    Departments FK_DepartmentID_DepartmentID )  
ON ( V.EmployeeID = SalesOrders.SalesRepresentative  
    AND V.DepartmentID =  
        FK_DepartmentID_DepartmentID.DepartmentID );
```


Siehe auch

- „Rekursive allgemeine Tabellenausdrücke“ auf Seite 543
- „Schlüssel-Joins bei mehrfachen Fremdschlüsselbeziehungen“ auf Seite 525
- „Schlüssel-Joins von Tabellenausdrücken“ auf Seite 528

Regeln, die die Verarbeitung von Schlüssel-Joins beschreiben

Folgende Regeln fassen die oben gegebenen Informationen zusammen:

Regel 1: Zwei Tabellen mit Schlüssel-Join verknüpfen

Diese Regel gilt für `A KEY JOIN B`, wobei A und B Basis- oder temporäre Tabellen sind.

1. Es werden alle Fremdschlüssel aus A gesucht, die B referenzieren.

Wenn ein Fremdschlüssel vorhanden ist, dessen Rollename der Korrelationsname von Tabelle B ist, wird er als bevorzugter Fremdschlüssel markiert.

2. Es werden alle Fremdschlüssel aus B gesucht, die A referenzieren.

Wenn ein Fremdschlüssel vorhanden ist, dessen Rollename der Korrelationsname von Tabelle A ist, wird er als bevorzugter Fremdschlüssel markiert.

3. Wenn mehrere bevorzugte Schlüssel vorhanden sind, ist der Join zweideutig. Der Syntaxfehler `SQL_AMBIGUOUS_JOIN (-147)` wird ausgegeben.
4. Wenn es einen einzigen bevorzugten Schlüssel gibt, wird dieser Fremdschlüssel ausgewählt, um die generierte Join-Bedingung für diesen `KEY JOIN`-Ausdruck festzulegen.
5. Wenn es keinen bevorzugten Schlüssel gibt, werden andere Fremdschlüssel von A und B verwendet:
 - Wenn mehrere Fremdschlüssel zwischen A und B vorhanden sind, ist der Join zweideutig. Der Syntaxfehler `SQL_AMBIGUOUS_JOIN (-147)` wird ausgegeben.
 - Wenn es einen einzigen Fremdschlüssel gibt, wird dieser Fremdschlüssel ausgewählt, um die generierte Join-Bedingung für diesen `KEY JOIN`-Ausdruck festzulegen.
 - Wenn es keinen Fremdschlüssel gibt, ist der Join ungültig, und es wird ein Fehler ausgegeben.

Regel 2: Tabellenausdrücke ohne Kommas mit Schlüssel-Join verknüpfen

Diese Regel gilt für `A KEY JOIN B`, wobei A und B Tabellenausdrücke sind, die keine Kommas enthalten.

1. Für jedes Tabellenpaar, jeweils eine Tabelle aus Ausdruck A und eine Tabelle aus Ausdruck B, werden alle Fremdschlüssel aufgelistet und alle bevorzugten Fremdschlüssel zwischen den Tabellen markiert. Die Regel zur Bestimmung eines bevorzugten Fremdschlüssel ist in Regel 1 beschrieben.
2. Wenn mehrere bevorzugte Schlüssel vorhanden sind, ist der Join zweideutig. Der Syntaxfehler `SQL_AMBIGUOUS_JOIN (-147)` wird ausgegeben.

3. Wenn es einen einzigen bevorzugten Schlüssel gibt, wird dieser Fremdschlüssel ausgewählt, um die generierte Join-Bedingung für diesen KEY JOIN-Ausdruck festzulegen.
4. Wenn es keinen bevorzugten Schlüssel gibt, werden andere Fremdschlüssel der Tabellenpaare verwendet:
 - Wenn mehrere Fremdschlüssel vorhanden sind, ist der Join zweideutig. Der Syntaxfehler `SQL_ambiguous_join` (-147) wird ausgegeben.
 - Wenn es einen einzigen Fremdschlüssel gibt, wird dieser Fremdschlüssel ausgewählt, um die generierte Join-Bedingung für diesen KEY JOIN-Ausdruck festzulegen.
 - Wenn es keinen Fremdschlüssel gibt, ist der Join ungültig, und es wird ein Fehler ausgegeben.

Regel 3: Tabellenausdruckslisten durch Schlüssel-Joins verknüpfen

Diese Regel gilt für `(A1, A2, ...) KEY JOIN (B1, B2, ...)`, wobei A1, B1 usw. Tabellenausdrücke sind, die keine Kommas enthalten.

1. Für jedes Paar von Tabellenausdrücken, A_i und B_j, wird eine eindeutige generierte Join-Bedingung für den Tabellenausdruck `(Ai KEY JOIN Bj)` gesucht. Dabei wird Regel 1 oder 2 angewendet. Falls ein KEY JOIN eines Tabellenausdrucks paars nach Regel 1 oder 2 zweideutig ist, wird ein Syntaxfehler ausgegeben.
2. Die generierte Join-Bedingung für diesen KEY JOIN-Ausdruck ist die Verbindung der in Schritt 1 gefundenen Join-Bedingungen.

Regel 4: Listen und Tabellenausdrücke ohne Kommas durch Schlüssel-Joins verknüpfen

Diese Regel gilt für `(A1, A2, ...) KEY JOIN (B1, B2, ...)`, wobei A1, B1 usw. Tabellenausdrücke sind, die Kommas enthalten können.

1. Für jedes Paar von Tabellenausdrücken, A_i und B_j, wird eine eindeutige generierte Join-Bedingung für den Tabellenausdruck `(Ai KEY JOIN Bj)` gesucht. Dabei wird Regel 1, 2 oder 3 angewendet. Falls ein KEY JOIN eines Tabellenausdrucks paars nach Regel 1, 2 oder 3 zweideutig ist, wird ein Syntaxfehler ausgegeben.
2. Die generierte Join-Bedingung für diesen KEY JOIN-Ausdruck ist die Verbindung der in Schritt 1 gefundenen Join-Bedingungen.

Siehe auch

- „Es gibt mehr als eine Möglichkeit für einen Join von '%1' mit '%2'“ [[Fehlermeldungen](#)]

Allgemeine Tabellenausdrücke

Allgemeine Tabellenausdrücke werden unter Verwendung der WITH-Klausel definiert, die dem SELECT-Schlüsselwort in einer SELECT-Anweisung vorangestellt wird. Der Inhalt der Klausel legt eine oder mehrere temporäre Ansichten fest, die nur innerhalb des Bereichs einer einzelnen SELECT-Anweisung bekannt sind und möglicherweise an anderer Stelle in der Anweisung referenziert werden. Die Syntax dieser Klausel ähnelt der einer CREATE VIEW-Anweisung.

Allgemeine Tabellenausdrücke sind nützlich oder können notwendig sein, wenn eine Abfrage mehrere Aggregatfunktionen beinhaltet oder eine Ansicht innerhalb einer gespeicherten Prozedur festlegt, die Programmvariablen referenziert. Allgemeine Tabellenausdrücke bieten auch eine bequeme Methode, um Wertemengen temporär zu speichern.

Beispiel

Betrachten Sie zum Beispiel den Fall, dass die Abteilung mit den meisten Mitarbeitern ermittelt werden soll. Die Tabelle "Employees" in der SQL-Anywhere-Beispieldatenbank listet alle Mitarbeiter in einer fiktiven Firma auf und gibt an, in welcher Abteilung jeder arbeitet. Die folgende Abfrage listet die ID-Codes der Abteilungen und die Gesamtzahl der Mitarbeiter in ihnen auf.

```
SELECT DepartmentID, COUNT( * ) AS n
FROM Employees
GROUP BY DepartmentID;
```

Mit der folgenden Abfrage können Sie die Abteilung extrahieren, die die meisten Mitarbeiter hat:

```
SELECT DepartmentID, n
FROM ( SELECT DepartmentID, COUNT( * ) AS n
      FROM Employees GROUP BY DepartmentID ) AS a
WHERE a.n =
      ( SELECT MAX( n )
        FROM ( SELECT DepartmentID, COUNT( * ) AS n
              FROM Employees GROUP BY DepartmentID ) AS b );
```

Zwar liefert diese Anweisung das korrekte Ergebnis, doch hat sie auch Nachteile. Der erste Nachteil ist, dass durch die wiederholte Unterabfrage die Anweisung weniger effizient ist. Der zweite Nachteil ist, dass diese Anweisung keine klare Verbindung zwischen den Unterabfragen herstellt.

Dies umgehen Sie, indem Sie eine Ansicht erstellen und sie dazu verwenden, die Abfrage neu zu formulieren. Dieses Vorgehen vermeidet die oben erwähnten Probleme.

```
CREATE VIEW CountEmployees( DepartmentID, n ) AS
  SELECT DepartmentID, COUNT( * ) AS n
  FROM Employees GROUP BY DepartmentID;

SELECT DepartmentID, n
FROM CountEmployees
WHERE n = ( SELECT MAX( n )
           FROM CountEmployees );
```

Der Nachteil dieser Vorgehensweise besteht darin, dass eine bestimmte Menge an Overhead erforderlich ist, weil der Datenbankserver die Systemtabellen aktualisieren muss, wenn die Ansicht erstellt wird. Wenn die Ansicht häufig verwendet wird, ist dieses Vorgehen vernünftig. Wenn die Ansicht jedoch nur einmal innerhalb einer bestimmten SELECT-Anweisung verwendet wird, besteht die bevorzugte Methode darin, stattdessen wie folgt einen allgemeinen Tabellenausdruck zu verwenden.

```
WITH CountEmployees( DepartmentID, n ) AS
  ( SELECT DepartmentID, COUNT( * ) AS n
    FROM Employees GROUP BY DepartmentID )
SELECT DepartmentID, n
FROM CountEmployees
WHERE n = ( SELECT MAX( n )
           FROM CountEmployees );
```

Wenn die Abfrage so geändert wird, dass nach der Abteilung mit den wenigsten Mitarbeitern gesucht wird, können möglicherweise mehrere Zeilen zurückgeben werden.

```
WITH CountEmployees( DepartmentID, n ) AS
( SELECT DepartmentID, COUNT( * ) AS n
  FROM Employees GROUP BY DepartmentID )
SELECT DepartmentID, n
FROM CountEmployees
WHERE n = ( SELECT MIN( n )
           FROM CountEmployees );
```

In der SQL Anywhere-Beispieldatenbank haben zwei Abteilungen die geringste Anzahl von Mitarbeitern, nämlich 9.

Siehe auch

- „Mehrere Korrelationsnamen“ auf Seite 538
- „Mehrere Tabellenausdrücke“ auf Seite 538
- „Wo allgemeine Tabellenausdrücke zulässig sind“ auf Seite 539

Mehrere Korrelationsnamen

Ähnlich wie bei der Verwendung von Tabellen können Sie mehreren Instanzen eines allgemeinen Tabellenausdrucks verschiedene Korrelationsnamen zuordnen. Dies ermöglicht es Ihnen, einen allgemeinen Tabellenausdruck mit sich selbst zu verknüpfen. Die nachfolgende Abfrage zum Beispiel liefert Paare von Abteilungen, die dieselbe Anzahl von Mitarbeitern haben, obwohl es in der SQL Anywhere-Beispieldatenbank nur zwei Abteilungen mit derselben Anzahl von Mitarbeitern gibt.

```
WITH CountEmployees( DepartmentID, n ) AS
( SELECT DepartmentID, COUNT( * ) AS n
  FROM Employees GROUP BY DepartmentID )
SELECT a.DepartmentID, a.n, b.DepartmentID, b.n
FROM CountEmployees AS a JOIN CountEmployees AS b
ON a.n = b.n AND a.DepartmentID < b.DepartmentID;
```

Siehe auch

- „Allgemeine Tabellenausdrücke“ auf Seite 536
- „Mehrere Tabellenausdrücke“ auf Seite 538
- „Wo allgemeine Tabellenausdrücke zulässig sind“ auf Seite 539

Mehrere Tabellenausdrücke

Eine einzelne WITH-Klausel kann mehr als einen allgemeinen Tabellenausdruck definieren. Diese Definitionen müssen durch Kommas getrennt sein. Das folgende Beispiel listet die Abteilung auf, die die kleinste Lohnsumme sowie die Abteilung, die die größte Anzahl von Mitarbeitern hat.

```
WITH
  CountEmployees( DepartmentID, n ) AS
  ( SELECT DepartmentID, COUNT( * ) AS n
    FROM Employees GROUP BY DepartmentID ),
  DepartmentPayroll( DepartmentID, amount ) AS
  ( SELECT DepartmentID, SUM( Salary ) AS amount
    FROM Employees GROUP BY DepartmentID )
```

```
SELECT count.DepartmentID, count.n, pay.amount
FROM CountEmployees AS count JOIN DepartmentPayroll AS pay
ON count.DepartmentID = pay.DepartmentID
WHERE count.n = ( SELECT MAX( n ) FROM CountEmployees )
OR pay.amount = ( SELECT MIN( amount ) FROM DepartmentPayroll );
```

Siehe auch

- „Allgemeine Tabellenausdrücke“ auf Seite 536
- „Mehrere Korrelationsnamen“ auf Seite 538
- „Wo allgemeine Tabellenausdrücke zulässig sind“ auf Seite 539

Wo allgemeine Tabellenausdrücke zulässig sind

Allgemeine Tabellenausdrücke sind nur an drei Stellen zulässig, auch wenn sie überall im Hauptteil der Abfrage oder in Unterabfragen referenziert werden können.

- **Oberste SELECT-Anweisung** Allgemeine Tabellenausdrücke sind innerhalb von SELECT-Anweisungen der obersten Stufe zulässig, aber nicht in Unterabfragen.

```
WITH DepartmentPayroll( DepartmentID, amount ) AS
( SELECT DepartmentID, SUM( Salary ) AS amount
FROM Employees GROUP BY DepartmentID )
SELECT DepartmentID, amount
FROM DepartmentPayroll
WHERE amount = ( SELECT MAX( amount )
FROM DepartmentPayroll );
```

- **Die obersten SELECT-Anweisungen in einer Ansichtsdefinition** Allgemeine Tabellenausdrücke sind innerhalb der SELECT-Anweisung der obersten Stufe zulässig, die eine Ansicht festlegt, aber nicht innerhalb von Unterabfragen.

```
CREATE VIEW LargestDept ( DepartmentID, Size, pay ) AS
WITH
CountEmployees( DepartmentID, n ) AS
( SELECT DepartmentID, COUNT( * ) AS n
FROM Employees GROUP BY DepartmentID ),
DepartmentPayroll( DepartmentID, amount ) AS
( SELECT DepartmentID, SUM( Salary ) AS amount
FROM Employees GROUP BY DepartmentID )
SELECT count.DepartmentID, count.n, pay.amount
FROM CountEmployees count JOIN DepartmentPayroll pay
ON count.DepartmentID = pay.DepartmentID
WHERE count.n = ( SELECT MAX( n ) FROM CountEmployees )
OR pay.amount = ( SELECT MAX( amount ) FROM DepartmentPayroll );
```

- **Eine oberste SELECT-Anweisung in einer INSERT-Anweisung** Allgemeine Tabellenausdrücke sind innerhalb einer SELECT-Anweisung der obersten Stufe in einer INSERT-Anweisung zulässig, aber nicht in Unterabfragen innerhalb der INSERT-Anweisung.

```
CREATE TABLE LargestPayrolls ( DepartmentID INTEGER, Payroll NUMERIC,
CurrentDate DATE );
INSERT INTO LargestPayrolls( DepartmentID, Payroll, CurrentDate )
WITH DepartmentPayroll( DepartmentID, amount ) AS
( SELECT DepartmentID, SUM( Salary ) AS amount
FROM Employees
GROUP BY DepartmentID )
```

```
SELECT DepartmentID, amount, CURRENT_TIMESTAMP
FROM DepartmentPayroll
WHERE amount = ( SELECT MAX( amount )
                FROM DepartmentPayroll );
```

Siehe auch

- „Allgemeine Tabellenausdrücke“ auf Seite 536
- „Mehrere Korrelationsnamen“ auf Seite 538
- „Mehrere Tabellenausdrücke“ auf Seite 538

Typische Anwendungen von allgemeinen Tabellenausdrücken

Generell sind allgemeine Tabellenausdrücke immer dann nützlich, wenn ein Tabellenausdruck mehrfach in einer einzelnen Abfrage erscheinen muss. Die folgenden typischen Situationen sind für allgemeine Tabellenausdrücke geeignet.

- Abfragen, die mehrere Aggregatfunktionen betreffen.
- Ansichten innerhalb einer Prozedur, die eine Referenz zu einer Programmvariable enthalten muss.
- Abfragen, die temporäre Tabellen zum Speichern einer Wertemenge verwenden.

Diese Liste ist nicht vollständig. Es gibt viele andere Situationen, in denen allgemeine Tabellenausdrücke nützlich sind.

Mehrere Aggregatfunktionen

Allgemeine Tabellenausdrücke sind nützlich, sobald mehrere Stufen von Aggregaten in einer einzigen Abfrage vorkommen müssen. Das ist der Fall im Beispiel des vorherigen Abschnitts. Die Aufgabe war es, die Abteilungs-ID der Abteilung abzufragen, die die meisten Mitarbeiter hat. Um das zu erreichen, wird die COUNT-Aggregatfunktion verwendet, um die Anzahl der Mitarbeiter in den einzelnen Abteilungen zu berechnen, und die MAX-Funktion, um die größte Abteilung auszuwählen.

Eine ähnliche Situation entsteht, wenn Sie eine Abfrage schreiben um festzustellen, welche Abteilung die höchste Lohnsumme hat. Die SUM-Aggregatfunktion wird verwendet, um die Lohnsumme der einzelnen Abteilungen zu berechnen, und die MAX-Funktion, um zu ermitteln, welche die größte ist. Das Vorhandensein beider Funktionen in der Abfrage weist darauf hin, dass ein allgemeiner Tabellenausdruck hilfreich sein könnte.

```
WITH DepartmentPayroll( DepartmentID, amount ) AS
( SELECT DepartmentID, SUM( Salary ) AS amount
  FROM Employees GROUP BY DepartmentID )
SELECT DepartmentID, amount
FROM DepartmentPayroll
WHERE amount = ( SELECT MAX( amount )
                FROM DepartmentPayroll );
```

Siehe auch

- „Fenster-Aggregatfunktionen“ auf Seite 571

Ansichten, die Variablen referenzieren

Manchmal kann es nützlich sein, eine Ansicht zu erstellen, die eine Referenz zu einer Programmvariablen enthält. Sie können zum Beispiel eine Variable innerhalb einer Prozedur definieren, die einen bestimmten Kunden kennzeichnet. Sie möchten das Kaufverhalten des Kunden abfragen, und da Sie auf ähnliche Daten mehrfach zugreifen, oder möglicherweise mehrere Aggregatfunktionen verwenden werden, empfiehlt es sich, eine Ansicht zu erstellen, die die Daten über den angegebenen Kunden enthält.

Sie können keine Ansicht erstellen, die eine Programmvariable referenziert, weil es keine Möglichkeit gibt, den Bereich einer Ansicht auf den Ihrer Prozedur zu beschränken. Sobald sie erstellt ist, kann eine Ansicht in anderen Kontexten verwendet werden. Sie können jedoch einen allgemeinen Tabellenausdruck innerhalb der Abfragen in Ihrer Prozedur verwenden. Da der Bereich eines allgemeinen Tabellenausdrucks auf die Anweisung beschränkt ist, erzeugt die Variablenreferenz keine Zweideutigkeit und ist zulässig.

Die folgende Anweisung wählt die Bruttoverkaufszahlen verschiedener Handelsvertreter in der SQL Anywhere-Beispieldatenbank aus:

```
SELECT GivenName || ' ' || Surname AS sales_rep_name,
       SalesRepresentative AS sales_rep_id,
       SUM( p.UnitPrice * i.Quantity ) AS total_sales
FROM Employees LEFT OUTER JOIN SalesOrders AS o
              INNER JOIN SalesOrderItems AS i
              INNER JOIN Products AS p
WHERE OrderDate BETWEEN '2000-01-01' AND '2001-12-31'
GROUP BY SalesRepresentative, GivenName, Surname;
```

Die oben stehende Abfrage ist die Basis des allgemeinen Tabellenausdrucks, der in der folgenden Prozedur erscheint: Die ID-Nummer eines Handelsvertreters und das in Frage kommende Jahr sind hereinkommende Parameter. Wie die folgende Prozedur zeigt, können Prozedurparameter und etwaige deklarierte lokale Variable innerhalb der WITH-Klausel referenziert werden.

```
CREATE PROCEDURE sales_rep_total (
    IN rep INTEGER,
    IN yyyy INTEGER )
BEGIN
    DECLARE StartDate DATE;
    DECLARE EndDate DATE;
    SET StartDate = YMD( yyyy, 1, 1 );
    SET EndDate = YMD( yyyy, 12, 31 );
    WITH total_sales_by_rep ( sales_rep_name,
                             sales_rep_id,
                             month,
                             order_year,
                             total_sales ) AS
    ( SELECT GivenName || ' ' || Surname AS sales_rep_name,
       SalesRepresentative AS sales_rep_id,
       month( OrderDate ),
       year( OrderDate ),
       SUM( p.UnitPrice * i.Quantity ) AS total_sales
    FROM Employees LEFT OUTER JOIN SalesOrders o
                     INNER JOIN SalesOrderItems i
```

```
                INNER JOIN Products p
WHERE OrderDate BETWEEN StartDate AND EndDate
      AND SalesRepresentative = rep
GROUP BY year( OrderDate ), month( OrderDate ),
        GivenName, Surname, SalesRepresentative )
SELECT sales_rep_name,
       monthname( YMD(yyyy, month, 1) ) AS month_name,
       order_year,
       total_sales
FROM total_sales_by_rep
WHERE total_sales =
      ( SELECT MAX( total_sales ) FROM total_sales_by_rep )
ORDER BY order_year ASC, month ASC;
END;
```

Die folgende Anweisung ruft die obige Prozedur auf.

```
CALL sales_rep_total( 129, 2000 );
```

Ansichten, die Werte speichern

Es kann nützlich sein, eine bestimmte Menge von Werten innerhalb einer SELECT-Anweisung oder einer Prozedur zu speichern. Nehmen wir zum Beispiel an, dass eine Firma es vorzieht, die Ergebnisse ihrer Handelsvertreter anstatt quartalsweise pro Jahresdrittel zu analysieren. Da es im Gegensatz zu den Quartalen keine integrierten Datumsteile für Jahresdrittel gibt, ist es notwendig, die Datumsangaben innerhalb der Prozedur zu speichern.

```
WITH thirds ( q_name, q_start, q_end ) AS
( SELECT 'T1', '2000-01-01', '2000-04-30' UNION
  SELECT 'T2', '2000-05-01', '2000-08-31' UNION
  SELECT 'T3', '2000-09-01', '2000-12-31' )
SELECT q_name,
       SalesRepresentative,
       count(*) AS num_orders,
       SUM( p.UnitPrice * i.Quantity ) AS total_sales
FROM thirds LEFT OUTER JOIN SalesOrders AS o
  ON OrderDate BETWEEN q_start and q_end
      KEY JOIN SalesOrderItems AS i
      KEY JOIN Products AS p
GROUP BY q_name, SalesRepresentative
ORDER BY q_name, SalesRepresentative;
```

Diese Methode sollte mit Umsicht verwendet werden, da die Werte möglicherweise periodische Aktualisierungen erfordern. So muss zum Beispiel die oben stehende Anweisung modifiziert werden, um für ein anderes Jahr zu gelten.

Sie können diese Methode auch innerhalb von Prozeduren verwenden. Das folgende Beispiel deklariert eine Prozedur, die das betreffende Jahr als Argument nimmt.

```
CREATE PROCEDURE sales_by_third ( IN y INTEGER )
BEGIN
  WITH thirds ( q_name, q_start, q_end ) AS
  ( SELECT 'T1', YMD( y, 01, 01), YMD( y, 04, 30 ) UNION
    SELECT 'T2', YMD( y, 05, 01), YMD( y, 08, 31 ) UNION
    SELECT 'T3', YMD( y, 09, 01), YMD( y, 12, 31 ) )
  SELECT q_name,
         SalesRepresentative,
         count(*) AS num_orders,
```



```

        SUM( p.UnitPrice * i.Quantity ) AS total_sales
FROM thirds LEFT OUTER JOIN SalesOrders AS o
  ON OrderDate BETWEEN q_start and q_end
   KEY JOIN SalesOrderItems AS i
   KEY JOIN Products AS p
GROUP BY q_name, SalesRepresentative
ORDER BY q_name, SalesRepresentative;
END;

```

Die folgende Anweisung ruft die obige Prozedur auf.

```
CALL sales_by_third (2000);
```

Rekursive allgemeine Tabellenausdrücke

Allgemeine Tabellenausdrücke sind dann rekursiv, wenn sie wiederholt ausgeführt und bei jeder Ausführung zusätzliche Zeilen zurückgegeben werden, bis die gesamte Ergebnismenge abgerufen wurde. Sie können einen allgemeinen Tabellenausdruck als rekursiv festlegen, indem Sie in der WITH-Klausel das RECURSIVE-Schlüsselwort unmittelbar nach WITH einfügen. Eine einzelne WITH-Klausel kann mehrere rekursive Ausdrücke enthalten, die sowohl rekursiv als auch nicht rekursiv sein können.

Mit einer Rekursion ist es einfacher, Tabellen zu durchsuchen, die Baum- oder baumartige Datenstrukturen haben. Die einzige Möglichkeit, eine solche Struktur mit einer einzigen Anweisung zu durchsuchen, ohne rekursive Ausdrücke zu verwenden, besteht darin, die Tabelle für jede mögliche Ebene einmal mit sich selbst zu verknüpfen.

Einschränkungen für rekursive allgemeine Tabellenausdrücke

- Referenzen zu anderen rekursiven allgemeinen Tabellenausdrücken können nicht innerhalb der Definition von rekursiven allgemeinen Tabellenausdrücken erscheinen, weil rekursive allgemeine Tabellenausdrücke nicht gegenseitig rekursiv sein können. Allerdings können rekursive allgemeine Tabellenausdrücke Referenzen zu nicht rekursiven allgemeinen Tabellenausdrücken enthalten und umgekehrt.
- Der einzige zulässige Mengenoperator zwischen der anfänglichen Unterabfrage und der rekursiven Unterabfrage ist UNION ALL.
- Innerhalb der Definition einer rekursiven Unterabfrage darf eine Eigenreferenz zum rekursiven allgemeinen Tabellenausdruck nur in der FROM-Klausel der rekursiven Unterabfrage auftreten und nicht auf der nullwertliefernden Seite eines Outer-Joins.
- Die rekursive Unterabfrage darf keine DISTINCT-, GROUP BY- oder ORDER BY-Klausel enthalten.
- In der rekursiven Unterabfrage kann keine Aggregatfunktion verwendet werden.
- Um endlos laufende rekursive Abfragen zu vermeiden, wird ein Fehler generiert, wenn die Anzahl der Rekursionsstufen den aktuellen Wert der Option "max_recursive_iterations" übersteigt. Der Standardwert für diese Option ist 100.

Beispiel

Bei einer Tabelle, die Unterstellungsbeziehungen in einer Firma darstellt, können Sie eine Abfrage schreiben, die alle einer bestimmten Person unterstellten Mitarbeiter zurückgibt.

Je nachdem, wie Sie die Abfrage schreiben, kann es sinnvoll sein, die Anzahl der Rekursionsstufen zu beschränken. Durch Beschränken der Anzahl der Stufen können Sie beispielsweise nur die obersten Managementebenen zurückgeben, was aber möglicherweise einige Mitarbeiter ausschließt, wenn die Hierarchie mehr Ebenen aufweist, als Sie angenommen haben. Wenn Sie für die Anzahl der Stufen keine Einschränkung festlegen, stellen Sie zwar sicher, dass kein Mitarbeiter ausgeschlossen wird, erzeugen aber möglicherweise unendliche Rekursionen, falls die Ausführung Schleifen erfordert, wenn z.B. ein Mitarbeiter direkt oder indirekt sich selbst unterstellt ist. Dies könnte in der Managementhierarchie einer Firma eintreten, wenn Mitarbeiter der Firma auch im Vorstand sitzt.

Die folgende Abfrage zeigt, wie Sie die Mitarbeiter anhand ihrer Managementebene auflisten. Ebene 0 stellt Mitarbeiter ohne Vorgesetzte dar. Ebene 1 stellt Mitarbeiter dar, die direkt einem Vorgesetzten der Ebene 0 unterstellt sind, Ebene 2 stellt Mitarbeiter dar, die direkt einem Vorgesetzten der Ebene 1 unterstellt sind, und so weiter.

```
WITH RECURSIVE
manager ( EmployeeID, ManagerID,
          GivenName, Surname, mgmt_level ) AS
( ( SELECT EmployeeID, ManagerID,      -- initial subquery
    GivenName, Surname, 0
  FROM Employees AS e
  WHERE ManagerID = EmployeeID )
  UNION ALL
  ( SELECT e.EmployeeID, e.ManagerID,  -- recursive subquery
    e.GivenName, e.Surname, m.mgmt_level + 1
  FROM Employees AS e JOIN manager AS m
    ON e.ManagerID = m.EmployeeID
    AND e.ManagerID <> e.EmployeeID
    AND m.mgmt_level < 20 ) )
SELECT * FROM manager
ORDER BY mgmt_level, Surname, GivenName;
```

Die Bedingung innerhalb der rekursiven Abfrage, mit der die Anzahl der Managementebenen auf weniger als 20 beschränkt wird (`m.mgmt_level < 20`), wird als Stop-Bedingung bezeichnet und ist eine wichtige Vorsichtsmaßnahme. Sie verhindert eine unendliche Rekursion, falls die Tabellendaten eine Schleife enthalten.

Die `max_recursive_iterations`-Option kann ebenfalls dazu verwendet werden, endlos laufende rekursive Abfragen zu verhindern. Der Standardwert dieser Option ist 100. Rekursive Abfragen, die diese Anzahl von Wiederholungen überschreiten, werden beendet, verursachen aber einen Fehler. Auch wenn diese Option scheinbar die Bedeutung einer Stop-Bedingung vermindert, ist das nicht der Fall. Die Anzahl der bei jeder Wiederholung ausgewählten Zeilen wächst möglicherweise exponentiell, was ernsthafte Auswirkungen auf die Datenbank-Performance haben kann, bevor das Maximum erreicht ist. Stop-Bedingungen in rekursiven Abfragen sind eine Möglichkeit, entsprechende Grenzen in solchen Situationen zu setzen.

Rekursive allgemeine Tabellenausdrücke enthalten eine **Anfangs-Unterabfrage**, einen sogenannten Initialwert (Seed-Wert), und eine **rekursive Unterabfrage**, die bei jeder Wiederholung zusätzliche Zeilen zur Ergebnismenge hinzufügt. Die zwei Teile können nur mit dem UNION ALL-Operator verbunden werden. Die Anfangs-Unterabfrage ist eine gewöhnliche nicht-rekursive Abfrage und wird zuerst abgearbeitet. Der rekursive Abschnitt enthält eine Referenz zu den Zeilen, die während der vorherigen Wiederholung hinzugefügt wurden. Die Rekursion stoppt automatisch, sobald eine Wiederholung keine neuen Zeilen mehr generiert. Es gibt keine Möglichkeit, Zeilen zu referenzieren, die vor der vorherigen Wiederholung ausgewählt wurden.

Die SELECT-Liste der rekursiven Unterabfrage muss der der anfänglichen Unterabfrage in Anzahl und Datentyp entsprechen. Wenn eine automatische Konvertierung der Datentypen nicht durchgeführt werden kann, passen Sie explizit die Ergebnisse der einen Unterabfrage an, damit sie jenen in der anderen Unterabfrage entsprechen.

Siehe auch

- „Mehrere rekursive allgemeine Tabellenausdrücke“ auf Seite 546
- „max_recursive_iterations-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Datentyp-Deklarationen in rekursiven allgemeinen Tabellenausdrücken

Die Datentypen der Spalten in der temporären Ansicht werden durch jene in der anfänglichen Unterabfrage definiert. Die Datentypen der Spalten von der rekursiven Unterabfrage müssen übereinstimmen. Der Datenbankserver versucht, die von der rekursiven Unterabfrage zurückgegebenen Werte automatisch zu konvertieren, damit sie mit jenen der anfänglichen Unterabfrage übereinstimmen. Wenn das nicht möglich ist oder wenn bei der Konvertierung Informationen verloren gehen, wird ein Fehler generiert.

Im Allgemeinen sind explizite Typenfestlegungen notwendig, wenn die anfängliche Unterabfrage einen Literal oder NULL ergibt. Explizite Typenfestlegungen können auch erforderlich sein, wenn die anfängliche Unterabfrage Werte aus anderen Spalten als die rekursive Unterabfrage auswählt.

Typenfestlegungen sind möglicherweise erforderlich, wenn die Spalten der anfänglichen Unterabfrage nicht dieselben Domänen wie die rekursive Unterabfrage haben. Typenfestlegungen in der anfänglichen Unterabfrage müssen immer auf NULL angewendet werden.

Das Stücklistenproblem funktioniert beispielsweise deshalb, weil die anfängliche Unterabfrage Zeilen aus der Bücherregal-Tabelle zurückgibt und die Datentypen der ausgewählten Spalten übernimmt.

Wenn diese Abfrage jedoch folgendermaßen umgeschrieben wird, sind explizite Typenfestlegungen erforderlich.

```
WITH RECURSIVE parts ( component, subcomponent, quantity ) AS
( SELECT NULL, 'bookcase', 1          -- ERROR! Wrong domains!
  UNION ALL
  SELECT b.component, b.subcomponent,
         p.quantity * b.quantity
  FROM parts p JOIN bookcase b
    ON p.subcomponent = b.component )
SELECT * FROM parts
ORDER BY component, subcomponent;
```

Ohne Casting ergeben sich Fehler, und zwar aus folgenden Gründen:

- Der korrekte Datentyp für Komponentennamen ist VARCHAR, aber die erste Spalte ist NULL.
- Die Ziffer 1 lässt auf SMALLINT schließen, doch der Datentyp der Mengenspalte ist INT.

Bei der zweiten Spalte ist keine Typenfestlegung erforderlich, weil diese Spalte der Anfangs-Abfrage bereits eine Zeichenfolge ist.

Ein Casting des Datentyps in der anfänglichen Unterabfrage ermöglicht es der Abfrage, sich wie vorgesehen zu verhalten.

```
WITH RECURSIVE parts ( component, subcomponent, quantity ) AS
( SELECT CAST( NULL AS VARCHAR ), -- CASTs must be used
    'bookcase', -- to declare the
    CAST( 1 AS INT ) -- correct datatypes
    UNION ALL
    SELECT b.component, b.subcomponent,
        p.quantity * b.quantity
    FROM parts p JOIN bookcase b
        ON p.subcomponent = b.component )
SELECT * FROM parts
ORDER BY component, subcomponent;
```

Siehe auch

- „Stücklistenproblem“ auf Seite 547

Mehrere rekursive allgemeine Tabellenausdrücke

Eine rekursive Abfrage kann mehrere rekursive Abfragen enthalten, solange sie voneinander unabhängig sind. Sie kann auch eine Mischung aus rekursiven und nicht-rekursiven allgemeinen Tabellenausdrücken enthalten. Das RECURSIVE-Schlüsselwort muss verwendet werden, wenn zumindest einer der allgemeinen Tabellenausdrücke rekursiv ist.

Die folgende Abfrage, die dasselbe Ergebnis wie die vorherige Abfrage zurückgibt, verwendet beispielsweise einen zweiten, nicht-rekursiven allgemeinen Tabellenausdruck, um die Länge der kürzesten Route auszuwählen. Die Definition des zweiten allgemeinen Tabellenausdrucks wird von der Definition des ersten durch ein Komma getrennt.

```
WITH RECURSIVE
trip ( route, destination, previous, distance, segments ) AS
( SELECT CAST( origin || ', ' || destination AS VARCHAR(256) ),
    destination, origin, distance, 1
    FROM travel
    WHERE origin = 'Kitchener'
    UNION ALL
    SELECT route || ', ' || v.destination,
        v.destination,
        v.origin,
        t.distance + v.distance,
        segments + 1
    FROM trip t JOIN travel v ON t.destination = v.origin
    WHERE v.destination <> 'Kitchener'
        AND v.destination <> t.previous
        AND v.origin <> 'Pembroke'
        AND segments
            < ( SELECT count(*)/2 FROM travel ) ),
shortest ( distance ) AS
( SELECT MIN(distance)
    FROM trip
    WHERE destination = 'Pembroke' )
SELECT route, distance, segments FROM trip
WHERE destination = 'Pembroke' AND
    distance < 1.5 * ( SELECT distance FROM shortest )
ORDER BY distance, segments, route;
```

Wie nicht-rekursive allgemeine Tabellenausdrücke können auch rekursive Ausdrücke, wenn sie in gespeicherten Prozeduren verwendet werden, Referenzen zu lokalen Variablen oder Prozedurparametern enthalten. Die unten definierte Prozedur "best_routes" findet beispielsweise die kürzeste Route zwischen den zwei benannten Städten heraus.

```
CREATE PROCEDURE best_routes (
    IN initial VARCHAR(10),
    IN final   VARCHAR(10)
)
BEGIN
    WITH RECURSIVE
        trip ( route, destination, previous, distance, segments ) AS
    ( SELECT CAST( origin || ', ' || destination AS VARCHAR(256) ),
        destination, origin, distance, 1
      FROM travel
     WHERE origin = initial
      UNION ALL
        SELECT route || ', ' || v.destination,
            v.destination,          -- current endpoint
            v.origin,              -- previous endpoint
            t.distance + v.distance, -- total distance
            segments + 1           -- total number of segments
      FROM trip t JOIN travel v ON t.destination = v.origin
     WHERE v.destination <> initial -- Don't return to start
        AND v.destination <> t.previous -- Prevent backtracking
        AND v.origin      <> final      -- Stop at the end
        AND segments      <> ( SELECT count(*)/2 FROM travel ) -- TERMINATE RECURSION!
    )
    SELECT route, distance, segments FROM trip
   WHERE destination = final AND
        distance < 1.4 * ( SELECT MIN( distance )
                          FROM trip
                         WHERE destination = final )
   ORDER BY distance, segments, route;
END;
```

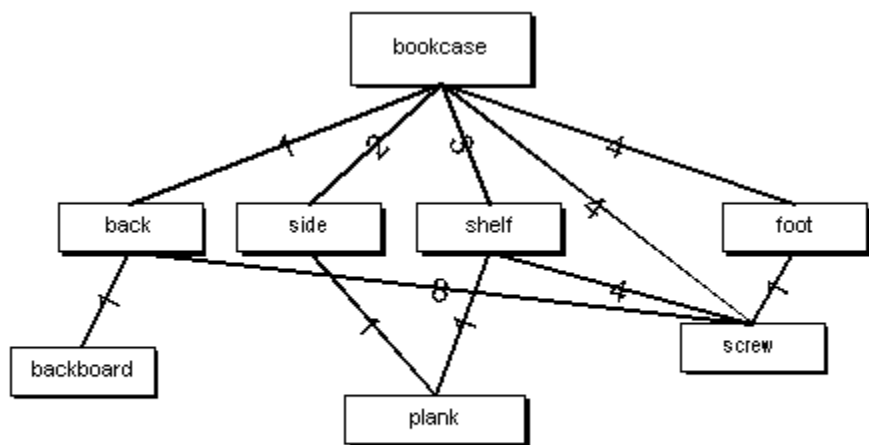
Die folgende Anweisung ruft die obige Prozedur auf.

```
CALL best_routes ( 'Pembroke', 'Kitchener' );
```

Stücklistenproblem

Das Stücklistenproblem ist eine klassische Anwendung einer Rekursion. In diesem Problem werden die Komponenten, die zum Zusammensetzen eines bestimmten Objekts notwendig sind, in einem Graphen dargestellt. Das Ziel ist es, diesen Graphen unter Verwendung einer Datenbanktabelle darzustellen, um dann die Gesamtzahl der notwendigen Elemente zu berechnen.

Der folgende Graph zum Beispiel stellt die Komponenten eines einfachen Bücherregals dar. Das Bücherregal (bookcase) besteht aus drei Fächern (shelves), einer Rückwand (back) und vier Füßen (feet), die mit vier Schrauben (screws) befestigt werden. Jedes Fach ist ein Brett (plank), das mit vier Schrauben befestigt wird. Die Rückwand ist ein weiteres Brett (backboard), das mit acht Schrauben befestigt wird.



Die Angaben in der unten stehenden Tabelle stellen die Ränder des Bücherregalgraphen dar. Die erste Spalte benennt eine Komponente, die zweite Spalte benennt eine der Unterkomponenten dieser Komponente, und die dritte Spalte legt fest, wie viele der Unterkomponenten erforderlich sind.

component	subcomponent	quantity
bookcase	back	1
bookcase	side	2
bookcase	shelf	3
bookcase	foot	4
bookcase	screw	4
back	backboard	1
back	screw	8
side	plank	1
shelf	plank	1
shelf	screw	4

Führen Sie die folgenden Anweisungen aus, um eine Buchregal-Tabelle (bookcase) zu erstellen und Komponenten- und Unterkomponenten-Daten einzufügen.

```
CREATE TABLE bookcase (  
  component VARCHAR(9),
```

```

        subcomponent    VARCHAR(9),
        quantity        INTEGER,
        PRIMARY KEY ( component, subcomponent )
    );
INSERT INTO bookcase
    SELECT 'bookcase', 'back',      1 UNION
    SELECT 'bookcase', 'side',     2 UNION
    SELECT 'bookcase', 'shelf',    3 UNION
    SELECT 'bookcase', 'foot',     4 UNION
    SELECT 'bookcase', 'screw',    4 UNION
    SELECT 'back',      'backboard', 1 UNION
    SELECT 'back',      'screw',      8 UNION
    SELECT 'side',      'plank',      1 UNION
    SELECT 'shelf',     'plank',      1 UNION
    SELECT 'shelf',     'screw',      4;

```

Führen Sie die folgende Anweisung aus, um eine Liste von Komponenten und Unterkomponenten sowie die Menge zu generieren, die für die Montage des Bücherregals erforderlich ist.

```

SELECT * FROM bookcase
ORDER BY component, subcomponent;

```

Führen Sie die folgende Anweisung aus, um eine Liste von Unterkomponenten und die Menge zu generieren, die für die Montage des Bücherregals erforderlich ist.

```

WITH RECURSIVE parts ( component, subcomponent, quantity ) AS
( SELECT component, subcomponent, quantity
  FROM bookcase WHERE component = 'bookcase'
  UNION ALL
  SELECT b.component, b.subcomponent, p.quantity * b.quantity
    FROM parts p JOIN bookcase b ON p.subcomponent = b.component )
SELECT subcomponent, SUM( quantity ) AS quantity
FROM parts
WHERE subcomponent NOT IN ( SELECT component FROM bookcase )
GROUP BY subcomponent
ORDER BY subcomponent;

```

Die Ergebnisse dieser Abfrage werden unten angezeigt.

subcomponent	quantity
backboard	1
foot	4
plank	5
screw	24

Oder Sie schreiben diese Abfrage um, um eine zusätzliche Stufe der Rekursion durchzuführen und die Unterabfrage in der Haupt-SELECT-Anweisung überflüssig zu machen. Die Ergebnisse der folgenden Abfrage sind mit jenen der vorherigen Abfrage identisch.

```

WITH RECURSIVE parts ( component, subcomponent, quantity ) AS
( SELECT component, subcomponent, quantity
  FROM bookcase WHERE component = 'bookcase'
  UNION ALL
  SELECT p.subcomponent, b.subcomponent,

```

```

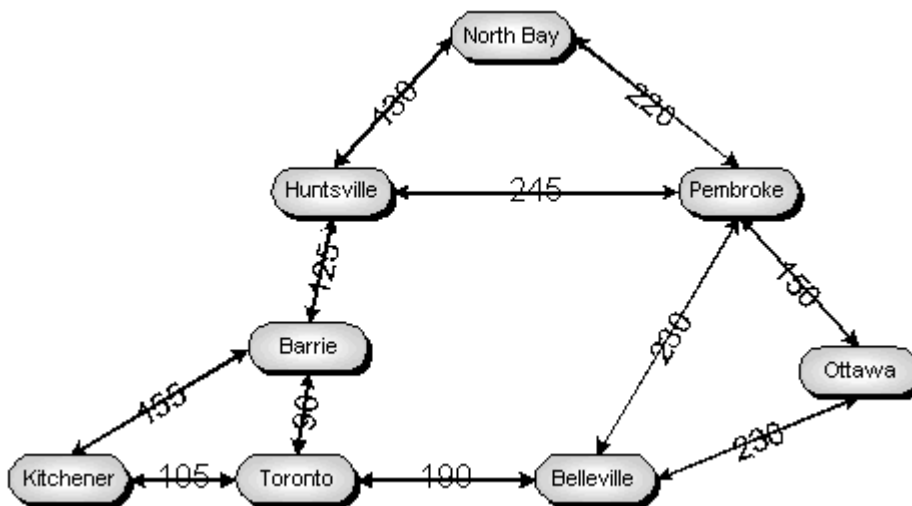
    IF b.quantity IS NULL
    THEN p.quantity
    ELSE p.quantity * b.quantity
    ENDIF
FROM parts p LEFT OUTER JOIN bookcase b
ON p.subcomponent = b.component
   WHERE p.subcomponent IS NOT NULL
)
SELECT component, SUM( quantity ) AS quantity
FROM parts
WHERE subcomponent IS NULL
GROUP BY component
ORDER BY component;

```

Das Problem der kürzesten Entfernung

Sie können rekursive allgemeine Tabellenausdrücke verwenden, um ideale Strecken in einem direktiven Graphen zu finden. Jede Zeile in einer Datenbank stellt eine gerichtete Kante dar. Jede Zeile spezifiziert einen Ausgangspunkt, ein Ziel und die Kosten der Reise vom Ausgangspunkt zum Ziel. Abhängig von der Problemstellung können die Kosten die Entfernung, die Reisezeit oder einen anderen Faktor darstellen. Die Rekursion ermöglicht es Ihnen, mögliche Routen durch diesen Graphen zu untersuchen. Aus der Menge der möglichen Routen können Sie dann jene auswählen, die Sie interessieren.

Betrachten Sie zum Beispiel die Problemstellung, einen idealen Weg zwischen den Städten Kitchener und Pembroke zu finden. Es gibt einige mögliche Routen, die Sie alle durch eine Menge von dazwischen liegenden Städten führen. Das Ziel ist es, die kürzesten Routen zu finden und sie miteinander zu vergleichen.



Als Erstes definieren Sie eine Tabelle, die die Kanten dieses Graphen darstellt, und fügen eine Zeile für jede Kante ein. Da alle Kanten dieses Graphen zwei Richtungen haben, müssen die Kanten, die die umgekehrte Richtung darstellen, ebenfalls eingefügt werden. Das wird erreicht, indem Sie die erste Menge von Zeilen auswählen, aber den Ausgangspunkt und das Ziel vertauschen. Eine Zeile muss

beispielsweise die Reise von Kitchener nach Toronto darstellen und eine andere die Reise von Toronto zurück nach Kitchener.

```
CREATE TABLE travel (
    origin      VARCHAR(10),
    destination VARCHAR(10),
    distance    INT,
    PRIMARY KEY ( origin, destination )
);
INSERT INTO travel
SELECT 'Kitchener', 'Toronto', 105 UNION
SELECT 'Kitchener', 'Barrie', 155 UNION
SELECT 'North Bay', 'Pembroke', 220 UNION
SELECT 'Pembroke', 'Ottawa', 150 UNION
SELECT 'Barrie', 'Toronto', 90 UNION
SELECT 'Toronto', 'Belleville', 190 UNION
SELECT 'Belleville', 'Ottawa', 230 UNION
SELECT 'Belleville', 'Pembroke', 230 UNION
SELECT 'Barrie', 'Huntsville', 125 UNION
SELECT 'Huntsville', 'North Bay', 130 UNION
SELECT 'Huntsville', 'Pembroke', 245;
INSERT INTO travel -- Insert the return trips
SELECT destination, origin, distance
FROM travel;
```

Die nächste Aufgabe ist es, die rekursiven allgemeinen Tabellenausdrücke zu schreiben. Da die Reise in Kitchener losgeht, beginnt die Anfangs-Unterabfrage, indem alle möglichen Strecken aus Kitchener heraus ausgewählt werden, zusammen mit den jeweiligen Entfernungen.

Die rekursive Unterabfrage erweitert diese Strecken. Bei jeder Strecke werden Abschnitte hinzugefügt, die von den Zielen der vorherigen Abschnitte ausgehen, wobei die Länge der neuen Abschnitte hinzuaddiert wird, um die aktuellen Gesamtkosten für die einzelnen Strecken zu ermitteln. Aus Gründen der Effizienz werden Strecken beendet, wenn sie eine der folgenden Bedingungen erfüllen:

- Die Strecke führt zum Ausgangspunkt zurück.
- Die Strecke führt zum vorherigen Standort zurück.
- Die Strecke erreicht das endgültige Ziel.

Im aktuellen Beispiel sollte keine Strecke nach Kitchener zurückführen, und alle Strecken sollten beendet werden, wenn sie Pembroke erreichen.

Wenn rekursive Abfragen verwendet werden, um zyklische Diagramme zu untersuchen, ist es wichtig zu überprüfen, dass sie korrekt beendet werden. In diesem Fall sind die oben stehenden Bedingungen unzureichend, da eine Route eine beliebig hohe Anzahl von Reisen hin und zurück zwischen zwei dazwischen liegenden Städten enthalten kann. Die untenstehende rekursive Abfrage garantiert ein Ende, indem die maximale Anzahl von Abschnitten bei jeder gegebenen Route auf sieben beschränkt wird.

Da der Zweck der Beispielsabfrage darin liegt, eine sinnvolle Route zu finden, wählt die Haupt-Abfrage nur jene Routen aus, die weniger als 50 % länger als die kürzeste Route sind.

```
WITH RECURSIVE
    trip ( route, destination, previous, distance, segments ) AS
( SELECT CAST( origin || ', ' || destination AS VARCHAR(256) ),
    destination, origin, distance, 1
```

```

FROM travel
WHERE origin = 'Kitchener'
  UNION ALL
SELECT route || ', ' || v.destination,
       v.destination,           -- current endpoint
       v.origin,                -- previous endpoint
       t.distance + v.distance, -- total distance
       segments + 1            -- total number of segments
FROM trip t JOIN travel v ON t.destination = v.origin
WHERE v.destination <> 'Kitchener' -- Don't return to start
  AND v.destination <> t.previous -- Prevent backtracking
  AND v.origin <> 'Pembroke'      -- Stop at the end
  AND segments                  -- TERMINATE RECURSION!
  < ( SELECT count(*)/2 FROM travel )
SELECT route, distance, segments FROM trip
WHERE destination = 'Pembroke' AND
  distance < 1.5 * ( SELECT MIN( distance )
                    FROM trip
                    WHERE destination = 'Pembroke' )
ORDER BY distance, segments, route;

```

Wenn diese Anweisung in Bezug auf die oben stehende Datenmenge ausgeführt wird, führt sie zu folgenden Ergebnissen:

route	distance	segments
Kitchener, Barrie, Huntsville, Pembroke	525	3
Kitchener, Toronto, Belleville, Pembroke	525	3
Kitchener, Toronto, Barrie, Huntsville, Pembroke	565	4
Kitchener, Barrie, Huntsville, North Bay, Pembroke	630	4
Kitchener, Barrie, Toronto, Belleville, Pembroke	665	4
Kitchener, Toronto, Barrie, Huntsville, North Bay, Pembroke	670	5
Kitchener, Toronto, Belleville, Ottawa, Pembroke	675	4

OLAP-Unterstützung

On-Line Analytical Processing (OLAP) ermöglicht die Durchführung komplexer Datenanalysen innerhalb einer einzelnen SQL-Anweisung, womit der Wert der Ergebnisse gesteigert und die Performance verbessert wird, da die Menge der Abfragen in der Datenbank reduziert wird. Die OLAP-Funktionalität wird durch den Einsatz von Erweiterungen für SQL-Anweisungen und Fensterfunktionen ermöglicht. Diese SQL-Erweiterungen und -Funktionen bieten die Möglichkeit, mehrdimensionale Datenanalysen, Data Mining, Zeitreihenanalysen, Trendanalysen, Kostenzuweisungen, Zielsuchläufe sowie Benachrichtigungen bei Ausnahmesituationen durchzuführen, und zwar kurz und prägnant und oft mit nur einer SQL-Anweisung.

- **Erweiterungen zur SELECT-Anweisung** Die Erweiterungen zur SELECT-Anweisung ermöglichen die Gruppierung der Eingabezeilen sowie die Analyse der Gruppen und sie stellen die Ergebnisse in der endgültigen Ergebnismenge bereit. Diese Erweiterungen umfassen Erweiterungen zur GROUP BY-Klausel (Unterklauseln GROUPING SETS, CUBE und ROLLUP) und zur WINDOW-Klausel.

Die Erweiterungen zur GROUP BY-Klausel ermöglichen Ihnen, die Eingabezeilen auf mehrere Arten zu partitionieren, und liefern ein Ergebnis, das alle verschiedenen Gruppen verkettet. Außerdem können Sie eine dünn besetzte, mehrdimensionale Ergebnismenge für Data Mining-Analysen erstellen (auch als **Datenwürfel** bezeichnet). Außerdem bieten die Erweiterungen Zeilen mit Zwischen- und Endsummen, um die Analyse zu vereinfachen.

Die WINDOW-Klausel wird mit Fensterfunktionen benutzt, um zusätzliche Analysemöglichkeiten für Gruppen von Eingabezeilen zu schaffen.

- **Fenster-Aggregatfunktionen** Die meisten Aggregatfunktionen unterstützen das Konzept eines konfigurierbaren, verschiebbaren **Fensters**, das sich durch die Eingabezeilen nach unten bewegt, während diese verarbeitet werden. Während sich das Fenster bewegt, können zusätzliche Berechnungen für die Daten im Fenster durchgeführt werden, was weitere Analysen auf eine Art ermöglicht, die effizienter ist als semantisch gleichwertige Selbst-Join-Abfragen oder korrelierte Unterabfragen.

Beispiel: Fenster-Aggregatfunktionen in Verbindung mit den Erweiterungen CUBE, ROLLUP und GROUPING SETS zur GROUP BY-Klausel bieten eine effiziente Möglichkeit zur Berechnung von Prozentsätzen, veränderlichen Durchschnitten und kumulierten Summen innerhalb einer einzelnen SQL-Anweisung. Andernfalls wären dafür Selbst-Joins, korrelierte Unterabfragen, temporäre Tabellen sowie Kombinationen dieser drei Elemente erforderlich.

Mit den Fenster-Aggregatfunktionen können Sie Daten wie etwa den sich im Quartal verändernden Durchschnitt des Dow Jones Industrial Average oder alle Angestellten und deren kumulative Gehälter nach Abteilungen berechnen. Sie können diese Funktionen auch zum Berechnen von Varianz, Standardabweichung, Korrelation und Regression einsetzen.

- **Fenster-Rangfunktionen** Mit den Fenster-Rangfunktionen können Sie SQL-Abfragen mit einzelnen Anweisungen bilden, um Informationen wie etwa die zehn Spitzenprodukte des Jahres nach Gesamtumsatz oder die obersten 5 % der Verkäufer, die an mindestens 15 unterschiedliche Unternehmen verkauft haben, abzurufen.

Siehe auch

- „GROUP BY-Klauselerweiterungen“ auf Seite 554
- „Fensterfunktionen“ auf Seite 564
- „Fenster-Rangfunktionen“ auf Seite 590
- „Fenster-Aggregatfunktionen“ auf Seite 571

Verbesserungen der OLAP-Performance

Um die OLAP-Performance zu verbessern, setzen Sie die Datenbankoption "optimization_workload" auf "OLAP", um den Optimierer anzuweisen, beim Prüfen seiner Möglichkeiten den Operator "Clustered

Group By Hash" zu berücksichtigen. Weiterhin können Sie Indizes für OLAP-Arbeitslasten optimieren, indem Sie beim Definieren des Indexes die Option FOR OLAP WORKLOAD benutzen. Diese Option veranlasst den Datenbankserver zum Durchführen bestimmter Optimierungen, wozu das Führen einer Statistik gehört, die vom "Clustered Group By Hash"-Operator verwendet wird und die maximale Seitenentfernung zwischen zwei Zeilen mit demselben Schlüssel betrifft.

Siehe auch

- „optimization_workload-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „CREATE INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

GROUP BY-Klauselerweiterungen

Die GROUP BY-Standardklausel einer SELECT-Anweisung ermöglicht Ihnen die Gruppierung von Zeilen in der Ergebnismenge gemäß den angegebenen Gruppierungsausdrücken. Wenn Sie beispielsweise GROUP BY columnA, columnB angeben, werden die Zeilen nach Kombinationen von eindeutigen Werten aus den Spalten A und B gruppiert. In der GROUP BY-Standardklausel reflektieren die Gruppen die Auswertung der Kombination aller angegebenen GROUP BY-Ausdrücke.

Sie können jedoch unterschiedliche Gruppierungen oder Untergruppierungen der Ergebnismenge festlegen. Möglicherweise wollen Sie die Ergebnisse Ihrer Daten gruppiert nach eindeutigen Werten von columnA und columnB anzeigen und anschließend noch einmal nach eindeutigen Werten von columnC zusammenfassen. Sie können dieses Ergebnis erreichen, indem Sie die Erweiterung GROUPING SETS zur GROUP BY-Klausel verwenden.

GROUP BY GROUPING SETS

Die GROUPING SETS-Klausel ist eine Erweiterung zur GROUP BY-Klausel einer SELECT-Anweisung. Die GROUPING SETS-Klausel ermöglicht die Gruppierung der Ergebnisse auf mehrere Arten, ohne mehrere SELECT-Anweisungen benutzen zu müssen. Damit wird die Reaktionszeit verringert und die Performance verbessert.

Die folgenden beiden Abfrageanweisungen sind beispielsweise semantisch gleichwertig: Die zweite Abfrage definiert die Gruppierungskriterien jedoch effizienter, da eine GROUP BY GROUPING SETS-Klausel verwendet wird.

Mehrfachgruppierungen mithilfe von SELECT-Anweisungen:

```
SELECT NULL, NULL, NULL, COUNT( * ) AS Cnt
FROM Customers
WHERE State IN ( 'MB' , 'KS' )
UNION ALL
SELECT City, State, NULL, COUNT( * ) AS Cnt
FROM Customers
WHERE State IN ( 'MB' , 'KS' )
GROUP BY City, State
UNION ALL
SELECT NULL, NULL, CompanyName, COUNT( * ) AS Cnt
FROM Customers
```

```
WHERE State IN ( 'MB' , 'KS' )
GROUP BY CompanyName;
```

Mehrfachgruppierungen mithilfe von GROUPING SETS:

```
SELECT City, State, CompanyName, COUNT( * ) AS Cnt
FROM Customers
WHERE State IN ( 'MB' , 'KS' )
GROUP BY GROUPING SETS( ( City, State ), ( CompanyName ), ( ) );
```

Beide Methoden liefern die gleichen Ergebnisse:

	City	State	CompanyName	Cnt
1	(NULL)	(NULL)	(NULL)	8
2	(NULL)	(NULL)	Cooper Inc.	1
3	(NULL)	(NULL)	Westend Dealers	1
4	(NULL)	(NULL)	Toto's Active Wear	1
5	(NULL)	(NULL)	North Land Trading	1
6	(NULL)	(NULL)	The Ultimate	1
7	(NULL)	(NULL)	Molly's	1
8	(NULL)	(NULL)	Overland Army Navy	1
9	(NULL)	(NULL)	Out of Town Sports	1
10	'Pembroke'	'MB'	(NULL)	4
11	'Petersburg'	'KS'	(NULL)	1
12	'Drayton'	'KS'	(NULL)	3

Die Zeilen 2 – 9 sind die Zeilen, die durch die Gruppierung nach "CompanyName" generiert wurden. Die Zeilen 10 – 12 sind die Zeilen, die durch die Gruppierung nach der Kombination von "City" und "State" generiert wurden. Die Zeile 1 ist die Gesamtsumme, die durch die leere Gruppierungskombination repräsentiert wird, welche durch ein Klammernpaar angegeben wird (). Die leere Gruppierungskombination repräsentiert eine einzelne Partition aller Zeilen in der Eingabe für die GROUP BY-Klausel.

Beachten Sie, dass NULL als Platzhalter für alle Ausdrücke verwendet wird, die nicht in einer Gruppierungskombination benutzt werden, da die Ergebnismengen kombinierbar sein müssen. Die Zeilen 2 – 9 ergeben sich beispielsweise aus der zweiten Gruppierungskombination in der Abfrage (CompanyName). Da diese Gruppierungskombination weder "City" noch "State" als Ausdrücke enthielt, enthalten die Werte für "City" und "State" für die Zeilen 2 – 9 den Platzhalter NULL, während die Werte in "CompanyName" die unterschiedlichen Werte enthalten, die in "CompanyName" vorkommen.

Da NULL-Werte als Platzhalter benutzt werden, besteht die Gefahr, diese Platzhalter mit den tatsächlichen NULL-Werten in den Daten zu verwechseln. Die GROUPING-Funktion hilft Ihnen bei der Unterscheidung des Platzhalters NULL vom Datenwert NULL.

Siehe auch

- [„Erkennen von NULL-Platzhaltern mit der GROUPING-Funktion“ auf Seite 562](#)

Beispiel

Das folgende Beispiel zeigt, wie Sie die von einer Abfrage gelieferten Ergebnisse mithilfe der GROUPING SETS-Klausel anpassen und mithilfe der ORDER BY-Klausel besser organisieren können. Die Abfrage liefert die Gesamtzahl der Bestellungen nach Quartal und die Gesamtzahl für das gesamte Jahr. Durch die Sortierung nach Jahr und Quartal sind die Ergebnisse einfacher zu verstehen:

```
SELECT Year( OrderDate ) AS Year,  
       Quarter( OrderDate ) AS Quarter,  
       COUNT ( * ) AS Orders  
FROM SalesOrders  
GROUP BY GROUPING SETS ( ( Year, Quarter ), ( Year ) )  
ORDER BY Year, Quarter;
```

Diese Abfrage liefert folgende Ergebnisse:

	Year	Quarter	Orders
1	2000	(NULL)	380
2	2000	1	87
3	2000	2	77
4	2000	3	91
5	2000	4	125
6	2001	(NULL)	268
7	2001	1	139
8	2001	2	119
9	2001	3	10

Die Zeilen 1 und 6 sind Zwischensummen der Bestellungen für 2000 und 2001. Die Zeilen 2 – 5 und 7 – 9 enthalten die Details für die Zwischensummen. Das heißt, sie enthalten die Bestellungen je Quartal.

In den Ergebnissen gibt es keine Gesamtsumme für alle Quartale aller Jahre. Um auch dieses Ergebnis zu erhalten, muss die Abfrage die leere Gruppierungsspezifikation "()" in der GROUPING SETS-Spezifikation enthalten.

Leere Gruppierungsspezifikation angeben

Wenn Sie die leere GROUPING SETS-Spezifikation "()" in der GROUP BY-Klausel benutzen, wird eine Gesamtsummenzeile für alle Elemente zurückgegeben, die in den Ergebnissen zusammengefasst werden. Wenn eine Gesamtsummenzeile vorhanden ist, enthalten die Werte für alle Gruppierungsausdrücke den Platzhalter NULL. Mithilfe der GROUPING-Funktion können Sie den Platzhalter NULL von den tatsächlichen NULL-Werten unterscheiden, die sich aus der Auswertung von Werten in den zugrunde liegenden Daten für die Zeile ergeben.

Doppelte Gruppierungskombinationen festlegen

In einer GROUPING SETS-Klausel können Sie doppelte Gruppierungsklauseln angeben. In einem solchen Fall enthält das Ergebnis der SELECT-Anweisung identische Zeilen.

Die folgende Abfrage enthält doppelte Gruppierungen:

```
SELECT City, COUNT( * ) AS Cnt
FROM Customers
WHERE State IN ( 'MB' , 'KS' )
GROUP BY GROUPING SETS( ( City ), ( City ) );
```

Diese Abfrage liefert folgende Ergebnisse. Die Zeilen 1 – 3 sind als Ergebnis der doppelten Gruppierungen identisch mit den Zeilen 4 – 6:

	City	Cnt
1	'Drayton'	3
2	'Petersburg'	1
3	'Pembroke'	4
4	'Drayton'	3
5	'Petersburg'	1
6	'Pembroke'	4

Empfehlungen zum Format

Die Gruppierungssyntax für eine GROUP BY GROUPING SETS-Klausel wird anders interpretiert als diejenige für eine einfache GROUP BY-Klausel. GROUP BY (X, Y) liefert z.B. Ergebnisse, die nach bestimmten Kombinationen der Werte von X und Y gruppiert sind. GROUP BY GROUPING SETS (X, Y) legt jedoch zwei individuelle Gruppierungskombinationen fest, und das Ergebnis der beiden Gruppierungskombinationen wird mit UNION vereinigt. Das heißt, die Ergebnisse werden nach X gruppiert und dann mit den gleichen Ergebnissen, die nach Y gruppiert sind, vereinigt.

Um ein gutes Format einzuhalten und um Zweideutigkeiten bei komplexen Ausdrücken zu vermeiden, sollten Sie Klammern um jede einzelne Gruppierungskombination in der Spezifikation setzen, wenn die Gefahr von Fehlern besteht. Beispiel: Wenngleich die beiden folgenden Anweisungen korrekt und semantisch gleichwertig sind, hat nur die zweite das empfohlene Format.

```
SELECT * FROM t GROUP BY GROUPING SETS ( X, Y );  
SELECT * FROM t GROUP BY GROUPING SETS( ( X ), ( Y ) );
```

Siehe auch

- [„Erkennen von NULL-Platzhaltern mit der GROUPING-Funktion“ auf Seite 562](#)

ROLLUP und CUBE als Abkürzung für GROUPING SETS

Die Verwendung von GROUPING SETS ist nützlich, wenn Sie mehrere unterschiedliche Datenpartitionen in einer einzelnen Ergebnismenge verketteten möchten. Falls Sie jedoch viele Gruppierungen festlegen müssen und Zwischensummen benötigen, können Sie die Erweiterungen ROLLUP und CUBE verwenden.

Die Klauseln ROLLUP und CUBE können als Abkürzungen für vordefinierte GROUPING SETS-Spezifikationen betrachtet werden.

ROLLUP ist gleichwertig mit dem Festlegen einer Reihe von GROUPING SET-Spezifikationen, beginnend mit der leeren Gruppierungskombination "()" und gefolgt von Gruppierungskombinationen, bei denen ein zusätzlicher Ausdruck mit dem vorherigen verkettet wird. Beispiel: Wenn es die drei Gruppierungsausdrücke a, b und c gibt und Sie die ROLLUP-Klausel angeben, hat dies die gleiche Wirkung wie die Angabe einer GROUPING SETS-Klausel mit der folgenden Menge (), (a), (a, b) und (a, b, c). Diese Konstruktion wird auch als hierarchische Gruppierung bezeichnet.

CUBE bietet sogar noch mehr Gruppierungen. CUBE hat die gleiche Wirkung wie die Verwendung aller möglichen GROUPING SETS. Beispiel: Wenn es die drei Gruppierungsausdrücke a, b und c gibt und Sie die CUBE-Klausel angeben, hat dies die gleiche Wirkung wie die Angabe einer GROUPING SETS-Klausel mit der folgenden Menge: (), (a), (a, b), (a, c), (b), (b, c), (c) und (a, b, c).

Wenn Sie ROLLUP oder CUBE angeben, benutzen Sie die GROUPING-Funktion, um den Platzhalter NULL in den Ergebnissen zu erkennen, der von den Zwischenergebnisseilen eingefügt wird, die von ROLLUP oder CUBE in der Ergebnismenge geliefert werden.

Siehe auch

- [„Erkennen von NULL-Platzhaltern mit der GROUPING-Funktion“ auf Seite 562](#)

ROLLUP-Klausel

Viele Anwendungen erfordern, dass die Zwischenergebnisse der Gruppierungsattribute der Reihe nach von links nach rechts berechnet werden. Dieses Muster wird als Hierarchie bezeichnet, da das Einfügen zusätzlicher Zwischenergebnisberechnungen zusätzliche Zeilen mit feinerer Granularität der Details erstellt. Sie können eine Hierarchie von Gruppierungsattributen festlegen, indem Sie mit dem Schlüsselwort ROLLUP eine ROLLUP-Klausel erstellen.

Eine Abfrage mit einer ROLLUP-Klausel erstellt eine hierarchische Reihe von Gruppierungskombinationen wie nachfolgend beschrieben. Falls die ROLLUP-Klausel n GROUP BY-Ausdrücke im Format (X_1, X_2, \dots, X_n) enthält, generiert die ROLLUP-Klausel $n + 1$ Gruppierungskombinationen wie folgt:

{(), (X1), (X1,X2), (X1,X2,X3), ... , (X1,X2,X3, ... , Xn)}

Beispiel

Die folgende Abfrage fasst die Bestellungen nach Jahr und Quartal zusammen und liefert die Ergebnismenge, die in der unten stehenden Tabelle gezeigt wird:

```
SELECT QUARTER( OrderDate ) AS Quarter,
       YEAR( OrderDate ) AS Year,
       COUNT( * ) AS Orders,
       GROUPING( Quarter ) AS GQ,
       GROUPING( Year ) AS GY
FROM SalesOrders
GROUP BY ROLLUP( Year, Quarter )
ORDER BY Year, Quarter;
```

Diese Abfrage liefert folgende Ergebnisse:

	Quarter	Year	Orders	GQ	GY
1	(NULL)	(NULL)	648	1	1
2	(NULL)	2000	380	1	0
3	1	2000	87	0	0
4	2	2000	77	0	0
5	3	2000	91	0	0
6	4	2000	125	0	0
7	(NULL)	2001	268	1	0
8	1	2001	139	0	0
9	2	2001	119	0	0
10	3	2001	10	0	0

Die erste Zeile in einer Ergebnismenge zeigt die Gesamtsumme (648) aller Bestellungen für alle Quartale in beiden Jahren.

Zeile 2 zeigt die gesamten Bestellungen (380) für 2000, während die Zeilen 3 – 6 die Zwischensummen für die Quartale dieses Jahres enthalten. Entsprechend zeigt Zeile 7 die gesamten Bestellungen (268) für 2001, während die Zeilen 8 – 10 die Zwischensummen für die Quartale dieses Jahres enthalten.

Beachten Sie, dass die von der GROUPING-Funktion gelieferten Werte benutzt werden können, um die Zwischensummenzeilen von der Zeile zu unterscheiden, die das Gesamtergebnis enthält. Für die Zeilen 2 und 7 wird durch NULL in der Spalte "Quarter" und den Wert 1 in der Spalte "GQ" (Zwischensumme je Quartal) angezeigt, dass die Zeile die Summe aller Bestellungen aller Quartale (pro Jahr) enthält.

Entsprechend wird in Zeile 1 durch NULL in den Spalten "Quarter" und "Year" sowie den Wert 1 in den Spalten "GQ" und "GY" angezeigt, dass die Zeile die Summe aller Bestellungen aller Quartale und aller Jahre enthält.

Unterstützung für WITH ROLLUP-Syntax in Transact-SQL

Sie können auch die Transact-SQL-kompatible Syntax WITH ROLLUP benutzen, um die gleichen Ergebnisse zu erzielen wie mit GROUP BY ROLLUP. Die Syntax ist jedoch leicht unterschiedlich und Sie können in der Syntax nur eine einfache GROUP BY-Ausdrucksliste angeben.

Die folgende Abfrage liefert das gleiche Ergebnis wie im vorherigen Beispiel zu GROUP BY ROLLUP:

```
SELECT QUARTER( OrderDate ) AS Quarter,
       YEAR( OrderDate ) AS Year,
       COUNT( * ) AS Orders,
       GROUPING( Quarter ) AS GQ,
       GROUPING( Year ) AS GY
FROM SalesOrders
GROUP BY Year, Quarter WITH ROLLUP
ORDER BY Year, Quarter;
```

Siehe auch

- „GROUP BY-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

CUBE-Klausel

Als Alternative zu den hierarchischen Gruppierungsmustern der ROLLUP-Klausel können Sie auch einen Datenwürfel erstellen. Dies ist eine n -dimensionale Zusammenfassung der Eingabedaten, wobei durch die CUBE-Klausel jede mögliche Kombination von GROUP BY-Ausdrücken verwendet wird. Das Ergebnis der CUBE-Klausel ist eine Produktmenge aller möglichen Kombinationen von Elementen aus jeder Wertegruppe. Dies kann bei komplexen Datenanalysen sehr hilfreich sein.

Wenn n GROUPING-Ausdrücke im Format (X_1, X_2, \dots, X_n) in einer CUBE-Klausel enthalten sind, generiert CUBE 2^n Gruppierungskombinationen wie folgt:

$$\{(), (X_1), (X_1, X_2), (X_1, X_2, X_3), \dots, (X_1, X_2, X_3, \dots, X_n), (X_2), (X_2, X_3), (X_2, X_3, X_4), \dots, (X_2, X_3, X_4, \dots, X_n), \dots, (X_n)\}.$$

Beispiel

Die folgende Abfrage fasst die Bestellungen nach Jahr und Quartal zusammen sowie nach Quartalen innerhalb eines Jahres, und liefert die Ergebnismenge, die in der unten stehenden Tabelle gezeigt wird:

```
SELECT QUARTER( OrderDate ) AS Quarter,
       YEAR( OrderDate ) AS Year,
       COUNT( * ) AS Orders,
       GROUPING( Quarter ) AS GQ,
       GROUPING( Year ) AS GY
FROM SalesOrders
GROUP BY CUBE ( Year, Quarter )
ORDER BY Year, Quarter;
```

Diese Abfrage liefert folgende Ergebnisse:

	Quarter	Year	Orders	GQ	GY
1	(NULL)	(NULL)	648	1	1
2	1	(NULL)	226	0	1
3	2	(NULL)	196	0	1
4	3	(NULL)	101	0	1
5	4	(NULL)	125	0	1
6	(NULL)	2000	380	1	0
7	1	2000	87	0	0
8	2	2000	77	0	0
9	3	2000	91	0	0
10	4	2000	125	0	0
11	(NULL)	2001	268	1	0
12	1	2001	139	0	0
13	2	2001	119	0	0
14	3	2000	10	0	0

Die erste Zeile in der Ergebnismenge zeigt die Gesamtsumme (648) aller Bestellungen für alle Quartale der Jahre 2000 und 2001.

In den Zeilen 2 – 5 werden die Bestellungen für jedes Jahr nach Quartalen zusammengefasst.

In den Zeilen 6 und 11 werden die gesamten Bestellungen für 2000 und 2001 gezeigt.

In den Zeilen 7 – 10 und 12 – 14 werden die Quartalssummen der Jahre 2000 und 2001 aufgeführt.

Beachten Sie, dass die von der GROUPING-Funktion gelieferten Werte benutzt werden können, um die Zwischensummenzeilen von der Zeile zu unterscheiden, die das Gesamtergebnis enthält. Für die Zeilen 6 und 11 wird durch NULL in der Spalte "Quarter" und den Wert 1 in der Spalte "GQ" (Zwischensumme je Quartal) angezeigt, dass die Zeile die Summe aller Bestellungen aller Quartale des Jahres enthält.

Hinweis

Die durch CUBE generierte Ergebnismenge kann sehr groß sein, da CUBE eine exponentielle Anzahl von Gruppierungskombinationen bildet. Aus diesem Grund werden keine GROUP BY-Klauseln mit mehr als 64 GROUP BY-Ausdrücken unterstützt. Wenn eine Anweisung diese Grenze überschreitet, schlägt sie fehl und gibt SQLCODE -944 (SQLSTATE 42WA1) zurück.

Unterstützung für WITH CUBE-Syntax in Transact-SQL

Sie können auch die Transact-SQL-kompatible Syntax WITH CUBE benutzen, um die gleichen Ergebnisse zu erzielen wie mit GROUP BY CUBE. Die Syntax ist jedoch leicht unterschiedlich und Sie können in der Syntax nur eine einfache GROUP BY-Ausdrucksliste angeben.

Die folgende Abfrage liefert das gleiche Ergebnis wie im vorherigen Beispiel zu GROUP BY CUBE:

```
SELECT QUARTER( OrderDate ) AS Quarter,
       YEAR( OrderDate ) AS Year,
       COUNT( * ) AS Orders,
       GROUPING( Quarter ) AS GQ,
       GROUPING( Year ) AS GY
FROM SalesOrders
GROUP BY Year, Quarter WITH CUBE
ORDER BY Year, Quarter;
```

Siehe auch

- „Zu viele Ausdrücke in der GROUP BY-Liste für ROLLUP-, CUBE- oder GROUPING SETS-Vorgang“ [[Fehlermeldungen](#)]
- „GROUP BY-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Erkennen von NULL-Platzhaltern mit der GROUPING-Funktion

Die von ROLLUP oder CUBE erstellten Summen- und Zwischensummenzeilen enthalten den Platzhalter NULL in den Spalten, die in der SELECT-Liste angegeben wurden, die aber nicht für die Gruppierung benutzt wurden. Wenn Sie die Ergebnisse prüfen, können Sie nicht unterscheiden, ob es sich bei NULL in einer Zwischensummenzeile um den Platzhalter NULL handelt oder ob sich NULL aus der Auswertung der zugrunde liegenden Daten für die Zeile ergeben hat. Daher ist es auch schwierig, zwischen einer Detailzeile, einer Zwischensummenzeile und einer Gesamtsummenzeile zu unterscheiden.

Mithilfe der GROUPING-Funktion können Sie zwischen dem Platzhalter NULL und NULL unterscheiden, die sich aus den zugrunde liegenden Daten ergeben haben. Falls Sie aus der Gruppierungskombinationsspezifikation eine GROUPING-Funktion mit einem *group-by-expression* spezifizieren, liefert die Funktion eine 1, wenn es sich um den Platzhalter NULL handelt, und eine 0, wenn es sich um einen Wert (möglicherweise NULL) handelt, der in den zugrunde liegenden Daten für diese Zeile vorhanden ist.

Die folgende Abfrage liefert beispielsweise die Ergebnismenge, die in der unten stehenden Tabelle gezeigt wird:

```
SELECT Employees.EmployeeID AS Employee,
       YEAR( OrderDate ) AS Year,
       COUNT( SalesOrders.ID ) AS Orders,
       GROUPING( Employee ) AS GE,
       GROUPING( Year ) AS GY
FROM Employees LEFT OUTER JOIN SalesOrders
ON Employees.EmployeeID = SalesOrders.SalesRepresentative
WHERE Employees.Sex IN ( 'F' )
AND Employees.State IN ( 'TX' , 'NY' )
GROUP BY GROUPING SETS ( ( Year, Employee ), ( Year ), ( ) )
ORDER BY Year, Employee;
```

Diese Abfrage liefert folgende Ergebnisse:

	Employees	Year	Orders	GE	GY
1	(NULL)	(NULL)	54	1	1
2	(NULL)	(NULL)	0	1	0
3	102	(NULL)	0	0	0
4	390	(NULL)	0	0	0
5	1062	(NULL)	0	0	0
6	1090	(NULL)	0	0	0
7	1507	(NULL)	0	0	0
8	(NULL)	2000	34	1	0
9	667	2000	34	0	0
10	(NULL)	2001	20	1	0
11	667	2001	20	0	0

In diesem Beispiel enthält Zeile 1 die Gesamtsumme der Bestellungen (54), da die leere Gruppierungskombination "()" angegeben wurde. Beachten Sie, dass sowohl "GE" als auch "GY" eine 1 enthalten, womit angezeigt wird, dass es sich bei NULL in den Spalten "Employees" und "Year" um den Platzhalter NULL für die Spalten "Employees" und "Year" handelt.

Zeile 2 ist eine Zwischensummenzeile. Die 1 in der Spalte "GE" zeigt an, dass NULL in der Spalte "Employee" der Platzhalter NULL ist. Die 0 in der Spalte "GY" zeigt an, dass NULL in der Spalte "Year" das Ergebnis der Auswertung der zugrunde liegenden Daten und nicht der Platzhalter NULL ist. In diesem Fall enthält diese Zeile diejenigen Angestellten, die keine Bestellungen haben.

Die Zeilen 3 – 7 zeigen die Gesamtzahl der Bestellungen pro Mitarbeiter, bei denen "Year" den Wert NULL enthält. Dabei handelt es sich um die weiblichen Angestellten, die in Texas und New York leben und die keine Bestellungen haben. Dies sind die Detailzeilen für Zeile 2, d.h., Zeile 2 ist eine Summenzeile der Zeilen 3 – 7.

Zeile 8 ist eine Zwischensummenzeile, welche die Gesamtzahl der Bestellungen aller Angestellten im Jahr 2000 enthält. Zeile 9 ist die einzelne Detailzeile für Zeile 8.

Zeile 10 ist eine Zwischensummenzeile, welche die Gesamtzahl der Bestellungen aller Angestellten im Jahr 2001 enthält. Zeile 11 ist die einzelne Detailzeile für Zeile 10.

Siehe auch

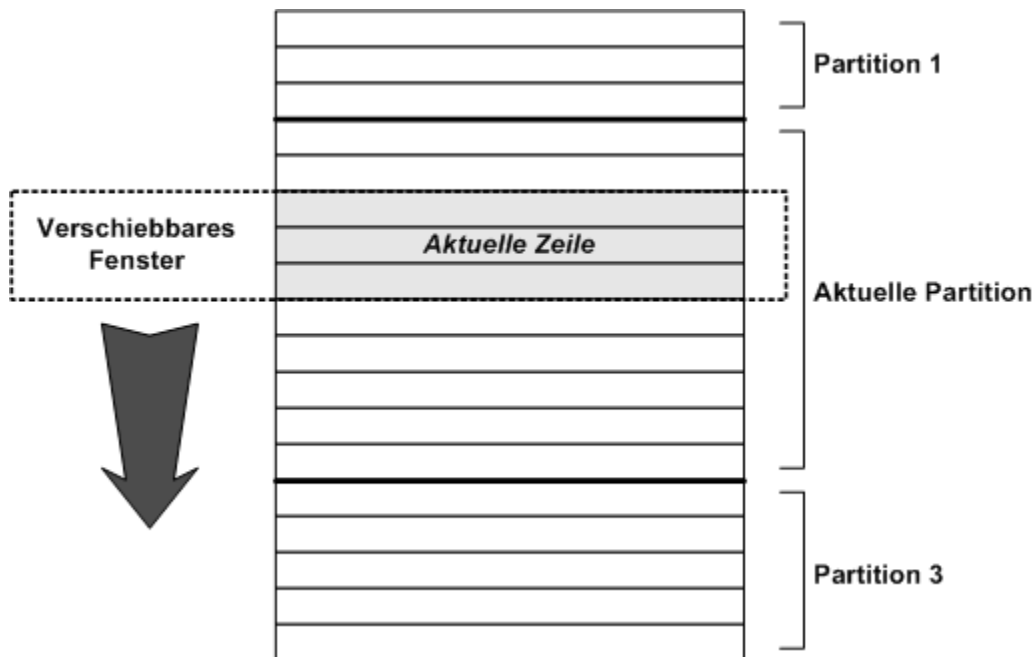
- „GROUPING-Funktion [Aggregat]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Fensterfunktionen

Die OLAP-Funktionalität enthält das Konzept eines verschiebbaren **Fensters**, das sich durch die Eingabezeilen nach unten bewegt, während diese verarbeitet werden. Während sich das Fenster bewegt, können zusätzliche Berechnungen für die Daten im Fenster durchgeführt werden, was weitere Analysen auf eine Art ermöglicht, die effizienter ist als semantisch gleichwertige Selbst-Join-Abfragen oder korrelierte Unterabfragen.

Sie konfigurieren die Grenzen des Fensters aufgrund der Informationen, die Sie aus den Daten extrahieren möchten. Ein Fenster kann aus einer, vielen oder allen Zeilen der Eingabedaten bestehen, die entsprechend der Gruppierungsspezifikationen aufgeteilt sind, die in der Fensterdefinition angegeben wurden. Das Fenster bewegt sich durch die Eingabedaten nach unten und umfasst die Zeilen, die zum Durchführen der angeforderten Berechnungen gebraucht werden.

Das nachstehende Diagramm zeigt die Bewegung des Fensters, während Eingabezeilen verarbeitet werden. Die Datenpartitionen reflektieren die Gruppierung von Eingabezeilen, die in der Fensterdefinition angegeben wurde. Wenn keine Gruppierung festgelegt wurde, werden alle Eingabezeilen als eine Partition betrachtet. Die Länge des Fensters (das ist die Anzahl der Zeilen im Fenster) und der Offset des Fensters relativ zur aktuellen Zeile reflektieren die Grenzen, die in der Fensterdefinition festgelegt wurden.



Fensterdefinitionen

Sie können die SQL-Fenstererweiterungen verwenden, um die Grenzen eines Fensters und die Partitionierung und Reihenfolge der Eingabezeilen zu konfigurieren. Logischerweise werden, im Rahmen der Semantik der Ergebnisberechnung einer Abfragespezifikation, die Partitionen erstellt, nachdem die durch die GROUP BY-Klausel definierten Gruppen erstellt wurden, jedoch vor der Auswertung der

endgültigen SELECT-Liste und der ORDER BY-Klausel der Abfrage. Die Reihenfolge der Auswertung der Klauseln in einer SQL-Anweisung lautet:

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. WINDOW
6. DISTINCT
7. ORDER BY

Wenn Sie Ihre Abfrage formulieren, sollten Sie die Auswirkung der Reihenfolge der Auflösung berücksichtigen. Es darf beispielsweise kein Prädikat für einen Ausdruck, der eine Fensterfunktion referenziert, in demselben SELECT-Abfrageblock verwendet werden. Wenn Sie allerdings den Abfrageblock in eine abgeleitete Tabelle platzieren, können Sie ein Prädikat für die abgeleitete Tabelle angeben. Die folgende Abfrage schlägt mit einer Meldung fehl, die angibt, dass der Fehler von einem Prädikat verursacht wurde, das für eine Fensterfunktion angegeben wird:

```
SELECT DepartmentID, Surname, StartDate, Salary,
       SUM( Salary ) OVER ( PARTITION BY DepartmentID
                           ORDER BY StartDate
                           RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW ) AS
"Sum_Salary"
FROM Employees
WHERE State IN ( 'CA', 'UT', 'NY', 'AZ' )
  AND DepartmentID IN ( '100', '200' )
GROUP BY DepartmentID, Surname, StartDate, Salary
HAVING Salary > 0 AND "Sum_Salary" > 200
ORDER BY DepartmentID, StartDate;
```

Verwenden Sie eine abgeleitete Tabelle (DT, derived table) und geben Sie für sie ein Prädikat an, um die gewünschten Ergebnisse zu erhalten.

```
SELECT * FROM ( SELECT DepartmentID, Surname, StartDate, Salary,
                      SUM( Salary ) OVER ( PARTITION BY DepartmentID
                                           ORDER BY StartDate
                                           RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW )
AS "Sum_Salary"
FROM Employees
WHERE State IN ( 'CA', 'UT', 'NY', 'AZ' )
  AND DepartmentID IN ( '100', '200' )
GROUP BY DepartmentID, Surname, StartDate, Salary
HAVING Salary > 0
ORDER BY DepartmentID, StartDate ) AS DT
WHERE DT.Sum_Salary > 200;
```

Da die Fensterpartitionierung nach einem GROUP BY-Operator erfolgt, stehen die Ergebnisse aller Aggregatfunktionen wie etwa SUM, AVG oder VARIANCE den Berechnungen zur Verfügung, die für

eine Partition durchgeführt werden. Fenster bieten daher eine weitere Möglichkeit, Gruppierungen und Sortierungen durchzuführen, zusätzlich zu den GROUP BY- und ORDER BY-Klauseln einer Abfrage.

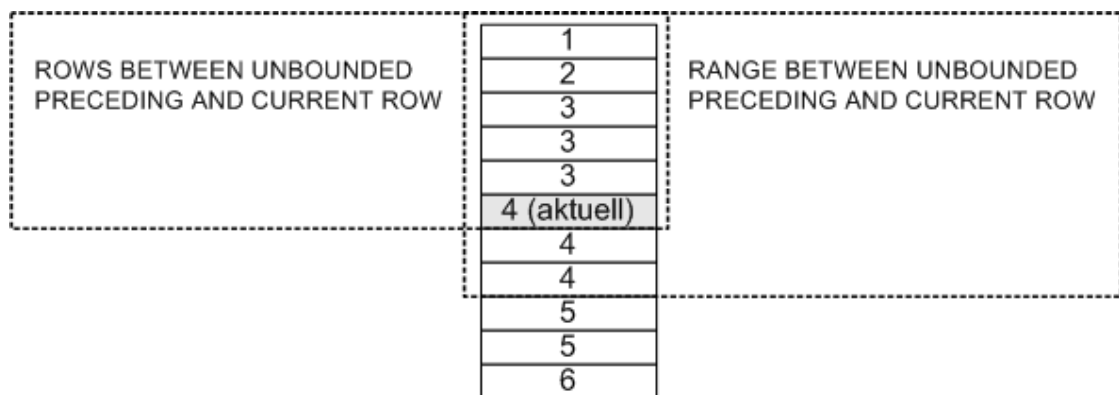
Eine Fensterspezifikation definieren

Bei der Definition eines Fensters, in dem eine Fensterfunktion ausgeführt wird, geben Sie eine oder mehrere der folgenden Einstellungen an:

- Partitionierung (PARTITION BY-Klausel)** Die PARTITION BY-Klausel legt fest, wie Eingabezeilen gruppiert werden. Ohne diese Einstellung wird die gesamte Eingabe als eine einzige Partition behandelt. Je nach Einstellung kann eine Partition aus einer, mehreren oder allen Eingabezeilen bestehen. Daten aus zwei Partitionen werden niemals vermischt. Dies bedeutet, wenn ein Fenster die Grenze zwischen zwei Partitionen erreicht, schließt es die Verarbeitung der Daten in der einen Partition ab, bevor es mit der Verarbeitung der Daten in der nächsten Partition beginnt. Die Größe eines Fensters kann am Beginn und am Ende einer Partition unterschiedlich sein, abhängig davon, wie die Grenzen für das Fenster definiert sind.
- Sortierung (ORDER BY-Klausel)** Die ORDER BY-Klausel legt fest, wie die Eingabezeilen sortiert werden, bevor sie durch die Fensterfunktion verarbeitet werden. Die ORDER BY-Klausel ist nur erforderlich, wenn Sie die Grenzen mithilfe einer RANGE-Klausel festlegen, oder wenn eine Rangfunktion auf das Fenster verweist. Andernfalls ist die ORDER BY-Klausel optional. Wenn sie weggelassen wird, verarbeitet der Datenbankserver die Eingabezeilen auf die effizienteste Art.
- Grenzen (RANGE- und ROWS-Klauseln)** Die aktuelle Zeile bietet den Referenzpunkt zum Bestimmen der Start- und Endzeilen eines Fensters. Sie können die RANGE- und ROWS-Klauseln der Fensterdefinition verwenden, um diese Grenzen festzulegen. RANGE definiert das Fenster als *Bereich von Datenwerten*, ausgehend vom Wert in der aktuellen Zeile. Wenn Sie daher RANGE angeben, müssen Sie auch eine ORDER BY-Klausel angeben, da Bereichsberechnungen erfordern, dass die Daten sortiert sind.

ROWS definiert das Fenster als *Anzahl von Zeilen*, ausgehend von der aktuellen Zeile.

Da RANGE eine Menge von Zeilen als Bereich von Datenwerten definiert, können die Zeilen, die in ein RANGE-Fenster einbezogen werden, Zeilen jenseits der aktuellen Zeile enthalten. Dies unterscheidet sich davon, wie ROWS gehandhabt wird. Das folgende Diagramm veranschaulicht den Unterschied zwischen den Klauseln ROWS und RANGE:



Innerhalb der Klauseln ROWS und RANGE können Sie (optional) die Start- und Endzeile des Fensters relativ zur aktuellen Zeile angeben. Um dies zu tun, verwenden Sie die Klauseln PRECEDING, BETWEEN und FOLLOWING. Diese Klauseln verwenden Ausdrücke und die Schlüsselwörter UNBOUNDED und CURRENT ROW. Wenn bei einem Fenster keine Grenzen definiert sind, werden die Standard-Fenstergrenzen folgendermaßen festgelegt:

- Wenn die Fensterspezifikation eine ORDER BY-Klausel enthält, ist dies gleichwertig mit der Angabe von RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW.
- Wenn die Fensterspezifikation keine ORDER BY-Klausel enthält, ist dies gleichwertig mit der Angabe von ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING.

Die folgende Tabelle enthält einige Beispiel-Fenstergrenzen und eine Beschreibung der Zeilen, die sie enthalten:

Spezifikation	Bedeutung
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW	Am Anfang der Partition starten und mit der aktuellen Zeile aufhören. Verwenden Sie dies, wenn Sie kumulative Ergebnisse berechnen, etwa kumulative Summen.
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING	Alle Zeilen in der Partition verwenden. Verwenden Sie dies, wenn der Wert einer Aggregatfunktion für jede Zeile einer Partition identisch sein soll.
ROWS BETWEEN x PRECEDING AND y FOLLOWING	Ein bewegliches Fenster fester Größe erstellen, beginnend bei einem Abstand von x von der aktuellen Zeile und endend bei einem Abstand von y von der aktuellen Zeile (inklusive). Verwenden Sie dieses Beispiel, wenn Sie einen veränderlichen Durchschnitt oder die Wertedifferenz zwischen angrenzenden Zeilen berechnen wollen. Bei einem beweglichen Fenster mit mehr als einer Zeile treten beim Berechnen der ersten und der letzten Zeile in der Partition Nullwerte auf. Der Grund dafür ist, dass es bei der ersten oder letzten Zeile der Partition keine vorhergehenden bzw. folgenden Zeilen gibt, die in der Berechnung verwendet werden können. Daher wird stattdessen NULL verwendet.
ROWS BETWEEN CURRENT ROW AND CURRENT ROW	Ein Fenster aus einer Zeile, nämlich der aktuellen Zeile.

Spezifikation	Bedeutung
RANGE BETWEEN 5 PRECEDING AND 5 FOLLOWING	Ein Fenster erstellen, das auf den Werten in den Zeilen basiert. Beispiel: Für die aktuelle Zeile enthält die Spalte, die in der ORDER BY-Klausel angegeben wurde, den Wert 10. Wenn Sie die Fenstergröße mit RANGE BETWEEN 5 PRECEDING AND 5 FOLLOWING festlegen, wird das Fenster so groß eingerichtet, dass die erste Zeile eine 5 in dieser Spalte enthält und die letzte Zeile im Fenster eine 15. Wenn sich das Fenster in der Partition nach unten bewegt, kann es größer oder kleiner werden, da es sich in der Größe der Bereichsspezifikation anpasst.

Formulieren Sie Ihre Fensterspezifikation so explizit wie möglich. Ansonsten geben die Standardwerte möglicherweise nicht die Ergebnisse zurück, die Sie erwarten.

Verwenden Sie die RANGE-Klausel, um Probleme zu vermeiden, die durch Lücken bei der Eingabe für eine Fensterfunktion bewirkt werden, wenn die Wertemenge nicht kontinuierlich ist. Wenn Fenstergrenzen mit einer RANGE-Klausel festgelegt werden, behandelt der Datenbankserver automatisch angrenzende Zeilen und Zeilen mit mehrfach vorhandenen Werten.

RANGE verwendet Ganzzahlen ohne Vorzeichen. Eine Kürzung des RANGE-Ausdrucks kann abhängig von der Domäne des ORDER BY-Ausdrucks und der Domäne des Wertes erfolgen, der in der RANGE-Klausel angegeben wurde.

Legen Sie die Grenzen eines Fensters nicht fest, wenn Sie eine Rang- oder Zeilennummerierungsfunktion benutzen.

Fensterdefinition: Inlining unter Verwendung der Klauseln OVER und WINDOW

Es gibt drei Methoden, ein Fenster zu definieren:

- Inline (innerhalb der OVER-Klausel einer Fensterfunktion)
- In einer WINDOW-Klausel
- Teilweise inline und teilweise in einer WINDOW-Klausel

Einige Methoden unterliegen allerdings Beschränkungen, wie in den folgenden Abschnitten beschrieben.

Inline-Definition (innerhalb der OVER-Klausel einer Fensterfunktion)

Eine Fensterdefinition kann in die OVER-Klausel einer Fensterfunktion eingefügt werden. Dies wird als *Inline*-Definieren des Fensters bezeichnet.

Beispiel: Die folgende Anweisung fragt aus der SQL Anywhere-Beispieldatenbank alle Produkte ab, die im Juli und August 2001 ausgeliefert wurden, und die kumulativen Auslieferungsmengen nach Lieferdatum. Das Fenster wird inline definiert.

```
SELECT p.ID, p.Description, s.Quantity, s.ShipDate,  
       SUM( s.Quantity ) OVER ( PARTITION BY s.ProductID
```

```

ORDER BY s.ShipDate
ROWS BETWEEN UNBOUNDED PRECEDING
AND CURRENT ROW ) AS Cumulative_qty
FROM SalesOrderItems s JOIN Products p
ON ( s.ProductID = p.ID )
WHERE s.ShipDate BETWEEN '2001-07-01' AND '2001-08-31'
ORDER BY p.ID;

```

Diese Abfrage liefert folgende Ergebnisse:

	ID	Description	Quantity	ShipDate	Cumulative_qty
1	301	V-neck	24	2001-07-16	24
2	302	Crew Neck	60	2001-07-02	60
3	302	Crew Neck	36	2001-07-13	96
4	400	Cotton Cap	48	2001-07-05	48
5	400	Cotton Cap	24	2001-07-19	72
6	401	Wool Cap	48	2001-07-09	48
7	500	Cloth Visor	12	2001-07-22	12
8	501	Plastic Visor	60	2001-07-07	60
9	501	Plastic Visor	12	2001-07-12	72
10	501	Plastic Visor	12	2001-07-22	84
11	601	Zippered Sweatshirt	60	2001-07-19	60
12	700	Cotton Shorts	24	2001-07-26	24

In diesem Beispiel erfolgt die Berechnung der Fensterfunktion SUM nach dem Join der beiden Tabellen und der Anwendung der WHERE-Klausel der Abfrage. Die Abfrage wird folgendermaßen verarbeitet:

1. Die Eingabezeilen werden aufgrund des Wertes in "ProductID" partitioniert (gruppiert).
2. Innerhalb jeder Partition werden die Zeilen aufgrund des Wertes in "ShipDate" sortiert.
3. Für jede Zeile in der Partition wird die SUM-Funktion auf die Mengenwerte angewendet, wobei ein verschiebbares Fenster benutzt wird, das die Zeilen von der ersten (sortierten) Zeile jeder Partition bis zur aktuellen Zeile enthält.

WINDOW-Klausel-Definition

Eine alternative Methode für die obige Abfrage besteht in der Verwendung einer WINDOW-Klausel, um das Fenster getrennt von den Funktionen, die es benutzen, zu definieren, und im Verweisen des Fensters aus der OVER-Klausel jeder Funktion.

In diesem Beispiel erstellt die WINDOW-Klausel ein Fenster namens "Cumulative", partitioniert die Daten nach "ProductID" und sortiert sie nach "ShipDate". Die SUM-Funktion verweist in ihrer OVER-Klausel auf das Fenster und definiert seine Größe mit einer ROWS-Klausel.

```
SELECT p.ID, p.Description, s.Quantity, s.ShipDate,
       SUM( s.Quantity ) OVER ( Cumulative
                               ROWS BETWEEN UNBOUNDED PRECEDING
                               AND CURRENT ROW ) AS cumulative_qty
FROM SalesOrderItems s
JOIN Products p ON ( s.ProductID = p.ID )
WHERE s.ShipDate BETWEEN '2001-07-01' AND '2001-08-31'
WINDOW Cumulative AS ( PARTITION BY s.ProductID ORDER BY s.ShipDate )
ORDER BY p.ID;
```

Wenn die Syntax der WINDOW-Klausel verwendet wird, gelten folgende Einschränkungen:

- Falls eine PARTITION BY-Klausel angegeben wird, muss sie sich innerhalb der WINDOW-Klausel befinden.
- Falls eine ROWS- oder RANGE-Klausel angegeben wird, muss sie sich innerhalb der OVER-Klausel der referenzierenden Funktion befinden.
- Falls eine ORDER BY-Klausel für das Fenster angegeben wird, kann sie sich entweder in der WINDOW-Klausel oder in der OVER-Klausel der referenzierenden Funktion befinden, nicht jedoch in beiden.
- Die WINDOW-Klausel muss sich vor der ORDER BY-Klausel der SELECT-Anweisung befinden.

Kombination aus Inline- und WINDOW-Klausel-Definition

Sie können einen Teil der Fensterdefinition inline setzen und dann den Rest in der WINDOW-Klausel definieren. Zum Beispiel:

```
AVG() OVER ( windowA
            ORDER BY expression )...
...
WINDOW windowA AS ( PARTITION BY expression )
```

Wenn Sie die Fensterdefinition auf diese Weise aufteilen, gelten die folgenden Beschränkungen:

- Sie können keine PARTITION BY-Klausel in der Fensterfunktionssyntax verwenden.
- Sie können eine ORDER BY-Klausel entweder in der Fensterfunktionssyntax oder in der WINDOW-Klausel verwenden, aber nicht in beiden.
- Sie können keine RANGE- oder ROWS-Klausel in die WINDOW-Klausel aufnehmen.

Siehe auch

- „WINDOW-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Fenster-Aggregatfunktionen“ auf Seite 571
- „Fenster-Rangfunktionen“ auf Seite 590
- „Fensterdefinitionen“ auf Seite 564

Fensterfunktionen in SQL Anywhere

Funktionen, mit denen Sie analytische Vorgänge für eine Gruppe von Eingabezeilen durchführen können, werden als Fensterfunktionen bezeichnet. Alle Rangfolgefunktionen und die meisten Aggregatfunktionen sind **Fensterfunktionen**. Sie können diese Funktionen einsetzen, um zusätzliche Analysen Ihrer Daten durchzuführen. Dazu werden die Eingabezeilen vor ihrer Verarbeitung partitioniert und sortiert. Anschließend werden die Zeilen in einem Fenster mit konfigurierbarer Größe verarbeitet, das sich durch die Eingabezeilen bewegt.

Es gibt drei Arten von Fensterfunktionen: Fenster-Aggregatfunktionen, Fenster-Rangfunktionen und Zeilennummerierungsfunktionen.

Fenster-Aggregatfunktionen

Fenster-Aggregatfunktionen liefern einen Wert für eine spezifizierte Gruppe der Eingabezeilen. Sie können Fensterfunktionen beispielsweise verwenden, um einen veränderlichen Durchschnitt der Absatzzahlen für ein Unternehmen über eine bestimmte Periode zu berechnen.

Fenster-Aggregatfunktionen werden in den folgenden drei Kategorien organisiert:

- **Basis-Aggregatfunktionen** In der folgenden Liste werden die unterstützten Basis-Aggregatfunktionen aufgeführt:
 - „SUM-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
 - „AVG-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
 - „MAX-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
 - „MIN-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
 - „MEDIAN-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
 - „FIRST_VALUE-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
 - „LAST_VALUE-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
 - „COUNT-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- **Funktionen für Standardabweichung und Varianz** In der folgenden Liste werden die unterstützten Funktionen für Standardabweichung und Varianz aufgeführt:
 - „STDDEV-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
 - „STDDEV_POP-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
 - „STDDEV_SAMP-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
 - „VAR_POP-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
 - „VAR_SAMP-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
 - „VARIANCE-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)

- **Funktionen für Korrelation und lineare Regression** In der folgenden Liste werden die unterstützten Funktionen für Korrelation und lineare Regression aufgeführt:
 - „COVAR_POP-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
 - „COVAR_SAMP-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
 - „REGR_AVGX-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
 - „REGR_AVGY-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
 - „REGR_COUNT-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
 - „REGR_INTERCEPT-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
 - „REGR_R2-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
 - „REGR_SLOPE-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
 - „REGR_SXX-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
 - „REGR_SXY-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
 - „REGR_SYY-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Siehe auch

- „Basis-Aggregatfunktionen“ auf Seite 572
- „Funktionen für Korrelation und lineare Regression“ auf Seite 588
- „Funktionen für Standardabweichung und Varianz“ auf Seite 584

Basis-Aggregatfunktionen

Komplexe Datenanalysen erfordern oft mehrere Aggregationsstufen. Die Partitionierung und Sortierung in Fenstern, zusätzlich oder anstelle einer GROUP BY-Klausel, bietet Ihnen erhebliche Flexibilität bei der Zusammenstellung von komplexen SQL-Abfragen. Wenn Sie beispielsweise eine Fensterkonstruktion mit einer einfachen Aggregatfunktion kombinieren, können Sie Werte wie veränderlichen Durchschnitt, veränderliche Summe, veränderliches Minimum oder Maximum und kumulative Summe berechnen.

Die folgenden Basis-Aggregatfunktionen werden unterstützt:

- **SUM-Funktion** Gibt die Gesamtsumme des angegebenen Ausdrucks für jede Zeilengruppe zurück
- **AVG-Funktion** Liefert für eine Zeilenmenge den Durchschnitt eines numerischen Ausdrucks oder einer Menge eindeutiger Werte.
- **MAX-Funktion** Gibt den maximalen Wert für den Ausdruck zurück, der in jeder Zeilengruppe gefunden wurde
- **MIN-Funktion** Liefert den minimalen Wert für den Ausdruck, der in jeder Zeilengruppe gefunden wurde
- **MEDIAN-Funktion** Gibt den Median eines numerischen Ausdrucks für eine Zeilenmenge zurück.
- **FIRST_VALUE-Funktion** Liefert Werte aus der ersten Zeile eines Fensters. Diese Funktion erfordert eine Fensterspezifikation.
- **LAST_VALUE-Funktion** Liefert Werte aus der letzten Zeile eines Fensters. Diese Funktion erfordert eine Fensterspezifikation.

- **COUNT-Funktion** Liefert die Anzahl der Zeilen, die dem angegebenen Ausdruck entsprechen.

Siehe auch

- „SUM-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- „AVG-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- „MAX-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- „MIN-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- „MEDIAN-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- „FIRST_VALUE-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- „LAST_VALUE-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- „COUNT-Funktion [Aggregat]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- „Fensterfunktionen“ auf Seite 564

Beispiel für SUM-Funktion

Im folgenden Beispiel wird die SUM-Funktion als eine Fensterfunktion verwendet. Die Abfrage gibt eine Ergebnismenge zurück, die die Daten nach "DepartmentID" partitioniert und dann eine kumulative Summe der Gehälter beginnend mit dem am längsten im Unternehmen beschäftigten Mitarbeiter anzeigt. Die Ergebnismenge enthält nur die Mitarbeiter, die in Kalifornien, Utah, New York oder Arizona leben. Die Spalte "Sum_Salary" liefert die kumulative Gesamtsumme der Mitarbeitergehälter.

```
SELECT DepartmentID, Surname, StartDate, Salary,
SUM( Salary ) OVER ( PARTITION BY DepartmentID
ORDER BY StartDate
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW )
AS "Sum_Salary"
FROM Employees
WHERE State IN ( 'CA', 'UT', 'NY', 'AZ' )
AND DepartmentID IN ( '100', '200' )
ORDER BY DepartmentID, StartDate;
```

Die nachstehende Tabelle stellt die Ergebnismenge dieser Abfrage dar. Die Ergebnismenge wird nach "DepartmentID" partitioniert.

	DepartmentID	Surname	StartDate	Salary	Sum_Salary
1	100	Whitney	1984-08-28	45700.00	45700.00
2	100	Cobb	1985-01-01	62000.00	107700.00
3	100	Shishov	1986-06-07	72995.00	180695.00
4	100	Driscoll	1986-07-01	48023.69	228718.69
5	100	Guevara	1986-10-14	42998.00	271716.69
6	100	Wang	1988-09-29	68400.00	340116.69
7	100	Soo	1990-07-31	39075.00	379191.69

	DepartmentID	Surname	StartDate	Salary	Sum_Salary
8	100	Diaz	1990-08-19	54900.00	434091.69
9	200	Overbey	1987-02-19	39300.00	39300.00
10	200	Martel	1989-10-16	55700.00	95000.00
11	200	Savarino	1989-11-07	72300.00	167300.00
12	200	Clark	1990-07-21	45000.00	212300.00
13	200	Goggin	1990-08-05	37900.00	250200.00

Für die DepartmentID 100 ist die kumulative Gesamtsumme der Gehälter von Mitarbeitern in Kalifornien, Utah, New York und Arizona insgesamt \$434,091.69 und die kumulative Gesamtsumme für die Mitarbeiter in DepartmentID 200 beträgt \$250,200.00.

Deltawerte zwischen angrenzenden Zeilen berechnen

Durch die Verwendung von zwei Fenstern - ein Fenster für die aktuelle Zeile, das andere für die nächste Zeile - können Sie Deltawerte bzw. Unterschiede zwischen zwei angrenzenden Zeilen berechnen. Die folgende Abfrage berechnet beispielsweise den Deltawert (Delta) zwischen den Gehältern eines Mitarbeiters und des nächsten Mitarbeiters in den Ergebnissen:

```
SELECT EmployeeID AS EmployeeNumber,
       Surname AS LastName,
       SUM( Salary ) OVER ( ORDER BY BirthDate
                           ROWS BETWEEN CURRENT ROW AND CURRENT ROW )
       AS CurrentRow,
       SUM( Salary ) OVER ( ORDER BY BirthDate
                           ROWS BETWEEN 1 PRECEDING AND 1 PRECEDING )
       AS PreviousRow,
       ( CurrentRow - PreviousRow ) AS Delta
FROM Employees
WHERE State IN ( 'NY' );
```

	EmployeeNumber	LastName	CurrentRow	PreviousRow	Delta
1	913	Martel	55700.000	(NULL)	(NULL)
2	1062	Blaikie	54900.000	55700.000	-800.000
3	249	Guevara	42998.000	54900.000	-11902.000
4	390	Davidson	57090.000	42998.000	14092.000
5	102	Whitney	45700.000	57090.000	-11390.000
6	1507	Wetherby	35745.000	45700.000	-9955.000
7	1751	Ahmed	34992.000	35745.000	-753.000

	EmployeeNumber	LastName	CurrentRow	PreviousRow	Delta
8	1157	Soo	39075.000	34992.000	4083.000

SUM wird nur auf die aktuelle Zeile für das CurrentRow-Fenster angewendet, da die Fenstergröße auf ROWS BETWEEN CURRENT ROW AND CURRENT ROW eingestellt wurde. Ebenso wird SUM nur auf die vorherige Zeile für das PreviousRow-Fenster angewendet, da die Fenstergröße auf ROWS BETWEEN 1 PRECEDING AND 1 PRECEDING eingestellt wurde. Der Wert von PreviousRow in der ersten Zeile ist NULL, da es keine vorherige Zeile gibt und somit auch der Deltawert NULL beträgt.

Komplexe Analysen

In der folgenden Abfrage werden die Spitzenverkäufer (definiert nach Gesamtumsatz) für die einzelnen Produkte in der Datenbank aufgeführt:

```
SELECT s.ProductID AS Products, o.SalesRepresentative,
       SUM( s.Quantity ) AS total_quantity,
       SUM( s.Quantity * p.UnitPrice ) AS total_sales
FROM SalesOrders o KEY JOIN SalesOrderItems s
  KEY JOIN Products p
GROUP BY s.ProductID, o.SalesRepresentative
HAVING total_sales = (
  SELECT First SUM( s2.Quantity * p2.UnitPrice )
    AS sum_sales
  FROM SalesOrders o2 KEY JOIN
    SalesOrderItems s2 KEY JOIN Products p2
  WHERE s2.ProductID = s.ProductID
  GROUP BY o2.SalesRepresentative
  ORDER BY sum_sales DESC )
ORDER BY s.ProductID;
```

Plananzeige 1

SQL

```
SELECT s.ProductID AS Products, o.SalesRepresentative,
SUM( s.Quantity ) AS total_quantity,
SUM( s.Quantity * p.UnitPrice ) AS total_sales
FROM SalesOrders o KEY JOIN SalesOrderItems s
KEY JOIN Products p
```

Statistikbene: Detaillierte und Knotenstatisti... Cursortyp: Asensitiv Update-Status: Schreibgesch... Plan abr...

Hauptabfrage

SELECT

Work

Sort

Filter

GrByH

JH*

JNL o

p s

Details Erweiterte Details

SELECT

```
SELECT s.ProductID AS Products, o.SalesRepresentative,
SUM( s.Quantity ) AS total_quantity,
SUM( s.Quantity * p.UnitPrice ) AS total_sales
FROM SalesOrders o KEY JOIN SalesOrderItems s
KEY JOIN Products p
GROUP BY s.ProductID, o.SalesRepresentative
HAVING total_sales = (
SELECT First SUM( s2.Quantity * p2.UnitPrice )
AS sum_sales
FROM SalesOrders o2 KEY JOIN
SalesOrderItems s2 KEY JOIN Products p2
WHERE s2.ProductID = s.ProductID
GROUP BY o2.SalesRepresentative
ORDER BY sum_sales DESC )
ORDER BY s.ProductID;
```

Knotenstatistiken

Öffnen... Speichern unter... Drucken... SQL ausblenden Schließen Hilfe

Diese Abfrage liefert das folgende Ergebnis:

	Products	SalesRepresentative	total_quantity	total_sales
1	300	299	660	5940.00
2	301	299	516	7224.00
3	302	299	336	4704.00
4	400	299	458	4122.00
5	401	902	360	3600.00
6	500	949	360	2520.00
7	501	690	360	2520.00

	Products	SalesRepresentative	total_quantity	total_sales
8	501	949	360	2520.00
9	600	299	612	14688.00
10	601	299	636	15264.00
11	700	299	1008	15120.00

Die ursprüngliche Abfrage wird durch eine korrelierte Unterabfrage gebildet, welche die höchsten Verkäufe für jedes einzelne Produkt ermittelt, da "ProductID" die korrelierte äußere Referenz der Unterabfrage ist. Wie auch in diesem Fall ist eine verschachtelte Abfrage jedoch oft eine kostenträchtige Option. Dies liegt daran, dass die Unterabfrage nicht nur eine GROUP BY-Klausel umfasst, sondern innerhalb dieser auch eine ORDER BY-Klausel. Dies macht es dem Abfrageoptimierer unmöglich, die verschachtelte Abfrage zu einem Join umzuschreiben und dabei die gleiche Semantik beizubehalten. Daher wird die Unterabfrage während der Abfrageausführung für jede abgeleitete Zeile ausgewertet, die im äußeren Block berechnet wird.

Plananzeige 1

SQL

```
SELECT s.ProductID AS Products, o.SalesRepresentative,
SUM( s.Quantity ) AS total_quantity,
SUM( s.Quantity * p.UnitPrice ) AS total_sales
FROM SalesOrders o KEY JOIN SalesOrderItems s
KEY JOIN Products p
```

Statistikebene: Detaillierte und Knotenstatisti... Cursorstyp: Asensitiv Update-Status: Schreibgesch... Plan abr...

Hauptabfrage

Details Erweiterte Details

Hash-Group By

Group by-Liste

s.ProductID int
o.SalesRepresentative int

Aggregate

sum(CAST(s.Quantity AS numeric(10,0)) * p.UnitPrice) numeric
sum(s.Quantity) int

Knotenstatistiken

	Schätzungen	Tatsächlich	Beschreibung
Invocations	-	1	Angabe, wie oft das Ergebnis berechnet wurde

Öffnen... Speichern unter... Drucken... SQL ausblenden Schließen Hilfe

Beachten Sie das kostenträchtige Filterprädikat im grafischen Plan: Der Optimierer schätzt, dass 99 % der Ausführungskosten der Abfrage auf diesen Plan-Operator zurückzuführen sind. Der Plan für die Unterabfrage zeigt deutlich, warum der Filter-Operator im Hauptblock so kostenträchtig ist: Die Unterabfrage enthält zwei Nested Loops Joins, einen Hash-basierten GROUP BY-Vorgang und einen Sortiervorgang.

Umschreiben mit einer Rangfunktion

Ein Umschreiben der gleichen Abfrage mithilfe einer Rangfunktion berechnet das identische Ergebnis viel effizienter:

```
SELECT v.ProductID, v.SalesRepresentative,
       v.total_quantity, v.total_sales
FROM ( SELECT o.SalesRepresentative, s.ProductID,
              SUM( s.Quantity ) AS total_quantity,
              SUM( s.Quantity * p.UnitPrice ) AS total_sales,
              RANK() OVER ( PARTITION BY s.ProductID
                           ORDER BY SUM( s.Quantity * p.UnitPrice ) DESC )
              AS sales_ranking
FROM SalesOrders o KEY JOIN SalesOrderItems s KEY JOIN Products p
GROUP BY o.SalesRepresentative, s.ProductID )
AS v
WHERE sales_ranking = 1
ORDER BY v.ProductID;
```

Die umgeschriebene Abfrage ergibt einen einfacheren Plan:

Plananzeige 1

SQL

```
SELECT v.ProductID, v.SalesRepresentative,
v.total_quantity, v.total_sales
FROM ( SELECT o.SalesRepresentative, s.ProductID,
SUM( s.Quantity ) AS total_quantity,
SUM( s.Quantity * p.UnitPrice ) AS total_sales,
RANK() OVER ( PARTITION BY s.ProductID
ORDER BY SUM( s.Quantity * p.UnitPrice ) DESC )
AS sales_ranking
FROM SalesOrders o KEY JOIN SalesOrderItems s KEY JOIN Products p
GROUP BY o.SalesRepresentative, s.ProductID )
AS v
WHERE sales_ranking = 1
ORDER BY v.ProductID;
```

Statistikebene: Detaillierte und Knotenstatistiken... Cursortyp: Asensitiv Update-Status: Schreibgesch... Plan abr...

Hauptabfrage

SELECT

Work

Filter

DT

Window

Sort

GrByH

JH*

JNL

o

Details

Erweiterte Details

SELECT

```
SELECT v.ProductID, v.SalesRepresentative,
v.total_quantity, v.total_sales
FROM ( SELECT o.SalesRepresentative, s.ProductID,
SUM( s.Quantity ) AS total_quantity,
SUM( s.Quantity * p.UnitPrice ) AS total_sales,
RANK() OVER ( PARTITION BY s.ProductID
ORDER BY SUM( s.Quantity * p.UnitPrice ) DESC )
AS sales_ranking
FROM SalesOrders o KEY JOIN SalesOrderItems s KEY JOIN Products p
GROUP BY o.SalesRepresentative, s.ProductID )
AS v
WHERE sales_ranking = 1
ORDER BY v.ProductID;
```

Knotenstatistiken

	Schätzungen	Tatsächlich	Beschreibung
Invocations	-	1	Angabe, wie oft das Ergebnis berechnet wurde
RowsReturned	54.85	11	Anzahl der zurückgegebenen Zeilen
PercentTotalCost	0	0.20928	Laufzeit als Prozentsatz der gesamten Abfragezeit
RunTime	0	0.00015784	Zeit für die Berechnung der Ergebnisse

Öffnen... Speichern unter... Drucken... SQL ausblenden Schließen Hilfe

Bedenken Sie, dass ein Fensteroperator nach der Verarbeitung einer GROUP BY-Klausel berechnet wird, aber vor der Auswertung der Elemente der SELECT-Liste sowie der ORDER BY-Klausel der Abfrage. Wie im grafischen Plan zu erkennen ist, werden die verknüpften Zeilen nach der Verknüpfung der drei Tabellen durch die Kombination der Attribute "SalesRepresentative" und "ProductID" gruppiert. Daher können die SUM-Aggregatfunktionen "total_quantity" und "total_sales" für alle Kombinationen von "SalesRepresentative" und "ProductID" berechnet werden.

Nach der Auswertung der GROUP BY-Klausel wird die RANK-Funktion berechnet, um mithilfe eines Fensters die Rangfolge der Zeilen im Zwischenergebnis absteigend nach "total_sales" darzustellen. Die WINDOW-Spezifikation umfasst eine PARTITION BY-Klausel. Dadurch wird das Ergebnis der GROUP BY-Klausel neu partitioniert (oder neu gruppiert) - dieses Mal nach "ProductID". Deshalb sortiert die RANK-Funktion die Zeilen für jedes Produkt absteigend nach "total_sales", jedoch nur für die Verkäufer, die dieses Produkt verkauft haben. Bei dieser Rangfolge ist zur Ermittlung der besten Vertriebsmitarbeiter erforderlich, das Ergebnis der abgeleiteten Tabelle so einzuschränken, dass Zeilen, deren Rang nicht 1 ist,

ausgeschlossen werden. Bei Gleichwertigkeit (Zeilen 7 und 8 in der Ergebnismenge) liefert RANK den gleichen Wert. Daher erscheinen die Verkäufer 690 und 949 beide im Endergebnis.

Siehe auch

- „SUM-Funktion [Aggregat]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Beispiel für AVG-Funktion

In diesem Beispiel wird AVG als Fensterfunktion verwendet, um den veränderlichen Durchschnitt aller Produktverkäufe pro Monat im Jahr 2000 zu berechnen. Die WINDOW-Spezifikation verwendet eine RANGE-Klausel, was bewirkt, dass die Fenstergrenzen basierend auf dem Monatswert berechnet werden, und nicht anhand der Anzahl von angrenzenden Zeilen wie bei der ROWS-Klausel. Die Verwendung von ROWS würde zu anderen Ergebnissen führen, wenn es z.B. in einem bestimmten Monat keine Verkäufe eines oder aller Produkte gibt.

```
SELECT *
FROM ( SELECT s.ProductID,
             Month( o.OrderDate ) AS julian_month,
             SUM( s.Quantity ) AS sales,
             AVG( SUM( s.Quantity ) )
             OVER ( PARTITION BY s.ProductID
                   ORDER BY Month( o.OrderDate ) ASC
                   RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING )
             AS average_sales
FROM SalesOrderItems s KEY JOIN SalesOrders o
WHERE Year( o.OrderDate ) = 2000
GROUP BY s.ProductID, Month( o.OrderDate ) )
AS DT
ORDER BY 1,2;
```

Siehe auch

- „AVG-Funktion [Aggregat]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Beispiel für MAX-Funktion

Korrelierte Unterabfragen eliminieren

In manchen Situationen kann es erforderlich sein, einen bestimmten Spaltenwert mit einem maximalen oder minimalen Wert zu vergleichen. Solche Abfragen werden oft in Form von verschachtelten Abfragen gebildet, die ein korreliertes Attribut enthalten (auch als äußere Referenz bezeichnet). Im Beispiel der folgenden Abfrage werden alle Bestellungen einschließlich der Produktdaten aufgeführt, bei denen der vorhandene Bestand nicht ausreicht, um die maximale Einzelbestellung für dieses Produkt zu erfüllen:

```
SELECT o.ID, o.OrderDate, p.*
FROM SalesOrders o, SalesOrderItems s, Products p
WHERE o.ID = s.ID AND s.ProductID = p.ID
      AND p.Quantity < ( SELECT MAX( s2.Quantity )
                        FROM SalesOrderItems s2
                        WHERE s2.ProductID = p.ID )
ORDER BY p.ID, o.ID;
```

Der grafische Plan für diese Abfrage wird in der Plananzeige angezeigt, wie unten dargestellt. Beachten Sie, wie der Abfrageoptimierer diese verschachtelte Abfrage in einen Join transformiert hat, der die

Tabellen "Products" und "SalesOrders" mit einer abgeleiteten Tabelle mit dem Korrelationsnamen DT verknüpft, in der eine Fensterfunktion enthalten ist.

The screenshot shows the 'Plananzeige 1' window with the following components:

- SQL Editor:** Contains the query:


```
SELECT o.ID, o.OrderDate, p.*
FROM SalesOrders o, SalesOrderItems s, Products p
WHERE o.ID = s.ID AND s.ProductID = p.ID
AND p.Quantity < ( SELECT MAX( s2.Quantity )
FROM SalesOrderItems s2 )
```
- Execution Plan:** A tree diagram showing the query execution flow:
 - SELECT (blue box)
 - Work (red box)
 - Sort (red box)
 - JNL (red box)
 - JH* (red box) connected to o (red box)
 - DT (red box) connected to p (red box)
 - DT (red box)
 - Window (red box)
 - s2 (red box)
- Node Statistics Table:**

	Schätzungen	Tatsächlich	Beschreibung
Invocations	-	1	Angabe, wie oft das Ergebnis berechnet wurde
RowsReturned	548.5	743	Anzahl der zurückgegebenen Zeilen
PercentTotalCost	4.9113	2.2785	Laufzeit als Prozentsatz der gesamten Abfragezeit
RunTime	0.0027425	0.0053535	Zeit für die Berechnung der Ergebnisse
FirstRowRunTime	-	0.22333	Zeit zum Abruf der ersten Zeile
CPUTime	0.0027425	-	Von der CPU benötigte Zeit
DiskReadTime	0	-	Zeit für Lesevorgänge von der Festplatte
DiskWriteTime	0	-	Zeit für Schreibvorgänge auf der Festplatte
DiskRead	0	0	Festplatten-Lesevorgänge
DiskWrite	0	0	Festplatten-Schreibvorgänge

Sie können solche Abfragen mit einer Fensterfunktion bilden, statt sich darauf zu verlassen, dass der Optimierer die korrelierte Unterabfrage in einen Join mit einer abgeleiteten Tabelle transformiert, was wegen der Komplexität der semantischen Analyse nur in einfachen Fällen möglich ist.

```
SELECT order_quantity.ID, o.OrderDate, p.*
FROM ( SELECT s.ID, s.ProductID,
      MAX( s.Quantity ) OVER (
        PARTITION BY s.ProductID
        ORDER BY s.ProductID )
      AS max_quantity
      FROM SalesOrderItems s )
AS order_quantity, Products p, SalesOrders o
WHERE p.ID = ProductID
AND o.ID = order_quantity.ID
```

```
AND p.Quantity < max_quantity  
ORDER BY p.ID, o.ID;
```

Siehe auch

- „MIN-Funktion [Aggregat]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „MAX-Funktion [Aggregat]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Beispiele für FIRST_VALUE- und LAST_VALUE-Funktion

Die Funktionen FIRST_VALUE und LAST_VALUE liefern Werte aus der ersten und der letzten Zeile eines Fensters. Dies ermöglicht es einer Abfrage, gleichzeitig auf Werte aus mehreren Zeilen zuzugreifen, ohne dass ein Selbst-Join erforderlich ist.

Diese beiden Funktionen unterscheiden sich von den anderen Fenster-Aggregatfunktionen, da sie mit einem Fenster benutzt werden müssen. Im Gegensatz zu den anderen Fenster-Aggregatfunktionen erlauben diese Funktionen die Klausel IGNORE NULLS. Falls IGNORE NULLS angegeben wird, wird der erste oder letzte Nicht-NULL-Wert des gewünschten Ausdrucks zurückgegeben. Ansonsten wird der erste oder der letzte Wert zurückgegeben.

Beispiel 1: Erster Eintrag in einer Gruppe

Die Funktion FIRST_VALUE kann benutzt werden, um den ersten Eintrag aus einer sortierten Gruppe von Werten abzurufen. Die folgende Abfrage liefert für jede Bestellung die Produkt-ID des ersten Elements der Bestellung. Dies ist die "ProductID" des Elements mit der niedrigsten "LineID" der jeweiligen Bestellung.

Beachten Sie, dass die Abfrage das Schlüsselwort DISTINCT benutzt, um Duplikate zu entfernen. Andernfalls würden für jedes Element in jeder Bestellung Duplikatzeilen zurückgegeben.

```
SELECT DISTINCT ID,  
FIRST_VALUE( ProductID ) OVER ( PARTITION BY ID ORDER BY LineID )  
FROM SalesOrderItems  
ORDER BY ID;
```

Beispiel 2: Prozentsatz der maximalen Verkäufe

Die Funktion FIRST_VALUE wird häufig dazu verwendet, einen Wert in jeder Zeile mit dem Maximum oder Minimum der aktuellen Gruppe zu vergleichen. Die folgende Abfrage berechnet den Gesamtumsatz für jeden Vertriebsmitarbeiter und vergleicht dann diesen Gesamtumsatz mit dem Maximum des Gesamtumsatzes des gleichen Produkts. Das Ergebnis wird als Prozentsatz des maximalen Gesamtumsatzes angezeigt.

```
SELECT s.ProductID AS prod_id, o.SalesRepresentative AS sales_rep,  
SUM( s.Quantity * p.UnitPrice ) AS total_sales,  
100 * total_sales / ( FIRST_VALUE( SUM( s.Quantity * p.UnitPrice ) )  
OVER Sales_Window ) AS total_sales_percentage  
FROM SalesOrders o KEY JOIN SalesOrderItems s KEY JOIN Products p  
GROUP BY o.SalesRepresentative, s.ProductID  
WINDOW Sales_Window AS ( PARTITION BY s.ProductID  
ORDER BY SUM( s.Quantity * p.UnitPrice ) DESC )  
ORDER BY s.ProductID;
```


Beispiel 3: Das Einlesen von NULL-Werten macht Daten dichter

Die Funktionen FIRST_VALUE und LAST_VALUE sind nützlich, wenn Sie die Daten verdichtet haben und Werte anstelle von NULL eintragen müssen. Angenommen, der Vertriebsmitarbeiter mit dem höchsten Gesamtumsatz eines Tages wird als Vertriebsmitarbeiter des Tages ausgezeichnet. Die folgende Abfrage listet die Gewinner für die erste Aprilwoche 2001 auf:

```
SELECT v.OrderDate, v.SalesRepresentative AS rep_of_the_day
FROM ( SELECT o.SalesRepresentative, o.OrderDate,
             RANK() OVER ( PARTITION BY o.OrderDate
                          ORDER BY SUM( s.Quantity *
                                         p.UnitPrice ) DESC ) AS sales_ranking
      FROM SalesOrders o KEY JOIN SalesOrderItems s KEY JOIN Products p
      GROUP BY o.SalesRepresentative, o.OrderDate ) AS v
WHERE v.sales_ranking = 1
AND v.OrderDate BETWEEN '2001-04-01' AND '2001-04-07'
ORDER BY v.OrderDate;
```

Diese Abfrage liefert folgende Ergebnisse:

OrderDate	rep_of_the_day
2001-04-01	949
2001-04-02	856
2001-04-05	902
2001-04-06	467
2001-04-07	299

Für die Tage ohne Umsatz werden keine Ergebnisse zurückgegeben. Die folgende Abfrage verdichtet die Daten und zeigt Datensätze für die Tage an, an denen es keine Umsätze gab. Außerdem verwendet sie die Funktion LAST_VALUE, um anstelle von NULL für "rep_of_the_day" (an Tagen ohne Gewinner) die ID des letzten Gewinners einzutragen, bis ein neuer Gewinner in den Ergebnissen erscheint.

```
SELECT d.dense_order_date,
       LAST_VALUE( v.SalesRepresentative IGNORE NULLS )
       OVER ( ORDER BY d.dense_order_date )
       AS rep_of_the_day
FROM ( SELECT o.SalesRepresentative, o.OrderDate,
             RANK() OVER ( PARTITION BY o.OrderDate
                          ORDER BY SUM( s.Quantity *
                                         p.UnitPrice ) DESC ) AS sales_ranking
      FROM SalesOrders o KEY JOIN SalesOrderItems s KEY JOIN Products p
      GROUP BY o.SalesRepresentative, o.OrderDate ) AS v
RIGHT OUTER JOIN ( SELECT DATEADD( day, row_num, '2001-04-01' )
                  AS dense_order_date
                  FROM sa_rowgenerator( 0, 6 )) AS d
ON v.OrderDate = d.dense_order_date AND sales_ranking = 1
ORDER BY d.dense_order_date;
```

Diese Abfrage liefert folgende Ergebnisse:

dense_order_date	rep_of_the_day
2001-04-01	949
2001-04-02	856
2001-04-03	856
2001-04-04	856
2001-04-05	902
2001-04-06	467
2001-04-07	299

Die abgeleitete Tabelle "v" aus der vorigen Abfrage wird durch einen Join mit der abgeleiteten Tabelle "d" verknüpft, die alle in Betracht kommenden Tage enthält. Dies ergibt eine Zeile für jeden gewünschten Tag, aber der Outer-Join enthält NULL in der Spalte "SalesRepresentative" für die Tage, an denen es keine Umsätze gab. Mit der Funktion LAST_VALUE wird dieses Problem gelöst, indem festgelegt wird, dass "rep_of_the_day" für die jeweilige Zeile den letzten Nicht-NULL-Wert aus "SalesRepresentative" bis zu dem Tag enthält, an dem die Spalte wieder einen Umsatzwert aufweist.

Siehe auch

- „FIRST_VALUE-Funktion [Aggregat]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „LAST_VALUE-Funktion [Aggregat]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Fensterfunktionen“ auf Seite 564

Funktionen für Standardabweichung und Varianz

Es werden zwei Versionen von Funktionen für Varianz und Standardabweichung unterstützt: eine Version für Stichproben und eine Version für Populationen. Die Auswahl einer Version hängt von dem statistischen Zusammenhang ab, in der die Funktion benutzt wird.

Alle Funktionen für Varianz und Standardabweichung sind echte Aggregatfunktionen, da sie Werte für eine Zeilenpartition berechnen können, die durch die GROUP BY-Klausel der Abfrage erstellt wurde. Ebenso wie andere Aggregatfunktionen, etwa MAX oder MIN, ignorieren auch diese Funktionen NULL in der Eingabe.

Um die Performance zu verbessern, berechnet der Datenbankserver den Mittelwert und die Abweichung vom Mittelwert in einem Schritt.

Außerdem werden unabhängig vom Datentyp des zu analysierenden Ausdrucks alle Berechnungen für Varianz und Standardabweichung im IEEE-Format einer Gleitkommazahl mit doppelter Genauigkeit durchgeführt. Falls die Eingabe einer Funktion für Varianz oder Standardabweichung leer ist, liefert die Funktion NULL als Ergebnis. Wenn VAR_SAMP für eine einzelne Zeile berechnet wird, ist das Ergebnis NULL, während es bei VAR_POP der Wert 0 ist.

Die folgenden Funktionen für Standardabweichung und Varianz werden unterstützt:

- STDDEV-Funktion
- STDDEV_POP-Funktion
- STDDEV_SAMP-Funktion
- VARIANCE-Funktion
- VAR_POP-Funktion
- VAR_SAMP-Funktion

STDDEV-Funktion

Diese Funktion ist ein Alias für die Funktion STDDEV_SAMP.

STDDEV_POP-Funktion

Diese Funktion berechnet die Standardabweichung für eine Normalverteilung, die aus einem numerischen Ausdruck besteht, als DOUBLE.

Beispiel 1

Die nachstehende Abfrage ergibt eine Ergebnismenge, die die Mitarbeiter zeigt, deren Gehalt um eine Standardabweichung über dem Durchschnittsgehalt ihrer Abteilung liegt. Die Standardabweichung ist ein Messwert für die Abweichung der Daten vom Mittelwert.

```
SELECT *
FROM ( SELECT
        Surname AS Employee,
        DepartmentID AS Department,
        CAST( Salary as DECIMAL( 10, 2 ) )
          AS Salary,
        CAST( AVG( Salary )
              OVER ( PARTITION BY DepartmentID ) AS DECIMAL ( 10, 2 ) )
          AS Average,
        CAST( STDDEV_POP( Salary )
              OVER ( PARTITION BY DepartmentID ) AS DECIMAL ( 10, 2 ) )
          AS StandardDeviation
      FROM Employees
      GROUP BY Department, Employee, Salary )
      AS DerivedTable
WHERE Salary > Average + StandardDeviation
ORDER BY Department, Salary, Employee;
```

Die nachstehende Tabelle stellt die Ergebnismenge dieser Abfrage dar. Jede Abteilung hat mindestens einen Mitarbeiter, dessen Gehalt deutlich vom Mittelwert abweicht.

	Employee	Department	Salary	Average	StandardDeviation
1	Lull	100	87900.00	58736.28	16829.60
2	Scheffield	100	87900.00	58736.28	16829.60
3	Scott	100	96300.00	58736.28	16829.60
4	Sterling	200	64900.00	48390.95	13869.60

	Employee	Department	Salary	Average	StandardDeviation
5	Savarino	200	72300.00	48390.95	13869.60
6	Kelly	200	87500.00	48390.95	13869.60
7	Shea	300	138948.00	59500.00	30752.40
8	Blaikie	400	54900.00	43640.67	11194.02
9	Morris	400	61300.00	43640.67	11194.02
10	Evans	400	68940.00	43640.67	11194.02
11	Martinez	500	55500.00	33752.20	9084.50

Der Mitarbeiter Scott verdient \$96,300.00, der Abteilungsdurchschnitt ist hingegen \$58,736.28. Die Standardabweichung für diese Abteilung ist 16,829.00. Dies bedeutet, dass Gehälter unter \$75,565.88 ($58736.28 + 16829.60 = 75565.88$) innerhalb einer Standardabweichung des Mittelwerts liegen. Bei \$96,300.00 ist der Mitarbeiter Scott deutlich über diesem Wert.

In diesem Beispiel wird davon ausgegangen, dass "Surname" und "Salary" für alle Mitarbeiter unterschiedlich sind, was aber nicht notwendigerweise der Fall ist. Um Eindeutigkeit zu gewährleisten, könnten Sie der GROUP BY-Klausel das Merkmal "EmployeeID" hinzufügen.

Beispiel 2

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 SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

Diese Abfrage liefert das folgende Ergebnis:

Year	Quarter	Average	Variance
2000	1	25.775148	14.2794...
2000	2	27.050847	15.0270...
...

STDDEV_SAMP-Funktion

Diese Funktion berechnet die Standardabweichung für eine Stichprobe, die aus einem numerischen Ausdruck besteht, als DOUBLE. Die folgende Anweisung liefert beispielsweise den Durchschnitt und die Varianz in der Anzahl der Elemente pro Auftrag in verschiedenen Quartalen:

```
SELECT YEAR( ShipDate ) AS Year,
       QUARTER( ShipDate ) AS Quarter,
       AVG( Quantity ) AS Average,
       STDDEV_SAMP( Quantity ) AS Variance
FROM SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

Diese Abfrage liefert das folgende Ergebnis:

Year	Quarter	Average	Variance
2000	1	25.775148	14.3218...
2000	2	27.050847	15.0696...
...

VARIANCE-Funktion

Diese Funktion ist ein Alias der VAR_SAMP-Funktion.

VAR_POP-Funktion

Diese Funktion berechnet die statistische Varianz einer Normalverteilung, die aus einem numerischen Ausdruck besteht, als DOUBLE. Die folgende Anweisung listet beispielsweise 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 SalesOrderItems
GROUP BY Year, Quarter
ORDER BY Year, Quarter;
```

Diese Abfrage liefert das folgende Ergebnis:

Year	Quarter	Average	Variance
2000	1	25.775148	203.9021...
2000	2	27.050847	225.8109...
...

Falls VAR_POP für eine einzelne Zeile berechnet wird, lautet das Ergebnis der Wert 0.

VAR_SAMP-Funktion

Diese Funktion berechnet die statistische Varianz einer Stichprobe, die aus einem numerischen Ausdruck besteht, als DOUBLE.

Die folgende Anweisung listet beispielsweise 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 SalesOrderItems  
GROUP BY Year, Quarter  
ORDER BY Year, Quarter;
```

Diese Abfrage liefert das folgende Ergebnis:

Year	Quarter	Average	Variance
2000	1	25.775148	205.1158...
2000	2	27.050847	227.0939...
...

Falls VAR_SAMP für eine einzelne Zeile berechnet wird, lautet das Ergebnis NULL.

Siehe auch

- „STDDEV_SAMP-Funktion [Aggregat]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „STDDEV_POP-Funktion [Aggregat]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „VAR_SAMP-Funktion [Aggregat]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „VAR_POP-Funktion [Aggregat]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „VAR_SAMP-Funktion [Aggregat]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Mathematische Formeln für die Aggregatfunktionen“ auf Seite 598

Funktionen für Korrelation und lineare Regression

Es wird eine Vielzahl von Statistikfunktionen unterstützt, deren Ergebnisse die Qualitätsanalyse linearer Regressionen unterstützen können.

Das erste Argument jeder Funktion ist der abhängige Ausdruck (als Y bezeichnet), das zweite Argument ist der unabhängige Ausdruck (als X bezeichnet).

- **COVAR_SAMP-Funktion** Die Funktion COVAR_SAMP liefert die Stichprobenkovarianz einer Gruppe von (Y, X)-Paaren.
- **COVAR_POP-Funktion** Die Funktion COVAR_POP liefert die Normalverteilungskovarianz einer Gruppe von (Y, X)-Paaren.

- **CORR-Funktion** Die Funktion CORR liefert den Korrelationskoeffizienten einer Gruppe von (Y, X)-Paaren.
- **REGR_AVGX-Funktion** Die Funktion REGR_AVGX liefert den Mittelwert der X-Werte aller Nicht-NULL-Paare von (Y, X)-Werten.
- **REGR_AVGY-Funktion** Die Funktion REGR_AVGY liefert den Mittelwert der Y-Werte aller Nicht-NULL-Paare von (Y, X)-Werten.
- **REGR_SLOPE-Funktion** Die Funktion REGR_SLOPE berechnet den Richtungskoeffizienten der linearen Regressionszeile für Nicht-NULL-Paare.
- **REGR_INTERCEPT-Funktion** Die Funktion REGR_INTERCEPT berechnet den y-Abschnitt der linearen Regressionszeile, die am besten zu den abhängigen und unabhängigen Variablen passt.
- **REGR_R2-Funktion** Die Funktion REGR_R2 berechnet den Koeffizienten der Determination (auch **R_zum_Quadrat** bzw. **Passgenauigkeit** genannt) für die Regressionszeile.
- **REGR_COUNT-Funktion** Die Funktion REGR_COUNT liefert die Anzahl der Nicht-NULL-Paare von (Y, X)-Werten in der Eingabe. Nur wenn im jeweiligen Paar sowohl X als auch Y nicht gleich NULL sind, wird dieses Ergebnis in Berechnungen linearer Regressionen benutzt.
- **REGR_SXX-Funktion** Diese Funktion liefert die Summe der Quadrate von x-Werten der (Y, X)-Paare.

Die Gleichung dieser Funktion ist äquivalent zum Zähler der Formel für die Stichproben- oder Populationsvarianz. Beachten Sie, dass REGR_SXX ebenso wie die anderen Funktionen zur linearen Regression alle Paare von (Y, X)-Werten in der Eingabe ignorieren, bei denen X oder Y gleich NULL ist.

- **REGR_SYY-Funktion** Diese Funktion liefert die Summe der Quadrate von Y-Werten der (Y, X)-Paare.
- **REGR_SXY-Funktion** Diese Funktion liefert die Differenz zweier Summenprodukte der Gruppe von (Y, X)-Paaren.

Siehe auch

- „COVAR_SAMP-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „COVAR_POP-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CORR-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REGR_AVGX-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REGR_AVGY-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REGR_SLOPE-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REGR_R2-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REGR_COUNT-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REGR_SXX-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REGR_SYY-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REGR_SXY-Funktion [Aggregat]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Fenster-Rangfunktionen

Fenster-Rangfunktionen liefern den Rang einer Zeile in Relation zu den anderen Zeilen einer Partition. Die unterstützten Rangfunktionen lauten:

- CUME_DIST
- DENSE_RANK
- PERCENT_RANK
- RANK

Rangfunktionen werden nicht als Aggregatfunktionen angesehen, da sie kein Ergebnis aus mehreren Eingabezeilen berechnen wie beispielsweise die SUM-Aggregatfunktion. Diese Funktionen berechnen den Rang oder die relative Sortierung einer Zeile innerhalb einer Partition aufgrund des Wertes eines bestimmten Ausdrucks. Jede Gruppe von Zeilen innerhalb einer Partition wird unabhängig von den anderen Gruppen nach Rang sortiert. Falls die OVER-Klausel keine PARTITION BY-Klausel enthält, wird die gesamte Eingabe als eine einzige Partition behandelt. Sie können daher keine ROWS- oder RANGE-Klausel für ein Fenster spezifizieren, das von einer Rangfunktion verwendet wird. Es ist möglich, eine Abfrage zu bilden, die mehrere Rangfunktionen enthält, von denen jede einzelne die Eingabezeilen unterschiedlich partitioniert oder sortiert.

Alle Rangfunktionen benötigen eine ORDER BY-Klausel, um die Reihenfolge der Eingabezeilen festzulegen, von der die Rangfunktionen abhängig sind. Falls die ORDER BY-Klausel mehrere Ausdrücke enthält, werden der zweite und die folgenden Ausdrücke benutzt, um Gleichwertigkeiten aufzulösen, falls der erste Ausdruck den gleichen Wert in angrenzenden Zeilen ergibt. NULL steht vor allen anderen Werten (bei aufsteigender Sortierfolge).

RANK-Funktion

Die RANK-Funktion liefert den Rang des Wertes in der aktuellen Zeile im Vergleich zu den Werten in anderen Zeilen. Der Rang eines Wertes reflektiert die Reihenfolge, in der er erscheinen würde, wenn die Werteliste sortiert wäre.

Die RANK-Funktion berechnet den Rang für den Ausdruck, der in der ORDER BY-Klausel des Fensters angegeben wurde. Falls die ORDER BY-Klausel mehrere Ausdrücke enthält, werden der zweite und die folgenden Ausdrücke benutzt, um Gleichwertigkeiten aufzulösen, falls der erste Ausdruck den gleichen Wert in angrenzenden Zeilen ergibt. NULL steht vor allen anderen Werten (bei aufsteigender Sortierfolge).

Beispiel 1

Die folgende Abfrage ermittelt die drei teuersten Produkte in der Datenbank. Für das Fenster wird eine absteigende Sortierfolge festgelegt, sodass das teuerste Produkt den niedrigsten Rang hat. Die Rangfolgen beginnen mit 1.

```
SELECT Top 3 *
FROM ( SELECT Description, Quantity, UnitPrice,
      RANK() OVER ( ORDER BY UnitPrice DESC ) AS Rank
FROM Products ) AS DT
ORDER BY Rank;
```

Diese Abfrage liefert das folgende Ergebnis:

	Description	Quantity	UnitPrice	Rank
1	Zipped Sweatshirt	32	24.00	1
2	Hooded Sweatshirt	39	24.00	1
3	Cotton Shorts	80	15.00	3

Die Zeilen 1 und 2 haben den gleichen Wert für "Unit Price" und daher auch den gleichen Rang. Dies wird als Gleichwertigkeit bezeichnet.

Bei der RANK-Funktion macht der Rangwert nach einer Gleichwertigkeit einen Sprung. So springt der Rangwert für Zeile 3 beispielsweise auf 3 statt auf 2. Dies ist unterschiedlich zur Funktion DENSE_RANK, bei der nach einer Gleichwertigkeit kein Sprung auftritt.

Beispiel 2

Die folgende SQL-Abfrage findet die Mitarbeiter und Mitarbeiterinnen in Utah und ordnet sie nach Gehaltshöhe in absteigender Reihenfolge.

```
SELECT Surname, Salary, Sex,
       RANK() OVER ( ORDER BY Salary DESC ) "Rank"
FROM Employees
WHERE State IN ( 'UT' );
```

Die nachstehende Tabelle stellt die Ergebnismenge dieser Abfrage dar:

	Surname	Salary	Sex	Rank
1	Shishov	72995.00	F	1
2	Wang	68400.00	M	2
3	Cobb	62000.00	M	3
4	Morris	61300.00	M	4
5	Diaz	54900.00	M	5
6	Driscoll	48023.69	M	6
7	Hildebrand	45829.00	F	7
8	Goggin	37900.00	M	8
9	Rebeiro	34576.00	M	9
10	Bigelow	31200.00	F	10
11	Lynch	24903.00	M	11

Beispiel 3

Sie können Ihre Daten partitionieren, um andere Ergebnisse anzuzeigen. Wenn Sie die Abfrage aus Beispiel 2 verwenden, können Sie die Daten ändern, indem Sie sie nach Geschlecht partitionieren. Im folgenden Beispiel werden Mitarbeiter in absteigender Rangfolge nach Gehalt angeordnet und nach Geschlecht aufgeteilt.

```
SELECT Surname, Salary, Sex,
       RANK ( ) OVER ( PARTITION BY Sex
                       ORDER BY Salary DESC ) "Rank"
FROM Employees
WHERE State IN ( 'UT' );
```

Die nachstehende Tabelle stellt die Ergebnismenge dieser Abfrage dar:

	Surname	Salary	Sex	Rank
1	Wang	68400.00	M	1
2	Cobb	62000.00	M	2
3	Morris	61300.00	M	3
4	Diaz	54900.00	M	4
5	Driscoll	48023.69	M	5
6	Goggin	37900.00	M	6
7	Rebeiro	34576.00	M	7
8	Lynch	24903.00	M	8
9	Shishov	72995.00	F	1
10	Hildebrand	45829.00	F	2
11	Bigelow	31200.00	F	3

Siehe auch

- „DENSE_RANK-Funktion“ auf Seite 592
- „RANK-Funktion [Rangfolge]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

DENSE_RANK-Funktion

Ähnlich wie die RANK-Funktion liefert DENSE_RANK den Rang des Wertes in der aktuellen Zeile im Vergleich zu den Werten in anderen Zeilen. Der Rang eines Wertes reflektiert die Reihenfolge, in der er in der Liste erscheinen würde, wenn die Werte sortiert wären. Der Rang wird für den Ausdruck berechnet, der in der ORDER BY-Klausel des Fensters festgelegt wurde.

Die Funktion DENSE_RANK liefert eine Reihe von Rangfolgen, die gleichmäßig und ohne Lücken oder Sprünge beim Rangwert ansteigen. Die Bezeichnung "dense" (= dicht) wird benutzt, da es anders als bei der RANK-Funktion keine Sprünge beim Rangwert gibt.

Wenn sich das Fenster durch die Eingabezeilen nach unten bewegt, wird der Rang für den Ausdruck berechnet, der in der ORDER BY-Klausel des Fensters angegeben wurde. Falls die ORDER BY-Klausel mehrere Ausdrücke enthält, werden der zweite und die folgenden Ausdrücke benutzt, um Gleichwertigkeiten aufzulösen, falls der erste Ausdruck den gleichen Wert in angrenzenden Zeilen ergibt. NULL steht vor allen anderen Werten (bei aufsteigender Sortierfolge).

Beispiel 1

Die folgende Abfrage ermittelt die drei teuersten Produkte in der Datenbank. Für das Fenster wird eine absteigende Sortierfolge festgelegt, sodass das teuerste Produkt den niedrigsten Rang hat. Die Rangfolgen beginnen mit 1.

```
SELECT Top 3 *
FROM ( SELECT Description, Quantity, UnitPrice,
      DENSE_RANK( ) OVER ( ORDER BY UnitPrice DESC ) AS Rank
FROM Products ) AS DT
ORDER BY Rank;
```

Diese Abfrage liefert das folgende Ergebnis:

	Description	Quantity	UnitPrice	Rank
1	Hooded Sweatshirt	39	24.00	1
2	Zippered Sweatshirt	32	24.00	1
3	Cotton Shorts	80	15.00	2

Die Zeilen 1 und 2 haben den gleichen Wert für "Unit Price" und daher auch den gleichen Rang. Dies wird als Gleichwertigkeit bezeichnet.

Mit der Funktion DENSE_RANK gibt es nach einer Gleichwertigkeit keinen Sprung beim Rangwert. Der Rangwert für Zeile 3 beträgt zum Beispiel 2. Dies ist unterschiedlich zur RANK-Funktion, bei der nach einer Gleichwertigkeit ein Sprung in den Rangwerten auftritt.

Beispiel 2

Da Fenster nach der GROUP BY-Klausel einer Abfrage ausgewertet werden, können Sie komplexe Kriterien festlegen, welche die Rangfolgen aufgrund der Werte einer Aggregatfunktion ermitteln.

Die folgende Abfrage liefert die drei Spitzenverkäufer jeder Region nach ihrem Gesamtumsatz zusammen mit dem Gesamtumsatz für jede einzelne Region:

```
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 Products 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;
```

Diese Abfrage liefert das folgende Ergebnis:

	SalesRepresentative	Region	total_sales	sales_rank
1	299	Canada	9312.00	1
2	(NULL)	Canada	24768.00	1
3	1596	Canada	3564.00	2
4	856	Canada	2724.00	3
5	299	Central	32592.00	1
6	(NULL)	Central	134568.00	1
7	856	Central	14652.00	2
8	467	Central	14352.00	3
9	299	Eastern	21678.00	1
10	(NULL)	Eastern	142038.00	1
11	902	Eastern	15096.00	2
12	690	Eastern	14808.00	3
13	1142	South	6912.00	1
14	(NULL)	South	45262.00	1
15	667	South	6480.00	2
16	949	South	5782.00	3
17	299	Western	5640.00	1
18	(NULL)	Western	37632.00	1
19	1596	Western	5076.00	2
20	667	Western	4068.00	3

Hier werden mehrere Gruppierungen durch die Verwendung von GROUPING SETS kombiniert. Daher verwendet die WINDOW PARTITION-Klausel für das Fenster die GROUPING-Funktion, um zwischen Detailzeilen, die bestimmte Verkäufer repräsentieren, und Zwischensummenzeilen, in denen der Gesamtumsatz für eine Region aufgeführt wird, zu unterscheiden. Die Zwischensummenzeilen nach Region, die für das Attribut "SalesRepresentative" den Wert NULL enthalten, haben alle den Rangwert 1, da die Rangfolge des Ergebnisses für jede Partition der Eingabe neu gestartet wird. Damit wird gewährleistet, dass die Detailzeilen korrekt mit dem Rangwert 1 beginnen.

Beachten Sie außerdem, dass die Funktion DENSE_RANK in diesem Beispiel die Rangfolge der Eingabe über die Aggregation des Gesamtumsatzes berechnet. Ein Element mit Alias in der SELECT-Liste wird als Abkürzung in der WINDOW ORDER-Klausel verwendet.

Siehe auch

- „RANK-Funktion“ auf Seite 590
- „DENSE_RANK-Funktion [Rangfolge]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

CUME_DIST-Funktion

Die kumulative Verteilungsfunktion CUME_DIST wird auch als umgekehrter Prozentsatz bezeichnet. CUME_DIST berechnet die normalisierte Position eines bestimmten Wertes relativ zur Gruppe der Werte im Fenster. Der Bereich der Funktion liegt zwischen 0 und 1.

Wenn sich das Fenster durch die Eingabezeilen nach unten bewegt, wird die kumulative Verteilung für den Ausdruck berechnet, der in der ORDER BY-Klausel des Fensters angegeben wurde. Falls die ORDER BY-Klausel mehrere Ausdrücke enthält, werden der zweite und die folgenden Ausdrücke benutzt, um Gleichwertigkeiten aufzulösen, falls der erste Ausdruck den gleichen Wert in angrenzenden Zeilen ergibt. NULL steht vor allen anderen Werten (bei aufsteigender Sortierfolge).

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 Employees
WHERE State IN ( 'CA' );
```

Diese Abfrage liefert das folgende Ergebnis:

DepartmentID	Surname	Salary	Rank
200	Savarino	72300.00	0.3333333333333333
200	Clark	45000.00	0.6666666666666667
200	Overbey	39300.00	1

Siehe auch

- „CUME_DIST-Funktion [Rangfolge]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

PERCENT_RANK-Funktion

Ähnlich wie die PERCENT-Funktion liefert die PERCENT_RANK-Funktion den Rang des Wertes in der Spalte, die in der ORDER BY-Klausel des Fensters angegeben wurde, jedoch als Bruch zwischen 0 und 1 ausgedrückt wurde, berechnet durch $(\text{RANK} - 1) / (-1)$.

Wenn sich das Fenster durch die Eingabezeilen nach unten bewegt, wird der Rang für den Ausdruck berechnet, der in der ORDER BY-Klausel des Fensters angegeben wurde. Falls die ORDER BY-Klausel mehrere Ausdrücke enthält, werden der zweite und die folgenden Ausdrücke benutzt, um Gleichwertigkeiten aufzulösen, falls der erste Ausdruck den gleichen Wert in angrenzenden Zeilen ergibt. NULL steht vor allen anderen Werten (bei aufsteigender Sortierfolge).

Beispiel 1

Das folgende Beispiel gibt eine Ergebnismenge zurück, die die Rangfolge der Gehälter von Mitarbeitern in New York anhand ihrer Geschlechtszugehörigkeit anzeigt. Die Ergebnisse sind in absteigender Reihenfolge unter Verwendung eines dezimalen Prozentsatzes gereiht und anhand der Geschlechtszugehörigkeit aufgeschlüsselt.

```
SELECT DepartmentID, Surname, Salary, Sex,  
       PERCENT_RANK( ) OVER ( PARTITION BY Sex  
                             ORDER BY Salary DESC ) AS PctRank  
FROM Employees  
WHERE State IN ( 'NY' );
```

Diese Abfrage liefert folgende Ergebnisse:

	DepartmentID	Surname	Salary	Sex	PctRank
1	200	Martel	55700.000	M	0.0
2	100	Guevara	42998.000	M	0.333333333
3	100	Soo	39075.000	M	0.666666667
4	400	Ahmed	34992.000	M	1.0
5	300	Davidson	57090.000	F	0.0
6	400	Blaikie	54900.000	F	0.333333333
7	100	Whitney	45700.000	F	0.666666667
8	400	Wetherby	35745.000	F	1.0

Da die Eingabe nach Geschlecht (Sex) partitioniert ist, wird PERCENT_RANK für Männer und Frauen getrennt ausgewertet.

Beispiel 2

Im folgenden Beispiel wird eine Liste der Mitarbeiterinnen in Utah und Arizona zurückgegeben, die nach Gehaltshöhe in absteigender Reihenfolge angeordnet werden. Hier wird die PERCENT_RANK-Funktion benutzt, um eine kumulative Gesamtsumme in absteigender Reihenfolge auszugeben.

```

SELECT Surname, Salary,
       PERCENT_RANK ( ) OVER ( ORDER BY Salary DESC ) "Rank"
FROM Employees
WHERE State IN ( 'UT', 'AZ' ) AND Sex IN ( 'F' );

```

Diese Abfrage liefert folgende Ergebnisse:

	Surname	Salary	Rank
1	Shishov	72995.00	0
2	Jordan	51432.00	0.25
3	Hildebrand	45829.00	0.5
4	Bigelow	31200.00	0.75
5	Bertrand	29800.00	1

Obere und untere Prozentsätze mit PERCENT_RANK finden

Sie können PERCENT_RANK verwenden, um die oberen oder unteren Prozentsätze in einer Datenmenge zu finden. Im folgenden Beispiel gibt die Abfrage männliche Mitarbeiter zurück, deren Gehalt in den oberen fünf Prozent der Datenmenge liegt.

```

SELECT *
FROM ( SELECT Surname, Salary,
       PERCENT_RANK ( ) OVER ( ORDER BY Salary DESC ) "Rank"
FROM Employees
WHERE Sex IN ( 'M' ) )
AS DerivedTable ( Surname, Salary, Percent )
WHERE Percent < 0.05;

```

Diese Abfrage liefert folgende Ergebnisse:

	Surname	Salary	Percent
1	Scott	96300.00	0
2	Sheffield	87900.00	0.025
3	Lull	87900.00	0.025

Siehe auch

- „PERCENT_RANK-Funktion [Rangfolge]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Zeilennummerierungsfunktionen

Zeilennummerierungsfunktionen vergeben eindeutige Zeilennummern für die Zeilen einer Partition. Zwei Zeilennummerierungsfunktionen werden unterstützt: NUMBER und ROW_NUMBER. Die Verwendung der ROW_NUMBER-Funktion wird empfohlen, weil sie eine ANSI-Standard-kompatible Funktion ist

und viele der Möglichkeiten der NUMBER(*)-Funktion bereitstellt. Zwar erfüllen beide Funktionen ähnliche Aufgaben, bei der Funktion NUMBER gibt es jedoch mehrere Einschränkungen, die für die Funktion ROW_NUMBER nicht gelten.

Siehe auch

- „NUMBER-Funktion [Verschiedene]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ROW_NUMBER-Funktion“ auf Seite 598

ROW_NUMBER-Funktion

Die Funktion ROW_NUMBER nummeriert die Zeilen im Ergebnis. Es handelt sich nicht um eine Rangfunktion. Sie können diese Funktion jedoch in jeder Situation benutzen, in der auch eine Rangfunktion verwendet werden kann, und sie funktioniert ähnlich wie eine Rangfunktion.

Beispielsweise können Sie ROW_NUMBER in einer abgeleiteten Tabelle verwenden, sodass zusätzliche Einschränkungen und sogar Joins auf die Werte von ROW_NUMBER angewendet werden können:

```
SELECT *
FROM ( SELECT Description, Quantity,
        ROW_NUMBER( ) OVER ( ORDER BY ID ASC ) AS RowNum
FROM Products ) AS DT
WHERE RowNum <= 3
ORDER BY RowNum;
```

Diese Abfrage liefert folgende Ergebnisse:

Description	Quantity	RowNum
Tank Top	28	1
V-neck	54	2
Crew Neck	75	3

Ebenso wie die Rangfunktionen erfordert ROW_NUMBER eine ORDER BY-Klausel.

Außerdem kann ROW_NUMBER nicht deterministische Ergebnisse liefern, wenn sich die ORDER BY-Klausel des Fensters auf nicht eindeutige Ausdrücke bezieht. Bei Gleichwertigkeit ist die Reihenfolge der Zeilen nicht vorhersehbar.

ROW_NUMBER ist für die Anwendung auf die gesamte Partition gedacht. Daher kann mit einer ROW_NUMBER-Funktion keine ROWS- oder RANGE-Klausel spezifiziert werden.

Mathematische Formeln für die Aggregatfunktionen

Zur Information finden Sie in den beiden folgenden Tabellen die äquivalenten mathematischen Formeln für die Fenster-Aggregatfunktionen.

Einfache Aggregatfunktionen

Function	Symbol	Formula
SUM(X)		$\sum_{i=1}^n x_i$
MAX(X)		$x_i : x_i \geq x_j, i \neq j \forall i, j \in n$
MIN(X)		$x_i : x_i \leq x_j, i \neq j \forall i, j \in n$
AVG(X)	\bar{x}	$\frac{\sum x_i}{n}$
COUNT(*)		n
VAR_SAMP(X)	s_x^2	$\frac{\sum (x_i - \bar{x})^2}{(n-1)}$
VAR_POP(X)	σ_x^2	$\frac{\sum (x_i - \bar{x})^2}{n}$
VARIANCE(X)		identical to VAR_SAMP(X)
STDDEV_SAMP(X)	s_x	$\sqrt{\frac{\sum (x_i - \bar{x})^2}{(n-1)}}$
STDDEV_POP(X)	σ_x	$\sqrt{\frac{\sum (x_i - \bar{x})^2}{n}}$
STDDEV(X)		identical to STDDEV_SAMP(X)

Statistische Aggregatfunktionen

COVAR_SAMP(Y,X)	Co-variance	$s_{xy} = \frac{\sum xy - \frac{(\sum x)(\sum y)}{n}}{(n-1)}$
COVAR_POP(Y,X)	Co-variance	$\sigma_{xy} = \frac{\sum xy - \frac{(\sum x)(\sum y)}{n}}{n}$
CORR(Y,X)	Correlation Coefficient	$r = \frac{\sum xy - \frac{1}{n}(\sum x)(\sum y)}{(n-1)s_x s_y}$
REGR_AVGX(Y,X)	Independent mean	\bar{x}
REGR_AVGY(Y,X)	Dependent mean	\bar{y}
REGR_SLOPE(Y,X)	Regression Slope	$b = r \frac{s_y}{s_x}$
REGR_INTERCEPT(Y,X)	Regression Intercept	$a = \bar{y} - b\bar{x}$
REGR_R2(Y,X)	'Goodness-of-fit'	r^2
REGR_COUNT(Y,X)	Sample size	n (non-null (Y, X) pairs)
REGR_SXX(Y,X)	Sum of squares (x)	$\sum x^2 - \frac{(\sum x)^2}{n}$
REGR_SYY(Y,X)	Sum of squares (y)	$\sum y^2 - \frac{(\sum y)^2}{n}$
REGR_SXY(Y,X)	Sum of products	$\sum xy - \frac{(\sum y)(\sum x)}{n}$

Verwendung von Unterabfragen

In einer relationalen Datenbank können Sie verknüpfte Daten in mehreren Tabellen speichern. Zusätzlich zum Extrahieren von Daten aus verbundenen Tabellen unter Verwendung eines Joins können Sie sie auch mit einer **Unterabfrage** extrahieren. Eine Unterabfrage ist eine SELECT-Anweisung, die in der SELECT-, WHERE- oder HAVING-Klausel einer übergeordneten SQL-Anweisung verschachtelt ist.

Durch Unterabfragen wird das Schreiben mancher Abfragen einfacher als das Schreiben von Joins. Außerdem gibt es auch Abfragen, die nicht ohne Unterabfragen geschrieben werden können.

Unterabfragen können auf verschiedene Weise kategorisiert werden:

- Ob sie eine oder mehrere Zeilen zurückgeben können (Einzeilen- ggü. Mehrzeilen-Unterabfragen)
- Ob sie korreliert oder nichtkorreliert sind
- Ob sie in einer anderen Unterabfrage verschachtelt sind

Einzeilige und mehrzeilige Unterabfragen

Unterabfragen, die nur eine oder null Zeilen an die äußere Anweisung zurückgeben können, werden **Einzeilen-Unterabfragen** genannt. Einzeilige Unterabfragen können in einer SQL-Anweisung mit oder ohne den Vergleichsoperator überall verwendet werden.

Eine einzeilige Unterabfrage kann innerhalb eines Ausdrucks in der SELECT-Klausel verwendet werden:

```
SELECT (select FIRST T.x FROM T) + 1 as ITEM_1, 2 as ITEM_2,...
```

Alternativ kann eine einzeilige Unterabfrage in einem Ausdruck in der SELECT-Klausel mit einem Vergleichsoperator verwendet werden.

Zum Beispiel:

```
SELECT IF (select FIRST T.x FROM T) >= 10 THEN 1 ELSE 0 ENDIF as ITEM_1, 2 as ITEM_2,...
```

Unterabfragen, die mehr als eine Zeile (aber nur eine Spalte) an die äußere Anweisung zurückgeben können, werden **Mehrzeilen-Unterabfragen** genannt. Mehrzeilen-Unterabfragen sind Unterabfragen, die in einer IN-, ANY-, ALL- oder EXISTS-Klausel verwendet werden.

Beispiel 1: Einzeilen-Unterabfrage

Sie speichern Produktinformationen in einer Tabelle namens "Products", und Informationen zu Bestellungen in der Tabelle "SalesOrdersItems". Die Tabelle "Products" enthält die Daten zu den einzelnen Produkten. Die Tabelle "SalesOrdersItems" enthält Daten zu den Bestellungen der Kunden. Wenn das Unternehmen Produkte nachbestellt, sobald der Lagerstand auf unter 50 Stück sinkt, kann die Frage "Welche Produkte sind unter dem Mindest-Lagerbestand?" mit folgender Abfrage beantwortet werden.

```
SELECT ID, Name, Description, Quantity
FROM Products
WHERE Quantity < 50;
```

Ein zielführenderes Ergebnis allerdings würde berücksichtigen, wie häufig ein Produkt bestellt wird, weil ein geringer Lagerbestand bei häufig nachgefragten Produkten bedenklicher ist als bei solchen, die kaum bestellt werden.

Sie können eine Unterabfrage verwenden, um die durchschnittliche Anzahl von Artikeln zu ermitteln, die ein Kunde bestellt, und dann diesen Mittelwert in der Hauptabfrage verwenden, um Produkte zu finden, für die fast nur der Mindest-Lagerbestand vorhanden ist. Folgende Abfrage sucht die Namen und Beschreibungen der Produkte, deren Anzahl weniger als doppelt so hoch ist wie die durchschnittliche Anzahl der Artikel, die von einem Kunden bestellt werden.

```
SELECT Name, Description
FROM Products WHERE Quantity < 2 * (
    SELECT AVG( Quantity )
    FROM SalesOrderItems
);
```

In der WHERE-Klausel helfen Unterabfragen bei der Auswahl der im Abfrageergebnis anzuzeigenden Zeilen aus den Tabellen, die in der FROM-Klausel angegeben werden. In der HAVING-Klausel helfen Sie bei der Auswahl von Zeilengruppen, die von der GROUP BY-Klausel der Hauptabfrage angegeben wurden und in den Ergebnissen der Abfrage erscheinen sollen.

Beispiel 2: Einzeilen-Unterabfrage

Das folgende Beispiel einer Einzeilen-Unterabfrage berechnet den durchschnittlichen Preis der Produkte in der Products-Tabelle. Der Mittelwert wird anschließend an die WHERE-Klausel der äußeren Abfrage übergeben. Die äußere Abfrage gibt ID, Name und UnitPrice aller Produkte zurück, die weniger teuer als der Mittelwert sind:

```
SELECT ID, Name, UnitPrice
FROM Products
WHERE UnitPrice <
    ( SELECT AVG( UnitPrice ) FROM Products )
ORDER BY UnitPrice DESC;
```

ID	Name	UnitPrice
401	Baseball Cap	10.00
300	Tee Shirt	9.00
400	Baseball Cap	9.00
500	Visor	7.00
501	Visor	7.00

Beispiel 3: Einfache Mehrzeilen-Unterabfrage mit IN

Angenommen, Sie wollen Artikel mit niedrigem Lagerbestand ermitteln und auch Bestellungen für diese Artikel erfassen. Sie könnten eine SELECT-Anweisung ausführen, die eine Unterabfrage in der WHERE-Klausel enthält, und zwar folgendermaßen:

```
SELECT *
FROM SalesOrderItems
```

```
WHERE ProductID IN
( SELECT ID
  FROM Products
  WHERE Quantity < 20 )
ORDER BY ShipDate DESC;
```

In diesem Beispiel erstellt die Unterabfrage eine Liste aller Werte in der Spalte ID der Tabelle "Products", welche die Suchbedingung der WHERE-Klausel erfüllen. Die Unterabfrage gibt anschließend eine Reihe von Zeilen, aber nur eine einzige Spalte zurück. Das Schlüsselwort IN behandelt jeden Wert als Teil einer Menge und testet, ob jede Zeile in der Hauptabfrage Teil der Menge ist.

Beispiel 4: Mehrzeilen-Unterabfragen, die die Verwendung von IN, ANY und ALL vergleichen

Zwei Tabellen in der SQL Anywhere-Beispieldatenbank enthalten Ergebnisse mit Finanzdaten. Bei der Tabelle "FinancialCodes" handelt es sich um eine Tabelle mit den verschiedenen Codes für Finanzdaten und deren Bedeutung. Um die Umsätze ("revenue") aus der Tabelle "FinancialData" aufzulisten, geben Sie Folgendes ein:

```
SELECT *
FROM FinancialData
WHERE Code IN
( SELECT Code
  FROM FinancialCodes
  WHERE type = 'revenue' );
```

Year	Quarter	Code	Amount
1999	Q1	r1	1023
1999	Q2	r1	2033
1999	Q3	r1	2998
1999	Q4	r1	3014
2000	Q1	r1	3114
...

Die Schlüsselwörter ANY und ALL können auf ähnliche Weise verwendet werden. Beispiel: Die folgende Abfrage liefert dieselben Ergebnisse wie die vorherige Abfrage, verwendet jedoch das Schlüsselwort ANY:

```
SELECT *
FROM FinancialData
WHERE FinancialData.Code = ANY
( SELECT FinancialCodes.Code
  FROM FinancialCodes
  WHERE type = 'revenue' );
```

Die Bedingung =ANY ist identisch mit der IN-Bedingung. Darüber hinaus kann ANY aber auch mit Ungleichheits-Operatoren, wie z.B. < oder > verwendet werden, um Unterabfragen flexibler zu gestalten.

Das Schlüsselwort ALL ist ähnlich dem Wort ANY. Folgende Abfrage listet zum Beispiel Finanzdaten auf, bei denen es sich nicht um Umsätze handelt:

```
SELECT *
FROM FinancialData
WHERE FinancialData.Code <> ALL
  ( SELECT FinancialCodes.Code
    FROM FinancialCodes
    WHERE type = 'revenue' );
```

Dies entspricht der folgenden Anweisung mit NOT IN:

```
SELECT *
FROM FinancialData
WHERE FinancialData.Code NOT IN
  ( SELECT FinancialCodes.Code
    FROM FinancialCodes
    WHERE type = 'revenue' );
```

Korrelierte und nichtkorrelierte Unterabfragen

Eine Unterabfrage kann eine Referenz auf ein Objekt enthalten, das in einer übergeordneten Anweisung definiert ist. Dies wird als **äußere Referenz** bezeichnet. Eine Unterabfrage, die eine äußere Referenz enthält, wird als **korrelierte Unterabfrage** bezeichnet. Korrelierte Unterabfragen können nicht unabhängig von der äußeren Abfrage ausgewertet werden, weil die Unterabfrage Werte der übergeordneten Anweisung verwendet. Das heißt, dass die Unterabfrage für jede Zeile in der übergeordneten Anweisung durchgeführt wird. Dadurch hängen die Ergebnisse der Unterabfrage von der aktiven Zeile ab, die in der übergeordneten Anweisung ausgewertet wird.

Die Unterabfrage in der untenstehenden Anweisung liefert beispielsweise einen Wert, der von der aktiven Zeile in der Tabelle "Products" abhängig ist:

```
SELECT Name, Description
FROM Products
WHERE Quantity < 2 * (
  SELECT AVG( Quantity )
  FROM SalesOrderItems
  WHERE Products.ID=SalesOrderItems.ProductID );
```

In diesem Beispiel ist die Spalte "Products ID" in dieser Unterabfrage die äußere Referenz. Die Abfrage holt die Namen und Beschreibungen der Produkte, deren Lagerbestand geringer als das Doppelte der durchschnittlich bestellten Menge des von der WHERE-Klausel in der Hauptabfrage getesteten Produkts ist. Die Unterabfrage führt dies durch, indem sie die Tabelle "SalesOrderItems" durchsucht. Die Spalte "Products.ID" in der WHERE-Klausel der Unterabfrage bezieht sich jedoch auf eine Spalte in der Tabelle aus der FROM-Klausel der *Haupt*-Abfrage, und nicht der Unterabfrage. Wenn der Datenbankserver durch die einzelnen Zeilen der Tabelle "Products" geht, verwendet er den ID-Wert der aktuellen Zeile bei der Auswertung der WHERE-Klausel der Unterabfrage.

Eine Abfrage wird ohne Fehler ausgeführt, wenn eine in einer Unterabfrage referenzierte Spalte in der Tabelle nicht existiert, die von der FROM-Klausel der Unterabfrage referenziert wird, hingegen in einer Tabelle vorhanden ist, die von der FROM-Klausel der äußeren Abfrage referenziert wird. SQL Anywhere qualifiziert implizit die Spalte in der Unterabfrage mit dem Tabellennamen der äußeren Abfrage.

Eine Unterabfrage, die keine Referenzen auf Objekte in einer übergeordneten Anweisung enthält, wird **nichtkorrelierte Unterabfrage** genannt. Im untenstehenden Beispiel berechnet die Unterabfrage genau einen Wert: die durchschnittliche Menge aus der Tabelle "SalesOrderItems". Bei der Auswertung der

Abfrage berechnet der Datenbankserver den Wert nur einmal und vergleicht jeden Wert im Feld "Quantity" der Tabelle "Products" damit, um zu ermitteln, ob die entsprechende Zeile ausgewählt werden soll.

```
SELECT Name, Description
FROM Products
WHERE Quantity < 2 * (
    SELECT AVG( Quantity )
    FROM SalesOrderItems );
```

Verschachtelte Unterabfragen

Eine **verschachtelte Unterabfrage** ist eine Unterabfrage, die in einer anderen Unterabfrage verschachtelt ist. Es gibt keine Beschränkung für die Stufen von Unterabfragen-Verschachtelungen, die Sie festlegen können, aber Abfragen mit drei oder mehr Stufen brauchen für die Ausführung deutlich länger als kleinere Abfragen.

Das folgende Beispiel verwendet verschachtelte Unterabfragen, um die Bestellungen-IDs und Zeilen-IDs von jenen Bestellungen zu bestimmen, die am selben Tag versandt wurden, an dem ein bestimmter Artikel in der Abteilung "Fees" bestellt wurde.

```
SELECT ID, LineID
FROM SalesOrderItems
WHERE ShipDate = ANY (
    SELECT OrderDate
    FROM SalesOrders
    WHERE FinancialCode IN (
        SELECT Code
        FROM FinancialCodes
        WHERE ( Description = 'Fees' ) ) );
```

ID	LineID
2001	1
2001	2
2001	3
2002	1
...	...

In diesem Beispiel erzeugt die innerste Unterabfrage eine Spalte mit Finanzcodes, deren Beschreibung "Fees" lautet.

```
SELECT Code
FROM FinancialCodes
WHERE ( Description = 'Fees' );
```

Die nächste Unterabfrage findet die Bestelldaten der Artikel, deren Code zu einem der in der innersten Unterabfrage ausgewählten Codes passen.

```
SELECT OrderDate
FROM SalesOrders
WHERE FinancialCode
IN ( subquery-expression );
```

Schließlich findet die äußerste Abfrage die Bestellungs-IDs und Zeilen-IDs der Artikel, die an einem der Daten versandt wurde, die in der Unterabfrage gefunden wurden.

```
SELECT ID, LineID
FROM SalesOrderItems
WHERE ShipDate = ANY ( subquery-expression );
```

Verwendung von Unterabfragen anstelle von Joins

Angenommen, Sie benötigen eine chronologische Liste der Aufträge und die Angabe der Firma, die sie erteilt hat, doch Sie möchten den Firmennamen anstelle der Kunden-ID erhalten. Um dieses Ergebnis zu erhalten, verwenden Sie einen Join.

Join verwenden

Um die Auftrags-ID, das Datum und den Firmennamen aller Aufträge seit Anfang 2001 aufzulisten, führen Sie folgende Abfrage aus:

```
SELECT SalesOrders.ID,
       SalesOrders.OrderDate,
       Customers.CompanyName
FROM SalesOrders
KEY JOIN Customers
WHERE OrderDate > '2001/01/01'
ORDER BY OrderDate;
```

Unterabfrage verwenden

Folgende Anweisung liefert dasselbe Ergebnis, wenn sie anstelle eines Joins eine Unterabfrage verwendet:

```
SELECT SalesOrders.ID,
       SalesOrders.OrderDate,
       ( SELECT CompanyName FROM Customers
         WHERE Customers.ID = SalesOrders.CustomerID )
FROM SalesOrders
WHERE OrderDate > '2001/01/01'
ORDER BY OrderDate;
```

Die Unterabfrage bezieht sich auf die Spalte "CustomerID" in der Auftrags-tabelle "SalesOrders", obwohl die Tabelle "SalesOrders" kein Bestandteil der Unterabfrage ist. Stattdessen bezieht sich die Spalte "SalesOrders.CustomerID" auf die Tabelle "SalesOrders" im Hauptteil der Anweisung.

Eine Unterabfrage kann immer dann anstelle eines Joins verwendet werden, wenn nur eine Spalte aus der anderen Tabelle benötigt wird. (Beachten Sie, dass Unterabfragen nur eine Spalte zurückgeben können.) In diesem Beispiel benötigen Sie nur die Spalte "CompanyName". Daher kann der Join in eine Unterabfrage geändert werden.

Outer-Joins verwenden

Um alle Kunden im US-Bundesstaat Washington zusammen mit ihren letzten Auftrags-IDs aufzulisten, führen Sie folgende Abfrage aus:

```
SELECT  CompanyName, State,
        ( SELECT MAX( ID )
          FROM SalesOrders
          WHERE SalesOrders.CustomerID = Customers.ID )
FROM Customers
WHERE State = 'WA';
```

CompanyName	State	MAX(SalesOrders.ID)
Custom Designs	WA	2547
It's a Hit!	WA	(NULL)

Die Firma "It's a Hit!" hat keine Aufträge erteilt, und die Unterabfrage gibt für diesen Kunden NULL zurück. Firmen, die keine Aufträge erteilt haben, werden bei der Verwendung eines Inner-Joins nicht aufgelistet.

Sie können auch einen Outer-Join explizit festlegen. In diesem Fall ist eine GROUP BY-Klausel erforderlich.

```
SELECT  CompanyName, State,
        MAX( SalesOrders.ID )
FROM Customers
KEY LEFT OUTER JOIN SalesOrders
WHERE State = 'WA'
GROUP BY CompanyName, State;
```

Unterabfragen in der WHERE-Klausel

Unterabfragen in der WHERE-Klausel funktionieren als Teil des Zeilenauswahlprozesses. Sie benutzen eine Unterabfrage in der WHERE-Klausel, wenn die Kriterien, die Sie für die Auswahl der Zeilen verwenden, von den Ergebnissen einer anderen Tabelle abhängen.

Beispiel

Sie suchen die Produkte, deren Lagermengen weniger als das Doppelte der durchschnittlich bestellten Menge betragen.

```
SELECT Name, Description
FROM Products WHERE Quantity < 2 * (
  SELECT AVG( Quantity )
  FROM SalesOrderItems );
```

Dies ist eine Abfrage in zwei Schritten: Zuerst wird die durchschnittliche Anzahl der Artikel pro Bestellung ermittelt und dann wird gesucht, welche Produkte weniger als das Doppelte dieser Menge betragen.

Die Abfrage in zwei Schritten

Die Spalte "Quantity" der Tabelle "SalesOrderItems" speichert die *Anzahl* der Artikel, die pro Artikeltyp, Kunde und Bestellung angefordert werden. Die Unterabfrage lautet wie folgt:

```
SELECT AVG( Quantity )
FROM SalesOrderItems;
```

Sie gibt die durchschnittliche Menge der Artikel in der Tabelle "SalesOrderItems" zurück, und zwar 25,851413.

Die nächste Abfrage gibt die Namen und Beschreibungen der Artikel zurück, deren Lagermengen weniger als doppelt so hoch sind wie der vorher abgefragte Wert.

```
SELECT Name, Description
FROM Products
WHERE Quantity < 2*25.851413;
```

Mit einer Unterabfrage werden die beiden Schritte in einem einzigen Vorgang zusammengefasst.

Zweck einer Unterabfrage in der WHERE-Klausel

Eine Unterabfrage in der WHERE-Klausel ist Teil einer Suchbedingung.

Siehe auch

- „Abfragen“ auf Seite 291

Unterabfragen in der HAVING-Klausel

Obwohl Unterabfragen normalerweise als Suchbedingungen in der WHERE-Klausel vorkommen, werden sie manchmal auch in der HAVING-Klausel einer Abfrage benutzt. Wenn eine Unterabfrage in der HAVING-Klausel erscheint, wird sie wie jeder andere Ausdruck in der HAVING-Klausel als Teil der Zeilengruppenauswahl benutzt.

Nachfolgend finden Sie eine Anforderung, die auf natürliche Weise zu einer Abfrage mit einer Unterabfrage in der HAVING-Klausel führt: "Die durchschnittliche Lagermenge welcher Artikel ist mehr als das Doppelte der durchschnittlichen Anzahl der pro Kunde bestellten Artikel?"

Beispiel

```
SELECT Name, AVG( Quantity )
FROM Products
GROUP BY Name
HAVING AVG( Quantity ) > 2* (
    SELECT AVG( Quantity )
    FROM SalesOrderItems
);
```

name	AVG(Products.Quantity)
Baseball Cap	62.000000
Shorts	80.000000

name	AVG(Products.Quantity)
Tee Shirt	52.333333

Die Abfrage wird folgendermaßen ausgeführt:

- Die Unterabfrage berechnet die durchschnittliche Menge der Artikel in der Tabelle "SalesOrderItems".
- Die Hauptabfrage durchläuft dann die Tabelle "Products", berechnet die Durchschnittsproduktmenge und gruppiert die Artikel nach Produktnamen.
- Die HAVING-Klausel prüft, ob jede Durchschnittsmenge mehr als das Doppelte der Menge ist, die von der Unterabfrage gefunden wurde. Wenn dies so ist, gibt die Hauptabfrage diese Zeilengruppe zurück, sonst nicht.
- Die SELECT-Klausel ergibt eine Summenzeile für jede Gruppe und zeigt die Namen der einzelnen Artikel und ihre durchschnittliche Lagermenge an.

Sie können auch äußere Referenzen in einer HAVING-Klausel verwenden, wie dies in der nachstehenden Abfrage gezeigt wird, die eine geringfügige Abweichung der oben gezeigten Anforderung darstellt.

Beispiel

In diesem Beispiel werden die Produkt-ID-Nummern und die Zeilen-ID-Nummern der Produkte gesucht, deren durchschnittlich bestellte Mengen mehr als die Hälfte der Lagermengen dieser Produkte betragen.

```
SELECT ProductID, LineID
FROM SalesOrderItems
GROUP BY ProductID, LineID
HAVING 2* AVG( Quantity ) > (
    SELECT Quantity
    FROM Products
    WHERE Products.ID = SalesOrderItems.ProductID );
```

ProductID	LineID
601	3
601	2
601	1
600	2
...	...

In diesem Beispiel muss die Unterabfrage die Lagermenge des Produkts ergeben, das der Zeilengruppe entspricht, die von der HAVING-Klausel getestet wurde. Die Unterabfrage wählt Datensätze für diesen Artikel aus, indem die äußere Referenz "SalesOrderItems.ProductID". verwendet wird.

Eine Unterabfrage mit einem Vergleich gibt einen einzelnen Wert zurück

Diese Abfrage benutzt den Vergleich ">" und nimmt an, dass die Unterabfrage exakt einen Wert zurückgeben muss. In diesem Fall trifft das zu. Da das ID-Feld der Tabelle "Products" ein Primärschlüssel ist, gibt es in der Tabelle "Products" jeweils nur einen Datensatz für eine bestimmte Produkt-ID.

Prädikate mit Unterabfragen

Da eine Unterabfrage nur ein Ausdruck ist, der in den Klauseln WHERE oder HAVING erscheinen kann, können Ihnen die Suchbedingungen in Unterabfragen bekannt vorkommen.

Dazu gehören:

- **Vergleichsprädikate mit Unterabfragen** Vergleicht den Wert eines Ausdrucks mit einem einzelnen Wert, der von der Unterabfrage für jeden Datensatz in den Tabellen der Hauptabfrage gebildet wird. Vergleichstests verwenden die Operatoren (=, <>, <, <=, >, >=), die mit der Unterabfrage übergeben werden.
- **Quantifizierter Vergleichstest** Damit wird der Wert eines Ausdrucks mit jedem Bestandteil einer Gruppe von Werten verglichen, die von einer Unterabfrage gebildet wird.
- **Test der Zugehörigkeit zur Wertegruppe einer Unterabfrage** Damit wird geprüft, ob der Wert eines Ausdrucks zu einem Wert in einer Gruppe von Werten passt, die von einer Unterabfrage gebildet wird.
- **Existenztest** Prüft, ob die Unterabfrage Zeilen zurückgibt.

Siehe auch

- [„Abfragen“ auf Seite 291](#)

Unterabfrage-Vergleichstest

Der Unterabfrage-Vergleichstest (=, <>, <, <=, >, >=) ist eine modifizierte Version des einfachen Vergleichstests. Der einzige Unterschied zwischen den beiden besteht darin, dass im Ersteren der Ausdruck, der dem Operator folgt, eine Unterabfrage ist. Dieser Test wird verwendet, um einen Wert aus einer Zeile in der Hauptabfrage mit einem *einzelnen* Wert zu vergleichen, der von der Unterabfrage erstellt wird.

Beispiel

Diese Abfrage enthält ein Beispiel für einen Unterabfrage-Vergleichstest:

```
SELECT Name, Description, Quantity
FROM Products
WHERE Quantity < 2 * (
    SELECT AVG( Quantity )
    FROM SalesOrderItems );
```

name	Description	Quantity
Tee Shirt	Tank Top	28
Baseball Cap	Wool cap	12
Visor	Cloth Visor	36
Visor	Plastic Visor	28
...

Folgende Unterabfrage ruft einen einzelnen Wert (die Durchschnittsmenge von Artikeln jedes Typs pro Kundenbestellung) aus der Tabelle "SalesOrderItems" ab.

```
SELECT AVG( Quantity )  
FROM SalesOrderItems;
```

Die Hauptabfrage vergleicht dann den Lagerbestand jedes Artikels mit diesem Wert.

Eine Unterabfrage in einem Vergleichstest gibt genau einen Wert zurück

Eine Unterabfrage in einem Vergleichstest muss genau einen Wert zurückgeben. Sehen Sie sich die folgende Abfrage an, deren Unterabfrage zwei Spalten aus der Tabelle "SalesOrderItems" filtert:

```
SELECT Name, Description, Quantity  
FROM Products  
WHERE Quantity < 2 * (  
    SELECT AVG( Quantity ), MAX( Quantity )  
    FROM SalesOrderItems);
```

Es wird ein Fehler zurückgegeben.

Siehe auch

- „Unterabfrage nur in einem Element der SELECT-Liste zulässig“ [[Fehlermeldungen](#)]

Unterabfragen und der IN-Test

Sie können den Test der Zugehörigkeit zur Wertegruppe einer Unterabfrage benutzen, um einen Wert aus der Hauptabfrage mit mehr als einem Wert für die Unterabfrage zu vergleichen.

Der Test der Zugehörigkeit zur Wertegruppe einer Unterabfrage vergleicht einen einzelnen Datenwert für jede Zeile in der Hauptabfrage mit einer einzelnen Spalte von Datenwerten, die von der Unterabfrage produziert werden. Wenn der Datenwert der Hauptabfrage zu *einem* der Datenwerte in der Spalte passt, ergibt die Unterabfrage TRUE.

Beispiel

Es werden die Namen der Mitarbeiter gesucht, die die Abteilungen "Shipping" oder "Finance" leiten.

```
SELECT GivenName, Surname  
FROM Employees
```

```

WHERE EmployeeID IN (
  SELECT DepartmentHeadID
  FROM Departments
  WHERE ( DepartmentName='Finance' OR
          DepartmentName = 'Shipping' ) );

```

GivenName	Surname
Mary Anne	Shea
Jose	Martinez

Die Unterabfrage in diesem Beispiel holt aus der Tabelle "Departments" die ID-Nummern, die den Leitern der Abteilungen "Shipping" und "Finance" entsprechen. Die Hauptabfrage gibt dann die Namen der Mitarbeiter zurück, deren ID-Nummern zu einem der beiden von der Unterabfrage gefundenen Namen passen.

```

SELECT DepartmentHeadID
FROM Departments
WHERE ( DepartmentName='Finance' OR
        DepartmentName = 'Shipping' );

```

Der Test der Zugehörigkeit zu einer Gruppe ist gleichwertig zum ANY-Test.

Der Test der Zugehörigkeit zu einer Gruppe ist gleichwertig zum ANY-Test. Die folgende Abfrage ist gleichwertig zu der Abfrage aus dem vorherigen Beispiel.

```

SELECT GivenName, Surname
FROM Employees
WHERE EmployeeID = ANY (
  SELECT DepartmentHeadID
  FROM Departments
  WHERE ( DepartmentName='Finance' OR
          DepartmentName = 'Shipping' ) );

```

Negativinterpretation des Tests der Zugehörigkeit zu einer Gruppe

Der Test der Zugehörigkeit zur Wertegruppe einer Unterabfrage kann auch benutzt werden, um jene Zeilen abzufragen, deren Spaltenwerte nicht denen entsprechen, die von einer Unterabfrage ausgewertet wurden. Um einen Test der Zugehörigkeit zu einer Gruppe negativ zu interpretieren, setzen Sie das Wort NOT vor das Schlüsselwort IN.

Beispiel

Die Unterabfrage in dieser Abfrage gibt Vornamen und Nachnamen der Mitarbeiter zurück, die nicht Abteilungsleiter von "Finance" oder "Shipping" sind.

```

SELECT GivenName, Surname
FROM Employees
WHERE EmployeeID NOT IN (
  SELECT DepartmentHeadID
  FROM Departments
  WHERE ( DepartmentName='Finance' OR
          DepartmentName = 'Shipping' ) );

```

Unterabfragen und der ANY-Test

Der ANY-Test, der mit einem der SQL-Vergleichsoperatoren (=, >, <, >=, <=, !=, <>, !>, !<) verwendet wird, vergleicht einen einzelnen Wert mit der von der Unterabfrage erzeugten Spalte von Datenwerten. Um den Test durchzuführen, benutzt SQL den angegebenen Vergleichsoperator, um den Testwert mit jedem Datenwert in der Spalte zu vergleichen. Wenn *irgendeiner* (*any*) der Vergleiche das Ergebnis TRUE ausgibt, wird vom ANY-Test der Wert TRUE übergeben.

Eine Unterabfrage, die mit ANY benutzt wird, muss eine einzelne Spalte zurückgeben.

Beispiel

Es sollen Bestellnummern und Kundennummern derjenigen Bestellungen gesucht werden, die aufgegeben wurden, nachdem das erste Produkt der Bestellung 2005 ausgeliefert wurde.

```
SELECT ID, CustomerID
FROM SalesOrders
WHERE OrderDate > ANY (
    SELECT ShipDate
    FROM SalesOrderItems
    WHERE ID=2005 );
```

ID	CustomerID
2006	105
2007	106
2008	107
2009	108
...	...

Beim Ausführen dieser Abfrage prüft die Hauptabfrage die Bestelldaten für jede Bestellung anhand der Versanddaten *aller* Produkte der Bestellung 2005. Wenn ein Bestelldatum nach dem Versanddatum für *eine* Lieferung der Bestellung 2005 liegt, werden die ID und Kunden-ID aus der Tabelle "SalesOrders" in die Ergebnismenge übernommen. Der ANY-Test gestaltet sich analog zum OR-Operator. Die oben genannte Abfrage kann wie folgt gelesen werden: "Wurde diese Bestellung aufgegeben, nachdem das erste Produkt der Bestellung 2005 versandt wurde, oder nachdem das zweite Produkt der Bestellung 2005 versandt wurde, oder..."

Funktion des ANY-Operators

Der ANY-Operator kann Verwirrung stiften. Man könnte versucht sein, die Abfrage wie folgt zu lesen: "Gib alle Bestellungen zurück, die aufgegeben wurden, nachdem ein beliebiges Produkt der Bestellung 2005 versandt wurde". Das bedeutet aber, dass die Abfrage die Nummern der Bestellungen und der Kunden für die Bestellungen ausgibt, die aufgegeben wurden, nachdem *alle* Produkte der Bestellung 2005 versandt wurden - dies wird allerdings von der Abfrage nicht erfüllt.

Sie sollten daher die Abfrage wie folgt lesen: "Gib die Nummern der Bestellungen und der Kunden für jene Bestellungen zurück, die aufgegeben wurden, nachdem *mindestens* ein Produkt der Bestellung 2005

versandt wurde". Wenn Sie das Schlüsselwort **SOME** (EINIGE) verwenden würden, könnte die Abfrage etwas einfacher verstanden werden. Die nachstehende Abfrage ist der vorherigen Abfrage gleichwertig.

```
SELECT ID, CustomerID
FROM SalesOrders
WHERE OrderDate > SOME (
    SELECT ShipDate
    FROM SalesOrderItems
    WHERE ID=2005 );
```

Das Schlüsselwort **SOME** hat denselben Wert wie das Schlüsselwort **ANY**.

Hinweise zum ANY-Operator

Es gibt zwei weitere wichtige Eigenheiten des ANY-Tests:

- **Leere Unterabfragen-Ergebnismenge** Wenn die Unterabfrage eine leere Ergebnismenge produziert, wird der ANY-Test als **FALSE** ausgewertet. Dies ist sinnvoll: Wenn es keine Ergebnisse gibt, dann ist es auch nicht wahr, dass mindestens ein Ergebnis den Vergleichstest bestanden hat.
- **NULL in der Ergebnismenge der Unterabfrage** Angenommen es gibt mindestens einen **NULL**-Wert in der Ergebnismenge der Unterabfrage. Wenn der Vergleichstest für alle Nicht-**NULL**-Werte in der Ergebnismenge **FALSE** ergibt, gibt die ANY-Suche **UNKNOWN** zurück. Dies ist deshalb so, weil Sie in dieser Situation nicht eindeutig folgern können, ob es einen Wert für die Unterabfrage gibt, für die der Vergleichstest ausgeführt wird. Es könnte einen Wert geben, oder aber auch nicht, abhängig von den *korrekten* Werten für die **NULL**-Daten in der Ergebnismenge.

Siehe auch

- „ANY- und SOME-Suchbedingungen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Unterabfragen und der ALL-Test

Der **ALL**-Test wird mit einem der SQL-Vergleichsoperatoren (**=**, **>**, **<**, **>=**, **<=**, **!=**, **<>**, **!>**, **!<**) verwendet, um einen einzelnen Wert mit den von der Unterabfrage erzeugten Datenwerten zu vergleichen. Um den Test durchzuführen, benutzt SQL den angegebenen Vergleichsoperator, um den Testwert mit jedem Datenwert in der Ergebnismenge zu vergleichen. Wenn alle Vergleiche mit einem **TRUE**-Ergebnis enden, gibt der **ALL**-Test den Wert **TRUE** zurück.

Beispiel

Dieses Beispiel sucht die Bestellnummern und Kundennummern derjenigen Bestellungen, die aufgegeben wurden, nachdem alle Produkte der Bestellung Nr. 2001 versandt wurden.

```
SELECT ID, CustomerID
FROM SalesOrders
WHERE OrderDate > ALL (
    SELECT ShipDate
    FROM SalesOrderItems
    WHERE ID=2001 );
```

ID	CustomerID
2002	102
2003	103
2004	104
2005	101
...	...

Beim Ausführen dieser Abfrage prüft die Hauptabfrage die Bestelldaten für jede Bestellung anhand der Versanddaten *aller* Produkte der Bestellung 2001. Wenn ein Bestelldatum nach dem Versanddatum für *alle* Lieferungen der Bestellung 2001 liegt, werden die ID und Kunden-ID aus der Tabelle "SalesOrders" in die Ergebnismenge übernommen. Der ALL-Test ist analog zum AND-Operator: Die oben stehende Abfrage kann wie folgt gelesen werden: "Wurde diese Bestellung aufgegeben, bevor das erste Produkt der Bestellung 2001 versandt wurde, und bevor das zweite Produkt der Bestellung 2001 versandt wurde, und ...".

Hinweise zum ALL-Operator

Es gibt drei weitere wichtige Eigenheiten des ALL-Tests:

- **Leere Unterabfragen-Ergebnismenge** Wenn die Unterabfrage eine leere Ergebnismenge produziert, wird der ALL-Test als TRUE ausgewertet. Dies ist sinnvoll, denn wenn es keine Ergebnisse gibt, dann ist es wahr, dass alle Werte in der Ergebnismenge den Vergleichstest bestanden haben.
- **NULL in der Ergebnismenge der Unterabfrage** Wenn der Vergleichstest für beliebige Werte in der Ergebnismenge FALSE ergibt, gibt die ALL-Suche ebenfalls FALSE zurück. Sie gibt TRUE zurück, wenn alle Werte TRUE sind. Sonst gibt sie UNKNOWN zurück - zum Beispiel kann dies vorkommen, wenn NULL in der Ergebnismenge der Unterabfrage vorhanden ist, die Suchbedingung aber TRUE für alle Nicht-NULL-Werte ausgibt.
- **Negieren des ALL-Tests** Die folgenden Ausdrücke sind *nicht* entsprechend.

```
NOT a = ALL (subquery)
a <> ALL (subquery)
```

Siehe auch

- „Unterabfrage, die auf ANY, ALL oder SOME folgt“ auf Seite 619

Unterabfragen und der EXISTS-Test

Unterabfragen, die im Unterabfrage-Vergleichstest und im Test der Zugehörigkeit zu einer Gruppe verwendet werden, geben Datenwerte aus der Tabelle der Unterabfrage zurück. Es gibt Situationen, in denen Sie herausfinden wollen, ob die Unterabfrage *irgendein* Ergebnis zurückgibt, und nicht *welche* Ergebnisse. Der Existenztest (EXISTS) prüft, ob eine Unterabfrage eine Zeile mit Abfrageergebnissen

ausgibt. Wenn die Unterabfrage eine oder mehrere Zeilen mit Ergebnissen produziert, gibt der EXISTS-Test den Wert TRUE zurück. Andernfalls wird der Wert FALSE zurückgegeben.

Beispiel

In diesem Beispiel sehen wir eine Anforderung, die mit einer Unterabfrage ausgedrückt wird: "Welche Kunden haben nach dem 13.07.01 Bestellungen aufgegeben?"

```
SELECT GivenName, Surname
FROM Customers
WHERE EXISTS (
    SELECT *
    FROM SalesOrders
    WHERE ( OrderDate > '2001-07-13' ) AND
          ( Customers.ID = SalesOrders.CustomerID ) );
```

GivenName	Surname
Almen	de Joie
Grover	Pendelton
Ling Ling	Andrews
Bubba	Murphy

Erklärung des Existenztests

Hier prüft die Unterabfrage für jede Zeile in der Tabelle "Customers", ob die Kunden-ID zu einer Bestellung nach dem 13.07.01 gehört. Wenn dies der Fall ist, selektiert die Abfrage die Vor- und Nachnamen des Kunden aus der Haupttabelle.

Der EXISTS-Test benutzt die Ergebnisse der Unterabfrage nicht für die Weiterverarbeitung. Er prüft nur, ob die Unterabfrage Zeilen selektieren wird. Somit liefert der Existenztest für die beiden folgenden Unterabfragen dieselben Ergebnisse: Dies sind Unterabfragen, die nicht für sich alleine verarbeitet werden können, da sie sich auf die Tabelle "Customers" beziehen, die Teil der Hauptabfrage ist, nicht aber Teil der Unterabfrage.

```
SELECT *
FROM Customers, SalesOrders
WHERE ( OrderDate > '2001-07-13' ) AND
      ( Customers.ID = SalesOrders.CustomerID )

SELECT OrderDate
FROM Customers, SalesOrders
WHERE ( OrderDate > '2001-07-13' ) AND
      ( Customers.ID = SalesOrders.CustomerID );
```

Es ist unerheblich, welche Spalten aus der Tabelle "SalesOrders" in der SELECT-Anweisung erscheinen, es wird aber üblicherweise die Schreibweise "SELECT *" verwendet.

Existenztest negieren

Sie können die Logik des EXISTS-Tests mit NOT EXISTS umkehren. In diesem Fall gibt der Test TRUE zurück, wenn die Unterabfrage keine Zeilen ergibt, und FALSE, wenn Zeilen gefunden werden.

Korrelierte Unterabfragen

Sie haben vielleicht festgestellt, dass die Unterabfrage eine Referenz auf die ID-Spalte der Tabelle "Customers" enthält. Eine Referenz auf Spalten oder Ausdrücke in der oder den Haupttabellen wird **äußere Referenz** genannt, und die Unterabfrage ist **korreliert**. Konzeptgemäß verarbeitet SQL die oben genannte Abfrage, indem die Tabelle "Customers" durchsucht und die Unterabfrage für jeden Kunden abgearbeitet wird. Wenn das Bestelldatum in der Tabelle "SalesOrders" nach dem 13.07.01 liegt und die Kunden-ID in den Tabellen "Customers" und "SalesOrders" zueinander passen, werden die Vor- und Nachnamen der Kunden aus der Tabelle "Customers" angezeigt. Da die Unterabfrage die Hauptabfrage referenziert, gibt die Unterabfrage in diesem Abschnitt - im Unterschied zu den Unterabfragen der vorherigen Abschnitte - einen Fehler aus, wenn Sie versuchen, die Unterabfrage allein auszuführen.

Siehe auch

- [„Korrelierte und nichtkorrelierte Unterabfragen“ auf Seite 603](#)

Optimierer: Automatische Konvertierung von Unterabfragen in Joins

Der Abfrageoptimierer erstellt viele der Abfragen, die Unterabfragen verwenden, automatisch neu als Joins. Die Umwandlung erfolgt ohne Eingriff des Benutzers. In diesem Abschnitt wird beschrieben, welche Unterabfragen in Joins umgewandelt werden können, sodass Sie die Performance von Abfragen in Ihrer Datenbank verstehen können.

Die Kriterien, die erfüllt werden müssen, damit eine vielstufige Abfrage mit Joins umgeschrieben werden kann, unterscheiden sich bei den verschiedenen Arten von Operatoren und bei den Strukturen der Abfrage und der Unterabfrage. Wie bereits gesagt: Wenn eine Unterabfrage in der WHERE-Klausel der Abfrage erscheint, hat sie folgendes Format:

```
SELECT select-list
FROM table
WHERE
[NOT] expression comparison-operator ( subquery-expression )
|[NOT] expression comparison-operator { ANY | SOME } ( subquery-expression )
|[NOT] expression comparison-operator ALL ( subquery-expression )
|[NOT] expression IN ( subquery-expression )
|[NOT] EXISTS ( subquery-expression )
GROUP BY group-by-expression
HAVING search-condition
```

Beispiel: Die Anforderung lautet "Wann haben Frau Clarke und Suresh ihre Bestellungen aufgegeben, und welcher Verkäufer hat sie bedient?" Diese Anforderung kann mit folgender Abfrage abgearbeitet werden:

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders
WHERE CustomerID IN (
    SELECT ID
    FROM Customers
    WHERE Surname = 'Clarke' OR GivenName = 'Suresh' );
```

OrderDate	SalesRepresentative
2001-01-05	1596
2000-01-27	667
2000-11-11	467
2001-02-04	195
...	...

Die Unterabfrage ergibt eine Liste von Kunden-IDs, die den beiden Kunden entsprechen, deren Namen in der WHERE-Klausel gelistet sind, und die Hauptabfrage findet die Bestelldaten und den Verkäufer, die den Bestellungen dieser beiden Kunden entsprechen.

Dieselbe Frage kann auch mit Joins beantwortet werden. Nachstehend finden Sie eine andere Form der Abfrage mit einem Join über zwei Tabellen:

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders, Customers
WHERE CustomerID=Customers.ID AND
      ( Surname = 'Clarke' OR GivenName = 'Suresh' );
```

Diese Form der Abfrage verknüpft die Tabelle "SalesOrders" mit der Tabelle "Customers", um die Bestellungen für die einzelnen Kunden zu finden, und gibt dann nur die Datensätze für "Suresh" und "Clarke" zurück.

Fall, bei dem eine Unterabfrage funktioniert, aber nicht ein Join

Es gibt Fälle, bei denen eine Unterabfrage funktioniert, aber ein Join es nicht tut. Zum Beispiel:

```
SELECT Name, Description, Quantity
FROM Products
WHERE Quantity < 2 * (
      SELECT AVG( Quantity )
      FROM SalesOrderItems );
```

name	Description	Quantity
Tee Shirt	Tank Top	28
Baseball Cap	Wool cap	12
Visor	Cloth Visor	36
...

In diesem Fall ist die innere Abfrage eine Zusammenfassung, die äußere Abfrage hingegen nicht, sodass die beiden Abfragen nicht in einem einfachen Join kombiniert werden können.

Siehe auch

- „Joins: Daten aus mehreren Tabellen abrufen“ auf Seite 492

Unterabfrage, die einem Vergleichsoperator folgt

Eine Unterabfrage, die einem Vergleichsoperator (=, >, <, >=, <=, !=, <>, !>, !<) folgt, wird ein Vergleich genannt. Der Optimierer konvertiert diese Unterabfragen in Joins, sofern die Unterabfrage folgende Bedingungen erfüllt:

- Sie gibt genau einen Wert für jede Zeile in der Hauptabfrage zurück.
- Sie enthält keine GROUP BY-Klausel.
- Die Unterabfrage enthält kein Schlüsselwort DISTINCT.
- Die Unterabfrage ist keine UNION-Abfrage.
- Die Unterabfrage ist keine Aggregatabfrage.

Beispiel

Nehmen Sie folgende Anforderung an: "Wann wurden die Produkte von Suresh bestellt und von welchem Verkäufer?"

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders
WHERE CustomerID = (
    SELECT ID
    FROM Customers
    WHERE GivenName = 'Suresh' );
```

Diese Abfrage entspricht dem Kriterium und kann daher in eine Abfrage umgewandelt werden, die einen Join benutzt:

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders, Customers
WHERE CustomerID=Customers.ID AND
    ( Surname = 'Clarke' OR GivenName = 'Suresh' );
```

Die Anforderung "Finde die Produkte, deren Lagermengen weniger als das Doppelte der durchschnittlich bestellten Menge betragen" kann nicht in einen Join umgewandelt werden, da die Unterabfrage eine Aggregatfunktion enthält, nämlich AVG:

```
SELECT Name, Description
FROM Products
WHERE Quantity < 2 * (
    SELECT AVG( Quantity )
    FROM SalesOrderItems );
```

Unterabfrage, die auf ANY, ALL oder SOME folgt

Eine Unterabfrage, die auf die Schlüsselwörter ANY, ALL oder SOME folgt, wird als quantifizierter Vergleich bezeichnet. Der Optimierer konvertiert diese Unterabfragen in Joins, sofern folgende Bedingungen erfüllt sind:

- Die Hauptabfrage enthält keine GROUP BY-Klausel und ist keine Aggregatabfrage, oder die Unterabfrage gibt genau einen Wert zurück.
- Die Unterabfrage enthält keine GROUP BY-Klausel.
- Die Unterabfrage enthält kein DISTINCT-Schlüsselwort.
- Die Unterabfrage ist keine UNION-Abfrage.
- Die Unterabfrage ist keine Aggregatabfrage.
- Die Verbindung '*expressioncomparison-operator* { **ANY** | **SOME** } (*subquery-expression*)' darf nicht negiert werden.
- Die Verbindung '*expressioncomparison-operator* (**ALL** (*subquery-expression*)' muss negiert werden.

Die ersten vier Bedingungen sind relativ klar.

Beispiel

Die Anforderung, "Wann haben Frau Clarke und Suresh ihre Bestellungen aufgegeben, und welcher Verkäufer hat sie bedient?", kann als Unterabfrage behandelt werden:

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders
WHERE CustomerID = ANY (
    SELECT ID
    FROM Customers
    WHERE Surname = 'Clarke' OR GivenName = 'Suresh' );
```

Sie kann alternativ auch in einem Join-Format verarbeitet werden:

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders, Customers
WHERE CustomerID=Customers.ID AND
    ( Surname = 'Clarke' OR GivenName = 'Suresh' );
```

Anders bei der Anforderung "Wann haben Frau Clarke, Suresh und ein Mitarbeiter, der auch Kunde ist, ihre Bestellungen aufgegeben?". Sie muss als Union-Abfrage formuliert werden, und eine Umwandlung in einen Join ist daher nicht möglich:

```
SELECT OrderDate, SalesRepresentative
FROM SalesOrders
WHERE CustomerID = ANY (
    SELECT ID
    FROM Customers
    WHERE Surname = 'Clarke' OR GivenName = 'Suresh'
    UNION
    SELECT EmployeeID
    FROM Employees );
```

Ähnlich würde die Anforderung "Finde die Bestellungen-IDs und die Kunden-IDs derjenigen Bestellungen, die nicht nach den ersten Versanddaten aller Produkte versandt wurden" als Aggregatabfrage ausgedrückt werden. Sie kann daher nicht in einen Join umgewandelt werden:

```
SELECT ID, CustomerID
FROM SalesOrders
WHERE NOT OrderDate > ALL (
    SELECT FIRST ( ShipDate )
    FROM SalesOrderItems
    ORDER BY ShipDate );
```

Negativ-Interpretation von Unterabfragen mit den Operatoren ANY und ALL

Das fünfte Kriterium ist etwas schwerer zu verstehen. Abfragen der folgenden Formate werden in Joins umgewandelt:

```
SELECT select-list
FROM table
WHERE NOT expression comparison-operator ALL ( subquery-expression )
```

```
SELECT select-list
FROM table
WHERE expression comparison-operator ANY ( subquery-expression )
```

Die folgenden Abfragen werden jedoch nicht in Joins umgewandelt:

```
SELECT select-list
FROM table
WHERE expression comparison-operator ALL ( subquery-expression )
```

```
SELECT select-list
FROM table
WHERE NOT expression comparison-operator ANY ( subquery-expression )
```

Die ersten beiden Abfragen sind gleichwertig, ebenso wie die letzten beiden. Es wurde schon erwähnt, dass der Operator ANY dem Operator OR gleichwertig ist, aber eine variable Anzahl von Argumenten hat. Der Operator ALL ist in diesem Sinne dem Operator AND gleichwertig. Die folgenden beiden Ausdrücke sind beispielsweise gleichwertig:

```
NOT ( ( X > A ) AND ( X > B ) )
( X <= A ) OR ( X <= B )
```

Ebenfalls sind die folgenden beiden Ausdrücke gleichwertig:

```
WHERE NOT OrderDate > ALL (
    SELECT FIRST ( ShipDate )
    FROM SalesOrderItems
    ORDER BY ShipDate )

WHERE OrderDate <= ANY (
    SELECT FIRST ( ShipDate )
    FROM SalesOrderItems
    ORDER BY ShipDate )
```

Negativinterpretationen der Ausdrücke ANY und ALL

Allgemein sind die folgenden Ausdrücke gleichwertig:

NOT *column-name operator ANY (subquery-expression)*

column-name inverse-operator ALL (subquery-expression)

Diese Ausdrücke sind allgemein ebenfalls gleichwertig:

NOT *column-name operator ALL (subquery-expression)*

column-name inverse-operator ANY (subquery-expression)

Dabei gilt: *inverse-operator* wird erlangt, indem *operator* negiert wird, wie die folgende Tabelle zeigt:

Operator	Inverser_Operator
=	<>
<	=>
>	=<
=<	>
=>	<
<>	=

Unterabfrage, die IN folgt

Der Optimierer konvertiert eine Unterabfrage, die einem Schlüsselwort IN folgt, nur, falls folgende Bedingungen erfüllt werden:

- Die Hauptabfrage enthält keine GROUP BY-Klausel und ist keine Aggregatabfrage, oder die Unterabfrage gibt genau einen Wert zurück.
- Die Unterabfrage enthält keine GROUP BY-Klausel.
- Die Unterabfrage enthält kein DISTINCT-Schlüsselwort.
- Die Unterabfrage ist keine UNION-Abfrage.
- Die Unterabfrage ist keine Aggregatabfrage.
- Die Verbindung '*expression IN (subquery-expression)*' darf nicht negiert werden.

Beispiel

Die Anforderung "Finde die Namen der Mitarbeiter, die auch Abteilungsleiter sind" wird mit der folgenden Abfrage ausgedrückt und in eine Join-Abfrage konvertiert, da sie die Bedingungen erfüllt.

```
SELECT GivenName, Surname
FROM Employees
```

```
WHERE EmployeeID IN (
  SELECT DepartmentHeadID
  FROM Departments
  WHERE ( DepartmentName = 'Finance' OR
    DepartmentName = 'Shipping' ) );
```

Anders die Anforderung "Finde die Namen der Mitarbeiter, die Abteilungsleiter oder Kunden sind". Sie wird nicht in einen Join umgewandelt, wenn sie durch die UNION-Abfrage ausgedrückt wird.

Eine UNION-Abfrage nach dem Operator IN kann nicht umgewandelt werden

```
SELECT GivenName, Surname
FROM Employees
WHERE EmployeeID IN (
  SELECT DepartmentHeadID
  FROM Departments
  WHERE ( DepartmentName='Finance' OR
    DepartmentName = 'Shipping' )
UNION
SELECT CustomerID
FROM SalesOrders);
```

Die Anforderung "Finde die Namen der Mitarbeiter, die keine Abteilungsleiter sind" wird als negativ interpretierte Unterabfrage (siehe unten) formuliert und nicht konvertiert.

```
SELECT GivenName, Surname
FROM Employees
WHERE NOT EmployeeID IN (
  SELECT DepartmentHeadID
  FROM Departments
  WHERE ( DepartmentName='Finance' OR
    DepartmentName = 'Shipping' ) );
```

Die Bedingungen, die zur Konvertierung einer IN- oder ANY-Unterabfrage in einen Join erforderlich sind, sind identisch. Der Grund hierfür ist, dass beide Ausdrücke logisch gleichwertig sind.

Konvertierung einer Abfrage mit dem Operator IN in eine Abfrage mit dem Operator ANY

In einigen Fällen konvertiert SQL Anywhere eine Abfrage mit dem IN-Operator in eine Abfrage mit dem ANY-Operator und entscheidet dann, ob die Unterabfrage in einen Join konvertiert wird. Die folgenden beiden Ausdrücke sind beispielsweise gleichwertig:

WHERE *column-name* **IN**(*subquery-expression*)

WHERE *column-name* = **ANY**(*subquery-expression*)

Die folgenden beiden Abfragen sind ebenfalls gleichwertig:

```
SELECT GivenName, Surname
FROM Employees
WHERE EmployeeID IN (
  SELECT DepartmentHeadID
  FROM Departments
  WHERE ( DepartmentName='Finance' OR
    DepartmentName = 'Shipping' ) );

SELECT GivenName, Surname
FROM Employees
WHERE EmployeeID = ANY (
  SELECT DepartmentHeadID
```



```
FROM Departments
WHERE ( DepartmentName='Finance' OR
       DepartmentName = 'Shipping' ) );
```

Unterabfrage, die EXISTS folgt

Der Optimierer konvertiert eine Unterabfrage, die einem Schlüsselwort folgt, nur, falls folgende Bedingungen erfüllt werden:

- Die Hauptabfrage enthält keine GROUP BY-Klausel und ist keine Aggregatabfrage, oder die Unterabfrage gibt genau einen Wert zurück.
- Die Verbindung 'EXISTS (Unterabfrage)' wird nicht negiert.
- Die Unterabfrage ist korreliert, d.h. sie enthält eine äußere Referenz.

Beispiel

Die Anforderung "Welche Kunden haben nach dem 13.07.01 Bestellungen aufgegeben?", die in einer Abfrage formuliert werden kann, deren nicht-negierte Unterabfrage die äußere Referenz **Customers.ID** = **SalesOrders.CustomerID** enthält, könnte durch den folgenden Join dargestellt werden:

```
SELECT GivenName, Surname
FROM Customers
WHERE EXISTS (
  SELECT *
  FROM SalesOrders
  WHERE ( OrderDate > '2001-07-13' ) AND
        ( Customers.ID = SalesOrders.CustomerID ) );
```

Durch das Schlüsselwort EXISTS wird der Datenbankserver angewiesen, nach leeren Ergebnismengen zu suchen. Wenn INNER-Joins verwendet werden, zeigt der Datenbankserver automatisch nur die Zeilen an, in denen sich Daten von allen Tabellen befinden, die in der FROM-Klausel enthalten sind. Diese Abfrage gibt daher dieselben Zeilen zurück wie diejenige ohne die Unterabfrage:

```
SELECT DISTINCT GivenName, Surname
FROM Customers, SalesOrders
WHERE ( SalesOrders.OrderDate > '2001-07-13' ) AND
      ( Customers.ID = SalesOrders.CustomerID );
```

Datenmanipulationsanweisungen

Die Anweisungen zum Hinzufügen, Ändern oder Löschen von Daten werden Datenmanipulationsanweisungen genannt. Sie sind eine DML-Teilmenge (Data Modification Language, Datenänderungssprache) von ANSI SQL.

Es gibt folgende DML-Hauptanweisungen:

- **INSERT-Anweisung** Damit werden neue Zeilen in eine Tabelle oder Ansicht eingefügt.
- **UPDATE-Anweisung** Ändert Zeilen in einer Gruppe von Tabellen oder Ansichten.

- **DELETE-Anweisung** Entfernt Zeilen aus einer Gruppe von Tabellen oder Ansichten.
- **MERGE-Anweisung** Fügt einer Tabelle oder Ansicht bestimmte Zeilen hinzu, ändert und entfernt sie.

Zusätzlich zu den obenstehenden Anweisungen sind die Anweisungen LOAD TABLE und TRUNCATE TABLE für den Massenimport und das Löschen großer Datenmengen nützlich.

Siehe auch

- „INSERT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „UPDATE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DELETE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „MERGE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Privilegien für die Datenmanipulation

Sie können Datenmanipulationsanweisungen nur ausführen, wenn Sie die entsprechenden Privilegien für die Datenbanktabellen haben, die Sie ändern möchten. Der Datenbankadministrator und die Eigentümer von Datenbankobjekten verwenden die Anweisungen GRANT und REVOKE, um festzulegen, wer Zugriff auf welche Datenmanipulationsfunktionen hat.

Privilegien können einzelnen Benutzer, Rollen und benutzererweiterten Rollen erteilt werden.

Siehe auch

- „Benutzersicherheit (Rollen und Privilegien)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Transaktionen und Datenmanipulation

Wenn Sie Daten ändern, wird eine Kopie des alten und des neuen Status der Zeile, die von den Datenmanipulationsanweisungen betroffen ist, in das Rollback-Log geschrieben. Wenn Sie eine Transaktion beginnen, einen Fehler erkennen und die Transaktion zurücksetzen, wird die Datenbank wieder in ihren früheren Zustand zurückversetzt.

Siehe auch

- „Transaktionen und Isolationsstufen“ auf Seite 881

Permanente Datenänderungen

Die Anweisung COMMIT schreibt alle Änderungen fest.

Die Anweisung COMMIT sollte immer nach einer sinnvoll zusammengestellten Gruppe von Anweisungen ausgeführt werden. Wenn Sie z. B. Geld vom Konto eines Kunden zu einem anderen transferieren, müssen Sie bei einem Konto den Betrag hinzufügen und beim anderen Konto diesen Betrag abziehen. Dann sollten Sie die Änderungen festschreiben, da es keinen Sinn macht, die Datenbank in einem Zustand zu belassen, in dem mehr oder weniger Geld gespeichert ist als vorher.

Sie können Interactive SQL anweisen, alle Änderungen automatisch festzuschreiben, indem Sie die Option "auto_commit" auf "On" setzen. Hierbei handelt es sich um eine Interactive SQL-Option. Wenn "auto_commit" auf "On" gesetzt wurde, gibt Interactive SQL nach jeder Ihrer INSERT-, UPDATE- und DELETE-Anweisungen eine COMMIT-Anweisung aus. Dadurch kann die Performance erheblich beeinträchtigt werden. Aus diesem Grund sollten Sie die Option "auto_commit" lieber in der Einstellung "Off" belassen.

Verwenden Sie COMMIT mit Umsicht

Wenn Sie die Beispiele in dieser praktischen Einführung testen, schreiben Sie Änderungen nur dann mit COMMIT fest, wenn Sie die Datenbank dauerhaft ändern möchten.

Siehe auch

- „Interactive SQL-Optionen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „COMMIT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Rückgängigmachen von Änderungen

Jede von Ihnen veranlasste und noch nicht festgeschriebene Änderung kann rückgängig gemacht werden. Mit SQL können Sie alle Änderungen seit dem letzten Festschreiben rückgängig machen, indem Sie die Anweisung ROLLBACK verwenden. Diese Anweisung macht alle Änderungen rückgängig, die seit dem letzten Festschreiben der Änderungen vorgenommen wurden.

Siehe auch

- „ROLLBACK-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Transaktionen und Datenwiederherstellung

SQL Anywhere schützt die Integrität Ihrer Datenbank im Fall eines System- oder Stromausfalls. Es stehen mehrere Optionen zur Verfügung, um den Datenbankserver wieder herzustellen. Die Logdatei, die SQL Anywhere auf einem separaten Device speichert, kann zur Wiederherstellung Ihrer Daten verwendet werden. Wenn eine Logdatei zur Wiederherstellung verwendet wird, muss SQL Anywhere Ihre Datenbank nicht so häufig aktualisieren und die Performance Ihres Datenbankservers wird gesteigert.

Der Datenbankserver kann mithilfe der Transaktionsverarbeitung Situationen identifizieren, in denen sich Ihre Daten in einem konsistenten Zustand befinden. Die Transaktionsverarbeitung gewährleistet, dass die gesamte Transaktion widerrufen bzw. zurückgesetzt wird, wenn sie aus irgendeinem Grund nicht erfolgreich abgeschlossen werden kann. Die Datenbank wird durch den Ausfall von Transaktionen in keinem Fall beeinflusst.

Die Transaktionsverarbeitung in SQL Anywhere gewährleistet, dass die Inhalte einer Transaktion auch bei einem Systemausfall während einer Transaktion sicher abgearbeitet werden.

Siehe auch

- „Sicherung und Datenwiederherstellung“ [[SQL Anywhere Server - Datenbankadministration](#)]

Referenzielle Integrität

SQL Anywhere überprüft automatisch die Gültigkeit der Daten auf häufig anzutreffende Fehler, wenn Sie Daten einfügen, aktualisieren oder löschen. Diese Art der Gültigkeitsüberprüfung wird **Erzwingen der referenziellen Integrität** genannt, weil sie die Integrität der Daten innerhalb der und zwischen den Tabellen in der Datenbank überprüft.

Siehe auch

- [„Entitätsintegrität und referenzielle Integrität“ auf Seite 868](#)

Daten mit INSERT hinzufügen

Mit der Anweisung INSERT werden Zeilen in eine Datenbank eingefügt. Die INSERT-Anweisung hat zwei Formen: Sie können das Schlüsselwort VALUES verwenden oder eine SELECT-Anweisung ausführen:

INSERT mit Werten

Das Schlüsselwort VALUES gibt Werte für einige oder alle Spalten in einer neuen Zeile an. Eine vereinfachte Version der Syntax für die Anweisung INSERT mit dem Schlüsselwort VALUES lautet wie folgt:

```
INSERT [ INTO ] table-name [ ( column-name, ... ) ]  
VALUES ( expression, ... )
```

Sie können die Liste der Spaltennamen weglassen, wenn Sie einen Wert für jede Spalte in der Tabelle in der Reihenfolge eingeben, in der sie bei der Durchführung einer Abfrage mit SELECT * erscheinen.

INSERT aus SELECT

Sie können SELECT in einer INSERT-Anweisung verwenden, um Werte aus einer oder mehreren Tabellen abzurufen. Wenn die Tabelle, in die Sie die Daten einfügen, eine große Anzahl von Spalten aufweist, können Sie auch WITH AUTO NAME zur Vereinfachung der Syntax verwenden. Mit WITH AUTO NAME müssen Sie die Spaltennamen nur in der SELECT-Anweisung angeben, und nicht jeweils in den INSERT- und SELECT-Anweisungen. Die Namen in der SELECT-Anweisung müssen Spaltenreferenzen oder Ausdrücke mit einem Alias sein.

Eine vereinfachte Version der Syntax für die INSERT-Anweisung mit einer SELECT-Anweisung lautet wie folgt:

```
INSERT [ INTO ] table-name  
[ WITH AUTO NAME ] select-statement
```

Siehe auch

- [„INSERT-Anweisung“ \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)

Einfügen von Werten in alle Spalten einer Zeile

Mit der INSERT-Anweisung können Sie Werte in alle Spalten einer Zeile einfügen.

Voraussetzungen

Sie müssen das INSERT-Privileg für die Tabelle haben. Wenn die ON EXISTING UPDATE-Klausel angegeben wird, ist das UPDATE-Privileg für die Tabelle ebenfalls erforderlich.

Geben Sie die Werte in derselben Reihenfolge ein wie die Spaltennamen in der ursprünglichen CREATE TABLE-Anweisung.

Schließen Sie diese Werte in Klammern ein.

Schließen Sie alle Zeichenfolgendaten in Apostrophe ein.

Verwenden Sie eine eigene INSERT-Anweisung für jede hinzuzufügende Zeile.

Aufgabe

- Führen Sie eine INSERT-Anweisung aus, die Werte für jede Spalte enthält.

Ergebnisse

Die angegebenen Werte werden in die einzelnen Spalten einer neuen Zeile eingefügt.

Beispiel

Die folgende INSERT-Anweisung fügt eine neue Zeile in die Departments-Tabelle ein und weist jeder Spalte in der Zeile einen Wert zu:

```
INSERT INTO GROUP0.Departments  
VALUES ( 702, 'Eastern Sales', 902 );
```

Siehe auch

- „INSERT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Werte in bestimmte Spalten einfügen

Eingefügte Werte für festgelegte und nicht festgelegte Spalten

Werte werden so in eine Zeile eingefügt, wie es in der INSERT-Anweisung festgelegt wird. Wenn für eine Spalte kein Wert festgelegt wird, hängt der eingegebene Wert von den Spalteneinstellungen ab, also z.B. ob NULL erlaubt ist, ein DEFAULT-Wert benutzt werden soll usw. Manchmal schlägt der Einfügevorgang fehl und es wird ein Fehler zurückgegeben. In der folgenden Tabelle werden die möglichen Ergebnisse abhängig vom eingefügten Wert (falls zutreffend) und den Spalteneinstellungen dargestellt:

Eingefügter Wert	Nullwertfähig	Nicht nullwertfähig	Nullwertfähig mit DEFAULT	Nicht nullwertfähig mit DEFAULT	Nicht nullwertfähig, mit DEFAULT AUTOINCREMENT oder DEFAULT [UTC] TIMESTAMP
<keiner>	NULL	SQL-Fehler	DEFAULT-Wert	DEFAULT-Wert	DEFAULT-Wert
NULL	NULL	SQL-Fehler	NULL	SQL-Fehler	DEFAULT-Wert
Angegebener Wert	Angegebener Wert	Angegebener Wert	Angegebener Wert	Angegebener Wert	Angegebener Wert

Standardmäßig lassen Spalten NULL zu, wenn Sie nicht bei der Erstellung der Tabelle ausdrücklich NOT NULL in der Spaltendefinition festlegen. Sie können den Standardwert mit der Option "allow_nulls_by_default" ändern. Sie können auch festlegen, ob eine bestimmte Spalte NULL enthalten darf, indem Sie die Anweisung ALTER TABLE benutzen.

Spaltendaten mit Integritätsregeln einschränken

Sie können Integritätsregeln für eine Spalte oder Domäne erstellen. Integritätsregeln bestimmen die Art der Daten, die hinzugefügt werden können oder nicht.

Ausdrückliche Einfügung von NULL

Sie können NULL explizit in eine Spalte einfügen, indem Sie NULL eingeben. Setzen Sie dies nicht in Anführungszeichen, da die Eingabe sonst als Zeichenfolge interpretiert wird. Die folgende Anweisung fügt beispielsweise explizit NULL in die Spalte "DepartmentHeadID" ein:

```
INSERT INTO Departments
VALUES ( 703, 'Western Sales', NULL );
```

Standardwerte für die Eingabe von Werten verwenden

Sie können eine Spalte so definieren, dass bei jedem Einfügen einer Zeile ein Standardwert in die Spalte eingefügt wird, auch wenn in der Anweisung diese Spalte keinen Wert erhält. Hierzu geben Sie einen Standardwert für die Spalte an.

Siehe auch

- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „allow_nulls_by_default-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Tabellen- und Spalten-Integritätsregeln“ auf Seite 859
- „Spalten-Standardwerte“ auf Seite 852

Einfügen von Werten in bestimmte Spalten

Sie können Daten in bestimmte Spalten einer Zeile einfügen, indem Sie nur diese Spalten und ihre Werte angeben.

Voraussetzungen

Sie müssen das INSERT-Privileg für die Tabelle haben. Wenn die ON EXISTING UPDATE-Klausel angegeben wird, ist das UPDATE-Privileg für die Tabelle ebenfalls erforderlich.

Kontext und Bemerkungen

Die von Ihnen angegebene Spaltenreihenfolge muss nicht unbedingt der Reihenfolge der Spalten in der Tabelle entsprechen, sondern der Reihenfolge, in der Sie die Werte angeben, die Sie einfügen.

Definieren Sie alle anderen Spalten, die nicht in der Spaltenliste enthalten sind, sodass sie NULL enthalten können oder Standardwerte enthalten. Wenn Sie eine Spalte überspringen, die einen Standardwert hat, wird dieser benutzt.

Aufgabe

- Führen Sie eine INSERT INTO-Anweisung aus, um Daten in bestimmte Spalten einzufügen.

Die folgende Anweisung fügt Daten nur in zwei Spalten ein: DepartmentID und DepartmentName:

```
INSERT INTO GROUPO.Departments ( DepartmentID, DepartmentName )  
VALUES ( 703, 'Western Sales' );
```

DepartmentHeadID hat keinen Standardwert, akzeptiert aber NULL. Daher wird der Spalte automatisch NULL zugewiesen.

Ergebnisse

Die Daten werden in die angegebenen Spalten eingefügt.

Siehe auch

- „INSERT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Hinzufügen neuer Zeilen mit SELECT

Um Werte aus anderen Tabellen in eine Tabelle einzufügen, können Sie eine SELECT-Klausel in der INSERT-Anweisung benutzen. Die SELECT-Klausel kann Werte in einige oder alle Spalten einer Zeile einfügen.

Es kann hilfreich sein, Werte nur für einige Spalten einzufügen, wenn Sie einige Werte aus einer vorhandenen Tabelle übernehmen möchten. Danach können Sie die UPDATE-Anweisung verwenden, um die Werte für die anderen Spalten hinzuzufügen.

Bevor Sie Werte für nur einige Spalten einer Tabelle einfügen, müssen Sie sicherstellen, dass für die Spalten, in die kein Wert eingefügt wird, entweder ein Standardwert vorhanden ist oder NULL angegeben wurde. Andernfalls wird ein Fehler gemeldet.

Wenn Sie Zeilen aus einer Tabelle in eine andere einfügen, müssen die beiden Tabellen kompatible Strukturen haben—das bedeutet, dass die passenden Spalten dieselben Datentypen aufweisen oder mit Datentypen definiert sein müssen, zwischen denen SQL Anywhere automatisch konvertieren kann.

Daten in einen Teil der Spalten einfügen

Sie können die SELECT-Anweisung verwenden, um genauso wie mit der VALUES-Klausel Daten nur in eine Teilmenge der Spalten einer Zeile einzufügen. Geben Sie einfach in der INSERT-Anweisung die Spalten an, in die Sie Daten einfügen wollen.

Daten aus derselben Tabelle einfügen

Sie können Daten in eine Tabelle unter Benutzung von anderen Daten in derselben Tabelle einfügen. Der Vorgang umfasst im Wesentlichen das Kopieren eines Teils oder der Gesamtheit einer Zeile.

Sie können beispielsweise basierend auf vorhandenen Produkten neue Produkte in die Tabelle "Products" einfügen. Die folgende Anweisung fügt den neuen Artikel "Extra Large Tee Shirts" (aus der Modellreihe "Tank Top", "V-neck" und "Crew Neck") in die Tabelle "Products" ein. Die Identifizierungsnummer ist um 30 größer als die bestehende T-Shirt-Größe:

```
INSERT INTO Products
SELECT ID + 30, Name, Description,
      'Extra large', Color, 50, UnitPrice, NULL
FROM Products
WHERE Name = 'Tee Shirt';
```

Beispiel

Wenn die Spalten in beiden Tabellen in derselben Reihenfolge sind, brauchen Sie in keiner Tabellen Spaltennamen anzugeben. Nehmen Sie beispielsweise an, Sie haben eine Tabelle namens NewProducts, die dasselbe Schema wie die Tabelle "Products" hat und einige Zeilen mit Produktinformationen enthält, die Sie der Tabelle "Products" hinzufügen möchten. Sie könnten die folgende Anweisung ausführen:

```
INSERT Products
SELECT *
FROM NewProducts;
```

Einfügen von Dokumenten und Grafiken

Wenn Sie Dokumente oder Grafiken in Ihrer Datenbank speichern möchten, können Sie eine Anwendung schreiben, die den Inhalt der Datei in eine Variable einliest und diese Variable als Wert für eine INSERT-Anweisung bereitstellt.

Sie können auch die xp_read_file-Systemprozedur benutzen, um den Dateiinhalt in eine Tabelle einzufügen. Diese Prozedur ist nützlich, wenn Sie Dateiinhalte aus Interactive SQL oder einer anderen Umgebung einfügen, die keine vollständige Programmiersprache bereitstellt.

Beispiel

In diesem Beispiel wird eine Tabelle erstellt und in einer Spalte der Tabelle wird eine Grafik eingefügt. Sie können diese Schritte aus Interactive SQL durchführen.

1. Erstellen Sie eine Tabelle, in die Grafiken eingefügt werden sollen.

```
CREATE TABLE Pictures
( C1 INT DEFAULT AUTOINCREMENT PRIMARY KEY,
  Filename VARCHAR(254),
  Picture LONG BINARY );
```


2. Fügen Sie den Inhalt von *portrait.gif* im aktuellen Arbeitsverzeichnis des Datenbankservers in die Tabelle ein.

```
INSERT INTO Pictures ( Filename, Picture )
VALUES ( 'portrait.gif',
        xp_read_file( 'portrait.gif' ) );
```

Siehe auch

- „xp_read_file-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- Verwenden von OPENXML mit xp_read_file auf Seite 683
- „Vorbereitete Anweisungen – Überblick“ [*SQL Anywhere Server - Programmierung*]
- „Hinweise zu BLOBs“ [*SQL Anywhere Server - Datenbankadministration*]
- „SET-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „INSERT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Fortgeschrittene Aufgaben: Plattenspeicherzuordnung für eingefügte Zeilen

SQL Anywhere speichert wenn möglich Zeilen zusammenhängend

Jede neue Zeile, die kleiner als die Seitengröße der Datenbankdatei ist, wird stets auf einer Seite gespeichert. Wenn keine vorhandene Seite genügend freien Speicherplatz für die neue Zeile hat, schreibt SQL Anywhere die Zeile auf eine neue Seite. Wenn die neue Zeile zum Beispiel 600 Byte an Speicherplatz benötigt, aber nur 500 Byte auf einer teilweise gefüllten Seite verfügbar sind, platziert SQL Anywhere die Zeile auf einer neuen Seite.

Damit Tabellenseiten auf der Festplatte zusammenhängender gespeichert werden, weist SQL Anywhere Tabellenseiten jeweils in Blöcken mit je acht Seiten zu. Wenn beispielsweise eine Seite zugewiesen werden muss, werden acht Seiten zugewiesen, die Seite in den Block eingefügt und dann der Block mit den nächsten sieben Seiten aufgefüllt. Außerdem wird eine Bitmap für freie Seiten verwendet, um zusammenhängende Seitenblöcke im DBSpace zu suchen. Um relevante Seiten aufzufinden, werden mithilfe der Bitmap sequenzielle Suchen durchgeführt, wobei jeweils Gruppen von 64 kB gelesen werden. Dadurch können effizientere sequenzielle Suchvorgänge erreicht werden.

Speichern der Zeilen in SQL Anywhere in beliebiger Reihenfolge möglich

SQL Anywhere ermittelt Speicherplatz auf den Seiten und fügt Zeilen in der Reihenfolge ein, in der es sie erhält. SQL Anywhere ordnet jede Zeile einer Seite zu, aber die Position, die in der Tabelle ausgewählt wird, muss nicht der Reihenfolge entsprechen, in der sie eingefügt wurde. Der Datenbankserver wird möglicherweise eine neue Seite beginnen müssen, um eine lange Zeile zusammenhängend speichern zu können. Sollte die nächste Reihe kürzer sein, passt sie eventuell an eine leere Stelle auf einer vorhergehenden Seite.

Die Zeilen aller Tabellen sind unsortiert. Wenn die Reihenfolge, mit der Sie Ihre Zeilen erhalten oder verarbeiten, wichtig ist, verwenden Sie eine ORDER BY-Klausel in Ihrer SELECT-Anweisung, um eine Sortierfolge im Ergebnis zu bewirken. Anwendungen, die sich auf die Reihenfolge von Zeilen in einer Tabelle verlassen, können ohne Warnung fehlschlagen.

Wenn Sie häufig die Zeilen einer Tabelle in einer bestimmten Reihenfolge benötigen, ist es sinnvoll, für diese in der ORDER BY-Klausel der Abfrage enthaltenen Spalten einen Index zu erstellen.

Speicher für Null-Spalten nicht reserviert

Immer wenn SQL Anywhere eine Zeile einfügt, reserviert es standardmäßig nur den Speicherplatz, der zum Speichern der Zeile mit ihren Werten erforderlich ist, die sie zum Zeitpunkt ihrer Erstellung enthält. SQL Anywhere reserviert keinen Speicherplatz, um NULL zu speichern oder um Felder wie zum Beispiel Textzeichenfolgen aufzunehmen, die sich vergrößern können.

Sie können SQL Anywhere zwingen, Speicherplatz zu reservieren, indem Sie beim Erstellen der Tabelle die PCTFREE-Option verwenden.

Einmal eingefügte Zeilenidentifizierer sind nicht zu verschieben

Sobald einer Zeile eine Position auf einer Seite zugeordnet ist, wird die Zeile nicht mehr von dieser Seite verschoben. Wenn eine Aktualisierung einen der Werte in der Zeile so ändert, dass er nicht mehr auf die ihm zugeordnete Seite passt, dann wird die Zeile geteilt, und die zusätzlichen Daten werden in eine andere Seite eingefügt.

Diese Eigenschaft verdient besondere Beachtung, vor allem weil SQL Anywhere keinen zusätzlichen Speicherplatz zu dem Zeitpunkt zuordnet, an dem die Zeile eingefügt wird. Nehmen wir zum Beispiel an, dass Sie eine große Anzahl von leeren Zeilen in eine Tabelle einfügen und dann die Werte in eine Spalte nach der anderen eingeben, wobei Sie UPDATE-Anweisungen verwenden. Das Ergebnis wäre, dass beinahe jeder Wert der einzelnen Zeilen auf einer separaten Seite gespeichert wird. Um alle Werte einer Zeile abzurufen, müsste der Datenbankserver möglicherweise mehrere Plattenspeicher-Seiten lesen. Dieser einfache Vorgang würde sich dadurch unnötigerweise extrem verlangsamen.

Es ist sinnvoll, neue Zeilen zum Zeitpunkt des Einfügens mit Werten anzufüllen. Sobald sie eingefügt sind, werden sie genügend Platz haben, um die erwarteten Daten aufzunehmen.

Eine Datenbankdatei wird niemals kleiner

Wenn Sie Zeilen in die Datenbank einfügen oder aus ihr löschen, wird der von ihnen belegte Speicherplatz von SQL Anywhere automatisch wieder verwendet. Das heißt, dass SQL Anywhere möglicherweise eine Zeile an einen Platz einfügt, der früher von einer anderen Zeile besetzt war.

SQL Anywhere führt Aufzeichnungen über den leeren Speicherplatz, den einzelne Seiten aufweisen. Wenn Sie SQL Anywhere auffordern, eine neue Zeile einzufügen, durchsucht es zuerst sein Protokoll über den Speicherplatz auf vorhandenen Seiten. Wenn SQL Anywhere genügend Platz auf einer vorhandenen Seite findet, platziert es die neue Zeile auf diese Seite, wobei nötigenfalls die Seite neu organisiert wird. Wenn nicht, beginnt SQL Anywhere eine neue Seite.

Wenn im Laufe der Zeit Zeilen gelöscht und keine neuen Zeilen eingefügt werden, die klein genug wären, um die leeren Plätze zu verwenden, werden die Informationen in der Datenbank sehr verstreut sein. Sie können die Tabelle neu laden oder mit der REORGANIZE TABLE-Anweisung defragmentieren.

Siehe auch

- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „REORGANIZE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Datenänderungen mit UPDATE

Die UPDATE-Anweisung gibt die zu ändernden Zeilen an sowie die Ausdrücke, die als neue Werte für bestimmte Spalten in diesen Zeilen verwendet werden sollen.

Sie können die UPDATE-Anweisung verwenden, um einzelne Zeilen, Gruppen von Zeilen oder alle Zeilen in einer Tabelle zu ändern. Im Gegensatz zu anderen Datenmanipulationsanweisungen (INSERT, MERGE und DELETE) können Sie mit der UPDATE-Anweisung auch Zeilen in mehreren Tabellen gleichzeitig ändern. In allen Fällen ist die Ausführung der UPDATE-Anweisung atomar, d.h., entweder werden alle Zeilen ohne Fehler geändert oder keine von ihnen. Wenn beispielsweise einer der zu ändernden Werte den falschen Datentyp hat oder wenn der neue Wert bewirkt, dass eine CHECK-Integritätsregel verletzt wird, schlägt die UPDATE-Anweisung fehl und der gesamte Vorgang wird zurückgesetzt.

Syntax von UPDATE

Eine vereinfachte Version der Syntax für die UPDATE-Anweisung lautet wie folgt:

```
UPDATE table-name
SET column_name = expression
WHERE search-condition
```

Wenn das Unternehmen Newton Ent. (in der Tabelle "Customers" der SQL Anywhere-Beispieldatenbank) von Einstein, Inc. übernommen wird, können Sie den Namen des Unternehmens mithilfe folgender Anweisung aktualisieren:

```
UPDATE Customers
SET CompanyName = 'Einstein, Inc.'
WHERE CompanyName = 'Newton Ent.';
```

Sie können in einer WHERE-Klausel jeden beliebigen Ausdruck verwenden. Wenn Sie nicht sicher sind, wie der Firmenname geschrieben wird, können Sie alle Firmen mit dem Namen "Newton" aktualisieren. Hierzu können Sie z.B. folgende Anweisung verwenden:

```
UPDATE Customers
SET CompanyName = 'Einstein, Inc.'
WHERE CompanyName LIKE 'Newton%';
```

Die Suchbedingung muss sich nicht auf die zu aktualisierende Spalte beziehen. Die Unternehmenskennung für Newton Entertainments ist 109. Da diese Kennung der Primärschlüssel für die Tabelle ist, können Sie sicher sein, dass die richtige Zeile aktualisiert wird, indem Sie folgende Anweisung verwenden:

```
UPDATE Customers
SET CompanyName = 'Einstein, Inc.'
WHERE ID = 109;
```

Tipp

Sie können auch Zeilen der Ergebnismenge in Interactive SQL ändern.

SET-Klausel

Die SET-Klausel gibt an, welche Spalten aktualisiert werden müssen und wie deren neue Werte lauten. Die WHERE-Klausel legt die Zeilen fest, die aktualisiert werden müssen. Wenn Sie keine WHERE-

Klausel eingeben, werden die angegebenen Spalten aller Zeilen mit den Werten aktualisiert, die in der SET-Klausel eingegeben wurden.

Die in der SET-Klausel angegebenen Ausdrücke können Literalkonstanten, Host- oder SQL-Variablen, Unterabfragen, Spezialwerte wie CURRENT_TIMESTAMP, aus einer anderen Tabelle übernommene Ausdruckswerte oder eine beliebige Kombination dieser Elemente sein. Sie können in einer SET-Klausel auch DEFAULT angeben, was für den Standardwert der betreffenden Spalte in der Basistabelle steht. Wenn sich der Datentyp des Ausdrucks vom Datentyp der zu ändernden Spalte unterscheidet, wird der Ausdruck vom Datenbankserver, sofern möglich, automatisch in den Datentyp der Spalte konvertiert. Wenn die Konvertierung nicht möglich ist, resultiert eine Datenausnahmebedingung und die UPDATE-Anweisung schlägt fehl.

Sie können die SET-Klausel verwenden, um zusätzlich zum Ändern von Spaltenwerten den Wert für eine Variable festzulegen. In diesem Beispiel wird zusätzlich zum Aktualisieren der Tabelle "T" der Variablen @var ein Wert zugeordnet:

```
UPDATE T
SET @var = expression1, coll = expression2
WHERE...;
```

Dies entspricht ungefähr dem seriellen Ausführen einer SELECT-Anweisung, gefolgt von einer UPDATE-Anweisung:

```
SELECT @var = expression1
FROM T
WHERE... ;
UPDATE T SET coll = expression2
WHERE...;
```

Der Vorteil der Variablenzuordnung innerhalb einer UPDATE-Anweisung besteht darin, dass der Wert der Variablen noch während der Ausführung der Anweisung festgelegt werden kann, solange diese Schreibsperrern besitzt. Dadurch wird verhindert, dass durch parallele Aktualisierungsaktivitäten anderer Verbindungen unerwartete Werte zugeordnet werden.

WHERE-Klausel

Die WHERE-Klausel gibt an, welche Zeilen aktualisiert werden sollen, indem *search-condition* auf die Tabelle oder das kartesische Produkt aus Tabellenausdrücken in der UPDATE-Anweisung angewendet wird. Die folgende Anweisung ersetzt beispielsweise das T-Shirt der Größe "One Size Fits All" durch ein T-Shirt der Größe "Extra Large":

```
UPDATE Products
SET Size = 'Extra Large'
WHERE Name = 'Tee Shirt'
AND Size = 'One Size Fits All';
```

Komplexe UPDATE-Anweisungen

Komplexere Formen der UPDATE-Anweisung ermöglichen Aktualisierungen über Joins und andere Arten von Tabellenausdrücken hinweg.

Syntax 1 der UPDATE-Anweisung lautet beispielsweise:

```
UPDATE [ row-limitation ] table-name
SET set-item[, ...]
```

```

FROM table-expression [, ...] ]
[ WHERE search-condition ]
[ ORDER BY expression [ ASC | DESC ] , ...]
[ OPTION( query-hint, ... ) ]

```

Die Semantik dieser Form der UPDATE-Anweisung besteht darin, dass zunächst eine Ergebnismenge berechnet wird, die aus allen Kombinationen von Zeilen aus jedem *table-expression* besteht, anschließend die *search-condition* in der WHERE-Klausel angewendet wird und schließlich die resultierenden Zeilen mit der ORDER BY-Klausel sortiert werden. Aus dieser Berechnung ergibt sich die Menge der Zeilen, die geändert werden. Jeder *table-expression* kann aus Joins von Basistabellen, Ansichten und abgeleiteten Tabellen bestehen. Die Syntax erlaubt es, eine oder mehrere Tabellen mit Werten aus Spalten in anderen Tabellen zu aktualisieren. Der Abfrageoptimierer ändert möglicherweise die Reihenfolge der Vorgänge, um eine effizientere Ausführungsstrategie für die UPDATE-Anweisung zu erstellen.

Wenn eine Basistabelle in einer Menge von mehrmals zu ändernden Zeilen erscheint, wird die Zeile mehrfach aktualisiert, sofern sich die neuen Werte bei jedem Manipulationsversuch unterscheiden. Wenn ein BEFORE ROW UPDATE-Trigger vorhanden ist, wird der BEFORE ROW UPDATE-Trigger für jede einzelne Zeilenmanipulation ausgelöst, gemäß der UPDATE OF *column-list*-Klausel des Triggers. AFTER ROW UPDATE-Trigger werden ebenfalls für jede einzelne Zeilenmanipulation ausgelöst, jedoch nur, wenn die Werte der Zeile tatsächlich geändert werden, gemäß der UPDATE OF *column-list*-Klausel des Triggers.

Trigger werden für jede aktualisierte Tabelle ausgelöst, basierend auf dem Typ des Triggers und dem Wert der ORDER-Klausel für jede Triggerdefinition. Wenn eine UPDATE-Anweisung mehr als eine Tabelle ändert, ist jedoch die Reihenfolge, in der die Tabellen aktualisiert werden, nicht gewährleistet.

Im folgenden Beispiel werden ein BEFORE ROW UPDATE-Trigger und ein AFTER STATEMENT UPDATE-Trigger in der Products-Tabelle erstellt, die jeweils eine Meldung im Meldungsfenster des Datenbankservers ausgeben:

```

CREATE OR REPLACE TRIGGER trigger0
BEFORE UPDATE
ON Products
REFERENCING OLD AS old_product NEW AS new_product
FOR EACH ROW
BEGIN
    PRINT ('BEFORE row: PK value: ' || old_product.ID || ' New Price: ' ||
new_product.UnitPrice );
END;

CREATE OR REPLACE TRIGGER trigger1
AFTER UPDATE
ON Products
REFERENCING NEW AS new_product
FOR EACH STATEMENT
BEGIN
    DECLARE @pk INTEGER;
    DECLARE @newUnitPrice DECIMAL(12,2);
    DECLARE @err_notfound EXCEPTION FOR SQLSTATE VALUE '02000';
    DECLARE new_curs CURSOR FOR
        SELECT ID, UnitPrice FROM new_product;
    OPEN new_curs;
    LoopGetRow:
    LOOP
        FETCH NEXT new_curs INTO @pk, @newUnitPrice;
        IF SQLSTATE = @err_notfound THEN

```

```
        LEAVE LoopGetRow
    END IF;
    PRINT ('AFTER stmt: PK value: ' || @pk || ' Unit price: ' ||
@newUnitPrice );
    END LOOP LoopGetRow;
    CLOSE new_curs
END;
```

Angenommen, Sie führen anschließend eine UPDATE-Anweisung über einen Join der Products-Tabelle mit der SalesOrderItems-Tabelle aus, um einen Rabatt von 5 % auf Produkte anzuwenden, die seit dem 1. April 001 versandt wurden und für die mindestens ein großer Auftrag vorliegt:

```
UPDATE Products p JOIN SalesOrderItems s ON (p.ID = s.ProductID)
SET p.UnitPrice = p.UnitPrice * 0.95
WHERE s.ShipDate > '2001-04-01' AND s.Quantity >= 72;
```

Das Meldungsfenster des Datenbankservers zeigt die folgenden Meldungen:

```
BEFORE row: PK value: 700 New Price: 14.25
BEFORE row: PK value: 302 New Price: 13.30
BEFORE row: PK value: 700 New Price: 13.54
AFTER stmt: PK value: 700 Unit price: 14.25
AFTER stmt: PK value: 302 Unit price: 13.30
AFTER stmt: PK value: 700 Unit price: 13.54
```

Die Meldungen zeigen, dass Produkt 700 zweimal aktualisiert wurde, weil es in zwei verschiedenen Aufträgen enthalten war, die die Suchbedingung in der UPDATE-Anweisung erfüllen. Die mehrfach vorhandenen Aktualisierungen sind sowohl für den BEFORE ROW-Trieger als auch für den AFTER STATEMENT-Trieger sichtbar. Bei jeder Zeilenmanipulation werden der **alte** und der **neue** Wert für jeden Triggeraufruf entsprechend geändert. Bei einem AFTER STATEMENT-Trieger stimmt die Reihenfolge der Zeilen in den durch die REFERENCING-Klausel gebildeten temporären Tabellen möglicherweise nicht mit der Reihenfolge überein, in der die Zeilen geändert wurden, und die genaue Reihenfolge der Zeilen ist nicht gewährleistet.

Aufgrund der mehrfach vorhandenen Aktualisierungen wurde der Rabatt zweimal auf den UnitPrice-Wert für Produkt 700 angewendet, sodass der Stückpreis von ursprünglich \$15.00 auf \$13.54 gesenkt wurde (was einem Rabatt von 9,75% entspricht) und nicht nur auf \$14.25. Zur Vermeidung dieser unbeabsichtigten Folge können Sie stattdessen die UPDATE-Anweisung so formulieren, dass statt eines Joins eine EXISTS-Unterabfrage verwendet wird, um sicherzustellen, dass jede Produktzeile höchstens einmal geändert wird. Die überarbeitete UPDATE-Anweisung enthält sowohl eine EXISTS-Unterabfrage als auch die alternative Syntax der UPDATE-Anweisung, die eine FROM-Klausel zulässt:

```
UPDATE Products AS p
SET p.UnitPrice = p.UnitPrice * 0.95
FROM Products AS p
WHERE EXISTS(
    SELECT *
    FROM SalesOrderItems s
    WHERE p.ID = s.ProductID
        AND s.ShipDate > '2001-04-01'
        AND s.Quantity >= 72);
```

UPDATE und Verletzungen von Integritätsregeln

Wenn eine UPDATE-Anweisung während der Ausführung eine Integritätsregel zur Erhaltung der referenziellen Integrität verletzt, wird das Verhalten der Anweisung durch die Einstellung der wait-for-

commit-Option gesteuert. Wenn die wait_for_commit-Option auf OFF gesetzt ist und eine Integritätsregel zur Erhaltung der referenziellen Integrität verletzt wird, werden die Auswirkungen der UPDATE-Anweisung sofort automatisch zurückgesetzt und eine Fehlermeldung wird angezeigt. Wenn die wait_for_commit-Option auf ON gesetzt ist, wird jede durch die UPDATE-Anweisung verursachte Verletzung einer Integritätsregel zur Erhaltung der referenziellen Integrität vorübergehend ignoriert und später überprüft, wenn die Verbindung eine COMMIT-Anweisung ausführt.

Wenn die Basistabelle oder zu ändernde Tabellen Primärschlüssel, UNIQUE-Integritätsregeln oder eindeutige Indizes enthalten, kann eine zeilenweise Ausführung der UPDATE-Anweisung dazu führen, dass eine Eindeutigkeits-Integritätsregel verletzt wird. Sie können beispielsweise eine UPDATE-Anweisung ausgeben, die die Werte aller Primärschlüsselspalten für eine Tabelle "T" inkrementiert:

```
UPDATE T SET PKcol = PKcol + 1;
```

Wenn während der Ausführung einer UPDATE-Anweisung eine Verletzung der Eindeutigkeit auftritt, führt der Datenbankserver automatisch die folgenden Schritte aus:

1. Alter und neuer Wert der geänderten Zeile werden in eine temporäre Tabelle mit demselben Schema wie die zu ändernde Basistabelle kopiert.
2. Die ursprüngliche Zeile wird aus der Basistabelle gelöscht. Als Folge dieses Löschvorgangs werden keine DELETE-Trigger ausgelöst.

Während der Ausführung der UPDATE-Anweisung hängt es von der Reihenfolge der Auswertung ab, welche Zeilen erfolgreich aktualisiert und welche vorübergehend gelöscht werden, und dies kann nicht gewährleistet werden. Das Verhalten von SQL-Anforderungen von anderen Verbindungen, die auf schwächeren Isolationsstufen (Isolationsstufe 0, 1 oder 2) ausgeführt werden, kann durch diese vorübergehend gelöschten Zeilen beeinflusst werden. An alle BEFORE ROW- oder AFTER ROW-Trigger der geänderten Tabelle werden die alten und neuen Werte der einzelnen Zeilen gemäß der REFERENCING-Klausel des Triggers übergeben, aber wenn der ROW-Trigger eine separate SQL-Anweisung in der geänderten Tabelle ausgibt, fehlen die in der temporären Tabelle gespeicherten Zeilen.

Nachdem die UPDATE-Anweisung die Änderungen an jeder Zeile abgeschlossen hat, werden die in der temporären Tabelle gespeicherten Zeilen wieder in die Basistabelle eingefügt. Wenn weiterhin eine Verletzung der Eindeutigkeit auftritt, wird die gesamte UPDATE-Anweisung zurückgesetzt. Erst wenn alle in der temporären Tabelle gespeicherten Zeilen erfolgreich wieder in die Basistabelle eingefügt wurden, werden AFTER STATEMENT-Trigger ausgelöst.

Der Datenbankserver verwendet keine Haltetabelle zum temporären Speichern von Zeilen, wenn die zu ändernde Basistabelle das Ziel einer durch eine Integritätsregel zur Erhaltung der referenziellen Integrität ausgelösten Aktion ist, einschließlich ON DELETE CASCADE, ON DELETE SET NULL, ON DELETE DEFAULT, ON UPDATE CASCADE, ON UPDATE SET NULL und ON UPDATE DEFAULT.

Siehe auch

- „UPDATE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Ergebnismengen in Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „ansi_update_constraints-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Integritätsprüfungen bei DELETE oder UPDATE“ auf Seite 875
- „Sperrern während einer Aktualisierung“ auf Seite 920

Datenänderungen mit INSERT

Sie können die Klausel ON EXISTING der INSERT-Anweisung verwenden, um existierende Zeilen einer Tabelle (basierend auf einem Nachschlagen der Primärschlüssel) mit neuen Werten zu aktualisieren. Diese Klausel kann nur bei Tabellen angewendet werden, die einen Primärschlüssel besitzen. Der Versuch, diese Klausel in Tabellen ohne Primärschlüssel oder in Proxy-Tabellen zu verwenden, generiert einen Syntaxfehler.

Die Angabe der ON EXISTING-Klausel veranlasst den Server, für jede Eingabezeile den Primärschlüssel nachzuschlagen. Wenn die entsprechende Zeile nicht existiert, wird die neue Zeile eingefügt. Für Zeilen, die bereits in der Tabelle existieren, können Sie folgende Optionen wählen:

- Einen Fehler für doppelte Schlüsselwerte generieren. Dies ist das Standardverhalten, wenn die ON EXISTING-Klausel nicht angegeben wird.
- Stillschweigend die Eingabezeile ignorieren, ohne irgendeinen Fehler zu generieren.
- Aktualisieren Sie die bestehende Zeile mit den Werten der Eingabezeile.

Siehe auch

- „INSERT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Löschen von Daten mit DELETE

Einfache DELETE-Anweisungen haben folgende Form:

```
DELETE [ FROM ] table-name  
WHERE column-name = expression
```

Sie können auch eine komplexere Form verwenden:

```
DELETE [ FROM ] table-name  
FROM table-list  
WHERE search-condition
```

WHERE-Klausel

Verwenden Sie die WHERE-Klausel, um anzugeben, welche Zeilen entfernt werden sollen. Wenn keine WHERE-Klausel vorhanden ist, entfernt die DELETE-Anweisung alle Zeilen aus der Tabelle.

FROM-Klausel

Die FROM-Klausel an der zweiten Position einer DELETE-Anweisung ist eine besondere Funktion, mit deren Hilfe Sie Daten aus Tabellen entnehmen und entsprechende Daten aus der erstgenannten Tabelle entfernen können. Die Zeilen, die Sie in der FROM-Klausel angeben, setzen die Bedingungen für den Löschvorgang fest.

Siehe auch

- „DELETE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Ergebnismengen in Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

In diesem Beispiel wird die Beispieldatenbank für SQL Anywhere benutzt. Um die Anweisungen in diesem Beispiel auszuführen, müssen Sie die Option "wait_for_commit" auf "On" setzen. Die folgende Anweisung führt diese Aufgabe nur für die laufende Verbindung durch:

```
SET TEMPORARY OPTION wait_for_commit = 'On';
```

Damit können Zeilen gelöscht werden, auch wenn sie Primärschlüssel enthalten, auf die ein Fremdschlüssel Bezug nimmt. Ein COMMIT ist erst zulässig, wenn der dazugehörige Fremdschlüssel ebenfalls gelöscht wurde.

In der folgenden Ansicht werden Produkte und der Wert der verkauften Produkte angezeigt:

```
CREATE VIEW ProductPopularity as
SELECT  Products.ID,
        SUM( Products.UnitPrice * SalesOrderItems.Quantity )
        AS "Value Sold"
FROM    Products JOIN SalesOrderItems
ON      Products.ID = SalesOrderItems.ProductID
GROUP BY Products.ID;
```

Mit dieser Ansicht können Sie die Produkte aus der Tabelle "Products" löschen, die weniger als 20.000 USD Umsatz gebracht haben.

```
DELETE
FROM Products
FROM Products NATURAL JOIN ProductPopularity
WHERE "Value Sold" < 20000;
```

Machen Sie diese Änderungen in der Datenbank rückgängig, indem Sie eine ROLLBACK-Anweisung ausführen:

```
ROLLBACK;
```

Tipp

Sie können auch Zeilen aus Datenbanktabellen der Interactive SQL-Ergebnismenge löschen.

Siehe auch

- „Ergebnismengen in Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]

Löschen aller Zeilen aus einer Tabelle

Sie können die Anweisung TRUNCATE TABLE als schnelle Methode zum Löschen aller Zeilen in einer Tabelle verwenden. Dies funktioniert schneller als eine DELETE-Anweisung ohne Bedingungen, da DELETE alle Änderungen ins Log schreibt, während die einzelnen gelöschten Zeilen mit TRUNCATE nicht aufgezeichnet werden.

Die Tabellendefinition für eine Tabelle, die mit der Anweisung TRUNCATE TABLE geleert wurde, bleibt zusammen mit ihren Indizes und anderen zugeordneten Objekten in der Datenbank, sofern Sie keine DROP TABLE-Anweisung ausführen.

Sie können TRUNCATE TABLE nicht benutzen, wenn eine andere Tabelle Zeilen enthält, die über eine Integritätsregel zur Aufrechterhaltung der referenziellen Integrität darauf zugreifen. Löschen Sie die Zeilen aus der Fremdtabelle oder kürzen Sie zuerst die Fremdtabelle und dann die Primärtabelle.

Das Kürzen von Basistabellen oder der Massenimport von Daten bewirken, dass Daten in Indizes (regulär oder Text) und in abhängigen materialisierten Ansichten veralten. Sie sollten zuerst die Daten in den Indizes und den abhängigen materialisierten Ansichten kürzen, die INPUT-Anweisung ausführen und dann die Indizes und materialisierten Ansichten neu aufbauen bzw. aktualisieren.

Syntax für TRUNCATE TABLE

Die Syntax für TRUNCATE TABLE lautet wie folgt:

TRUNCATE TABLE *table-name*

Um z.B. alle Daten aus der Tabelle "SalesOrders" zu entfernen, geben Sie folgende Anweisung ein:

```
TRUNCATE TABLE SalesOrders;
```

Eine TRUNCATE TABLE-Anweisung löst keine Trigger aus, die für die Tabelle definiert wurden.

Machen Sie diese Änderungen in der Datenbank rückgängig, indem Sie eine ROLLBACK-Anweisung ausführen:

```
ROLLBACK;
```

Siehe auch

- „TRUNCATE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „TRUNCATE TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

SQL-Dialekte und Kompatibilität

Dieser Abschnitt beschreibt die Kompatibilität mit Transact-SQL und die Funktionen von SQL Anywhere, die im Allgemeinen in anderen SQL-Implementierungen nicht zu finden sind.

SQL Anywhere entspricht der SQL-92-basierten States Federal Information Processing Standard Publication (FIPS PUB) 127. Mit geringfügigen Ausnahmen entspricht SQL Anywhere der ISO/ANSI SQL/2008-Kernspezifikation, die in den 9 Teilen von ISO/IEC JTC 1/SC 32 9075-2008 dokumentiert ist. Informationen zur Kompatibilität können Sie in der Referenzdokumentation zu jeder Funktion von SQL Anywhere finden.

Testen der SQL-Konformität mit dem SQL Flagger

In SQL Anywhere können der Datenbankserver und der SQL-Präprozessor (sqlpp) SQL-Anweisungen erkennen, die Erweiterungen des Herstellers sind, die nicht kompatibel mit bestimmten ISO/ANSI SQL-Standards sind oder die nicht von UltraLite unterstützt werden. Diese Funktionalität wird als SQL Flagger bezeichnet. Sie wurde als optionale SQL-Sprachfunktion F812 des ISO/ANSI 9075-1999 SQL-Standards eingeführt, die in diesem Dokument als SQL/1999 bezeichnet wird. Der SQL Flagger hilft Anwendungsentwicklern, SQL-Sprachkonstrukte zu erkennen, die eine angegebene Teilmenge der SQL-Sprache verletzen. Der SQL Flagger kann benutzt werden, um die Kompatibilität mit Kernfunktionen eines SQL-Standards oder mit einer Kombination aus Kernfunktionen und optionalen Funktionen zu gewährleisten. Der SQL Flagger kann auch benutzt werden, um einen Prototyp einer UltraLite-Anwendung mit SQL Anywhere zu erstellen und damit sicherzustellen, dass die verwendete SQL-Version von UltraLite unterstützt wird.

Da die Unterstützung räumlicher Daten als Teil 3 des SQL/MM-Standards (ISO/IEC 13249-3) standardisiert ist, werden die räumlichen Funktionen, Vorgänge und Syntax vom SQL Flagger nicht unterstützt und als Erweiterungen des Herstellers markiert.

Der SQL Flagger ist dazu gedacht, eine statische Prüfung der Konformität zur Kompilierungszeit zu bieten, obwohl sowohl syntaktische als auch semantische Elemente einer SQL-Anweisung Kandidaten für die Analyse durch den SQL Flagger sind. Ein Beispiel für syntaktische Kompatibilität ist das Fehlen des optionalen Schlüsselwortes INTO in einer INSERT-Anweisung (z.B. `INSERT Products VALUES(. . .)`), wobei es sich um eine SQL Anywhere-Grammatikerweiterung der SQL-Sprache handelt. Die Verwendung einer INSERT-Anweisung ohne das Schlüsselwort INTO wird als Erweiterung des Herstellers gekennzeichnet, da der ANSI-Standard SQL/2008 die Verwendung des Schlüsselworts INTO verlangt. Beachten Sie jedoch, dass das Schlüsselwort INTO für UltraLite-Anwendungen optional ist.

Schlüssel-Joins werden ebenfalls als Erweiterung des Herstellers gekennzeichnet. Ein Schlüssel-Join wird standardmäßig verwendet, wenn das JOIN-Schlüsselwort ohne ON-Klausel benutzt wird. Ein Schlüssel-Join verwendet vorhandene Fremdschlüssel-Beziehungen, um die Tabellen zu verknüpfen. Schlüssel-Joins werden von UltraLite nicht unterstützt. Die folgende Abfrage legt beispielsweise eine implizite Join-Bedingung zwischen den Tabellen "Products" und "SalesOrderItems" fest. Diese Abfrage wird vom SQL Flagger als Erweiterung des Herstellers gekennzeichnet.

```
SELECT * FROM Products JOIN SalesOrderItems;
```

Die Funktionalität des SQL Flagger ist nicht von der Ausführung einer SQL-Anweisung abhängig. Die gesamte Flagger-Logik wird lediglich als statischer Prozess zur Kompilierungszeit ausgeführt.

Siehe auch

- „SQLFLAGGER-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Konformität mit Standards für räumliche Daten“ [*SQL Anywhere Server - Unterstützung für räumliche Daten*]
- „Der SQL-Präprozessor“ [*SQL Anywhere Server - Programmierung*]
- „INSERT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Schlüssel-Joins“ auf Seite 523

Aufrufen des SQL Flagger

SQL Anywhere bietet verschiedene Methoden, um den SQL Flagger zum Prüfen einer SQL-Anweisung oder eines Batches mit SQL-Anweisungen aufzurufen:

- **Funktion SQLFLAGGER** Die Funktion SQLFLAGGER analysiert eine einfache SQL-Anweisung oder einen Batch mit SQL-Anweisungen, die als Zeichenfolgenargument übergeben wurden, um die Kompatibilität mit einem bestimmten SQL-Standard zu überprüfen. Die Anweisung oder der Batch werden syntaktisch analysiert, aber nicht ausgeführt.
- **Systemprozedur sa_ansi_standard_packages** Die sa_ansi_standard_packages-Systemprozedur analysiert eine Anweisung oder einen Batch für die Verwendung von optionalen SQL-Sprachfunktionen oder Paketen gemäß der internationalen ANSI-Standards SQL/2008, SQL/2003 oder SQL/1999. Die Anweisung oder der Batch werden syntaktisch analysiert, aber nicht ausgeführt.
- **Optionen sql_flagger_error_level und sql_flagger_warning_level** Die Optionen sql_flagger_error_level und sql_flagger_warning_level rufen den SQL Flagger für alle Anweisungen auf, die für die Verbindung vorbereitet oder ausgeführt wurden. Falls eine Anweisung der Optionseinstellung, d.h., einem bestimmten ANSI-Standard oder UltraLite, nicht entspricht, wird die Anweisung mit einem Fehler beendet (SQLSTATE 0AW03), oder sie gibt abhängig von der festgelegten Option eine Warnung zurück (SQLSTATE 01W07). Falls die Anweisung mit dem Standard übereinstimmt, wird die Anweisung normal ausgeführt.
- **SQL-Präprozessor (sqlpp)** Der SQL-Präprozessor (sqlpp) ist in der Lage, statische SQL-Anweisungen in einer eingebetteten SQL-Anwendung zur Kompilierungszeit zu kennzeichnen. Diese Funktion kann besonders beim Entwickeln einer UltraLite-Anwendung nützlich sein, um SQL-Anweisungen auf UltraLite-Kompatibilität zu prüfen.

Siehe auch

- „Batchanweisungen“ auf Seite 112
- „SQLFLAGGER-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sql_flagger_error_level-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „sql_flagger_warning_level-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „sa_ansi_standard_packages-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Der SQL-Präprozessor“ [*SQL Anywhere Server - Programmierung*]
- „Der SQL-Präprozessor“ [*SQL Anywhere Server - Programmierung*]

Standards und Kompatibilität

Die Flagger-Funktionalität, die im Datenbankserver und im SQL-Präprozessor verwendet wird, entspricht der SQL Flagger-Funktionalität, die in Teil 1 (Framework) des internationalen Standards ANSI/ISO SQL/2008 definiert wird. Der SQL Flagger unterstützt die folgenden ANSI SQL-Standards bei der Ermittlung der Konformität mit SQL-Sprachkonstrukten:

- SQL/1992 Entry Level, Intermediate Level und Full Level
- SQL/1999 Core sowie optionale Pakete von SQL/1999
- SQL/2003 Core sowie optionale Pakete von SQL/2003
- SQL/2008 Core sowie optionale Pakete von SQL/2008

Hinweis

Die SQL Flagger-Unterstützung für SQL/1992 (alle Stufen) wird nicht mehr weiterentwickelt.

Außerdem kann der SQL Flagger Anweisungen ermitteln, die nicht mit UltraLite SQL kompatibel sind. UltraLite hat beispielsweise nur begrenzte Fähigkeiten, CREATE und ALTER auf Schema-Objekte anzuwenden.

Alle SQL-Anweisungen können vom SQL Flagger analysiert werden. Die meisten Anweisungen, die Schema-Objekte erstellen oder ändern - darunter Anweisungen zum Erstellen von Tabellen, Indizes, materialisierten Ansichten, Publikationen, Subskriptionen und Proxy-Tabellen - sind jedoch Herstellererweiterungen des ANSI SQL-Standards und werden als nicht konform gekennzeichnet.

Die Anweisung SET OPTION einschließlich ihrer optionalen Komponenten wird nie aufgrund der Nichtkonformität mit SQL-Standards oder der Kompatibilität mit UltraLite gekennzeichnet.

Siehe auch

- „UltraLite-SQL-Sprachelemente“ [*UltraLite - Datenbankverwaltung*]
- „SET OPTION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

SQL Anywhere-Funktionen, die sich von anderen SQL-Implementierungen unterscheiden

SQL Anywhere bietet umfassende SQL-Funktionen, einschließlich: Trigger pro Zeile, pro Anweisung und INSTEAD OF-Trigger, gespeicherte SQL-Prozeduren und benutzerdefinierte Funktionen,

RECURSIVE UNION-Abfragen, allgemeine Tabellenausdrücke, Tabellenfunktionen, von LATERAL abgeleitete Tabellen, integrierte Volltextsuche, WINDOW-Aggregatfunktionen, Suche mit regulären Ausdrücken, XML-Unterstützung, materialisierte Ansichten, Snapshot-Isolation und referenzielle Integrität. Dieser Abschnitt beschreibt einige spezifische Funktionen, die von SQL Anywhere unterstützt werden und sich von anderen SQL-Datenbankimplementierungen unterscheiden.

Datumsangaben

SQL Anywhere hat Datums-, Zeit- und Zeitstempeltypen, die Jahr, Monat, Tag, Stunde, Minute, Sekunde und Sekundenbruchteile umfassen. Für Einfügungen oder Aktualisierungen in Datumsfelder oder Vergleiche mit Datumsfeldern wird ein freies Datumsformat unterstützt.

Zusätzlich sind die folgenden Vorgänge mit Datumsangaben zulässig:

- **Datum + Ganzzahl** Fügt die angegebene Anzahl von Tagen zu einem Datum hinzu
- **Datum - Ganzzahl** Zieht die angegebene Anzahl von Tagen von einem Datum ab
- **Datum - Datum** Berechnet die Anzahl von Tagen zwischen zwei Datumsangaben
- **Datum + Zeit** Macht aus einem Datum und einer Zeit einen Zeitstempel

SQL Anywhere bietet keine Unterstützung für einen INTERVAL-Datentyp, bei dem es sich um die SQL-Sprachfunktion F052 des SQL/2008-Standards handelt. SQL Anywhere bietet jedoch zahlreiche Funktionen für die Bearbeitung von Datums- und Uhrzeitangaben, wie etwa DATEADD.

Entitätsintegrität und referenzielle Integrität

SQL Anywhere unterstützt sowohl Entitätsintegrität als auch referenzielle Integrität über die PRIMARY KEY- und FOREIGN KEY-Klauseln der Anweisungen CREATE TABLE und ALTER TABLE.

```
PRIMARY KEY [ CLUSTERED ] ( column-name [ ASC | DESC ], ... )  
[NOT NULL] FOREIGN KEY [role-name]  
    [(column-name [ ASC | DESC ], ...)]  
    REFERENCES table-name [(column-name, ...)]  
    [ MATCH [ UNIQUE | SIMPLE | FULL ] ]  
    [ ON UPDATE [ CASCADE | RESTRICT | SET DEFAULT | SET NULL ] ]  
    [ ON DELETE [ CASCADE | RESTRICT | SET DEFAULT | SET NULL ] ]  
    [ CHECK ON COMMIT ] [ CLUSTERED ]
```

Die PRIMARY KEY-Klausel deklariert den Primärschlüssel für die Tabelle. SQL Anywhere erzwingt dann die Eindeutigkeit des Primärschlüssels durch die Erstellung eines eindeutigen Indexes für die Primärschlüsselspalte(n). Zwei SQL Anywhere-Erweiterungen ermöglichen die Anpassung dieses Indexes:

- **CLUSTERED** Das Schlüsselwort CLUSTERED bedeutet, dass der Primärschlüssel-Index ein Clustered-Index ist und daher angrenzende Indexeinträge im Index auf physisch angrenzende Zeilen in der Tabelle zeigen.
- **ASC | DESC** Die Sortierung - aufsteigend oder absteigend - jeder indizierten Spalte im Primärschlüssel-Index kann angepasst werden. Durch diese Anpassung kann gewährleistet werden, dass die Sortierung des Primärschlüsselindexes mit der für spezielle SQL-Abfragen erforderlichen

Sortierung, die in den ORDER BY-Klauseln der betreffenden Anweisungen angegeben ist, übereinstimmt.

Die FOREIGN KEY-Klausel legt eine Beziehung zwischen zwei Tabellen fest. Diese Beziehung wird von einer Spalte (oder Spalten) in dieser Tabelle repräsentiert, die Werte im Primärschlüssel einer anderen Tabelle enthalten muss. SQL Anywhere erstellt automatisch einen Index für die einzelnen FOREIGN KEY-Klauseln, die zum Erzwingen der Integritätsregel zur Erhaltung der referenziellen Integrität definiert sind. Die Semantik der Integritätsregel und die physischen Eigenschaften dieses Indexes können wie folgt angepasst werden:

- **CLUSTERED** Das Schlüsselwort CLUSTERED bedeutet, dass der Fremdschlüssel-Index ein Clustered-Index ist und daher angrenzende Indexeinträge im Index auf physisch angrenzende Zeilen in der Fremdtabelle zeigen.
- **ASC | DESC** Die Sortierung - aufsteigend oder absteigend - jeder indizierten Spalte im Fremdschlüsselindex kann angepasst werden. Die Sortierung des Fremdschlüsselindexes unterscheidet sich möglicherweise von der des Primärschlüsselindexes. Die Anpassung der Sortierung kann verwendet werden, um sicherzustellen, dass die Sortierung des Fremdschlüsselindexes mit der für spezielle SQL-Abfragen in der Anwendung erforderlichen Sortierung, die in den ORDER BY-Klauseln der betreffenden Anweisungen angegeben ist, übereinstimmt.
- **MATCH-Klausel** SQL Anywhere unterstützt die MATCH-Klausel, bei der es sich um die SQL-Sprachfunktion F741 des SQL/2008-Standards handelt. Zusätzlich wird in SQL Anywhere MATCH UNIQUE unterstützt, das eine Eins-zu-Eins-Relation zwischen der Primärtabelle und der entfernten Tabelle erzwingt, ohne die Notwendigkeit eines zusätzlichen UNIQUE-Indexes.

Eindeutige Indizes

SQL Anywhere unterstützt die Erstellung von eindeutigen Indizes, die auch als eindeutige Sekundärindizes bezeichnet werden, für nullwertfähige Spalten. Jeder Indexschlüssel muss standardmäßig eindeutig sein oder in mindestens einer Spalte NULL enthalten. Beispiel: Zwei Indexeinträge ('a', NULL) und ('a', NULL) werden jeweils als eindeutige Indexwerte betrachtet. SQL Anywhere unterstützt auch eindeutige Sekundärindizes, in denen Nullwerte als Spezialwerte in jeder Domäne behandelt werden. Dies wird mithilfe der WITH NULLS NOT DISTINCT-Klausel erreicht. Bei einem solchen Index werden die beiden Wertepaare ('a', NULL) und ('a', NULL) als Duplikate betrachtet.

Joins

SQL Anywhere unterstützt INNER-, LEFT OUTER-, RIGHT OUTER- und FULL OUTER-Joins. Zusätzlich zu expliziten Join-Prädikaten werden in SQL Anywhere natürliche Joins unterstützt sowie eine als KEY-Joins bekannte Erweiterung des Herstellers, womit ein implizites Join-Prädikat basierend auf den Fremdschlüsselbeziehungen der Tabellen angegeben wird.

CHAR-, NCHAR- und BINARY-Datentypen

Die internen Vorgänge von SQL Anywhere unterscheiden nicht zwischen Zeichenfolgentypen mit fester und variabler Länge (CHAR, NCHAR oder BINARY). SQL Anywhere kürzt keine nachgestellten Leerzeichen in Zeichenfolgentypen, wenn diese Werte in die Datenbank eingefügt werden. SQL Anywhere unterscheidet zwischen dem Nullwert und der leeren Zeichenfolge. Standardmäßig verwenden SQL Anywhere-Datenbanken eine Kollation ohne Berücksichtigung von Groß- und Kleinschreibung zur Unterstützung von Zeichenfolgenvergleichen, die nicht zwischen Groß- und Kleinschreibung

unterscheiden. In SQL Anywhere werden Zeichenfolgendatentypen mit fester Länge nie mit Leerzeichen aufgefüllt. Stattdessen wird die Leerzeichen-Auffüllsemantik während der Ausführung der einzelnen Zeichenfolgenvergleiche simuliert. Diese Semantik unterscheidet sich möglicherweise geringfügig von Zeichenfolgenvergleichen anderer SQL-Implementierungen.

UPDATE-Anweisungen

SQL Anywhere unterstützt teilweise die optionale SQL-Sprachfunktion T111, die es einer UPDATE-Anweisung gestattet, eine Ansicht zu referenzieren, die einen Join enthält. Außerdem gestatten es die Anweisungen UPDATE und UPDATE WHERE CURRENT OF, mehr als eine Tabelle in der SET-Klausel der Anweisung zu referenzieren, und die FROM-Klausel einer UPDATE-Anweisung kann aus einem beliebigen Tabellenausdruck mit Joins und abgeleiteten Tabellen bestehen.

SQL Anywhere gestattet es auch, die Anweisungen UPDATE, INSERT, MERGE und DELETE in eine andere SQL-Anweisung als abgeleitete Tabelle einzubetten. Einer der Vorteile dieser Unterstützung ist, dass Sie auf einfache Weise eine Abfrage erstellen können, die die Gruppe von Zeilen liefert, die von einer UPDATE-Anweisung geändert wurde.

Tabellenfunktionen

Sie können in SQL Anywhere die Ergebnismenge einer gespeicherten Prozedur in der FROM-Klausel einer Anweisung als Tabelle referenzieren. Dieses Feature wird in der Regel als Tabellenfunktion bezeichnet. Bei Tabellenfunktionen handelt es sich um die SQL-Sprachfunktion T326 des SQL/2008-Standards. In dem Standard werden Tabellenfunktionen mit dem TABLE-Schlüsselwort angegeben. In SQL Anywhere ist die Verwendung des TABLE-Schlüsselworts nicht erforderlich. Eine gespeicherte Prozedur kann direkt in der FROM-Klausel referenziert werden, optional mit einem Korrelationsnamen und der Angabe des Schemas der Ergebnismenge, die von der Prozedur zurückgegeben wird.

Das folgende Beispiel verknüpft das Ergebnis der gespeicherten Prozedur ShowCustomerProducts mit der Basistabelle "Products". Die Referenz der gespeicherten Prozedur wird von einer expliziten Deklaration des Schemas des Prozedurergebnisses unter Verwendung der WITH-Klausel begleitet:

```
SELECT sp.ident, sp.quantity, Products.name
FROM ShowCustomerProducts( 149 )
    WITH ( ident INT, description CHAR(20), quantity INT ) sp
JOIN Products ON sp.ident = Products.ID
```

Materialisierte Ansichten

SQL Anywhere unterstützt materialisierten Ansichten, bei denen es sich um vorberechnete Ergebnismengen handelt, die direkt oder indirekt innerhalb einer SQL-Abfrage referenziert werden können. In SQL Anywhere können sofort und manuell verwaltete Ansichten mit der VIEW MATERIALIZED CREATE-Anweisung erstellt werden. Andere Datenbankprodukte verwenden möglicherweise andere Begriffe zur Beschreibung dieser Funktion.

Cursor

SQL Anywhere unterstützt die optionale SQL-Sprachenfunktion F431 des SQL/2008-Standards. In SQL Anywhere sind alle Cursor bidirektional scrollfähig, es sei denn, sie werden explizit als FORWARD ONLY deklariert. Anwendungen können durch einen Cursor blättern, indem sie in der FETCH-Anweisung die relative oder absolute Positionierung oder deren Äquivalent anderer Schnittstellen für die Anwendungsprogrammierung wie ODBC verwenden.

SQL Anywhere unterstützt wertempfindliche Cursor sowie Cursor, die nach Zeilenmitgliedschaft unterscheiden. Allgemein übliche Cursorarten, einschließlich INSENSITIVE-, KEYSET-DRIVEN- und SENSITIVE-Cursor, werden ebenfalls unterstützt. Bei der Verwendung von Embedded SQL können Cursor-Positionen frei in der Anweisung FETCH verschoben werden. Cursor können vorwärts und rückwärts, relativ zur aktuellen Position oder einer angegebenen Anzahl von Datensätzen vom Anfang oder Ende des Cursors verschoben werden.

Standardmäßig sind Cursor in Embedded SQL- und SQL-Prozeduren, benutzerdefinierten Funktionen und Triggern aktualisierbar. Sie können mit der FOR UPDATE-Klausel explizit aktualisierbar gemacht werden. Durch die Angabe der FOR UPDATE-Klausel alleine werden jedoch keine Sperren für die Zeilen in der Ergebnismenge des Cursors erworben. Um sicherzustellen, dass die Zeilen in der Ergebnismenge nicht durch andere Transaktionen geändert werden können, können Sie eine der folgenden beiden Klauseln verwenden:

- **FOR UPDATE BY LOCK** Diese Klausel bewirkt, dass der Datenbankserver Absichtszeilensperren für abgerufene Zeilen der Ergebnismenge erwirbt. Dies sind langfristige Sperren, die gehalten werden, bis die Transaktion festgeschrieben oder zurückgesetzt wird.
- **FOR UPDATE BY { VALUES | TIMESTAMP }** Der SQL Anywhere-Datenbankserver verwendet einen Keyset-gesteuerten Cursor, damit die Anwendung benachrichtigt werden kann, wenn Zeilen geändert oder gelöscht werden, während die Ergebnismenge durchlaufen wird.

Alias-Referenzen

SQL Anywhere erlaubt, dass Aliasausdrücke in der SELECT-Liste einer Abfrage in anderen Teilen der Abfrage referenziert werden. Die meisten anderen SQL-Implementierungen und der Standard SQL/2008 lassen dieses Verhalten nicht zu. Sie können z.B. folgende SQL-Abfrage verwenden:

```
SELECT column-or-expression AS alias-name
FROM table-reference
WHERE alias-name = expression
```

Aliasnamen können an einer beliebigen Stelle im SELECT-Block verwendet werden, auch in anderen SELECT-Listenausdrücken, die wiederum zusätzliche Aliase festlegen. Zyklische Aliasreferenzen sind nicht zulässig. Wenn der für einen Ausdruck angegebene Alias mit dem Namen einer Spalte oder Variablen im Namespace des SELECT-Blocks identisch ist, schließt die Aliasdefinition die Spalte oder Variable aus. Spaltennamen können jedoch in solchen Fällen explizit mit Tabellennamen qualifiziert werden.

Snapshot-Isolation

SQL Anywhere unterstützt die Snapshot-Isolation, die auch als Multi-Version Concurrency Control (MVCC) bezeichnet wird. In anderen SQL-Implementierungen, die die Snapshot-Isolation unterstützen, werden Schreibvorgangskonflikte, d.h. gleichzeitige Aktualisierungen durch zwei oder mehr Transaktionen für dieselbe Zeile, erst zum Zeitpunkt des COMMIT erkannt. In diesen Fällen gewinnt in der Regel die erste COMMIT-Anweisung und die anderen am Konflikt beteiligten Transaktionen müssen abgebrochen werden.

In SQL Anywhere werden bei Schreibvorgängen für Zeilen Schreibsperren für Zeilen erworben, sodass Snapshot-Transaktionen neben Transaktionen vorhanden sein können, die auf ANSI-Isolationsstufen ausgeführt werden. Aus diesem Grund führt ein Schreibvorgangskonflikt in SQL Anywhere zu einer

Blockierung. Das genaue Verhalten kann durch die Verbindungsoptionen BLOCKING und BLOCKING_TIMEOUT gesteuert werden.

Siehe auch

- „Datums- und Zeitfunktionen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Zeichendatentypen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Schlüssel-Joins“ auf Seite 523
- „FROM-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Materialisierte Ansichten“ auf Seite 56
- „SQL Anywhere-Cursor“ [[SQL Anywhere Server - Programmierung](#)]
- „DECLARE CURSOR-Anweisung [ESQL] [SP]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Snapshot-Isolation“ auf Seite 888

Watcom SQL

Der von SQL Anywhere unterstützte SQL-Dialekt wird als Watcom SQL bezeichnet. Die ursprüngliche Version von SQL Anywhere wurde bei seiner Einführung 1992 Watcom SQL genannt. Der Begriff Watcom SQL wird immer noch für den von SQL Anywhere unterstützten SQL-Dialekt verwendet.

SQL Anywhere unterstützt auch eine große Untergruppe von Transact-SQL, einem SQL-Dialekt, der von Sybase Adaptive Server Enterprise unterstützt wird.

Siehe auch

- „Transact-SQL-Kompatibilität“ auf Seite 648

Transact-SQL-Kompatibilität

SQL Anywhere unterstützt eine große Untergruppe von Transact-SQL, einem SQL-Dialekt, der von Sybase Adaptive Server Enterprise unterstützt wird. In diesem Abschnitt wird die SQL-Kompatibilität zwischen SQL Anywhere und Adaptive Server Enterprise beschrieben.

Ziele

Die Transact-SQL-Unterstützung in SQL Anywhere hat folgenden Zweck:

- **Anwendungsportierbarkeit** Viele Anwendungen, gespeicherte Prozeduren und Batchdateien können so geschrieben werden, dass sie sowohl in Adaptive Server Enterprise- als auch in SQL Anywhere-Datenbanken verwendet werden können.
- **Datenportierbarkeit** SQL Anywhere- und Adaptive Server Enterprise-Datenbanken können mit wenig Aufwand Daten untereinander austauschen und replizieren.

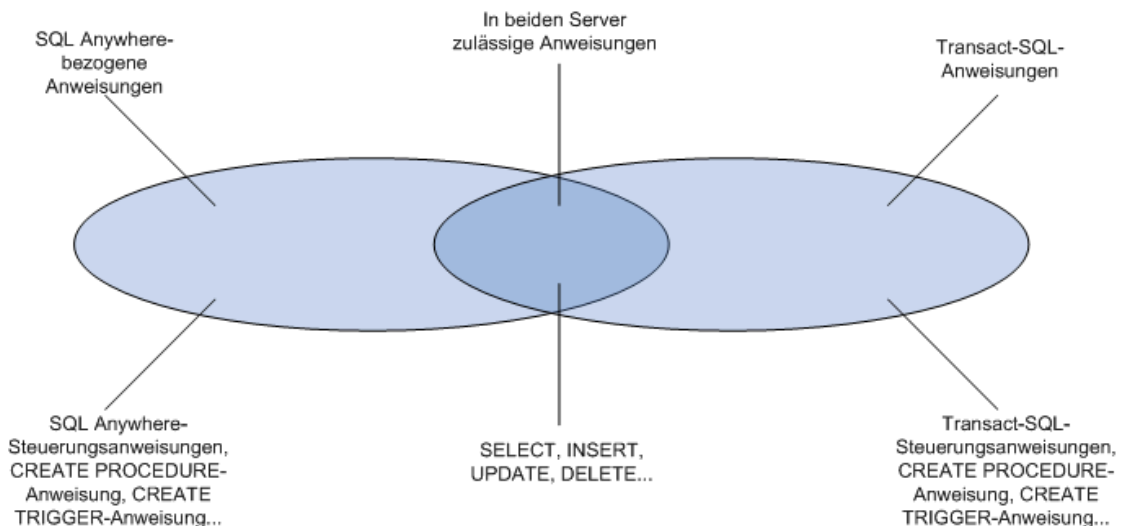
Ziel ist es, Anwendungen zu schreiben, die sowohl mit Adaptive Server Enterprise als auch mit SQL Anywhere arbeiten. Bestehende Adaptive Server Enterprise-Anwendungen erfordern im Allgemeinen einige Änderungen, damit sie in einer SQL Anywhere-Datenbank laufen können.

Wie Transact-SQL unterstützt wird

Die Transact-SQL-Unterstützung in SQL Anywhere hat die folgende Form:

- Viele SQL-Anweisungen sind zwischen SQL Anywhere und Adaptive Server Enterprise kompatibel.
- Für einige Anweisungen, insbesondere in der in Prozeduren, Triggern und Batches verwendeten Prozedursprache, wird zusätzlich zur Syntax, die von Vorgängerversionen von SQL Anywhere unterstützt wird, eine separate Transact-SQL-Anweisung unterstützt. Bei diesen Anweisungen unterstützt SQL Anywhere zwei Dialekte von SQL. Diese Dialekte werden als Transact-SQL (der Dialekt von Adaptive Server Enterprise) und Watcom SQL (der Dialekt von SQL Anywhere) bezeichnet.
- Eine Prozedur, ein Trigger oder ein Batch werden entweder im Transact-SQL- oder im Watcom SQL-Dialekt ausgeführt. Sie dürfen innerhalb des Batches oder der Prozedur nur Steueranweisungen eines der beiden Dialekte verwenden. Jeder Dialekt hat beispielsweise verschiedene Steueranweisungen.

Das folgende Diagramm veranschaulicht, wie sich die beiden Dialekte überlappen.



Ähnlichkeiten und Unterschiede

SQL Anywhere unterstützt einen großen Anteil der Sprachelemente, Funktionen und Anweisungen in Transact-SQL, um vorhandene Daten zu verarbeiten. SQL Anywhere unterstützt beispielsweise alle numerischen, Aggregat- und Datums- bzw. Uhrzeitfunktionen sowie alle Zeichenfolgenfunktionen (bis auf eine). Außerdem unterstützt SQL Anywhere erweiterte DELETE- und UPDATE-Anweisungen, die Joins verwenden.

Außerdem unterstützt SQL Anywhere auch einen großen Anteil der Transact-SQL-Sprache für gespeicherte Prozeduren (CREATE PROCEDURE- und CREATE TRIGGER-Syntax, Steueranweisungen usw.) sowie viele Aspekte der Anweisungen für die Datendefinitionssprache in Transact-SQL.

Es bestehen Unterschiede im Entwurf der architektonischen und Konfigurationseinrichtungen, die von jedem Produkt unterstützt werden. Deviceverwaltung, Benutzerverwaltung und Wartungstasks, wie z.B. Sicherungen, sind eher systemspezifisch. Sogar hier stehen in SQL Anywhere Transact-SQL-Systemtabellen als Ansichten zur Verfügung, wenn die Tabellen, die in SQL Anywhere bedeutungslos sind, keine Zeilen haben. Außerdem bietet SQL Anywhere eine Reihe von Systemprozeduren für einige der gewöhnlichen Administrationsaufgaben.

Dieser Abschnitt befasst sich zuerst mit einigen Aspekten auf Systemebene, bei denen die Unterschiede am deutlichsten sind, und erläutert dann Sprachaspekte der Datenmanipulation und Datendefinition der Dialekte, bei denen eine hohe Kompatibilität vorliegt.

Gilt nur für Transact-SQL

Es gibt SQL-Anweisungen, die von SQL Anywhere unterstützt werden und lediglich Teil des einen und nicht des anderen Dialekts sind. Die beiden Dialekte können innerhalb einer Prozedur, eines Triggers oder eines Batches nicht gemischt werden. SQL Anywhere unterstützt z.B. die folgenden Anweisungen, jedoch nur als Teil des Transact-SQL-Dialekts:

- Transact-SQL-Steueranweisungen IF und WHILE
- Transact-SQL-Anweisung EXECUTE
- Transact-SQL-Anweisungen CREATE PROCEDURE und CREATE TRIGGER
- Transact-SQL-Anweisung BEGIN TRANSACTION
- SQL-Anweisungen, die *nicht* durch Semikolons getrennt werden, sind Teil einer Transact-SQL-Prozedur oder eines Transact-SQL-Batches.

Nur SQL Anywhere

Adaptive Server Enterprise unterstützt nicht die folgenden Anweisungen:

- LOOP- und FOR-Steueranweisungen
- SQL Anywhere-Versionen von IF und WHILE
- CALL-Anweisung
- SIGNAL-Anweisung
- SQL Anywhere-Versionen der Anweisungen CREATE PROCEDURE, CREATE FUNCTION und CREATE TRIGGER
- SQL-Anweisungen, getrennt durch Semikolons

Hinweise

Die beiden Dialekte können nicht innerhalb einer Prozedur, eines Triggers oder eines Batches gemischt werden.

- Sie können für Transact-SQL geltende Anweisungen in einem Batch, einer Prozedur oder einem Trigger mit Anweisungen kombinieren, die Teil beider Dialekte sind.

- Sie können von Adaptive Server Enterprise nicht unterstützte Anweisungen zusammen mit Anweisungen, die von beiden Servern unterstützt werden, in einem Batch, einer Prozedur oder einem Trigger verwenden.
- Sie können nur für Transact-SQL geltende Anweisungen nicht zusammen mit nur für SQL Anywhere geltenden Anweisungen in einem Batch, einer Prozedur oder einem Trigger verwenden.

Adaptive Server Enterprise-Architekturen

Adaptive Server Enterprise und SQL Anywhere sind Komplementärprodukte, deren Architekturen so entworfen wurden, dass sie ihren speziellen Aufgaben gerecht werden.

In diesem Abschnitt werden architektonische Unterschiede zwischen Adaptive Server Enterprise und SQL Anywhere beschrieben. Außerdem werden die Adaptive Server Enterprise-ähnlichen Tools beschrieben, die SQL Anywhere für eine kompatible Datenbankverwaltung zur Verfügung stellt.

Server und Datenbanken

In Adaptive Server Enterprise ist die Beziehung zwischen Servern und Datenbanken anders als in SQL Anywhere.

In Adaptive Server Enterprise existiert jede Datenbank innerhalb eines Servers, und jeder Server kann mehrere Datenbanken enthalten. Den Benutzern können Login-Rechte für den Server erteilt werden, sodass sie eine Verbindung zum Server herstellen können. Sie können dann jede Datenbank auf diesem Server je nach den ihnen erteilten Berechtigungen nutzen. Systemweite Systemtabellen in der Master-Datenbank enthalten Informationen, die allen Datenbanken auf dem Server gemein sind.

Keine Master-Datenbank in SQL Anywhere

In SQL Anywhere gibt es keine Stufe, die der Master-Datenbank in Adaptive Server Enterprise entspricht. Stattdessen ist jede Datenbank eine unabhängige Entität, die sämtliche Systemtabellen enthält. Den Benutzern können Verbindungsrechte zur Datenbank, jedoch nicht zum Server erteilt werden. Wenn ein Benutzer eine Verbindung herstellt, handelt es sich dabei um eine Verbindung zu einer individuellen Datenbank. Es gibt keinen systemweiten Systemtabellen-Satz, der in einer Master-Datenbank abgelegt wird. Jeder SQL Anywhere-Datenbankserver kann dynamisch mehrere Datenbanken laden und entladen, und die Benutzer können jeweils eine unabhängige Verbindung aufrechterhalten.

SQL Anywhere bietet sowohl in seiner Transact-SQL- als auch seiner Open Server-Unterstützung Tools, mit denen einige Aufgaben in einer dem Adaptive Server Enterprise entsprechenden Weise ausgeführt werden können. Beispielsweise bietet SQL Anywhere eine Implementierung der Adaptive Server Enterprise-Systemprozedur `sp_addlogin`, welche die am ehesten entsprechende Aktion ausführt: Einen Benutzer der Datenbank hinzufügen.

Dateibearbeitungs-Anweisungen

SQL Anywhere bietet keine Unterstützung für die Transact-SQL-Anweisungen `DUMP DATABASE` und `LOAD DATABASE` zum Sichern und Wiederherstellen von Daten. Stattdessen verfügt SQL Anywhere

über die eigenen Anweisungen BACKUP DATABASE und RESTORE DATABASE, die eine andere Syntax haben.

Siehe auch

- „SQL Anywhere als Open Server“ [[SQL Anywhere Server - Datenbankadministration](#)]

Deviceverwaltung

SQL Anywhere und Adaptive Server Enterprise verwenden unterschiedliche Modelle für die Device- und Plattenspeicherverwaltung, wodurch die unterschiedlichen Einsatzmöglichkeiten der beiden Produkte zum Ausdruck kommen. Während Adaptive Server Enterprise ein vollständiges Ressourcenverwaltungsschema unter Verwendung einer Vielzahl von Transact-SQL-Anweisungen erstellt, verwaltet SQL Anywhere seine eigenen Ressourcen selbst, und seine Datenbanken sind normale Betriebssystemdateien.

SQL Anywhere unterstützt keine DISK-Anweisungen in Transact-SQL, wie DISK INIT, DISK MIRROR, DISK REFIT, DISK REINIT, DISK REMIRROR und DISK UNMIRROR.

Siehe auch

- „Datenbankdateitypen“ [[SQL Anywhere Server - Datenbankadministration](#)]

Standardwerte und Regeln

SQL Anywhere unterstützt nicht die Transact-SQL-Anweisungen CREATE DEFAULT oder CREATE RULE. Mithilfe der CREATE DOMAIN-Anweisung kann ein Standardwert und eine Regel (die so genannte CHECK-Bedingung) in die Definition einer Domäne eingefügt werden und so ähnliche Funktionen wie die Transact-SQL-Anweisungen CREATE DEFAULT und CREATE RULE bieten.

In SQL Anywhere können einer Domäne ein Standardwert und eine CHECK-Bedingung zugeordnet sein, die für alle Spalten gelten, welche mit diesem Datentyp definiert wurden. Sie erstellen die Domäne mit der Anweisung CREATE DOMAIN.

Standardwerte und Regeln oder CHECK-Bedingungen können mit der CREATE TABLE-Anweisung oder der ALTER TABLE-Anweisung für individuelle Spalten definiert werden.

In Adaptive Server Enterprise erstellt die CREATE DEFAULT-Anweisung einen benannten Standardwert. Dieser Standardwert kann als Standardwert für Spalten verwendet werden, indem der Standardwert an eine bestimmte Spalte gebunden wird, oder als Standardwert für alle Spalten einer Domäne, indem der Standardwert an den Datentyp gebunden wird, mit der Systemprozedur "sp_bindefault". Die CREATE RULE-Anweisung erstellt eine benannte Regel, mit der die Domäne für Spalten definiert werden kann, indem die Regel an eine bestimmte Spalte gebunden wird, oder als eine Regel, die für alle Spalten einer Domäne verwendet werden kann, indem die Regel an den Datentyp gebunden wird. Eine Regel wird mit der Systemprozedur "sp_bindrule" an einen Datentyp oder eine Spalte gebunden.

Siehe auch

- „CREATE DOMAIN-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Suchbedingungen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Systemtabellen

Zusätzlich zu seinen eigenen Systemtabellen bietet SQL Anywhere eine Gruppe von Systemansichten, die relevante Teile der Systemtabellen von Adaptive Server Enterprise simulieren.

Die Systemtabellen von SQL Anywhere werden vollständig innerhalb jeder Datenbank gehalten, während die Systemtabellen von Adaptive Server Enterprise teils in jeder Datenbank und teils in der Master-Datenbank gehalten werden. Die Architektur von SQL Anywhere enthält keine Master-Datenbank.

In Adaptive Server Enterprise ist der Datenbankeigentümer (Benutzer dbo) der Eigentümer der Systemtabellen. In SQL Anywhere ist der Systemeigentümer (Benutzer SYS) der Eigentümer der Systemtabellen. Der Benutzer "dbo" ist Eigentümer der von SQL Anywhere bereitgestellten Adaptive Server Anywhere-kompatiblen Systemansichten.

Siehe auch

- „Ansichten für Transact-SQL-Kompatibilität“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Administrative Rollen

Adaptive Server Enterprise bietet eine umfangreichere Gruppe administrativer Rollen als SQL Anywhere. In Adaptive Server Enterprise gibt es eine Reihe verschiedener Rollen, obwohl mehr als einem Login-Konto für einen Adaptive Server Enterprise eine beliebige Rolle erteilt werden und ein Konto mehr als eine Rolle besitzen kann.

Adaptive Server Enterprise Rollen

In Adaptive Server Enterprise gibt es folgende Rollen:

- **Systemadministrator** Verantwortlich für allgemeine administrative Tasks, die nicht mit spezifischen Anwendungen im Zusammenhang stehen. Er hat Zugang zu sämtlichen Datenbankobjekten.
- **Sicherheitsadministrator** Verantwortlich für sicherheitsrelevante Tasks in Adaptive Server Enterprise. Er hat jedoch keine speziellen Berechtigungen für Datenbankobjekte.
- **Datenbankeigentümer** Hat sämtliche Privilegien für Objekte innerhalb der Datenbank, deren Eigentümer er ist, kann Benutzer zur Datenbank hinzufügen und anderen Benutzern die erforderlichen Privilegien erteilen, um innerhalb der Datenbank Objekte zu erstellen und Anweisungen auszuführen.
- **Datendefinitionsanweisungen** Benutzern können Privilegien für spezifische Datendefinitionsanweisungen, z.B. CREATE TABLE oder CREATE VIEW, erteilt werden, sodass der Benutzer Datenbankobjekte erstellen kann.

- **Objekteigentümer** Jedes Datenbankobjekt hat einen Eigentümer, der anderen Benutzern Privilegien für den Zugriff auf das Objekt erteilen kann. Der Eigentümer eines Objekts besitzt automatisch alle Privilegien für dieses Objekt.
- Die Datenbankadministratorrolle (SYS_AUTH_DBA_ROLE-Kompatibilitätsrolle) hat, wie der Datenbankadministrator in Adaptive Server Enterprise, sämtliche Privilegien für alle Objekte innerhalb der Datenbank (außer Objekten mit dem Eigentümer SYS) und kann anderen Benutzern die erforderlichen Privilegien erteilen, um innerhalb der Datenbank Objekte zu erstellen und Anweisungen auszuführen. Der Standard-Datenbankadministrator ist der Benutzer DBA.
- Mit der RESOURCE-Rolle (Kompatibilitätsrolle SYS_AUTH_RESOURCE_ROLE) kann der Benutzer innerhalb einer Datenbank jede Art von Objekt erstellen. Dies ersetzt das Schema des Adaptive Server Enterprise, bei dem Berechtigungen für individuelle CREATE-Anweisungen erteilt werden.
- Die Objekteigentümer von SQL Anywhere entsprechen den Objekteigentümern von Adaptive Server Enterprise. Der Eigentümer eines Objekts hat automatisch alle Privilegien für dieses Objekt, einschließlich des Rechts, Privilegien zu erteilen.

Um einen nahtlosen Zugriff auf Daten zu gewährleisten, die sowohl in Adaptive Server Enterprise als auch in SQL Anywhere gespeichert sind, sollten Sie Benutzer-IDs mit den entsprechenden Privilegien in der Datenbank erstellen und diese Benutzer-ID zum Erstellen von Objekten verwenden. Wenn in jeder Umgebung dieselbe Benutzer-ID verwendet wird, können Objektnamen und Objektqualifizierer in beiden Datenbanken identisch sein, sodass kompatibler Zugriff gewährleistet ist.

Benutzer und Gruppen

SQL Anywhere unterstützt die folgenden Systemprozeduren von Adaptive Server Enterprise für die Verwaltung von Benutzern und Gruppen.

Systemprozedur	Beschreibung
sp_addlogin	Damit wird in Adaptive Server Enterprise dem Server ein Benutzer hinzugefügt. In SQL Anywhere wird einer Datenbank ein Benutzer hinzugefügt.
sp_adduser	Damit wird sowohl in Adaptive Server Enterprise als auch in SQL Anywhere einer Datenbank ein Benutzer hinzugefügt. Während sich diese Prozedur in Adaptive Server Enterprise von "sp_addlogin" unterscheidet, sind sie in SQL Anywhere gleich.
sp_addgroup	Fügt einer Datenbank eine Gruppe hinzu
sp_changegroup	Fügt einer Gruppe einen Benutzer hinzu oder verschiebt einen Benutzer von einer Gruppe in eine andere
sp_droplogin	Entfernt in Adaptive Server Enterprise einen Benutzer aus dem Server. Entfernt in SQL Anywhere einen Benutzer aus der Datenbank.

Systemprozedur	Beschreibung
sp_dropuser	Entfernt einen Benutzer aus der Datenbank
sp_dropgroup	Entfernt eine Gruppe aus der Datenbank

In Adaptive Server Enterprise sind Login-IDs serverweit gültig. In SQL Anywhere gehören Benutzer zu einzelnen Datenbanken.

Datenbankobjektprivilegien

Die GRANT- und REVOKE-Anweisungen, die in Adaptive Server Enterprise und SQL Anywhere zum Erteilen von Privilegien für einzelne Datenbankobjekte verwendet werden, sind sehr ähnlich. Beide lassen SELECT-, INSERT-, DELETE-, UPDATE- und REFERENCE-Privilegien für Datenbanktabellen und Ansichten zu sowie UPDATE-Privilegien für ausgewählte Spalten in Datenbanktabellen. Beide lassen das Erteilen von EXECUTE-Privilegien für gespeicherte Prozeduren zu.

Die folgende Anweisung ist beispielsweise sowohl in Adaptive Server Enterprise als auch in SQL Anywhere gültig:

```
GRANT INSERT, DELETE
ON Employees
TO MARY, SALES;
```

Diese Anweisung erteilt der Benutzerin MARY und der Gruppe SALES die erforderlichen Privilegien, um die Anweisungen INSERT und DELETE in der Tabelle "Employees" zu verwenden.

Sowohl SQL Anywhere als auch Adaptive Server Enterprise unterstützen die WITH GRANT OPTION-Klausel, die es dem Empfänger von Privilegien ermöglicht, diese seinerseits zu erteilen, obwohl SQL Anywhere es nicht zulässt, dass WITH GRANT OPTION für eine GRANT EXECUTE-Anweisung verwendet wird. In SQL Anywhere können Sie WITH GRANT OPTION nur für Benutzer angeben. Mitglieder von Gruppen erben nicht WITH GRANT OPTION, wenn dies einer Gruppe erteilt wird.

Datenbankweite Privilegien

Adaptive Server Enterprise und SQL Anywhere verwenden unterschiedliche Modelle für datenbankweite Privilegien. SQL Anywhere verwendet die DBA-Rolle, um einem Benutzer die volle Berechtigung innerhalb einer Datenbank zu geben. Der Systemadministrator in Adaptive Server Enterprise hat dieses Privileg für alle Datenbanken auf einem Server. Die DBA-Rolle für eine Datenbank in SQL Anywhere unterscheidet sich jedoch von den Berechtigungen eines Datenbankeigentümers in Adaptive Server Enterprise, der die Adaptive Server Enterprise-Anweisung SETUSER verwenden muss, um Berechtigungen für Objekte zu erlangen, deren Eigentümer andere Benutzer sind.

Siehe auch

- „System- und Katalogprozeduren von Adaptive Server Enterprise“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Benutzer und Gruppen“ auf Seite 654

Transact-SQL-kompatible Datenbanken

Einige Verhaltensunterschiede zwischen SQL Anywhere und Adaptive Server Enterprise können vermieden werden, indem Sie geeignete Optionen auswählen, wenn Sie die Datenbank erstellen oder wenn Sie eine bereits bestehende Datenbank neu aufbauen. Weitere Unterschiede können durch Optionen auf Verbindungsebene unter Verwendung der SET TEMPORARY OPTION-Anweisung in SQL Anywhere oder der SET-Anweisung in Adaptive Server Enterprise verwaltet werden.

Beachtung der Groß- und Kleinschreibung in der Datenbank aktivieren

Standardmäßig wird bei Vergleichen von Zeichenfolgen in Adaptive Server Enterprise-Datenbanken die Groß- und Kleinschreibung berücksichtigt, während das in SQL Anywhere nicht der Fall ist.

Wenn Sie eine Adaptive Server Enterprise-kompatible Datenbank mit SQL Anywhere erstellen, sollten Sie die Option so festlegen, dass die Groß- und Kleinschreibung berücksichtigt wird.

- Wenn Sie Sybase Central benutzen, finden Sie diese Option im **Assistenten zum Erstellen einer Datenbank**.
- Wenn Sie das Dienstprogramm "dbinit" verwenden, benutzen Sie die Option -c.
- Wenn Sie die CREATE DATABASE-Anweisung verwenden, geben Sie die CASE RESPECT-Klausel an.

Bei Vergleichen nachgestellte Leerzeichen ignorieren

Wenn Sie eine Adaptive Server Enterprise-kompatible Datenbank mit SQL Anywhere erstellen, sollten Sie die Option so festlegen, dass nachgestellte Leerzeichen bei Vergleichen ignoriert werden.

- Wenn Sie Sybase Central benutzen, finden Sie diese Option im **Assistenten zum Erstellen einer Datenbank**.
- Wenn Sie das Dienstprogramm "dbinit" verwenden, benutzen Sie die Option -b.
- Wenn Sie die CREATE DATABASE-Anweisung verwenden, geben Sie die BLANK PADDING ON-Klausel an.

Wenn Sie diese Option wählen, betrachten SQL Anywhere und Adaptive Server Enterprise die folgenden beiden Zeichenfolgen als gleich:

```
'ignore the trailing blanks '
'ignore the trailing blanks'
```

Wenn Sie diese Option nicht wählen, betrachtet SQL Anywhere die beiden obigen Zeichenfolgen als verschieden.

Ein Nebeneffekt dieser Option ist, dass die Zeichenfolgen mit Leerzeichen aufgefüllt werden, wenn sie von einer Clientanwendung abgerufen werden.

Alte Systemansichten entfernen

Ältere Versionen von SQL Anywhere haben zwei Systemansichten verwendet, deren Namen mit den für die Kompatibilität bereitgestellten Systemansichten von Adaptive Server Enterprise in Konflikt standen.

Diese Ansichten sind SYSCOLUMNS und SYSINDEXES. Wenn Sie Open Client- oder JDBC-Interfaces benutzen, sollten Sie bei der Erstellung Ihrer Datenbank diese Ansichten ausschließen. Das erreichen Sie mit der Befehlszeilenoption "dbinit -k".

Falls Sie diese Option beim Erstellen der Datenbank nicht verwenden, wird durch das Ausführen der Anweisung `SELECT * FROM SYSCOLUMNS;` der SQLE_AMBIGUOUS_TABLE_NAME-Fehler erzeugt.

Siehe auch

- „SET OPTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Datenbankoptionen“ [[SQL Anywhere Server - Datenbankadministration](#)]

Transact-SQL-kompatible Datenbanken erstellen (Sybase Central)

Sie können Sybase Central verwenden, um eine Transact-SQL-kompatible Datenbank zu erstellen.

Voraussetzungen

Standardmäßig müssen Sie das SERVER OPERATOR-Systemprivileg haben. Die erforderlichen Privilegien können mithilfe der Datenbankserveroption -gu geändert werden.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie auf **Extras » SQL Anywhere 16 » Datenbank erstellen**.
3. Befolgen Sie die Anweisungen des Assistenten.

Klicken Sie auf der Seite **Zusätzliche Einstellungen angeben** auf **Adaptive Server Enterprise emulieren** und anschließend auf **Weiter**.

4. Folgen Sie den weiteren Anweisungen des Assistenten.

Ergebnisse

Eine Transact-SQL-kompatible Datenbank wird erstellt. Die Datenbank ist mit Leerzeichen aufgefüllt und berücksichtigt die Groß- und Kleinschreibung. Sie enthält nicht die Systemansichten SYS.SYSCOLUMNS und SYS.SYSINDEXES.

Siehe auch

- „Transact-SQL- und SQL/2008-Kompatibilitätsoptionen“ [[SQL Anywhere Server - Datenbankadministration](#)]

Transact-SQL-kompatible Datenbanken erstellen(Befehlszeile)

Sie können die Befehlszeile verwenden, um eine Transact-SQL-kompatible Datenbank zu erstellen. Im verbleibenden Teil des Abschnitts wird beschrieben, welche Optionen festgelegt werden müssen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

- Führen Sie den folgenden dbinit-Befehl aus:

```
dbinit -b -c -k -dba DBA,sql db-name.db
```

In diesem Befehl wird mit -b die Datenbank mit Leerzeichen aufgefüllt, mit -c berücksichtigt die Datenbank die Groß- und Kleinschreibung und mit -k werden die Systemansichten SYS.SYSCOLUMNS und SYS.SYSINDEXES nicht erstellt.

Ergebnisse

Eine Transact-SQL-kompatible Datenbank wird erstellt.

Siehe auch

- „Dienstprogramm Initialisierung (dbinit)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Transact-SQL- und SQL/2008-Kompatibilitätsoptionen“ [[SQL Anywhere Server - Datenbankadministration](#)]

Transact-SQL-kompatible Datenbanken erstellen (SQL)

Sie können SQL verwenden, um eine Transact-SQL-kompatible Datenbank zu erstellen.

Voraussetzungen

Standardmäßig müssen Sie das SERVER OPERATOR-Systemprivileg haben. Die erforderlichen Privilegien können mithilfe der Datenbankserveroption -gu geändert werden.

Aufgabe

1. Stellen Sie eine Verbindung mit einer beliebigen SQL Anywhere-Datenbank her.
2. Führen Sie die folgende Anweisung in Interactive SQL aus:

```
CREATE DATABASE 'db-name.db'  
DBA USER 'DBA' DBA PASSWORD 'sql'  
ASE COMPATIBLE  
CASE RESPECT  
BLANK PADDING ON;
```

In dieser Anweisung verhindert die ASE COMPATIBLE-Klausel das Erstellen der Systemansichten SYS.SYSCOLUMNS und SYS.SYSINDEXES.

Ergebnisse

Eine Transact-SQL-kompatible Datenbank wird erstellt. Die Datenbank ist mit Leerzeichen aufgefüllt und berücksichtigt die Groß- und Kleinschreibung. Sie enthält nicht die Systemansichten SYS.SYSCOLUMNS und SYS.SYSINDEXES.

Siehe auch

- „CREATE DATABASE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Transact-SQL- und SQL/2008-Kompatibilitätsoptionen“ [[SQL Anywhere Server - Datenbankadministration](#)]

Optionen für die Transact-SQL-Kompatibilität

Datenbankoptionen werden in SQL Anywhere mit der Anweisung SET OPTION festgelegt. Einige Datenbankoptionseinstellungen sind für Transact-SQL-Verhalten relevant.

Option allow_nulls_by_default einstellen

Standardmäßig lässt Adaptive Server Enterprise NULL in neuen Spalten nicht zu, es sei denn, sie genehmigen diese ausdrücklich in den Spalteneinstellungen. SQL Anywhere lässt NULL in neuen Spalten standardmäßig zu, was mit dem SQL/2008 ISO-Standard kompatibel ist.

Damit das Verhalten von Adaptive Server Enterprise SQL/2008-kompatibel wird, müssen Sie die sp_dboption-Systemprozedur verwenden, um die allow_nulls_by_default-Option auf TRUE zu setzen.

Damit das Verhalten von SQL Anywhere Transact-SQL-kompatibel wird, müssen Sie die Option "allow_nulls_by_default" auf "Off" setzen. Das erreichen Sie, indem Sie die SET OPTION-Anweisung wie folgt verwenden:

```
SET OPTION PUBLIC.allow_nulls_by_default = 'Off';
```

Option quoted_identifier einstellen

Standardmäßig behandelt Adaptive Server Enterprise Bezeichner und Zeichenfolgen anders als SQL Anywhere, dessen Verhalten dem SQL/2008 ISO-Standard entspricht.

Die quoted_identifier-Option ist sowohl in Adaptive Server Enterprise als auch in SQL Anywhere verfügbar. Sie müssen sicherstellen, dass die Option in beiden Datenbanken auf den gleichen Wert gesetzt ist, damit Bezeichner und Zeichenfolgen in kompatibler Weise behandelt werden.

Um das Verhalten von SQL/2008 zu erreichen, setzen Sie die quoted_identifier-Option sowohl in Adaptive Server Enterprise als auch in SQL Anywhere auf "On".

Um das Verhalten von Transact-SQL zu erreichen, setzen Sie die Option "quoted_identifier" sowohl in Adaptive Server Enterprise als auch in SQL Anywhere auf "Off". Wenn Sie sich hierfür entscheiden, können Sie Bezeichner in Anführungsstrichen, die die selben sind wie Schlüsselwörter, nicht mehr verwenden. Eine Alternative zum Festlegen der Option "quoted_identifier" auf "Off" besteht darin, alle

Zeichenfolgen in SQL-Anweisungen der Anwendung in Apostrophe zu setzen, jedoch nicht in Anführungszeichen.

Option `string_rtruncation` einstellen

Sowohl Adaptive Server Enterprise als auch SQL Anywhere unterstützen die Option "`string_rtruncation`", welche sich auf die Fehlermeldung auswirkt, wenn eine INSERT- oder UPDATE-Zeichenfolge verkürzt ist. Achten Sie darauf, dass für jede einzelne Datenbank derselbe Wert festgelegt wird.

Siehe auch

- „Kompatibilitätsoptionen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „quoted_identifier-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „string_rtruncation-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Groß-/Kleinschreibung

Berücksichtigung von Groß- und Kleinschreibung in Datenbanken bezieht sich auf Folgendes:

- **Daten** Die Berücksichtigung von Groß- und Kleinschreibung der Daten wirkt sich auf Indizes etc. aus.
- **Bezeichner** Bezeichner beinhalten Tabellennamen, Spaltennamen usw.
- **Kennwörter** In SQL Anywhere-Datenbanken wird bei Kennwörtern immer die Groß- und Kleinschreibung berücksichtigt.

Groß- und Kleinschreibung von Daten

Über die Berücksichtigung der Groß- und Kleinschreibung von SQL Anywhere-Daten in Vergleichen entscheiden Sie bei Erstellung der Datenbank. Standardmäßig wird in SQL Anywhere-Datenbanken bei Vergleichen die Groß- und Kleinschreibung nicht berücksichtigt, obwohl Daten stets in der Schreibweise abgelegt werden, in der sie eingegeben wurden.

In Adaptive Server Enterprise hängt die Berücksichtigung der Groß- und Kleinschreibung der Daten von der im Adaptive Server Enterprise-System installierten Sortierreihenfolge ab. Die Berücksichtigung der Groß- und Kleinschreibung kann für Ein-Byte-Zeichensätze durch Umkonfigurieren der Sortierreihenfolge in Adaptive Server Enterprise geändert werden.

Berücksichtigung von Groß- und Kleinschreibung der Bezeichner

SQL Anywhere unterstützt keine Bezeichner, die die Groß- und Kleinschreibung berücksichtigen. In Adaptive Server Enterprise folgt die Berücksichtigung von Groß- und Kleinschreibung der Bezeichner der Berücksichtigung von Groß- und Kleinschreibung bei den Daten. Die Standard-Benutzer-ID für SQL Anywhere-Datenbanken ist "DBA".

In Adaptive Server Enterprise wird bei benutzerdefinierten Datentypen die Groß- und Kleinschreibung berücksichtigt. In SQL Anywhere wird außer bei den Java-Datentypen die Groß- und Kleinschreibung nicht berücksichtigt.

Groß- und Kleinschreibung von Kennwörtern

In SQL Anywhere berücksichtigen Kennwörter immer die Groß- und Kleinschreibung. Das Standardkennwort für die Benutzer-ID DBA lautet **sql** in Kleinbuchstaben.

In Adaptive Server Enterprise folgt die Berücksichtigung von Groß- und Kleinschreibung von Kennwörtern der Berücksichtigung von Groß- und Kleinschreibung des Servers.

Kompatible Objektnamen

Jedes Datenbankobjekt muss innerhalb eines bestimmten **Namespace** einen eindeutigen Namen haben. Außerhalb dieses Namensbereichs sind Duplikatnamen zulässig. Es gibt einige Datenbankobjekte, die in Adaptive Server Enterprise und in SQL Anywhere jeweils unterschiedliche Namensbereiche belegen.

Adaptive Server Enterprise hat einen restriktiveren Namensbereich für Triggernamen als SQL Anywhere. Triggernamen müssen in der Datenbank eindeutig sein. Für kompatible SQL sollten Sie die Einschränkung des Adaptive Server Enterprise berücksichtigen und in einer Datenbank nur eindeutige Triggernamen verwenden.

Spezielle TIMESTAMP-Spalte und TIMESTAMP-Datentyp in Transact-SQL

SQL Anywhere unterstützt die spezielle TIMESTAMP-Spalte in Transact-SQL. Die TIMESTAMP-Spalte wird zusammen mit der TSEQUAL-Systemfunktion verwendet, um zu überprüfen, ob eine Zeile aktualisiert wurde.

Zwei Bedeutungen von timestamp

SQL Anywhere verfügt über einen TIMESTAMP-Datentyp, in dem genaue Datums- und Zeitangaben enthalten sind. Dieser unterscheidet sich von der speziellen Transact-SQL-Spalte TIMESTAMP und dem speziellen Transact-SQL-Datentyp TIMESTAMP.

Erstellen einer Transact-SQL-TIMESTAMP-Spalte in SQL Anywhere

Um eine Transact-SQL-TIMESTAMP-Spalte zu erstellen, müssen Sie zuerst eine Spalte erstellen, die (in SQL Anywhere) den Datentyp TIMESTAMP und die Standardeinstellung "timestamp" hat. Die Spalte kann einen beliebigen Namen haben, obwohl der Name timestamp gebräuchlich ist.

Die folgende CREATE TABLE-Anweisung enthält beispielsweise eine Transact-SQL-TIMESTAMP-Spalte:

```
CREATE TABLE tablename (
    column_1 INTEGER,
    column_2 TIMESTAMP DEFAULT TIMESTAMP
);
```

Die folgende ALTER TABLE-Anweisung fügt der SalesOrders-Tabelle eine Transact-SQL-TIMESTAMP-Spalte hinzu:

```
ALTER TABLE SalesOrders
ADD timestamp TIMESTAMP DEFAULT TIMESTAMP;
```

In Adaptive Server Enterprise wird einer Tabelle mit dem Namen timestamp, für die kein Datentyp angegeben ist, automatisch der Datentyp TIMESTAMP zugeordnet. In SQL Anywhere müssen Sie den Datentyp explizit zuordnen.

Der Datentyp einer TIMESTAMP-Spalte

Adaptive Server Enterprise behandelt eine TIMESTAMP-Spalte als Domäne, die VARBINARY(8) ist und NULL zulässt, während SQL Anywhere eine TIMESTAMP-Spalte als den Datentyp TIMESTAMP behandelt, welcher aus Datum und Uhrzeit besteht, wobei Bruchteile einer Sekunde bis zur sechsten Dezimalstelle gespeichert werden.

Wenn bei späteren Aktualisierungen Daten aus der Tabelle abgerufen werden, muss die Variable, in die der TIMESTAMP-Wert abgerufen wird, der Beschreibung der Spalte entsprechen.

In Interactive SQL müssen Sie möglicherweise die Option "timestamp_format" festlegen, um die Differenz der Zeilenwerte anzuzeigen. Die folgende Anweisung legt die Option "timestamp_format" so fest, dass alle sechs Stellen der Bruchteile einer Sekunde angezeigt werden:

```
SET OPTION timestamp_format='YYYY-MM-DD HH:NN:SS.SSSSSS';
```

Wenn nicht alle sechs Stellen angezeigt werden, kann es sein, dass einige TIMESTAMP-Spaltenwerte gleich zu sein scheinen. Sie sind es aber nicht.

Aktualisierung mit TSEQUAL

Mit der TSEQUAL-Systemfunktion können Sie feststellen, ob eine TIMESTAMP-Spalte aktualisiert wurde.

Eine Anwendung kann mit SELECT eine TIMESTAMP-Spalte in eine Variable einfügen. Wenn mit UPDATE eine Aktualisierung einer der ausgewählten Zeilen vorgenommen wird, kann diese die TSEQUAL-Funktion verwenden, um zu überprüfen, ob die Zeile geändert wurde. Die TSEQUAL-Funktion vergleicht den TIMESTAMP-Wert in der Tabelle mit dem TIMESTAMP-Wert, den die SELECT-Anweisung geliefert hat. Identische Zeitstempel weisen darauf hin, dass keine Änderungen vorgenommen wurden. Wenn die Zeitstempel unterschiedlich sind, wurde die Zeile seit der Ausführung der SELECT-Anweisung geändert. Beispiel:

```
CREATE VARIABLE old_ts_value TIMESTAMP;

SELECT timestamp into old_ts_value
FROM publishers
WHERE pub_id = '0736';

UPDATE publishers
SET city = 'Springfield'
WHERE pub_id = '0736'
AND TSEQUAL(timestamp, old_ts_value);
```

Die spezielle IDENTITY-Spalte

In der IDENTITY-Spalte werden aufeinander folgende Nummern gespeichert, wie zum Beispiel Rechnungsnummern oder Mitarbeiternummern, die automatisch erstellt werden. Der Wert der IDENTITY-Spalte identifiziert jede Zeile in einer Tabelle eindeutig.

In Adaptive Server Enterprise kann jede Tabelle einer Datenbank eine IDENTITY-Spalte besitzen. Der Datentyp muss numerisch sein und null Dezimalstellen haben, und die IDENTITY-Spalte sollte NULL nicht zulassen.

In SQL Anywhere ist die IDENTITY-Spalte eine Spalten-StandardEinstellung. Werte, die nicht Teil der Sequenz sind, können explizit mit der INSERT-Anweisung in die Spalte eingefügt werden. Adaptive Server Enterprise lässt Einfügungen mit INSERT in die Identity-Spalten nicht zu, es sei denn, die Option `identity_insert` ist auf *ON* eingestellt. In SQL Anywhere müssen Sie die NOT NULL-Eigenschaft festlegen und sicherstellen, dass nur eine Spalte eine IDENTITY-Spalte ist. SQL Anywhere lässt die Verwendung eines beliebigen numerischen Datentyps für die IDENTITY-Spalte zu. Für eine bessere Performance wird die Verwendung von integer-Datentypen empfohlen.

In SQL Anywhere ist die IDENTITY-Spalte identisch mit der StandardEinstellung AUTOINCREMENT für eine Spalte.

Um eine IDENTITY-Spalte zu erstellen, verwenden Sie die folgende CREATE TABLE-Syntax, wobei *n* groß genug sein muss, um den Wert der maximalen Anzahl an Zeilen zu halten, die in die Tabelle eingefügt werden kann:

```
CREATE TABLE table-name (
    ...
    column-name numeric(n,0) IDENTITY NOT NULL,
    ...
)
```

Abrufen von IDENTITY-Spaltenwerten mit @@identity

Wenn Sie zum ersten Mal eine Zeile in eine Tabelle einfügen, wird einer IDENTITY-Spalte ein Wert von 1 zugewiesen. Bei jeder darauf folgenden Einfügung wird der Wert der Spalte um 1 erhöht. Der zuletzt in eine Identity-Spalte eingefügte Wert steht in der globalen Variable @@identity zur Verfügung.

Siehe auch

- „Globale Variable @@identity“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Kompatible SQL-Anweisungen

In diesem Abschnitt finden Sie allgemeine Richtlinien zum Schreiben von SQL für die Verwendung in mehr als einem Datenbankverwaltungssystem, und es werden Fragen der Kompatibilität zwischen Adaptive Server Enterprise und SQL Anywhere auf SQL-Anweisungsebene behandelt.

Allgemeine Richtlinien zum Schreiben von portierbarer SQL

Beim Schreiben von SQL, die für mehr als ein Datenbankverwaltungssystem benutzt werden soll, sollten Sie in Ihren SQL-Anweisungen so explizit wie möglich sein. Auch wenn eine gegebene SQL-Anweisung von mehr als einem Server unterstützt wird, sollte man nicht voraussetzen, dass das Standardverhalten für jedes System das gleiche ist, wenn keine Option angegeben wurde.

In SQL Anywhere können der Datenbankserver und der SQL-Präprozessor (sqlpp) SQL-Anweisungen erkennen, die Erweiterungen des Herstellers sind, die nicht kompatibel mit bestimmten ISO/ANSI SQL-Standards sind oder die nicht von UltraLite unterstützt werden. Diese Funktionalität wird der "SQL Flagger" genannt.

Allgemeine Richtlinien für das Schreiben von kompatibler SQL:

- Geben Sie alle verfügbaren Optionen an und verwenden Sie kein Standardverhalten.
- Machen Sie die Reihenfolge der Ausführung innerhalb von Anweisungen durch die Verwendung von Klammern explizit, und setzen Sie keine identische standardmäßige Vorrangreihenfolge für Operatoren voraus.
- Verwenden Sie die Transact-SQL-Konvention des Zeichens @, das Variablenamen vorangestellt wird, für Adaptive Server Enterprise-Portierbarkeit.
- Deklarieren Sie Variable und Cursor in Prozeduren, Triggern und Batches sofort nach einer BEGIN-Anweisung. Dies ist in SQL Anywhere erforderlich, obwohl Adaptive Server Enterprise Deklarationen an einer beliebigen Stelle in einer Prozedur, einem Trigger oder einem Batch zulässt.
- Vermeiden Sie es, reservierte Wörter aus Adaptive Server Enterprise oder SQL Anywhere als Bezeichner in Ihrer Datenbank zu verwenden.
- Nehmen Sie große Namensbereiche an. Vergewissern Sie sich, dass jeder Index über einen eindeutigen Namen verfügt.

Siehe auch

- [„Testen der SQL-Konformität mit dem SQL Flagger“ auf Seite 641](#)

Mit Transact-SQL kompatible Tabellen

SQL Anywhere unterstützt Domänen, mit denen Integritätsregel- und Standarddefinitionen in der Datentypdefinition verpackt werden können. Er unterstützt auch explizite Standardwerte und CHECK-Bedingungen in der CREATE TABLE-Anweisung. Er unterstützt jedoch keine benannten Standardwerte.

NULL

SQL Anywhere und Adaptive Server Enterprise verhalten sich unterschiedlich im Umgang mit NULL. In Adaptive Server Enterprise wird NULL gelegentlich wie ein Wert behandelt.

Zum Beispiel darf ein eindeutiger Index in Adaptive Server Enterprise keine Zeilen aufweisen, die NULL enthalten und ansonsten identisch sind. In SQL Anywhere kann ein eindeutiger Index solche Zeilen enthalten.

Standardmäßig werden Spalten in Adaptive Server Enterprise auf NOT NULL gesetzt, während in SQL Anywhere die Standardeinstellung NULL ist. Sie können diese Einstellung mit der Option "allow_nulls_by_default" kontrollieren. Sie sollten NULL oder NOT NULL explizit angeben, damit Ihre Datendefinitions-Anweisungen übertragbar sind.

Temporäre Tabellen

Sie können eine temporäre Tabelle erstellen, indem Sie ein Rautenzeichen (#) vor den Tabellennamen in einer Anweisung CREATE TABLE stellen. Diese temporären Tabellen sind in SQL Anywhere deklarierte temporäre Tabellen und stehen lediglich während der aktuellen Verbindung zur Verfügung.

Die physische Platzierung einer Tabelle wird in Adaptive Server Enterprise anders ausgeführt als in SQL Anywhere. SQL Anywhere unterstützt die **ONsegment-name**-Klausel, aber *segment-name* bezieht sich auf einen SQL Anywhere-dbspace.

Siehe auch

- „Testen der SQL-Konformität mit dem SQL Flagger“ auf Seite 641
- „Optionen für die Transact-SQL-Kompatibilität“ auf Seite 659
- „DECLARE LOCAL TEMPORARY TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Mit Transact-SQL kompatible Abfragen

Es gibt zwei Kriterien für das Schreiben einer Abfrage, die für Datenbanken sowohl in SQL Anywhere als auch in Adaptive Server Enterprise ausgeführt werden kann:

- Datentypen, Ausdrücke und Suchbedingungen in der Abfrage müssen kompatibel sein.
- Die Syntax der Anweisung selbst muss kompatibel sein.

In diesem Abschnitt wird von kompatiblen Datentypen, Ausdrücken und Suchbedingungen ausgegangen. Die Beispiele setzen voraus, dass die Option "quoted_identifier" auf "Off" gesetzt ist. Das ist die Standardeinstellung in Adaptive Server Enterprise, aber nicht die Standardeinstellung in SQL Anywhere.

Die Implementierung des Transact-SQL-Dialekts von SQL Anywhere unterstützt den Großteil der Abfragensyntax des Watcom SQL-Dialekts, auch wenn einige dieser SQL-Konstruktionen nicht von Adaptive Server Enterprise unterstützt werden. In einer Transact-SQL-Abfrage unterstützt SQL Anywhere die folgenden SQL-Konstruktionen:

- Abschließender Apostroph ` , Anführungszeichen " und eckige Klammern [] zum Angeben von IDs
- UNION-, EXCEPT- und INTERSECT-Abfrageausdrücke
- Abgeleitete Tabellen
- Tabellenfunktionen
- CONTAINS-Tabellenausdrücke für Volltextsuche
- REGEXP-, SIMILAR-, IS DISTINCT FROM- und CONTAINS-Prädikate
- Benutzerdefiniertes SQL oder externe Funktionen
- LEFT-, RIGHT- und FULL-Outer-Joins
- GROUP BY ROLLUP, CUBE und GROUPING SETS
- TOP N START AT M
- Fenster-Aggregatfunktionen und andere analytischen Funktionen, einschließlich statistische Analyse und Funktionen für lineare Regression

Der Transact-SQL-Dialekt von SQL Anywhere unterstützt also Folgendes:

Syntax

query-expression:
{ *query-expression* **EXCEPT** [**ALL**] *query-expression*
| *query-expression* **INTERSECT** [**ALL**] *query-expression*
| *query-expression* **UNION** [**ALL**] *query-expression*
| *query-specification* }
[**ORDER BY** { *expression* | *integer* }
 [**ASC** | **DESC**], ...]
[**FOR READ ONLY** | *for-update-clause*]
[**FOR XML** *xml-mode*]

query-specification:
SELECT [**ALL** | **DISTINCT**] [*cursor-range*] *select-list*
[**INTO** #*temporary-table-name*]
[**FROM** *table-expression*, ...]
[**WHERE** *search-condition*]
[**GROUP BY** *group-by-term*, ...]
[**HAVING** *search-condition*]
[**WINDOW** *window-specification*, ...]

Parameter

select-list:
table-name. *
| *
| *expression*
| *alias-name* = *expression*
| *expression* *as identifier*
| *expression* *as string*

table-expression: See „FROM-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)].

group-by-term: See „GROUP BY-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)].

for-update-clause: See **FOR UPDATE-** oder **FOR READ ONLY-Klausel**, **SELECT-Anweisung** [[SQL Anywhere Server - SQL-Referenzhandbuch](#)].

xml-mode: See „**SELECT-Anweisung**“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)].

alias-name:
identifier | 'string' | "string" | `string`

cursor-range:
{ **FIRST** | **TOP** *constant-or-variable* } [**START AT** *constant-or-variable*]

Transact-SQL-table-reference:
[*owner* .] *table-name* [[**AS**] *correlation-name*]
[(**INDEX** *index_name* [**PREFETCH** *size*] [**LRU** | **MRU**])]

Hinweise

- Zusätzlich zu der Watcom SQL-Syntax für die FROM-Klausel unterstützt SQL Anywhere auch die Transact-SQL-Syntax für bestimmte Adaptive Server Enterprise-Tabellen-Hints. Bei einer Tabellenreferenz unterstützt die *Transact-SQL-table-reference* das Hint-Schlüsselwort **INDEX**, zusammen mit den **PREFETCH**-, **MRU**- und **LRU**-Caching-Hints. **PREFETCH**, **MRU** und **LRU** werden in SQL Anywhere ignoriert.

- SQL Anywhere unterstützt die Transact-SQL-Erweiterung der GROUP BY-Klausel nicht, sodass Bezugnahmen auf Spalten und Ausdrücke, die nicht in der GROUP BY-Klausel enthalten sind, möglich sind.

SQL Anywhere unterstützt auch nicht die Transact-SQL GROUP BY ALL-Konstruktion.

- SQL Anywhere unterstützt eine Teilmenge von Transact-SQL Outer-Join-Konstruktionen unter Verwendung der Vergleichsoperatoren *= und =*.
- Der Transact-SQL-Dialekt von SQL Anywhere unterstützt keine allgemeinen Tabellenausdrücke, außer wenn sie innerhalb einer abgeleiteten Tabelle eingebettet sind. Aus diesem Grund unterstützt der Transact-SQL-Dialekt von SQL Anywhere keine rekursiven UNION-Abfragen. Verwenden Sie den Watcom SQL-Dialekt, wenn Sie diese Funktionalität benötigen.
- Der Performance-Parameter-Teil der Tabellen-Spezifikation wird syntaktisch analysiert, hat jedoch keine Auswirkungen.
- Das Schlüsselwort HOLDLOCK wird von SQL Anywhere unterstützt. Mit HOLDLOCK ist eine gemeinsame Sperre auf einer angegebenen Tabelle oder Ansicht restriktiver, weil die gemeinsame Sperre nicht freigegeben wird, wenn die Datenseite nicht mehr benötigt wird. Die Abfrage wird auf Isolationsstufe 3 für eine Tabelle ausgeführt, für die HOLDLOCK angegeben ist.
- Die Option HOLDLOCK gilt nur für die Tabelle oder Ansicht, für die sie angegeben ist, und nur für die Dauer der Transaktion, die durch die Anweisung, in der sie verwendet wird, definiert ist. Bei einer Isolationsstufe 3 wird ein Sperrenbesitz für jedes SELECT innerhalb einer Transaktion angewendet. Sie können in einer Abfrage nicht die beiden Optionen HOLDLOCK und NOHOLDLOCK gleichzeitig verwenden.
- Das Schlüsselwort NOHOLDLOCK wird von SQL Anywhere erkannt, hat jedoch keine Auswirkungen.
- Transact-SQL verwendet die SELECT-Anweisung, um lokalen Variablen Werte zuzuordnen:

```
SELECT @localvar = 42;
```

Die entsprechende Anweisung in SQL Anywhere ist die SET-Anweisung:

```
SET @localvar = 42;
```

- Adaptive Server Enterprise unterstützt nicht die folgenden Anweisungen:
 - SELECT...INTO *host-variable-list*
 - SELECT... INTO *variable-list*
 - EXCEPT [ALL] oder INTERSECT [ALL]
 - START AT-Klausel
 - Von SQL Anywhere definierte Tabellen-Hints
 - Tabellenfunktionen
 - FULL OUTER JOIN
 - FOR UPDATE BY { LOCK | TIMESTAMP }
 - Fenster-Aggregatfunktionen und lineare Regressionsfunktionen

- Die folgenden Schlüsselwörter und Klauseln der Adaptive Server Enterprise Transact-SQL SELECT-Anweisungssyntax werden von SQL Anywhere nicht unterstützt:
 - SHARED-Schlüsselwort
 - PARTITION-Schlüsselwort
 - COMPUTE-Klausel
 - FOR BROWSE-Klausel
 - GROUP BY ALL-Klausel
 - PLAN-Klausel
 - ISOLATION-Klausel
- SQL Anywhere unterstützt die folgenden Zeichen in Bezeichnern oder Aliasen nicht:
 - Anführungszeichen
 - Steuerzeichen (alle Zeichen unter 0x20)
 - Backslashes
 - Eckige Klammern
 - Invertierte Hochkommata

Siehe auch

- „Testen der SQL-Konformität mit dem SQL Flagger“ auf Seite 641
- „SELECT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „OLAP-Unterstützung“ auf Seite 552
- „GROUP BY und der Standard SQL/2008“ auf Seite 478
- „FROM-Klausel“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Transact-SQL-Outer-Joins (*= oder =*)“ auf Seite 509

Kompatibilität von Joins

In der Implementierung von Transact-SQL von SQL Anywhere können Sie die Join-Syntax des Standards SQL/2008 mit den Schlüsselwörtern JOIN, LEFT OUTER JOIN und RIGHT OUTER JOIN sowie FULL OUTER JOIN angeben, zusammen mit der alten Transact-SQL Outer-Join-Syntax, die die speziellen Vergleichsoperatoren *= und =* in der WHERE-Klausel der Anweisung verwendet.

Hinweis

Die Unterstützung für die Transact-SQL-Outer-Join-Operatoren *= und =* wird nicht weiterentwickelt und in einer zukünftigen Version entfernt werden.

Siehe auch

- „Joins: Daten aus mehreren Tabellen abrufen“ auf Seite 492
- „FROM-Klausel“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Transact-SQL-Outer-Joins (*= oder =*)“ auf Seite 509
- „Testen der SQL-Konformität mit dem SQL Flagger“ auf Seite 641

Transact-SQL-Prozedursprache

Die **Sprache für gespeicherte Prozeduren** ist der Teil der SQL, der in gespeicherten Prozeduren, Triggern und Batches verwendet wird.

SQL Anywhere unterstützt einen Großteil der Transact-SQL-Sprache für gespeicherte Prozeduren zusätzlich zum Watcom SQL-Dialekt, der auf SQL/2008 basiert.

Gespeicherte Transact-SQL-Prozeduren

Der native SQL Anywhere-Dialekt, Watcom SQL, basiert auf dem ISO/ANSI SQL/2008-Standard. Aus diesem Grund unterscheidet sich der Watcom SQL-Dialekt für gespeicherte Prozeduren in vielerlei Hinsicht vom Transact-SQL-Dialekt. Viele der Konzepte und Funktionen sind ähnlich, die Syntax ist jedoch anders. Die SQL Anywhere-Unterstützung für Transact-SQL nutzt ähnliche Konzepte, indem sie die automatische Konvertierung zwischen den Dialekten bereitstellt. Eine Prozedur muss jedoch ausschließlich in einem der beiden Dialekte geschrieben worden sein, und nicht in einer Mischung der Dialekte.

SQL Anywhere-Unterstützung für gespeicherte Prozeduren in Transact-SQL

Die SQL Anywhere-Unterstützung für gespeicherte Prozeduren in Transact-SQL umfasst mehrere Aspekte:

- Parameter übergeben
- Ergebnismengen liefern
- Statusangaben zurückgeben
- Standardwert für Parameter bereitstellen
- Steueranweisungen
- Fehlerbehandlung
- Benutzerdefinierte Funktionen

Transact-SQL Trigger

Trigger-Kompatibilität erfordert die Kompatibilität der Triggerfunktionen und der Triggersyntax. Dieser Abschnitt bietet einen Überblick über die Funktionskompatibilität von Transact-SQL- und SQL Anywhere-Trigger.

Adaptive Server Enterprise unterstützt AFTER-Trigger auf Anweisungsebene, also Trigger, die ausgelöst werden, nachdem die Trigger-auslösende Anweisung abgeschlossen ist. Der von SQL Anywhere unterstützte Watcom SQL-Dialekt unterstützt BEFORE-, AFTER- und INSTEAD OF-Trigger auf Zeilenebene und AFTER- und INSTEAD OF-Trigger auf Anweisungsebene.

Trigger auf Zeilenebene sind nicht Teil der Transact-SQL-Kompatibilitätsfunktionen.

Beschreibung von nicht unterstützten oder anderen Transact-SQL-Triggern

In der folgenden Liste werden einige Funktionen von Transact-SQL-Triggern beschrieben, die entweder nicht unterstützt werden oder in SQL Anywhere anders sind:

- **Trigger, die andere Trigger auslösen** Angenommen, ein Trigger führt eine Aktion aus, die, wenn sie direkt von einem Benutzer ausgeführt wird, einen weiteren Trigger auslösen würde. SQL Anywhere und Adaptive Server Enterprise verhalten sich in diesem Fall etwas unterschiedlich. Das Standardverhalten von Adaptive Server Enterprise besteht darin, dass Trigger weitere Trigger bis zu einer konfigurierbaren Verschachtelungsebene auslösen, welche den Standardwert 16 hat. Sie können die Verschachtelungsebene mit der Option "nested triggers" in Adaptive Server Enterprise steuern. In SQL Anywhere lösen Trigger unbegrenzt weitere Trigger aus, bis der Speicher aufgebraucht ist.
- **Trigger, die sich selbst auslösen** Angenommen, ein Trigger führt eine Aktion aus, die, wenn sie direkt von einem Benutzer ausgeführt wird, denselben Trigger auslösen würde. SQL Anywhere und Adaptive Server Enterprise verhalten sich in diesem Fall etwas unterschiedlich. In SQL Anywhere lösen sich Nicht-Transact-SQL-Trigger standardmäßig selbst rekursiv aus, während sich Trigger im Transact-SQL-Dialekt nicht selbst rekursiv auslösen. Für Trigger im Transact-SQL-Dialekt können Sie jedoch die Option "self_recursion" der Anweisung SET [T-SQL] verwenden, um einem Trigger eine rekursive Selbstauslösung zu erlauben.

Das Standardverhalten von Adaptive Server Enterprise besteht darin, dass ein Trigger sich nicht selbst rekursiv aufruft. Sie können jedoch die Option "self_recursion" verwenden, um die Rekursion zu ermöglichen.

- **ROLLBACK-Anweisung in Triggern nicht unterstützt** Innerhalb eines Triggers lässt Adaptive Server Enterprise die ROLLBACK TRANSACTION-Anweisung zu, welche die gesamte Transaktion, deren Bestandteil der Trigger ist, wieder zurücksetzt. SQL Anywhere lässt ROLLBACK- (oder ROLLBACK TRANSACTION-) Anweisungen in Triggern nicht zu, weil ein Trigger-Vorgang und sein Trigger zusammen eine atomare Anweisung bilden.

SQL Anywhere bietet die Adaptive Server Enterprise-kompatible ROLLBACK TRIGGER-Anweisung zum Zurücksetzen von Vorgängen, die innerhalb von Triggern ausgeführt wurden.

- **ORDER-Klausel nicht unterstützt** Transact-SQL-Trigger lassen keine ORDER *nn*-Klausel zu. Der Wert von trigger_order wird automatisch auf 1 gesetzt. Dies kann beim Erstellen eines T-SQL-Triggers dazu führen, dass ein Fehler zurückgegeben wird, wenn bereits ein Trigger auf Anweisungsebene vorhanden ist. Dies ist deshalb der Fall, weil die SYSTRIGGER-Systemtabelle einen eindeutigen Index für table_id, event, trigger_time und trigger_order hat. Für ein bestimmtes Ereignis (insert, update, delete) werden Trigger auf Anweisungsebene immer mit AFTER verwendet und trigger_order kann nicht festgelegt werden, sodass nur ein Trigger pro Tabelle vorhanden sein kann. Dabei wird davon ausgegangen, dass die anderen Trigger keine andere Reihenfolge als 1 festlegen.

Siehe auch

- „Trigger“ auf Seite 98
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ auf Seite 81
- „SET-Anweisung [T-SQL]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ROLLBACK TRIGGER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Transact-SQL Batches

In Transact-SQL ist ein Batch ein Satz von SQL-Anweisungen, die gemeinsam abgesetzt und als eine Gruppe, eine nach der anderen, ausgeführt werden. Batches können in SQL-Skriptdateien gespeichert werden. Interactive SQL kann verwendet werden, um Batches interaktiv auszuführen.

Die in Prozeduren verwendeten Steueranweisungen können ebenfalls in Batches benutzt werden. SQL Anywhere unterstützt die Verwendung von Steueranweisungen in Batches und die Transact-SQL-ähnliche Verwendung von nicht getrennten Anweisungsgruppen, die mit einer GO-Anweisung enden und so das Ende des Batches anzeigen.

Bei Batches, die in SQL-Skriptdateien gespeichert sind, unterstützt SQL die Verwendung von Parametern in diesen Dateien.

Siehe auch

- „PARAMETERS-Anweisung [Interactive SQL]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Automatische Konvertierung von gespeicherten Prozeduren

Zusätzlich zur Unterstützung der Transact-SQL-Alternativsyntax bietet SQL Anywhere Hilfsmittel für die Konvertierung von Anweisungen zwischen den Dialekten Watcom SQL und Transact-SQL. Integrierte SQL-Funktionen geben Informationen über SQL-Anweisungen zurück und ermöglichen die automatische Konvertierung von SQL-Anweisungen:

- **SQLDIALECT(Anweisung)** Liefert Watcom-SQL oder Transact-SQL
- **WATCOMSQL(Anweisung)** Gibt die Watcom-SQL-Syntax für die Anweisung zurück.
- **TRANSACTSQL(Anweisung)** Gibt die Transact-SQL-Syntax für die Anweisung zurück.

Hierbei handelt es sich um Funktionen, auf die daher auch mit einer Select-Anweisung aus Interactive SQL zugegriffen werden kann. Beispielsweise liefert die folgende Anweisung den Wert "Watcom-SQL":

```
SELECT SQLDIALECT( 'SELECT * FROM Employees' );
```

Übersetzen einer gespeicherten Prozedur

Übersetzen Sie gespeicherte Prozeduren zwischen SQL-Dialekten, beispielsweise zwischen Watcom-SQL und Transact-SQL.

Voraussetzungen

Sie müssen Eigentümer der Prozedur sein oder eines der folgenden Privilegien haben:

- ALTER ANY PROCEDURE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie auf den Ordner **Prozeduren und Funktionen** und wählen Sie in der Liste eine der gespeicherten Prozeduren aus.
3. Klicken Sie im rechten Fensterausschnitt auf die Registerkarte **SQL** und klicken Sie dann das Textfenster.
4. Klicken Sie auf **Datei** und anschließend auf eine der Optionen für **Übersetzen in**.

Die Prozedur erscheint im rechten Fensterausschnitt im gewählten Dialekt. Wenn der ausgewählte Dialekt nicht derjenige ist, in dem die Prozedur gespeichert wurde, wird sie vom Datenbankserver in diesen Dialekt konvertiert. Nicht konvertierte Zeilen erscheinen als Kommentare.

5. Schreiben Sie alle nicht übersetzten Zeilen neu.
6. Klicken Sie auf **Datei** » **Speichern**.

Ergebnisse

Die gespeicherte Prozedur wird übersetzt und in der Datenbank gespeichert.

Siehe auch

- „Automatische Konvertierung von gespeicherten Prozeduren“ auf Seite 671
- „SQLDIALECT-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „TRANSACTSQL-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „WATCOMSQL-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Aus Transact-SQL-Prozeduren zurückgegebene Ergebnismengen

SQL Anywhere 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.

Beispiel für eine Transact-SQL-Prozedur

Die folgende Transact-SQL-Prozedur veranschaulicht, wie Ergebnismengen von gespeicherten Prozeduren in Transact-SQL zurückgegeben werden:

```
CREATE PROCEDURE ShowDepartment (@deptname VARCHAR(30))
AS
    SELECT Employees.Surname, Employees.GivenName
    FROM Departments, Employees
    WHERE Departments.DepartmentName = @deptname
    AND Departments.DepartmentID = Employees.DepartmentID;
```

Beispiel für eine Watcom SQL-Prozedur

Nachfolgend wird die entsprechende SQL Anywhere-Prozedur aufgeführt:

```
CREATE PROCEDURE ShowDepartment(in deptname VARCHAR(30))
RESULT ( LastName CHAR(20), FirstName CHAR(20))
BEGIN
    SELECT Employees.Surname, Employees.GivenName
    FROM Departments, Employees
    WHERE Departments.DepartmentName = deptname
    AND Departments.DepartmentID = Employees.DepartmentID
END;
```

Siehe auch

- [„Ergebnismengen“ auf Seite 121](#)

Variable in Transact-SQL-Prozeduren

SQL Anywhere verwendet die SET-Anweisung, um Variablen in einer Prozedur Werte zuzuordnen. In Transact-SQL werden Werte entweder unter Verwendung der SELECT-Anweisung mit einer leeren Tabellenliste, oder mit der SET-Anweisung zugeordnet. Die folgende einfache Prozedur veranschaulicht die Funktionsweise der Transact-SQL-Syntax:

```
CREATE PROCEDURE multiply
    @mult1 int,
    @mult2 int,
    @result int output
AS
SELECT @result = @mult1 * @mult2;
```

Die Prozedur kann wie folgt aufgerufen werden:

```
CREATE VARIABLE @product int
go
EXECUTE multiply 5, 6, @product OUTPUT
go
```

Die Variable @product hat einen Wert von 30, nachdem die Prozedur ausgeführt wurde.

Siehe auch

- [„Mit Transact-SQL kompatible Abfragen“ auf Seite 665](#)
- [„SET-Anweisung“ \[*SQL Anywhere Server - SQL-Referenzhandbuch*\]](#)

Fehlerbehandlung in Transact-SQL-Prozeduren

Die standardmäßige Fehlerbehandlung bei Prozeduren ist in den Dialekten Watcom SQL und Transact-SQL verschieden. Standardmäßig werden Prozeduren im Watcom SQL-Dialekt beendet, wenn ein Fehler auftritt, und an die aufrufende Umgebung werden SQLSTATE- und SQLCODE-Werte zurückgegeben.

Explizite Fehlerbehandlung kann mit der EXCEPTION-Anweisung in gespeicherte Watcom SQL-Prozeduren integriert werden, oder die Prozedur kann mit der EXCEPTION RESUME-Anweisung

angewiesen werden, die Ausführung nach einem Fehler bei der nächsten Anweisung wieder aufzunehmen.

Wenn in einer Prozedur im Transact-SQL-Dialekt ein Fehler auftritt, dann wird die Ausführung bei der nächsten Anweisung fortgesetzt. Die globale Variable @@error enthält den Fehlerstatus der zuletzt ausgeführten Anweisung. Nachdem die Anweisung abgearbeitet worden ist, können Sie diese Variable überprüfen, um eine Rückgabe aus der Prozedur zu erzwingen. Beispielsweise verursacht die folgende Anweisung eine Beendigung, wenn ein Fehler auftritt.

```
IF @@error != 0 RETURN
```

Wenn die Prozedur die Ausführung abschließt, gibt ein Rückgabewert den Erfolg oder den Misserfolg der Prozedur an. Dieser Rückgabestatus ist eine Ganzzahl, auf die wie folgt zugegriffen werden kann:

```
DECLARE @Status INT
EXECUTE @Status = proc_sample
IF @Status = 0
    PRINT 'procedure succeeded'
ELSE
    PRINT 'procedure failed'
```

In der folgenden Tabelle werden die integrierten Prozedur-Rückgabewerte und ihre Bedeutungen beschrieben:

Wert	Definition	SQL Anywhere SQLSTATE
0	Prozeduren, die ohne Fehler ausgeführt werden	
-1	Fehlendes Objekt	42W33, 52W02, 52003, 52W07, 42W05
-2	Datentypfehler	53018
-3	Prozess wurde als Dead-lock-Opfer ausgewählt	40001, 40W06
-4	Berechtigungsfehler	42501
-5	Syntaxfehler	42W04
-6	Sonstige Benutzerfehler	
-7	Ressourcenfehler, wie z.B. Speichermangel	08W26
-10	Schwerwiegende interne Inkonsistenz	40W01
-11	Schwerwiegende interne Inkonsistenz	40000

Wert	Definition	SQL Anywhere SQLSTATE
-13	Datenbank ist beschädigt	WI004
-14	Hardwarefehler	08W17, 40W03, 40W04

Wenn ein SQL Anywhere SQLSTATE nicht anwendbar ist, wird der Standardwert -6 zurückgegeben.

Die RETURN-Anweisung kann verwendet werden, um andere Ganzzahlen als diese mit ihren eigenen benutzerdefinierten Bedeutungen zurückzugeben.

Prozeduren mit der RAISERROR-Anweisung

Mit der RAISERROR-Anweisung können Sie benutzerdefinierte Fehler generieren. Die RAISERROR-Anweisung funktioniert so ähnlich wie die SIGNAL-Anweisung.

Die RAISERROR-Anweisung an sich verursacht nicht die Beendigung der Prozedur, sie kann jedoch mit einer RETURN-Anweisung oder einem Test der globalen Variable @@error gekoppelt werden, um die Ausführung nach einem benutzerdefinierten Fehler zu steuern.

Wenn Sie die Datenbankoption "on_tsql_error" auf "Continue" setzen, meldet die Anweisung RAISERROR keinen Fehler mehr, der die Ausführung beendet. Stattdessen läuft die Prozedur zu Ende, speichert den RAISERROR-Statuscode sowie die Meldung und gibt den letzten RAISERROR zurück. Wenn die Prozedur, die den RAISERROR verursacht hat, aus einer anderen Prozedur aufgerufen wurde, wird der RAISERROR erst zurückgegeben, wenn die äußerste aufgerufene Prozedur beendet ist. Wenn Sie die Option on_tsql_error auf den Standardwert (Bedingt) setzen, steuert die Option continue_after_raiserror das Verhalten, das der Ausführung einer RAISERROR-Anweisung folgt. Wenn Sie die on_tsql_error-Option auf "Stop" oder "Continue" setzen, hat die on_tsql_error-Einstellung Vorrang vor der continue_after_raiserror-Einstellung.

Dazwischenliegende RAISERROR-Statusangaben und -Codes gehen verloren, nachdem die Prozedur abschließt. Falls zur Rückgabezeit ein Fehler zusammen mit RAISERROR auftritt, werden die Informationen für den neuen Fehler zurückgegeben, und die RAISERROR-Informationen sind verloren. Die Anwendung kann dazwischenliegende RAISERROR-Statusangaben abfragen, indem sie die globale Variable @@error zu verschiedenen Ausführungspunkten überprüft.

Siehe auch

- „RAISERROR-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Transact-SQL-ähnliche Fehlerbehandlung im Watcom SQL-Dialekt

Sie können eine Prozedur im Watcom SQL-Dialekt dazu veranlassen, dass sie Fehler in einer Transact-SQL-ähnlichen Weise behandelt, indem Sie die ON EXCEPTION RESUME-Klausel für die CREATE PROCEDURE-Anweisung bereitstellen:

```
CREATE PROCEDURE sample_proc()  
ON EXCEPTION RESUME  
BEGIN  
    ...  
END
```

Das Vorhandensein einer ON EXCEPTION RESUME-Klausel verhindert, dass expliziter Code für Verarbeitungsroutinen bei Ausnahmefehlern ausgeführt wird. Vermeiden Sie daher diese Klausel bei der expliziten Fehlerbehandlung.

Siehe auch

- „CREATE PROCEDURE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

XML in der Datenbank

Die "Extensible Markup Language" (XML, erweiterbare Markup-Sprache) stellt strukturierte Daten im Textformat dar. XML wurde gezielt entwickelt, um den Anforderungen von Electronic Publishing in großem Umfang gerecht zu werden.

XML ist eine einfache Markup-Sprache wie HTML, aber auch flexibel wie SGML. XML ist hierarchisch, und ihr Hauptzweck ist es, die Struktur von Daten auf eine Weise zu beschreiben, die sowohl für den Benutzer als auch für die Software lesbar und bearbeitbar ist.

Anstatt einen statischen Satz von Elementen zu liefern, die die verschiedenen Datenformate beschreiben, ermöglicht es Ihnen XML, die Elemente zu definieren. Daher können viele Typen von strukturierten Daten mit XML beschrieben werden. XML-Dokumente können optional eine Dokumenttypendefinition (DTD) oder ein XML-Schema verwenden, um die Struktur, die Elemente und die Attribute zu definieren, die in einer XML-Datei verwendet werden.

Es gibt verschiedene Möglichkeiten, XML mit SQL Anywhere zu verwenden:

- XML-Dokumente in der Datenbank speichern
- Relationale Daten als XML exportieren
- XML in die Datenbank importieren
- Relationale Daten als XML abfragen

Weitere Hinweise zu XML finden Sie unter <http://www.w3.org/XML/>.

Speichern von XML-Dokumenten in relationalen Datenbanken

SQL Anywhere unterstützt zwei Datentypen, die zum Speichern von XML-Dokumenten in Ihrer Datenbank verwendet werden können: Den XML-Datentyp und den LONG VARCHAR-Datentyp. Beide Datentypen speichern das XML-Dokument als Zeichenfolge in der Datenbank.

Der XML-Datentyp verwendet die Zeichensatzkodierung des Datenbankservers. Das XML-Kodierungsattribut sollte der Kodierung entsprechen, die vom Datenbankserver verwendet wird. Das XML-Kodierungsattribut gibt nicht an, wie die automatische Zeichensatzkonvertierung abgeschlossen wird.

Sie können zwischen dem XML-Datentyp und jedem anderen Datentyp, der in oder aus einer Zeichenfolge umgewandelt werden kann, eine Umwandlung durchführen. Die Zeichenfolge wird nicht auf fehlerfreie Formulierung überprüft, wenn sie zu XML umgewandelt wird.

Wenn Sie Elemente aus relationalen Daten generieren, werden alle in XML ungültigen Zeichen durch ein Escapezeichen ausgeglichen, außer die Daten sind vom Typ XML. Angenommen, Sie möchten ein `<product>`-Element generieren, das die Zeichen für kleiner als und größer als enthält:

```
<hat>bowler</hat>
```

Wenn Sie eine Abfrage schreiben, in der festgelegt wird, dass der Elementinhalt vom Typ XML ist, werden die Zeichen für kleiner als und größer als nicht in Apostrophe gesetzt:

```
SELECT XMLFOREST( CAST( '<hat>bowler</hat>' AS XML ) AS product );
```

Sie erhalten folgendes Ergebnis:

```
<product><hat>bowler</hat></product>
```

Die folgende Abfrage legt beispielsweise nicht fest, dass der Elementinhalt vom Typ XML ist:

```
SELECT XMLFOREST( '<hat>bowler</hat>' AS product );
```

In diesem Fall werden die Größer- und Kleiner-Zeichen durch Entitätsreferenzen ersetzt, und zwar folgendermaßen:

```
<product>&lt;hat&gt;bowler&lt;/hat&gt;</product>
```

Attribute werden immer in Anführungszeichen gesetzt, unabhängig vom Datentyp.

Siehe auch

- „Regeln für die Codierung unzulässiger XML-Namen“ auf Seite 689
- „XML-Datentyp“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Als XML exportierte relationale Daten

SQL Anywhere bietet zwei Möglichkeiten, relationale Daten als XML zu exportieren: die Interactive SQL OUTPUT-Anweisung und das ADO.NET DataSet-Objekt.

Mit der FOR XML-Klausel und den SQL/XML-Funktionen können Sie eine Ergebnismenge als XML aus den relationalen Daten in Ihrer Datenbank generieren. Sie können dann das generierte XML in eine Datei exportieren, indem Sie die UNLOAD-Anweisung oder die Systemprozedur "xp_write_file" verwenden.

Aus Interactive SQL exportierte relationale Daten

Die Interactive SQL OUTPUT-Anweisung unterstützt ein XML-Format, das Abfrageergebnisse in eine generierte XML-Datei ausgibt.

Diese erstellte XML-Datei wird in UTF-8 kodiert und enthält eine eingebettete DTD. In der XML-Datei werden Binärdatenwerte in Zeichendaten-(CDATA-)Blocks kodiert, wobei die Binärdaten als 2-hex-digit-Zeichenfolgen dargestellt werden.

Die INPUT-Anweisung akzeptiert XML nicht als Dateiformat. Sie können jedoch XML importieren, indem Sie den Operator OPENXML oder das ADO.NET DataSet-Objekt verwenden.

Siehe auch

- „OUTPUT-Anweisung [Interactive SQL]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Methoden zum Importieren von XML-Dokumenten als relationale Daten“ auf Seite 679

Als XML mit dem DataSet-Objekt exportierte Daten

Mit dem ADO.NET DataSet-Objekt können Sie den Inhalt des DataSet-Objekts in einem XML-Dokument speichern. Sobald Sie das DataSet-Objekt angefüllt haben (zum Beispiel mit den Ergebnissen einer Abfrage in Ihrer Datenbank), können Sie entweder nur das Schema oder sowohl das Schema als auch die Daten aus dem DataSet-Objekt in einer XML-Datei speichern. Die WriteXml-Methode speichert das Schema und die Daten in einer XML-Datei, während die WriteXmlSchema-Methode nur das Schema in einer XML-Datei speichert. Sie können ein DataSet-Objekt füllen, indem Sie den SQL Anywhere .NET-Datenprovider verwenden.

Weitere Hinweise über das Exportieren von relationalen Daten als XML mit einem DataSet-Objekt finden Sie unter „[SACommand: Zeilen mit ExecuteNonQuery einfügen, löschen und aktualisieren](#)“ [[SQL Anywhere Server - Programmierung](#)].

Methoden zum Importieren von XML-Dokumenten als relationale Daten

SQL Anywhere unterstützt zwei verschiedene Methoden zum Importieren von XML in Ihre Datenbank:

- Die Prozedur OPENXML verwenden, um eine Ergebnismenge aus einem XML-Dokument zu generieren
- Das ADO.NET DataSet-Objekt verwenden, um die Daten bzw. das Schema von einem XML-Dokument in ein DataSet-Objekt einzulesen

Importieren von XML mit dem OPENXML-Operator

Der OPENXML-Operator wird in der FROM-Klausel einer Abfrage benutzt, um eine Ergebnismenge aus einem XML-Dokument zu generieren. OPENXML verwendet eine Teilmenge der XPath-Abfragesprache, um Knoten in einem XML-Dokument auszuwählen.

XPath-Ausdrücke verwenden

Wenn Sie OPENXML verwenden, wird das XML-Dokument syntaktisch analysiert und das Ergebnis wird in einer Baumstruktur dargestellt. Der Baum besteht aus Knoten. XPath-Ausdrücke werden verwendet, um Knoten im Baum auszuwählen. Die folgende Liste beschreibt einige häufig benutzte XPath-Ausdrücke:

- **/** Gibt den Stammknoten des XML-Dokuments an
- **//** Gibt alle dem Knoten untergeordneten Knoten an, einschließlich des Stammknotens
- **.** (**einzelner Punkt**) Gibt den aktuellen Knoten des XML-Dokuments an
- **./** Gibt alle dem aktuellen Knoten untergeordnete Knoten an, einschließlich des aktuellen Knotens
- **..** Gibt den dem aktuellen Knoten übergeordneten Knoten an

- ***.@attributename*** Gibt an, dass das Attribut des aktuellen Knotens den Namen *attributename* hat
- ***.childname*** Gibt die Child-Objekte des aktuellen Knotens an, die Elemente mit dem Namen *childname* sind

Nehmen wir das folgende XML-Dokument:

```
<inventory>
  <product ID="301" size="Medium">Tee Shirt
    <quantity>54</quantity>
  </product>
  <product ID="302" size="One Size fits all">Tee Shirt
    <quantity>75</quantity>
  </product>
  <product ID="400" size="One Size fits all">Baseball Cap
    <quantity>112</quantity>
  </product>
</inventory>
```

Das Element `<inventory>` ist der Wurzelknoten. Sie können es referenzieren, indem Sie den folgenden XPath-Ausdruck verwenden:

```
/inventory
```

Nehmen wir an, der aktuelle Knoten ist ein `<quantity>`-Element. Sie können sich auf diesen Knoten beziehen, indem Sie den folgenden XPath-Ausdruck verwenden:

```
.
```

Um alle `<product>`-Elemente zu finden, die Child-Objekte des `<inventory>`-Elements sind, verwenden Sie den folgenden XPath-Ausdruck:

```
/inventory/product
```

Wenn der aktuelle Knoten ein `<product>`-Element ist und Sie sich auf das Größenattribut beziehen wollen, verwenden Sie den folgenden XPath-Ausdruck:

```
./@size
```

Informationen über die Abfragesprache XPath finden Sie unter <http://www.w3.org/TR/xpath>.

Ergebnismengen mit OPENXML generieren

Jede Übereinstimmung des ersten *xpath-query*-Arguments mit OPENXML generiert eine Zeile in der Ergebnismenge. Die WITH-Klausel gibt das Schema der Ergebnismenge an und wie der Wert für die einzelnen Spalten in der Ergebnismenge gefunden wird. Nehmen wir als Beispiel folgende Abfrage:

```
SELECT * FROM OPENXML(
  '<inventory>
    <product>Tee Shirt
      <quantity>54</quantity>
      <color>Orange</color>
    </product>
    <product>Baseball Cap
      <quantity>112</quantity>
      <color>Black</color>
    </product>
  </inventory>',
```

```

'/inventory/product' )
WITH ( Name CHAR (25) './text()',
      Quantity CHAR(3) 'quantity',
      Color CHAR(20) 'color');

```

Das erste *xpath-query*-Argument ist */inventory/product* und es gibt zwei *<product>*-Elemente in XML, daher generiert diese Abfrage zwei Zeilen.

Die WITH -Klausel legt fest, dass es drei Spalten gibt: Name, Quantity und Color. Die Werte für diese Spalten stammen von den *<product>*, *<quantity>*- und *<color>*-Elementen. Die oben stehende Abfrage generiert das folgende Ergebnis:

Name	Quantity	Color
Tee Shirt	54	Orange
Baseball Cap	112	Black

Mit OPENXML eine Edge-Tabelle generieren

Sie können mit dem OPENXML-Operator eine Edge-Tabelle generieren, eine Tabelle, die eine Zeile für jedes Element im XML-Dokument enthält. Sie können eine Edge-Tabelle generieren, wenn Sie die Daten unter Verwendung von SQL abfragen wollen.

Die folgenden SQL-Anweisungen erstellen eine Tabelle mit nur einem XML-Dokument. Das von der Abfrage generierte XML enthält ein Wurzelement namens *<root>*, das mit der *XMLELEMENT*-Funktion generiert wird. Außerdem werden unter Verwendung von *FOR XML AUTO* mit dem *ELEMENTS*-Modifizierer Elemente für jede Spalte in den Tabellen "Employees", "SalesOrders" und "Customers" generiert.

```

CREATE TABLE IF NOT EXISTS xmldata (xmldoc XML);
INSERT INTO xmldata WITH AUTO NAME
  SELECT XMLELEMENT( NAME root,
    (SELECT EmployeeID, Employees.GivenName, Employees.Surname,
      Customers.ID, Customers.GivenName, Customers.Surname,
      Customers.Phone, CompanyName,
      SalesOrders.ID, OrderDate, Region
    FROM Employees
    KEY JOIN SalesOrders
    KEY JOIN Customers
    ORDER BY EmployeeID, Customers.ID, SalesOrders.ID
    FOR XML AUTO, ELEMENTS)) AS xmldoc;
SELECT xmldoc FROM xmldata;

```

Der generierte XML-Code sieht folgendermaßen aus (das Ergebnis wurde formatiert, damit es leichter lesbar ist—das von der Abfrage zurückgegebene Ergebnis ist eine kontinuierliche Zeichenfolge):

```

<root>
  <Employees>
    <EmployeeID>129</EmployeeID>
    <GivenName>Philip</GivenName>
    <Surname>Chin</Surname>

    <Customers>
      <ID>101</ID>
      <GivenName>Michaels</GivenName>

```

```
<Surname>Devlin</Surname>
<Phone>2015558966</Phone>
<CompanyName>The Power Group</CompanyName>
  <SalesOrders>
    <ID>2560</ID>
    <OrderDate>2001-03-16</OrderDate>
    <Region>Eastern</Region>
  </SalesOrders>
</Customers>

<Customers>
  <ID>103</ID>
  <GivenName>Erin</GivenName>
  <Surname>Niedringhaus</Surname>
  <Phone>2155556513</Phone>
  <CompanyName>Darling Associates</CompanyName>
    <SalesOrders>
      <ID>2451</ID>
      <OrderDate>2000-12-15</OrderDate>
      <Region>Eastern</Region>
    </SalesOrders>
</Customers>

<Customers>
  <ID>104</ID>
  <GivenName>Meghan</GivenName>
  <Surname>Mason</Surname>
  <Phone>6155555463</Phone>
  <CompanyName>P.S.C.</CompanyName>
    <SalesOrders>
      <ID>2331</ID>
      <OrderDate>2000-09-17</OrderDate>
      <Region>South</Region>
    </SalesOrders>

    <SalesOrders>
      <ID>2342</ID>
      <OrderDate>2000-09-28</OrderDate>
      <Region>South</Region>
    </SalesOrders>
  </Customers>
  ...
</Employees>
  ...
<Employees>
  ...
</Employees>
</root>
```

Die folgende Abfrage verwendet den XPath-Ausdruck "descendant-or-self" (//*), um jedes Element im oben stehenden XML-Dokument zu vergleichen, und bei jedem Element wird die ID-Meta-Eigenschaft verwendet, um eine ID für den Knoten zu finden, wobei der übergeordnete XPath-Ausdruck (../) mit der ID-Meta-Eigenschaft verwendet wird, um den übergeordneten Knoten zu erhalten. Die Meta-Eigenschaft "localname" wird verwendet, um den Namen der einzelnen Elemente zu erhalten. Bei Meta-Eigenschaftsnamen wird die Groß- und Kleinschreibung berücksichtigt. Daher können ID oder LOCALNAME nicht als Meta-Eigenschaftsnamen benutzt werden.

```
CREATE OR REPLACE VARIABLE x XML;
SELECT xmldoc INTO x FROM xmldata;

SELECT *
```

```

FROM OPENXML( x, '//*' )
WITH (ID INT '@mp:id',
      parent INT '../@mp:id',
      name CHAR(25) '@mp:localname',
      text LONG VARCHAR 'text()' )
ORDER BY ID;

```

Die durch diese Abfrage generierte Ergebnismenge zeigt die ID jedes Knotens, die ID des übergeordneten Knotens sowie den Namen und Inhalt für jedes Element im XML-Dokument.

ID	parent	name	text
5	(NULL)	root	(NULL)
16	5	Employees	(NULL)
28	16	EmployeeID	129
55	16	GivenName	Phillip
82	16	Surname	Chin
...

Verwenden von OPENXML mit xp_read_file

Bisher wurde XML benutzt, das mit einer Prozedur wie XMLELEMENT generiert wurde. Sie können XML auch aus einer Datei lesen und mit der Prozedur "xp_read_file" syntaktisch analysieren.

Angenommen, die Datei `c:\temp\inventory.xml` wurde mithilfe der unten stehenden Abfrage geschrieben.

```

SELECT xp_write_file( 'c:\\temp\\inventory.xml',
  '<inventory>
    <product>Tee Shirt
      <quantity>54</quantity>
      <color>Orange</color>
    </product>
    <product>Baseball Cap
      <quantity>112</quantity>
      <color>Black</color>
    </product>
  </inventory>'
);

```

Sie können die folgende Anweisung benutzen, um XML in der Datei zu lesen und syntaktisch zu analysieren:

```

SELECT *
FROM OPENXML( xp_read_file( 'c:\\temp\\inventory.xml' ),
  '//*' )
WITH (ID INT '@mp:id',
      parent INT '../@mp:id',
      name CHAR(128) '@mp:localname',
      text LONG VARCHAR 'text()' )
ORDER BY ID;

```

XML in einer Spalte abfragen

Wenn Sie eine Tabelle mit einer Spalte haben, die XML enthält, können Sie OPENXML verwenden, um alle XML-Werte in der Spalte auf einmal abzufragen. Das wird erreicht, indem Sie eine lateral abgeleitete Tabelle verwenden.

Die folgenden Anweisungen erstellen eine Tabelle mit zwei Spalten, "ManagerID" und "Reports". Die Spalte "Reports" enthält XML-Daten, die von der Tabelle "Employees" generiert wurden.

```
CREATE TABLE IF NOT EXISTS xmltest (ManagerID INT, Reports XML);
INSERT INTO xmltest
  SELECT ManagerID, XMLELEMENT( NAME reports,
    XMLAGG( XMLELEMENT( NAME e, EmployeeID)))
  FROM Employees
  GROUP BY ManagerID;
```

Führen Sie die folgende Abfrage aus, um die Daten in der Testtabelle anzuzeigen:

```
SELECT *
FROM xmltest
ORDER BY ManagerID;
```

Diese Abfrage erzeugt das folgende Ergebnis:

ManagerID	Reports
501	<reports> <e>102</e> <e>105</e> <e>160</e> <e>243</e> ... </reports>
703	<reports> <e>191</e> <e>750</e> <e>868</e> <e>921</e> ... </reports>
902	<reports> <e>129</e> <e>195</e> <e>299</e> <e>467</e> ... </reports>
1293	<reports> <e>148</e> <e>390</e> <e>586</e> <e>757</e> ... </reports>
...	...

Die folgende Abfrage verwendet eine lateral abgeleitete Tabelle, um eine Ergebnismenge mit zwei Spalten zu generieren: Eine, die die ID für jeden Vorgesetzten auflistet, und eine, die die ID für jeden Mitarbeiter auflistet, der diesem Vorgesetzten zugeordnet ist:

```
SELECT ManagerID, EmployeeID
FROM xmltest, LATERAL( OPENXML( xmltest.Reports, '//e' )
WITH (EmployeeID INT '.') ) DerivedTable
ORDER BY ManagerID, EmployeeID;
```

Diese Abfrage erzeugt das folgende Ergebnis:

ManagerID	EmployeeID
501	102
501	105
501	160
501	243
...	...

Siehe auch

- „XMLELEMENT-Funktion [Zeichenfolge]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „FOR XML AUTO“ auf Seite 693
- „OPENXML-Operator“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „FROM-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Importieren von XML mit dem DataSet-Objekt

Mit dem ADO.NET DataSet-Objekt können Sie die Daten bzw. das Schema von einem XML-Dokument in ein DataSet-Objekt einlesen.

- Die ReadXml-Methode füllt ein DataSet-Objekt aus einem XML-Dokument an, das sowohl ein Schema als auch Daten enthält.
- Die ReadXmlSchema-Methode liest nur das Schema aus einem XML-Dokument ein. Sobald das DataSet-Objekt mit Daten vom XML-Dokument angefüllt ist, können Sie die Tabellen in Ihrer Datenbank anhand der Änderungen im DataSet-Objekt aktualisieren.

DataSet-Objekte können auch mithilfe des SQL Anywhere .NET-Datenproviders in SQL Anywhere verändert werden.

Siehe auch

- „SADataAdapter: Übersicht“ [[SQL Anywhere Server - Programmierung](#)]

Definition von Standard-Namespaces für XML

Sie legen einen Standard-Namespace in einem Element eines XML-Dokuments mit einem Attribut der Form `xmlns="URI"` fest. Im folgenden Beispiel hat ein Dokument den Standard-Namespace, der an den URI `http://www.iAnywhere.com/EmployeeDemo` gebunden ist:

```
<x xmlns="http://www.iAnywhere.com/EmployeeDemo"/>
```

Wenn das Element im Namen kein Präfix hat, wird ein Standard-Namespace auf das Element und seine untergeordneten Klassen angewendet, sofern er festgelegt ist. Ein Doppelpunkt trennt das Präfix vom Rest des Elementnamens. Beispiel: `<x/>` hat kein Präfix, während `<p:x/>` das Präfix `p` hat. Sie legen einen Namespace, der an ein Präfix gebunden ist, mit einem Attribut der Form `xmlns:prefix="URI"` fest. Im folgenden Beispiel bindet ein Dokument das Präfix `p` an denselben URI wie das vorherige Beispiel:

```
<x xmlns:p="http://www.iAnywhere.com/EmployeeDemo"/>
```

Standard-Namespaces werden nie auf Attribute angewendet. Ein Attribut wird immer an den NULL-Namespace-URI gebunden, es sei denn, es hat ein Präfix. Im folgenden Beispiel haben die Wurzel- und untergeordneten Elemente den `iAnywhere1`-Namespace, während das `x`-Attribut den NULL-Namespace-URI und das `y`-Attribut den `iAnywhere2`-Namespace hat:

```
<root xmlns="iAnywhere1" xmlns:p="iAnywhere2">
  <child x='1' p:y='2' />
</root>
```

Die Namespaces, die im Wurzelement des Dokuments festgelegt sind, werden in der Abfrage angewendet, wenn Sie ein XML-Dokument als *namespace-declaration*-Argument von einer OPENXML-Abfrage übergeben. Alle Abschnitte des Dokuments nach dem Wurzelement werden ignoriert. Im folgenden Beispiel ist `p1` an `iAnywhere1` im Dokument und an `p2` im *namespace-declaration*-Argument gebunden, und die Abfrage ist in der Lage, das Präfix `p2` zu verwenden:

```
SELECT *
FROM OPENXML('<p1:x xmlns:p1="iAnywhere1">123</p1:x>', '/p2:x', 1, '<root
xmlns:p2="iAnywhere1"/>')
WITH ( c1 int '.');
```

Für den Vergleich eines Elements müssen Sie den URI korrekt angeben, an den ein Präfix gebunden ist. Im obenstehenden Beispiel stimmt der `x`-Name in der xpath-Abfrage mit dem `x`-Element im Dokument überein, weil sie beide den `iAnywhere1`-Namespace haben.

Für den Vergleich eines Elements müssen Sie den URI korrekt angeben, an den ein Präfix gebunden ist. Im obenstehenden Beispiel stimmt der `x`-Name in der xpath-Abfrage mit dem `x`-Element im Dokument überein, weil sie beide den `iAnywhere1`-Namespace haben. Das Präfix des `x`-Elements der xpath-Abfrage bezieht sich auf den in der *namespace-declaration* festgelegten `iAnywhere1`-Namespace, der mit dem für das `x`-Element in den *xml-data* festgelegten Namespace übereinstimmt.

Verwenden Sie keinen Standard-Namespace in der *namespace-declaration* des OPENXML-Operators. Verwenden Sie eine Wildcard-Abfrage der Form `/*:x` (entspricht einem `x`-Element, das an einen URI gebunden ist, der den NULL-Namespace enthält) oder binden Sie den gewünschten URI an ein bestimmtes Präfix und verwenden Sie dieses in der Abfrage.

Siehe auch

- „OPENXML-Operator“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Abfrageergebnisse als XML

SQL Anywhere unterstützt zwei verschiedene Methoden, um Abfrageergebnisse von Ihren relationalen Daten als XML zu erhalten:

- **FOR XML-Klausel** Die FOR XML-Klausel können Sie in einer SELECT-Anweisung verwenden, um ein XML-Dokument zu generieren.
- **SQL/XML** SQL Anywhere unterstützt Funktionen, die auf dem Draft-Standard SQL/XML basieren und XML-Dokumente aus relationalen Daten generieren.

Die FOR XML-Klausel und die von SQL Anywhere unterstützten SQL/XML-Funktionen bieten Ihnen zwei Alternativen, um XML aus Ihren relationalen Daten zu generieren. Sie können normalerweise eine der beiden Möglichkeiten zum Generieren desselben XML-Codes verwenden.

Beispiel: Diese Abfrage verwendet FOR XML AUTO, um XML zu generieren:

```
SELECT ID, Name
FROM Products
WHERE Color='black'
FOR XML AUTO;
```

Die folgende Abfrage hingegen verwendet die XMLELEMENT-Funktion, um XML zu generieren:

```
SELECT XMLELEMENT(NAME product,
                  XMLATTRIBUTES(ID, Name))
FROM Products
WHERE Color='black';
```

Beide Abfragen generieren den folgenden XML-Code (die Ergebnismenge wurde formatiert, damit sie leichter lesbar wird):

```
<product ID="302" Name="Tee Shirt"/>
<product ID="400" Name="Baseball Cap"/>
<product ID="501" Name="Visor"/>
<product ID="700" Name="Shorts"/>
```

Tipp

Wenn Sie tief-verschachtelte Dokumente generieren, wird eine FOR XML EXPLICIT-Abfrage wahrscheinlich effizienter als eine SQL/XML-Abfrage sein, weil Abfragen im EXPLICIT-Modus normalerweise mit UNION die Verschachtelung generieren, während SQL/XML Unterabfragen verwendet, um die erforderliche Verschachtelung zu erzeugen.

Siehe auch

- „Abrufen von Abfrageergebnissen als XML mit der FOR XML-Klausel“ auf Seite 688
- „Verwendung von SQL/XML zum Abrufen von Abfrageergebnissen als XML“ auf Seite 706
- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Abrufen von Abfrageergebnissen als XML mit der FOR XML-Klausel

In SQL Anywhere können Sie eine SQL-Abfrage für Ihre Datenbank ausführen und die Ergebnisse als ein XML-Dokument zurückgeben, indem Sie die FOR XML-Klausel in Ihrer SELECT-Anweisung verwenden. Das XML-Dokument hat den Typ XML.

Die FOR XML-Klausel kann in jeder SELECT-Anweisung verwendet werden, einschließlich Unterabfragen, Abfragen mit einer GROUP BY-Klausel oder mit Aggregatfunktionen und Ansichtsdefinitionen.

SQL Anywhere erzeugt kein Schema für XML-Dokumente, die durch die FOR XML-Klausel generiert worden sind.

Innerhalb der FOR XML-Klausel können Sie einen von drei XML-Modi festlegen, der das Format der XML-Datei steuert, die generiert wird.

- **RAW** Stellt jede Zeile, die der Abfrage entspricht, als ein XML <row>-Element dar, und jede Spalte als ein Attribut.
- **AUTO** Gibt die Abfrageergebnisse als verschachtelte XML-Elemente zurück. Jede in der SELECT-Liste referenzierte Tabelle wird in der XML-Datei als Element dargestellt. Die Reihenfolge für die Verschachtelung der Elemente basiert auf der Reihenfolge der Spalten in der SELECT-Liste.
- **EXPLICIT** Damit können Sie Abfragen schreiben, die Informationen über die erwartete Verschachtelung enthalten, wodurch Sie das Format der resultierenden XML-Datei bestimmen können.

Die folgenden Abschnitte beschreiben das Verhalten der drei Modi der FOR XML-Klausel in Bezug auf Binärdaten, NULL-Werte und ungültige XML-Namen. Die Abschnitte enthalten auch Beispiele für die Verwendung der FOR XML-Klausel.

Siehe auch

- „XML-Datentyp“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „FOR XML-Beispiele“ auf Seite 690
- „FOR XML RAW“ auf Seite 691
- „FOR XML AUTO“ auf Seite 693
- „FOR XML EXPLICIT“ auf Seite 696

FOR XML und Binärdaten

Wenn Sie die FOR XML-Klausel in einer SELECT-Anweisung verwenden, werden unabhängig vom verwendeten Modus die BINARY-, LONG BINARY-, IMAGE- oder VARBINARY-Spalten als Attribute oder Elemente ausgegeben, die automatisch im base64-kodierten Format dargestellt werden.

Wenn Sie OPENXML verwenden, um eine Ergebnismenge aus XML zu generieren, nimmt OPENXML an, dass die Typen BINARY, LONG BINARY, IMAGE und VARBINARY im base64-kodierten Format sind, und dekodiert sie automatisch.

Siehe auch

- „OPENXML-Operator“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

FOR XML und NULL

Standardmäßig werden Elemente und Attribute, die NULL enthalten, aus dem Ergebnis ausgeklammert. Dieses Verhalten wird von der Option "for_xml_null_treatment" gesteuert.

Angenommen, ein Eintrag in der Tabelle "Customers" enthält für einen Firmennamen NULL.

```
INSERT INTO Customers( ID, Surname, GivenName, Street, City, Phone)
VALUES (100,'Robert','Michael', '100 Anywhere
Lane', 'Smallville', '519-555-3344');
```

Wenn Sie die folgende Abfrage ausführen und die Option "for_xml_null_treatment" auf "Omit" (Standardeinstellung) setzen, wird für NULL in der Spalte kein Attribut generiert.

```
SELECT ID, GivenName, Surname, CompanyName
FROM Customers
WHERE GivenName LIKE 'Michael%'
ORDER BY ID
FOR XML RAW;
```

In diesem Fall wird kein "CompanyName"-Attribut für Michael Robert generiert.

```
<row ID="100" GivenName="Michael" Surname="Robert"/>
<row ID="101" GivenName="Michaels" Surname="Devlin" CompanyName="The Power
Group"/>
<row ID="110" GivenName="Michael" Surname="Agliori" CompanyName="The Pep
Squad"/>
```

Wenn die Option "for_xml_null_treatment" auf "Empty" gesetzt ist, wird ein leeres Attribut in das Ergebnis aufgenommen:

```
<row ID="100" GivenName="Michael" Surname="Robert" CompanyName="" />
<row ID="101" GivenName="Michaels" Surname="Devlin" CompanyName="The Power
Group"/>
<row ID="110" GivenName="Michael" Surname="Agliori" CompanyName="The Pep
Squad"/>
```

In diesem Fall wird ein leeres "CompanyName"-Attribut für Michael Robert generiert.

Siehe auch

- „for_xml_null_treatment-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Regeln für die Codierung unzulässiger XML-Namen

SQL Anywhere verwendet die folgenden Regeln zum Kodieren von Namen, die keine zulässigen XML-Namen sind (zum Beispiel Spaltennamen mit Leerstellen):

XML hat Regeln für Namen, die sich von den Regeln für SQL-Namen unterscheiden. So etwa sind Leerstellen in XML-Namen nicht erlaubt. Wenn ein SQL-Name, wie z.B. ein Spaltenname, in einen

XML-Namen konvertiert wird, werden Zeichen, die für XML nicht zulässig sind, kodiert oder mit Escapezeichen versehen.

Bei jedem kodierten Zeichen basiert die Kodierung auf dem Unicode-Codepunktwert des Zeichens, der als hexadezimale Zahl dargestellt wird.

- Für die meisten Zeichen kann der Codepunktwert mit 16 Bit oder 4 Hex-Ziffern dargestellt werden, wobei die Kodierung `_xHHHH_` benutzt wird. Diese Zeichen entsprechen den Unicode-Zeichen, deren UTF-16-Wert ein 16-Bit-Wort ist.
- Für Zeichen, deren Codepunktwert mehr als 16 Bit benötigt, werden acht Hex-Ziffern in der Kodierung `_xHHHHHHHH_` benutzt. Diese Zeichen entsprechen den Unicode-Zeichen, deren UTF-16-Wert aus zwei 16-Bit-Wörtern besteht. Für die Kodierung wird jedoch der normalerweise aus 5 oder 6 Hex-Ziffern bestehende Unicode-Codepunktwert benutzt, nicht der UTF-16-Wert.

Die folgende Abfrage zum Beispiel enthält einen Spaltennamen mit einer Leerstelle:

```
SELECT EmployeeID AS "Employee ID"
FROM Employees
FOR XML RAW;
```

Sie gibt das folgende Ergebnis zurück:

```
<row Employee_x0020_ID="102"/>
<row Employee_x0020_ID="105"/>
<row Employee_x0020_ID="129"/>
<row Employee_x0020_ID="148"/>
...
```

- Unterstriche (`_`) werden mit Escapezeichen versehen, wenn ihnen das Zeichen `x` folgt. Der Namen `Linu_x` wird beispielsweise als `Linu_x005F_x` kodiert.
- Doppelpunkte (`:`) werden nicht mit Escapezeichen versehen, damit Namespace-Deklarationen und qualifizierte Element- und Attributnamen unter Verwendung einer `FOR XML`-Abfrage generiert werden können.

Tipp

Wenn Sie Abfragen ausführen, die eine `FOR XML`-Klausel in Interactive SQL enthalten, empfiehlt es sich, die Spaltenlänge zu vergrößern, indem Sie die Option `"truncation_length"` entsprechend einstellen.

Siehe auch

- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „truncation_length-Option [Interactive SQL]“ [[SQL Anywhere Server - Datenbankadministration](#)]

FOR XML-Beispiele

Die folgenden Beispiele zeigen, wie die `FOR XML`-Klausel in einer `SELECT`-Anweisung verwendet wird:

- Das folgende Beispiel zeigt, wie die `FOR XML`-Klausel in einer Unterabfrage verwendet wird:

```
SELECT XMLELEMENT( NAME root,
  (SELECT * FROM Employees FOR XML RAW) );
```

- Das folgende Beispiel zeigt, wie die FOR XML-Klausel in einer Abfrage mit einer GROUP BY-Klausel und Aggregatfunktion verwendet wird:

```
SELECT Name, AVG(UnitPrice) AS Price
FROM Products
GROUP BY Name
FOR XML RAW;
```

- Das folgende Beispiel zeigt, wie die FOR XML-Klausel in einer Ansichtsdefinition verwendet wird:

```
CREATE VIEW EmployeesDepartments
AS SELECT Surname, GivenName, DepartmentName
FROM Employees JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID
FOR XML AUTO;
```

FOR XML RAW

Wenn Sie FOR XML RAW in einer Abfrage angeben, wird jede Zeile als ein <row>-Element dargestellt, und jede Spalte ist ein Attribut dieses <row>-Elements.

Syntax

FOR XML RAW[, ELEMENTS]

Parameter

ELEMENTS Fordert FOR XML RAW auf, im Ergebnis ein XML-Element anstatt eines Attributs für jede Spalte zu generieren. Wenn NULL vorhanden ist, wird das Element im generierten XML-Dokument weggelassen. Die folgende Abfrage generiert <EmployeeID>- und <DepartmentName>-Elemente:

```
SELECT Employees.EmployeeID, Departments.DepartmentName
FROM Employees JOIN Departments
ON Employees.DepartmentID=Departments.DepartmentID
FOR XML RAW, ELEMENTS;
```

Diese Abfrage liefert das folgende Ergebnis:

```
<row>
  <EmployeeID>102</EmployeeID>
  <DepartmentName>R & D</DepartmentName>
</row>
<row>
  <EmployeeID>105</EmployeeID>
  <DepartmentName>R & D</DepartmentName>
</row>
<row>
  <EmployeeID>160</EmployeeID>
  <DepartmentName>R & D</DepartmentName>
</row>
<row>
  <EmployeeID>243</EmployeeID>
  <DepartmentName>R & D</DepartmentName>
</row>
...
```

Verwendung

Daten in BINARY-, LONG BINARY-, IMAGE- und VARBINARY-Spalten werden automatisch im base64-kodierten Format zurückgegeben, wenn Sie eine Abfrage ausführen, die FOR XML RAW enthält.

Standardmäßig wird NULL im Ergebnis weggelassen. Dieses Verhalten wird von der Option "for_xml_null_treatment" gesteuert.

FOR XML RAW gibt kein wohlgeformtes XML-Dokument zurück, weil das Dokument keinen einzelnen Stammknoten hat. Wenn ein <root>-Element erforderlich ist, können Sie eines einfügen, indem Sie die XMLELEMENT-Funktion verwenden. Beispiel:

```
SELECT XMLELEMENT( NAME root,
                   (SELECT EmployeeID AS id, GivenName AS name
                    FROM Employees FOR XML RAW) );
```

Die Attribut- oder Elementnamen, die im XML-Dokument verwendet werden, können durch die Angabe von Aliasnamen geändert werden. Die folgende Abfrage benennt das ID-Attribut in "product_ID" um:

```
SELECT ID AS product_ID
FROM Products
WHERE Color='black'
FOR XML RAW;
```

Diese Abfrage liefert das folgende Ergebnis:

```
<row product_ID="302"/>
<row product_ID="400"/>
<row product_ID="501"/>
<row product_ID="700"/>
```

Die Reihenfolge der Ergebnisse hängt vom Plan ab, den der Optimierer ausgewählt hat, außer Sie fordern etwas anderes an. Wenn Sie wollen, dass die Ergebnisse in einer bestimmten Reihenfolge angezeigt werden, müssen Sie eine ORDER BY-Klausel in die Abfrage aufnehmen, wie z.B.:

```
SELECT Employees.EmployeeID, Departments.DepartmentName
FROM Employees JOIN Departments
    ON Employees.DepartmentID=Departments.DepartmentID
ORDER BY EmployeeID
FOR XML RAW;
```

Beispiel

Nehmen wir an, Sie wollen Informationen darüber abrufen, zu welcher Abteilung ein Mitarbeiter gehört; und zwar folgendermaßen:

```
SELECT Employees.EmployeeID, Departments.DepartmentName
FROM Employees JOIN Departments
    ON Employees.DepartmentID=Departments.DepartmentID
FOR XML RAW;
```

Das folgende XML-Dokument wird zurückgegeben:

```
<row EmployeeID="102" DepartmentName="R & D"/>
<row EmployeeID="105" DepartmentName="R & D"/>
<row EmployeeID="160" DepartmentName="R & D"/>
<row EmployeeID="243" DepartmentName="R & D"/>
...
```

Siehe auch

- „FOR XML und NULL“ auf Seite 689
- „XMLELEMENT-Funktion [Zeichenfolge]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

FOR XML AUTO

Der AUTO-Modus generiert verschachtelte Elemente innerhalb des XML-Dokuments.

Wenn die ELEMENTS-Klausel weggelassen wurde, wird jede in der SELECT-Liste referenzierte Tabelle in der generierten XML-Datei als Element dargestellt. Die Reihenfolge der Verschachtelung basiert auf der Reihenfolge, in der die Spalten in der SELECT-Liste referenziert wurden. Für jede Spalte in der SELECT-Liste wird ein Attribut erstellt.

Wenn die ELEMENTS-Klausel vorhanden ist, wird jede in der SELECT-Liste referenzierte Tabelle und Spalte in der generierten XML-Datei als Element dargestellt. Die Reihenfolge der Verschachtelung basiert auf der Reihenfolge, in der die Spalten in der SELECT-Liste referenziert wurden. Für jede Spalte in der SELECT-Liste wird ein Element erstellt.

Syntax

FOR XML AUTO[, ELEMENTS]

Parameter

ELEMENTS Fordert FOR XML AUTO auf, im Ergebnis ein XML-Element anstatt eines Attributs für jede Spalte zu generieren. Beispiel:

```
SELECT Employees.EmployeeID, Departments.DepartmentName
FROM Employees JOIN Departments
    ON Employees.DepartmentID=Departments.DepartmentID
ORDER BY EmployeeID
FOR XML AUTO, ELEMENTS;
```

In diesem Fall wird jede Spalte in der Ergebnismenge als separates Element zurückgegeben und nicht als Attribut des <Employees>- oder <Departments>-Elements. Wenn NULL vorhanden ist, wird das Element im generierten XML-Dokument weggelassen.

```
<Employees>
  <EmployeeID>102</EmployeeID>
  <Departments>
    <DepartmentName>R & D</DepartmentName>
  </Departments>
</Employees>
<Employees>
  <EmployeeID>105</EmployeeID>
  <Departments>
    <DepartmentName>R & D</DepartmentName>
  </Departments>
</Employees>
<Employees>
  <EmployeeID>129</EmployeeID>
  <Departments>
    <DepartmentName>Sales</DepartmentName>
  </Departments>
</Employees>
...
```

Verwendung

Wenn Sie eine Abfrage ausführen, die FOR XML AUTO sowie Daten in BINARY-, LONG BINARY-, IMAGE- und VARBINARY-Spalten verwendet, werden Spalten automatisch im base64-kodiertem Format zurückgegeben. Standardmäßig wird NULL im Ergebnis weggelassen. Sie können NULL als leere Attribute zurückgeben, indem Sie die Option "for_xml_null_treatment" auf EMPTY setzen.

Sofern nicht anders angegeben, gibt der Datenbankserver die Zeilen einer Tabelle in einer beliebigen Reihenfolge zurück. Wenn Sie wollen, dass die Ergebnisse in einer bestimmten Reihenfolge angezeigt werden oder dass ein übergeordnetes Objekt (Parent) mehrere untergeordnete Objekte (Child) hat, nehmen Sie eine ORDER BY-Klausel in die Abfrage auf, damit alle Child-Objekte benachbart sind. Wenn Sie keine ORDER BY-Klausel angeben, hängt die Verschachtelung der Ergebnisse vom Plan ab, der vom Optimierer ausgesucht wurde, was möglicherweise nicht zu der von Ihnen gewünschten Verschachtelung führt.

FOR XML AUTO gibt kein wohlgeformtes XML-Dokument zurück, weil das Dokument keinen einzelnen Stammknoten hat. Wenn ein <root>-Element erforderlich ist, können Sie eines einfügen, indem Sie die XMLELEMENT-Funktion verwenden. Beispiel:

```
SELECT XMLELEMENT( NAME root,
                   (SELECT EmployeeID AS id, GivenName AS name
                    FROM Employees FOR XML AUTO ) );
```

Sie können die Attribut- oder Elementnamen ändern, die im XML-Dokument verwendet werden, indem Sie Aliasnamen angeben. Die folgende Abfrage benennt das ID-Attribut in "product_ID" um:

```
SELECT ID AS product_ID
FROM Products
WHERE Color='Black'
FOR XML AUTO;
```

Der folgende XML-Inhalt wird generiert:

```
<Products product_ID="302"/>
<Products product_ID="400"/>
<Products product_ID="501"/>
<Products product_ID="700"/>
```

Sie können auch die Tabelle mit einem Alias umbenennen. Die folgende Abfrage ändert den Namen der Tabelle zu "product_info":

```
SELECT ID AS product_ID
FROM Products AS product_info
WHERE Color='Black'
FOR XML AUTO;
```

Der folgende XML-Inhalt wird generiert:

```
<product_info product_ID="302"/>
<product_info product_ID="400"/>
<product_info product_ID="501"/>
<product_info product_ID="700"/>
```


Beispiel

Die folgende Abfrage generiert XML, das sowohl das <employee>- als auch das <department>-Element enthält, wobei das <employee>-Element (die Tabelle, die als erste in der SELECT-Liste vorkommt) das übergeordnete Objekt des <department>-Elements darstellt.

```
SELECT EmployeeID, DepartmentName
FROM Employees AS employee
JOIN Departments AS department
    ON employee.DepartmentID=department.DepartmentID
ORDER BY EmployeeID
FOR XML AUTO;
```

Das folgende XML wird durch die oben stehende Abfrage generiert:

```
<employee EmployeeID="102">
  <department DepartmentName="R & D" />
</employee>
<employee EmployeeID="105">
  <department DepartmentName="R & D" />
</employee>
<employee EmployeeID="129">
  <department DepartmentName="Sales;" />
</employee>
<employee EmployeeID="148">
  <department DepartmentName="Finance;" />
</employee>
...
```

Ändern Sie die Reihenfolge der Spalten in der SELECT-Liste wie folgt:

```
SELECT DepartmentName, EmployeeID
FROM Employees AS employee JOIN Departments AS department
    ON employee.DepartmentID=department.DepartmentID
ORDER BY 1, 2
FOR XML AUTO;
```

Das Ergebnis wird folgendermaßen verschachtelt:

```
<department DepartmentName="Finance">
  <employee EmployeeID="148" />
  <employee EmployeeID="390" />
  <employee EmployeeID="586" />
  ...
</department>
<department DepartmentName="Marketing">
  <employee EmployeeID="184" />
  <employee EmployeeID="207" />
  <employee EmployeeID="318" />
  ...
</department>
...
```

Wieder enthält das für diese Abfrage generierte XML sowohl das <employee> als auch das <department>-Element, aber in diesem Fall ist das <department>-Element das übergeordnete Objekt des <employee>-Elements.

Siehe auch

- „for_xml_null_treatment-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „XMLELEMENT-Funktion [Zeichenfolge]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

FOR XML EXPLICIT

Mit FOR XML EXPLICIT können Sie die Struktur des XML-Dokuments steuern, das von der Abfrage zurückgegeben wird. Die Abfrage muss auf eine bestimmte Art geschrieben werden, um die Informationen über die gewünschte Verschachtelung innerhalb des Abfrageergebnisses anzugeben. Mit den von FOR XML EXPLICIT unterstützten optionalen Direktiven können Sie die Behandlung einzelner Spalten konfigurieren. So können Sie zum Beispiel steuern, ob eine Spalte als Element- oder Attributinhalt erscheint, oder ob eine Spalte nur zur Reihung des Ergebnisses verwendet wird und nicht im generierten XML vorkommt.

Parameter

Im EXPLICIT-Modus müssen die ersten beiden Spalten in der SELECT-Anweisung **Tag** bzw. **Parent** benannt sein. "Tag" und "Parent" sind Metadaten-Spalten, deren Werte dazu verwendet werden, in diesen Spalten die Parent-/Child-Objektbeziehungen (Verschachtelung) der Elemente im XML-Dokument festzulegen, das von der Abfrage zurückgegeben wird.

- **Tag-Spalte** Dies ist die erste in der SELECT-Liste angegebene Spalte. Die Tag-Spalte speichert die Tag-Nummer des aktuellen Elements. Zulässige Werte für Tag-Nummern sind 1 bis 255.
- **Parent-Spalte** Diese Spalte speichert die Tag-Nummer für das übergeordnete Objekt des aktuellen Elements. Wenn der Wert in dieser Spalte NULL ist, wird die Zeile auf die oberste Ebene in der XML-Hierarchie platziert.

Nehmen wir zum Beispiel eine Abfrage, die die folgende Ergebnismenge zurückgibt, wobei FOR XML EXPLICIT nicht angegeben ist:

Tag	Parent	GivenName!1	ID!2
1	NULL	'Beth'	NULL
2	NULL	NULL	'102'

In diesem Beispiel sind die Werte in der Tag-Spalte die Nummern der jeweiligen Elemente in der Ergebnismenge. Die Parent-Spalte für beide Zeilen enthält den Wert NULL. Beide Elemente werden in der obersten Ebene der Hierarchie generiert, was das folgende Ergebnis ergibt, wenn die Abfrage die FOR XML EXPLICIT-Klausel enthält:

```
<GivenName>Beth</GivenName>  
<ID>102</ID>
```

Wenn allerdings die zweite Zeile in der Parent-Spalte den Wert 1 hat, sieht das Ergebnis folgendermaßen aus:

```
<GivenName>Beth
  <ID>102</ID>
</GivenName>
```

Datenspalten einer Abfrage hinzufügen

Zusätzlich zu den Tag- und Parent-Spalten muss die Abfrage eine oder mehrere Datenspalten enthalten. Die Namen dieser Datenspalten steuern, wie die Spalten während der Tag-Zuordnung interpretiert werden. Jeder Spaltenname wird in Felder aufgeteilt, die durch ein Ausrufezeichen (!) voneinander getrennt sind. Die folgenden Felder können für Datenspalten angegeben werden:

ElementName!TagNumber!AttributeName!Directive

ElementName Der Name des Elements. Bei einer gegebenen Zeile wird der Name des Elements, das für die Zeile generiert wird, dem Feld *ElementName* der ersten Spalte mit der übereinstimmenden Tag-Nummer entnommen. Wenn es mehrere Spalten mit derselben *TagNumber* gibt, wird der *ElementName* bei nachfolgenden Spalten mit derselben *TagNumber* ignoriert. Im vorigen Beispiel generiert die erste Zeile ein Element mit dem Namen `<GivenName>`.

TagNumber Die Tag-Nummer des Elements. Bei einer Zeile mit einem gegebenen Tag-Wert werden alle Spalten mit demselben Wert in ihrem *TagNumber*-Feld Inhalte dem Element beisteuern, das der Zeile entspricht.

AttributeName Gibt an, dass der Spaltenwert ein Attribut des *ElementName*-Elements ist. Wenn zum Beispiel eine Datenspalte den Namen "productID!!Color" hätte, dann würde "Color" (Farbe) als ein Attribut des `<productID>`-Elements erscheinen.

Directive Mit diesem optionalen Feld können Sie das Format des XML-Dokuments zusätzlich steuern. Sie können einen der folgenden Werte für *Directive* angeben:

- **hide** Gibt an, dass diese Spalte bei der Generierung des Ergebnisses ignoriert wird. Mit dieser Direktive können Sie Spalten aufnehmen, die nur zur Reihung der Tabelle verwendet werden. Der Attributname wird ignoriert und kommt im Ergebnis nicht vor.
- **element** Gibt an, dass die Spalte als ein verschachteltes Element mit dem Namen *AttributeName* eingefügt ist, und nicht als Attribut.
- **xml** Gibt an, dass die Spalte ohne Anführungszeichen eingefügt wird. Wenn der *AttributeName* angegeben ist, wird der Wert als ein Element mit diesem Namen eingefügt. Ansonsten wird er ohne Einfassungselement eingefügt. Wenn diese Direktive nicht verwendet wird, werden Markup-Zeichen in Escapezeichen gesetzt, außer die Spalte ist vom Typ XML. Beispiel: Der Wert `<a/>` würde als `<a/>` eingefügt werden.
- **cdata** Gibt an, dass der Spaltenwert als ein CDATA-Abschnitt eingefügt werden soll. Der *AttributeName* wird ignoriert.

Verwendung

Daten in BINARY-, LONG BINARY-, IMAGE- und VARBINARY-Spalten werden automatisch im base64-kodierten Format zurückgegeben, wenn Sie eine Abfrage ausführen, die FOR XML EXPLICIT enthält. Standardmäßig wird NULL in der Ergebnismenge weggelassen. Sie können dieses Verhalten ändern, indem Sie die Einstellungen der Option "for_xml_null_treatment" ändern.

Siehe auch

- „for_xml_null_treatment-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „FOR XML und NULL“ auf Seite 689
- Das cdata-Direktiv verwenden auf Seite 704
- Die xml-Direktive verwenden auf Seite 703
- Die Element-Direktive verwenden auf Seite 701
- Die Hide-Direktive verwenden auf Seite 702
- Eine Abfrage im EXPLICIT-Modus schreiben auf Seite 698
- Datenspalten einer Abfrage hinzufügen auf Seite 697
- Parameter auf Seite 696

Eine Abfrage im EXPLICIT-Modus schreiben

Nehmen wir an, Sie wollen eine Abfrage unter Verwendung von FOR XML EXPLICIT schreiben, die das folgende XML-Dokument generiert:

```
<employee employeeID='129'>
  <customer customerID='107' region='Eastern' />
  <customer customerID='119' region='Western' />
  <customer customerID='131' region='Eastern' />
</employee>
<employee employeeID='195'>
  <customer customerID='109' region='Eastern' />
  <customer customerID='121' region='Central' />
</employee>
```

Sie erreichen dies, indem Sie eine SELECT-Anweisung schreiben, die die folgende Ergebnismenge exakt in der angegebenen Reihenfolge zurückgibt, und dann FOR XML EXPLICIT an die Abfrage anhängen.

Tag	Parent	employee!1!employeeID	customer!2!customerID	customer!2!region
1	NULL	129	NULL	NULL
2	1	129	107	Eastern
2	1	129	119	Western
2	1	129	131	Central
1	NULL	195	NULL	NULL
2	1	195	109	Eastern
2	1	195	121	Central

Wenn Sie Ihre Abfrage schreiben, werden nur einige Spalten einer gegebenen Zeile Teil des generierten XML-Dokuments. Eine Spalte wird in das XML-Dokument nur aufgenommen, wenn der Wert im *TagNumber*-Feld (das zweite Feld im Spaltennamen) mit dem Wert in der Tag-Spalte übereinstimmt.

In diesem Beispiel wird die dritte Spalte für die zwei Zeilen verwendet, die den Wert 1 in ihrer Tag-Spalte haben. In der vierten und fünften Spalte werden die Werte für die Zeilen verwendet, die den Wert 2 in

ihrer Tag-Spalte haben. Die Elementnamen werden dem ersten Feld im Spaltennamen entnommen. In diesem Fall werden <employee>- und <customer>-Elemente erstellt.

Die Attributnamen kommen aus dem dritten Feld im Spaltennamen. Daher wird für <employee>-Elemente ein **employeeID**-Attribut erstellt, während für <customer>-Elemente **customerID**- und **region**-Attribute generiert werden.

Die folgenden Schritte beschreiben, wie Sie unter Verwendung der SQL Anywhere-Beispieldatenbank die FOR XML EXPLICIT-Abfrage konstruieren, die ein XML-Dokument generiert, das jenem am Beginn dieses Abschnitts ähnelt.

Beispiel: Schreiben einer FOR XML EXPLICIT-Abfrage

1. Schreiben Sie eine SELECT-Anweisung, um die obersten Elemente zu generieren.

In diesem Beispiel generiert die erste SELECT-Anweisung in der Abfrage die <employee>-Elemente. Die ersten zwei Werte in der Abfrage müssen die Tag- und Parent-Spaltenwerte sein. Das <employee>-Element ist an der Spitze der Hierarchie und bekommt daher einen Tag-Wert von 1 und einen Parent-Wert von NULL zugeordnet.

Hinweis

Wenn Sie eine Abfrage im EXPLICIT-Modus schreiben, die einen UNION verwendet, dann werden nur die Spaltennamen verwendet, die in der ersten SELECT-Anweisung angegeben werden. Spaltennamen, die als Element- oder Attributnamen verwendet werden sollen, müssen in der ersten SELECT-Anweisung angegeben werden, weil Spaltennamen ignoriert werden, die in nachfolgenden SELECT-Anweisungen angegeben werden.

2. Um die <employee>-Elemente für die oben stehende Tabelle zu generieren, sieht Ihre erste SELECT-Anweisung folgendermaßen aus:

```
SELECT
    1                AS tag,
    NULL             AS parent,
    EmployeeID AS [employee!1!employeeID],
    NULL           AS [customer!2!customerID],
    NULL           AS [customer!2!region]
FROM Employees;
```

3. Schreiben Sie eine SELECT-Anweisung, um die untergeordneten Elemente zu generieren.

Die zweite Abfrage generiert die <customer>-Elemente. Da es sich um eine Abfrage im EXPLICIT-Modus handelt, müssen die ersten zwei in allen SELECT-Anweisungen angegebenen Werte die Tag- und Parent-Werte sein. Das <customer>-Element erhält die Tag-Nummer 2, und da es dem <employee>-Element untergeordnet ist, ist sein Parent-Wert 1. Die erste SELECT-Anweisung hat bereits festgelegt, dass "EmployeeID", "CustomerID" und "Region" Attribute sind.

```
SELECT
    2,
    1,
    EmployeeID,
    CustomerID,
    Region
FROM Employees KEY JOIN SalesOrders
```

4. Fügen Sie der Abfrage UNION DISTINCT hinzu, um die beiden SELECT-Anweisungen miteinander zu vereinen.

```
SELECT
    1          AS tag,
    NULL       AS parent,
    EmployeeID AS [employee!1!employeeID],
    NULL       AS [customer!2!customerID],
    NULL       AS [customer!2!region]
FROM Employees
UNION DISTINCT
SELECT
    2,
    1,
    EmployeeID,
    CustomerID,
    Region
FROM Employees KEY JOIN SalesOrders
```

5. Fügen Sie eine ORDER BY-Klausel hinzu, um die Reihenfolge der Zeilen im Ergebnis festzulegen. Die Reihenfolge der Zeilen ist jene, die im resultierenden Dokument verwendet wird.

```
SELECT
    1          AS tag,
    NULL       AS parent,
    EmployeeID AS [employee!1!employeeID],
    NULL       AS [customer!2!customerID],
    NULL       AS [customer!2!region]
FROM Employees
UNION DISTINCT
SELECT
    2,
    1,
    EmployeeID,
    CustomerID,
    Region
FROM Employees KEY JOIN SalesOrders
ORDER BY 3, 1
FOR XML EXPLICIT;
```

FOR XML EXPLICIT-Beispiele

Die folgende Beispielabfrage ruft Informationen über die von Mitarbeitern getätigten Bestellungen ab. In diesem Beispiel gibt es drei Elemente: <employee>, <order> und <department>. Das <employee>-Element hat ID- und Namensattribute, das <order>-Element hat ein Datumsattribut, und das <department>-Element hat ein Namensattribut.

```
SELECT
    1          tag,
    NULL       parent,
    EmployeeID [employee!1!id],
    GivenName  [employee!1!name],
    NULL       [order!2!date],
    NULL       [department!3!name]
FROM Employees
UNION DISTINCT
SELECT
    2,
    1,
    EmployeeID,
    NULL,
```

```

        OrderDate,
        NULL
FROM Employees KEY JOIN SalesOrders
UNION DISTINCT
SELECT
        3,
        1,
        EmployeeID,
        NULL,
        NULL,
        DepartmentName
FROM Employees e JOIN Departments d
        ON e.DepartmentID=d.DepartmentID
ORDER BY 3, 1
FOR XML EXPLICIT;

```

Diese Abfrage erzeugt das folgende Ergebnis:

```

<employee id="102" name="Fran">
  <department name="R & D"/>
</employee>
<employee id="105" name="Matthew">
  <department name="R & D"/>
</employee>
<employee id="129" name="Philip">
  <order date="2000-07-24"/>
  <order date="2000-07-13"/>
  <order date="2000-06-24"/>
  <order date="2000-06-08"/>
  ...
  <department name="Sales"/>
</employee>
<employee id="148" name="Julie">
  <department name="Finance"/>
</employee>
...

```

Die Element-Direktive verwenden

Um Unterelemente statt Attributen zu generieren, können Sie die "element"-Direktive zur Abfrage hinzufügen, und zwar folgendermaßen:

```

SELECT
        1,
        NULL,
        EmployeeID,
        GivenName,
        NULL,
        NULL,
        tag,
        parent,
        [employee!1!id!element],
        [employee!1!name!element],
        [order!2!date!element],
        [department!3!name!element]
FROM Employees
UNION DISTINCT
SELECT
        2,
        1,
        EmployeeID,
        NULL,
        OrderDate,
        NULL
FROM Employees KEY JOIN SalesOrders
UNION DISTINCT
SELECT
        3,
        1,

```

```
        EmployeeID,  
        NULL,  
        NULL,  
        DepartmentName  
FROM Employees e JOIN Departments d  
  ON e.DepartmentID=d.DepartmentID  
ORDER BY 3, 1  
FOR XML EXPLICIT;
```

Diese Abfrage erzeugt das folgende Ergebnis:

```
<employee>  
  <id>102</id>  
  <name>Fran</name>  
  <department>  
    <name>R & D</name>  
  </department>  
</employee>  
<employee>  
  <id>105</id>  
  <name>Matthew</name>  
  <department>  
    <name>R & D</name>  
  </department>  
</employee>  
<employee>  
  <id>129</id>  
  <name>Philip</name>  
  <order>  
    <date>2000-07-24</date>  
  </order>  
  <order>  
    <date>2000-07-13</date>  
  </order>  
  <order>  
    <date>2000-06-24</date>  
  </order>  
  ...  
  <department>  
    <name>Sales</name>  
  </department>  
</employee>  
...
```

Die Hide-Direktive verwenden

In der folgenden Abfrage wird die EmployeeID zur Reihung des Ergebnisses verwendet, aber die EmployeeID selbst erscheint nicht im Ergebnis, weil die hide-Direktive (Ausblenden) angegeben ist:

```
SELECT  
  1,          tag,  
  NULL        parent,  
  EmployeeID  [employee!1:id!hide],  
  GivenName   [employee!1:name],  
  NULL        [order!2:date],  
  NULL        [department!3:name]  
FROM Employees  
UNION DISTINCT  
SELECT  
  2,  
  1,  
  EmployeeID,  
  NULL,
```



```

        OrderDate,
        NULL
FROM Employees KEY JOIN SalesOrders
UNION DISTINCT
SELECT
        3,
        1,
        EmployeeID,
        NULL,
        NULL,
        DepartmentName
FROM Employees e JOIN Departments d
        ON e.DepartmentID=d.DepartmentID
ORDER BY 3, 1
FOR XML EXPLICIT;

```

Diese Abfrage liefert das folgende Ergebnis:

```

<employee name="Fran">
  <department name="R & D" />
</employee>
<employee name="Matthew">
  <department name="R & D" />
</employee>
<employee name="Philip">
  <order date="2000-04-21" />
  <order date="2001-07-23" />
  <order date="2000-12-30" />
  <order date="2000-12-20" />
  ...
  <department name="Sales" />
</employee>
<employee name="Julie">
  <department name="Finance" />
</employee>
...

```

Die xml-Direktive verwenden

Wenn das Ergebnis einer FOR XML EXPLICIT-Abfrage Zeichen enthält, die keine gültigen XML-Zeichen sind, werden die unzulässigen Zeichen standardmäßig mit Escapezeichen versehen, außer die Spalte ist vom Typ XML. Hinweise finden Sie unter [„Regeln für die Codierung unzulässiger XML-Namen“ auf Seite 689](#).

Beispiel: Die folgende Abfrage generiert XML, das ein kaufmännisches Und (&) enthält:

```

SELECT
        1                AS tag,
        NULL             AS parent,
        ID               AS [customer!!id!element],
        CompanyName      AS [customer!!company!element]
FROM Customers
WHERE ID = '115'
FOR XML EXPLICIT;

```

Im von dieser Abfrage generierten Ergebnis wird das kaufmännische Und in Escapezeichen gesetzt, weil die Spalte nicht vom Typ XML ist:

```

<customer><id>115</id>
<company>Sterling & Co.</company>
</customer>

```

Die xml-Direktive gibt an, dass der Spaltenwert ohne Escape-Zeichen in das generierte XML eingefügt wird. Führen Sie dieselbe Abfrage wie oben mit der xml-Direktive aus:

```
SELECT
    1                AS tag,
    NULL             AS parent,
    ID                AS [customer!1!id!element],
    CompanyName       AS [customer!1!company!xml]
FROM Customers
WHERE ID = '115'
FOR XML EXPLICIT;
```

Für das kaufmännische Und-Zeichen wird im Ergebnis kein Escape-Zeichen gesetzt:

```
<customer>
  <id>115</id>
  <company>Sterling & Co.</company>
</customer>
```

Dieses XML ist nicht wohlgeformt, weil es ein kaufmännisches Und enthält, das ein Sonderzeichen in XML ist. Wenn XML von einer Abfrage generiert wird, liegt es in Ihrer Verantwortung, dass das XML wohlgeformt und gültig ist. SQL Anywhere überprüft nicht, ob das generierte XML wohlgeformt und gültig ist.

Wenn Sie die xml-Direktive angeben, wird das *AttributeName*-Feld verwendet, um Elemente statt Attribute zu generieren.

Das cdata-Direktiv verwenden

Die folgende Abfrage verwendet die cdata-Direktive, um den Kundennamen in einen CDATA-Abschnitt zurückzugeben:

```
SELECT
    1                AS tag,
    NULL             AS parent,
    ID                AS [product!1!id],
    Description       AS [product!1!!cdata]
FROM Products
FOR XML EXPLICIT;
```

Das von dieser Abfrage erzeugte Ergebnis listet die Beschreibung für jedes Produkt in einem CDATA-Abschnitt auf. Im CDATA-Abschnitt enthaltene Daten werden nicht in Anführungszeichen gesetzt:

```
<product id="300">
  <![CDATA[Tank Top]]>
</product>
<product id="301">
  <![CDATA[V-neck]]>
</product>
<product id="302">
  <![CDATA[Crew Neck]]>
</product>
<product id="400">
  <![CDATA[Cotton Cap]]>
</product>
...
```

Verwendung von Interactive SQL zum Anzeigen von Ergebnissen

Das Ergebnis einer FOR XML-Abfrage wird als Zeichenfolge zurückgegeben. In vielen Fällen ist die Ergebnis-Zeichenfolge relativ lang. In Interactive SQL haben Sie die Möglichkeit, mithilfe der Option **In Fenster anzeigen** die Struktur eines wohlgeformten XML-Dokuments anzuzeigen.

Das Ergebnis einer FOR XML-Abfrage kann in ein wohlgeformtes XML-Dokument umgewandelt werden, einschließlich eines <?xml?>-Tags und eines beliebigen umschließenden Tag-Paars (z.B. <root>...</root>). Die folgende Abfrage zeigt, wie Sie hierzu vorgehen.

```
SELECT XMLCONCAT( CAST('<?xml version="1.0"?>' AS XML),
  XMLELEMENT( NAME root, (
    SELECT
      1          AS tag,
      NULL       AS parent,
      EmployeeID AS [employee!1!employeeID],
      NULL       AS [customer!2!customerID],
      NULL       AS [customer!2!region],
      NULL       AS [custname!3!given_name!element],
      NULL       AS [custname!3!surname!element]
    FROM Employees
    UNION DISTINCT
    SELECT
      2,
      1,
      EmployeeID,
      CustomerID,
      Region,
      NULL,
      NULL
    FROM Employees KEY JOIN SalesOrders
    UNION DISTINCT
    SELECT
      3,
      2,
      EmployeeID,
      CustomerID,
      NULL,
      Customers.GivenName,
      Customers.SurName
    FROM SalesOrders
    JOIN Customers
      ON SalesOrders.CustomerID = Customers.ID
    JOIN Employees
      ON SalesOrders.SalesRepresentative = Employees.EmployeeID
    ORDER BY 3, 4, 1
    FOR XML EXPLICIT
  ) )
);
```

Der Wert der Interactive SQL-Spalte **Kürzungslänge** muss groß genug festgelegt werden, um die gesamte Spalte abrufen zu können. Dies ist über das Menü **Extras » Optionen** möglich oder durch Ausführen einer Interactive SQL-Anweisung ähnlich der folgenden.

```
SET OPTION truncation_length = 80000;
```

Um das XML-Ergebnis anzuzeigen, doppelklicken Sie im Fensterausschnitt **Ergebnisse** auf den Spalteninhalt und wählen Sie die Registerkarte **XML-Rahmen** aus.

Siehe auch

- „HTML- und XML-Daten in Interactive SQL anzeigen“ [[SQL Anywhere Server - Datenbankadministration](#)]

Verwendung von SQL/XML zum Abrufen von Abfrageergebnissen als XML

SQL/XML ist ein Draft-Standard, der eine funktionale Integration von XML in die SQL-Sprache beschreibt. Er beschreibt die Methoden, die verwendet werden können, um SQL mit XML anzuwenden. Mit den unterstützten Funktionen können Sie Abfragen schreiben, die XML-Dokumente aus relationalen Daten konstruieren.

Ungültige Namen und SQL/XML

In SQL/XML werden Ausdrücke, die keine zulässigen Namen sind (z.B. Ausdrücke mit Leerstellen), auf dieselbe Weise wie bei der FOR XML-Klausel in Escapezeichen gesetzt. Elementinhalte vom Typ XML werden nicht in Anführungszeichen gesetzt.

Weitere Hinweise zum XML-Datentyp finden Sie unter „[Speichern von XML-Dokumenten in relationalen Datenbanken](#)“ auf Seite 677.

Siehe auch

- „Regeln für die Codierung unzulässiger XML-Namen“ auf Seite 689

Verwendung der XMLAGG-Funktion

Die XMLAGG-Funktion wird verwendet, um verzweigte Baumstrukturen (Wald) von XML-Elementen aus einer Sammlung von XML-Elementen zu erzeugen. XMLAGG ist eine Aggregatfunktion und erzeugt ein einzelnes zusammengesetztes XML-Ergebnis für alle Zeilen in der Abfrage.

In der folgenden Abfrage wird XMLAGG verwendet, um ein <name>-Element für jede Zeile zu generieren, wobei die <name>-Elemente anhand des Mitarbeiternamens gereiht werden. Die ORDER BY-Klausel wird angegeben, um die XML-Elemente zu reihen:

```
SELECT XMLELEMENT( NAME Departments,
                   XMLATTRIBUTES ( DepartmentID ),
                   XMLAGG( XMLELEMENT( NAME name,
                                       Surname )
                           ORDER BY Surname )
                   ) AS department_list
FROM Employees
GROUP BY DepartmentID
ORDER BY DepartmentID;
```

Diese Abfrage erzeugt das folgende Ergebnis:

department_list
<pre><Departments DepartmentID="100"> <name>Breault</name> <name>Cobb</name> <name>Diaz</name> <name>Driscoll</name> ... </Departments></pre>
<pre><Departments DepartmentID="200"> <name>Chao</name> <name>Chin</name> <name>Clark</name> <name>Dill</name> ... </Departments></pre>
<pre><Departments DepartmentID="300"> <name>Bigelow</name> <name>Coe</name> <name>Coleman</name> <name>Davidson</name> ... </Departments></pre>
...

Siehe auch

- „XMLAGG-Funktion [Aggregat]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Verwendung der XMLCONCAT-Funktion

Die XMLCONCAT-Funktion erstellt verzweigte Baumstrukturen (Wald) von XML-Elementen, indem alle hereinkommenden XML-Werte verkettet werden. Die folgende Abfrage verkettet die <given_name>- und <surname>-Elemente für jeden Mitarbeiter in der Employees-Tabelle:

```
SELECT XMLCONCAT( XMLELEMENT( NAME given_name, GivenName ),
                  XMLELEMENT( NAME surname, Surname )
                ) AS "Employee_Name"
FROM Employees;
```

Diese Abfrage liefert das folgende Ergebnis:

Employee_Name
<pre><given_name>Fran</given_name> <surname>Whitney</surname></pre>
<pre><given_name>Matthew</given_name> <surname>Cobb</surname></pre>
<pre><given_name>Philip</given_name> <surname>Chin</surname></pre>

Employee_Name
<given_name>Julie</given_name> <surname>Jordan</surname>
...

Siehe auch

- „XMLCONCAT-Funktion [Zeichenfolge]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Verwendung der XMLELEMENT-Funktion

Die XMLELEMENT-Funktion konstruiert ein XML-Element aus relationalen Daten. Sie können den Inhalt des generierten Elements festlegen und gegebenenfalls auch Attribute und Attributinhalt für dieses Element angeben.

Verschachtelte Elemente generieren

Die folgende Abfrage generiert verschachteltes XML, wobei ein <product_info>-Element für jedes Produkt mit Elementen erzeugt wird, die den Namen, die Menge und die Beschreibung für jedes Produkt liefern:

```
SELECT ID,
  XMLELEMENT( NAME product_info,
    XMLELEMENT( NAME item_name, Products.name ),
    XMLELEMENT( NAME quantity_left, Products.Quantity ),
    XMLELEMENT( NAME description, Products.Size || ' ' ||
      Products.Color || ' ' || Products.name )
  ) AS results
FROM Products
WHERE Quantity > 30;
```

Diese Abfrage erzeugt das folgende Ergebnis:

ID	results
301	<product_info> <item_name>Tee Shirt </item_name> <quantity_left>54 </quantity_left> <description>Medium Orange Tee Shirt</description> </product_info>
302	<product_info> <item_name>Tee Shirt </item_name> <quantity_left>75 </quantity_left> <description>One Size fits all Black Tee Shirt </description> </product_info>

ID	results
400	<pre><product_info> <item_name>Baseball Cap </item_name> <quantity_left>112 </quantity_left> <description>One Size fits all Black Baseball Cap </description> </product_info></pre>
...	...

Element-Inhalt angeben

Mit der XMLELEMENT-Funktion können Sie den Inhalt eines Elements festlegen. Die folgende Anweisung erzeugt ein XML-Element mit dem Inhalt **hat**.

```
SELECT ID, XMLELEMENT( NAME product_type, 'hat' )
FROM Products
WHERE Name IN ( 'Baseball Cap', 'Visor' );
```

Elemente mit Attributen generieren

Sie können den Elementen Attribute hinzufügen, indem Sie das XMLATTRIBUTES-Argument in Ihrer Abfrage verwenden. Dieses Argument legt den Attributnamen und Inhalt fest. Die folgende Anweisung erzeugt ein Attribut für den Namen, die Farbe und den Preis für jeden Artikel.

```
SELECT ID, XMLELEMENT( NAME item_description,
                      XMLATTRIBUTES( Name,
                                      Color,
                                      UnitPrice )
                      ) AS item_description_element
FROM Products
WHERE ID > 400;
```

Attribute können benannt werden, indem die AS-Klausel angegeben wird:

```
SELECT ID, XMLELEMENT( NAME item_description,
                      XMLATTRIBUTES ( Color AS color,
                                      UnitPrice AS price ),
                      Products.Name
                      ) AS products
FROM Products
WHERE ID > 400;
```

Beispiel

Das folgende Beispiel verwendet XMLELEMENT mit einem HTTP-Webservice.

```
CREATE OR REPLACE PROCEDURE "DBA"."http_header_example_with_table_proc"()
RESULT ( res LONG VARCHAR )
BEGIN
  DECLARE var LONG VARCHAR;
  DECLARE varval LONG VARCHAR;
  DECLARE i INT;
  DECLARE res LONG VARCHAR;
  DECLARE htmltable XML;
```

```
        SET var = NULL;
loop_h:
    LOOP
        SET var = NEXT_HTTP_HEADER( var );
        IF var IS NULL THEN LEAVE loop_h END IF;
        SET varval = http_header( var );
        -- ... do some action for <var,varval> pair...
        SET htmltable = htmltable ||
            XMLELEMENT( name "tr",
                XMLATTRIBUTES( 'left' AS "align", 'top' AS "valign" ),
                XMLELEMENT( name "td", var ),
                XMLELEMENT( name "td", varval ) ) ;
    END LOOP;

    SET res = XMLELEMENT( NAME "table",
        XMLATTRIBUTES( ' ' AS "BORDER", '10' AS "CELLPADDING", '0' AS
"CELLSPACING" ),

        XMLELEMENT( NAME "th",
            XMLATTRIBUTES( 'left' AS "align", 'top' AS "valign" ),
            'Header Name' ),

        XMLELEMENT( NAME "th",
            XMLATTRIBUTES( 'left' AS "align", 'top' AS "valign" ),
            'Header Value' ),

        htmltable);
    SELECT res;
END;
```

Siehe auch

- „XMLELEMENT-Funktion [Zeichenfolge]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Verwendung der XMLFOREST-Funktion

XMLFOREST konstruiert verzweigte Baumstrukturen (Wald) von XML-Elementen. Es wird ein Element für jedes XMLFOREST-Argument erzeugt.

Die folgende Abfrage erzeugt ein <item_description>-Element mit Name- (<name>), Farbe- (<color>) und Preis- (<price>) Elementen:

```
SELECT ID, XMLELEMENT( NAME item_description,
                        XMLFOREST( Name AS name,
                                    Color AS color,
                                    UnitPrice AS price )
                        ) AS product_info
FROM Products
WHERE ID > 400;
```

Das folgende Ergebnis wird durch diese Abfrage generiert:

ID	product_info
401	<pre><item_description> <name>Baseball Cap</name> <color>White</color> <price>10.00</price> </item_description></pre>
500	<pre><item_description> <name>Visor</name> <color>White</color> <price>7.00</price> </item_description></pre>
501	<pre><item_description> <name>Visor</name> <color>Black</color> <price>7.00</price> </item_description></pre>
...	...

Siehe auch

- „XMLFOREST-Funktion [Zeichenfolge]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Verwendung der XMLGEN-Funktion

Die XMLGEN-Funktion wird verwendet, um einen XML-Wert zu generieren, der auf einem XQuery-Konstruktor basiert.

Das von der folgenden Abfrage generierte XML enthält Informationen über die Kundenbestellungen in der SQL Anywhere-Beispieldatenbank. Es verwendet die folgenden variablen Referenzen:

- **{ \$ID }** Generiert den Inhalt für das <ID>-Element, wobei Werte aus der ID-Spalte in der SalesOrders-Tabelle verwendet werden.
- **{ \$OrderDate }** Generiert den Inhalt für das <date>-Element, wobei Werte aus der OrderDate-Spalte in der SalesOrders-Tabelle verwendet werden.
- **{ \$Customers }** Generiert den Inhalt für das <customer>-Element anhand der CompanyName-Spalte in der Customers-Tabelle.

```
SELECT XMLGEN ( ' <order>
  <ID>{ $ID }</ID>
  <date>{ $OrderDate }</date>
  <customer>{ $Customers }</customer>
</order>' ,
  SalesOrders.ID,
  SalesOrders.OrderDate,
  Customers.CompanyName AS Customers
) AS order_info
FROM SalesOrders JOIN Customers
ON Customers.ID = SalesOrders.CustomerID
ORDER BY SalesOrders.CustomerID;
```

Diese Abfrage erzeugt das folgende Ergebnis:

order_info
<pre><order> <ID>2001</ID> <date>2000-03-16</date> <customer>The Power Group</customer> </order></pre>
<pre><order> <ID>2005</ID> <date>2001-03-26</date> <customer>The Power Group</customer> </order></pre>
<pre><order> <ID>2125</ID> <date>2001-06-24</date> <customer>The Power Group</customer> </order></pre>
<pre><order> <ID>2206</ID> <date>2000-04-16</date> <customer>The Power Group</customer> </order></pre>
...

Attribute generieren

Wenn die ID-Nummer der Bestellung als ein Attribut des <order>-Elements erscheinen soll, schreiben Sie die folgende Abfrage (die variable Referenz steht in doppelten Anführungszeichen, weil sie einen Attributwert angibt):

```
SELECT XMLGEN ( ' <order ID="{ $ID} ">
                  <date>{ $OrderDate}</date>
                  <customer>{ $Customers}</customer>
                  </order>',
                  SalesOrders.ID,
                  SalesOrders.OrderDate,
                  Customers.CompanyName AS Customers
                ) AS order_info
FROM SalesOrders JOIN Customers
ON Customers.ID = SalesOrders.CustomerID
ORDER BY SalesOrders.OrderDate;
```

Diese Abfrage erzeugt das folgende Ergebnis:

order_info
<pre><order ID="2131"> <date>2000-01-02</date> <customer>BoSox Club</customer> </order></pre>

order_info
<pre><order ID="2065"> <date>2000-01-03</date> <customer>Bloomfield&apos;s</customer> </order></pre>
<pre><order ID="2126"> <date>2000-01-03</date> <customer>Leisure Time</customer> </order></pre>
<pre><order ID="2127"> <date>2000-01-06</date> <customer>Creative Customs Inc.</customer> </order></pre>
...

In beiden Ergebnismengen wird der Kundenname "Bloomfield's" als Bloomfield's angeführt, weil der Apostroph in XML ein Sonderzeichen ist und das <customer>-Element nicht von einem XML-Typ generiert wurde.

Weitere Hinweise darüber, wie Sie unzulässige Zeichen in XMLGEN in Anführungszeichen setzen, finden Sie unter [Ungültige Namen und SQL/XML auf Seite 706](#).

Header-Daten für XML-Dokumente angeben

Die FOR XML-Klausel und die von SQL Anywhere unterstützten SQL/XML-Funktionen fügen keine Informationen über die Versionsdeklaration in die XML-Dokumente ein, die sie generieren. Sie können die XMLGEN-Funktion verwenden, um Header-Daten zu generieren.

```
SELECT XMLGEN( '<?xml version="1.0"
               encoding="ISO-8859-1" ?>
               <r>{$x}</r>',
               (SELECT GivenName, Surname
                FROM Customers FOR XML RAW) AS x );
```

Diese Abfrage bewirkt das folgende Ergebnis:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<r>
  <row GivenName="Michaels" Surname="Devlin"/>
  <row GivenName="Beth" Surname="Reiser"/>
  <row GivenName="Erin" Surname="Niedringhaus"/>
  <row GivenName="Meghan" Surname="Mason"/>
  ...
</r>
```

Siehe auch

- „XMLGEN-Funktion [Zeichenfolge]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

JSON in der Datenbank

JSON (JavaScript Object Notation) ist ein sprachenunabhängiges, textbasiertes Datenaustauschformat, das für die Serialisierung von JavaScript-Daten entwickelt wurde.

JSON stellt vier Basistypen dar: Zeichenfolgen, Zahlen, Boolesche Werte und NULL. Außerdem stellt JSON zwei strukturierte Typen dar: Objekte und Arrays.

Weitere Hinweise zum JSON-Format finden Sie unter <http://www.json.org>.

Abfrageergebnisse mit der FOR JSON-Klausel als JSON abrufen

In SQL Anywhere können Sie eine SQL-Abfrage in Ihrer Datenbank ausführen und die Ergebnisse als JSON-Dokument zurückgeben, indem Sie die FOR JSON-Klausel in einer SELECT-Anweisung verwenden.

Die FOR JSON-Klausel kann in jeder SELECT-Anweisung verwendet werden, einschließlich Unterabfragen und Abfragen mit GROUP BY-Klausel oder mit Aggregatfunktionen und Ansichtsdefinitionen. Wenn Sie die FOR JSON-Klausel verwenden, werden relationale Daten als JSON-Array dargestellt, das aus Arrays, Objekten und skalaren Elementen besteht.

Innerhalb der FOR JSON-Klausel können Sie einen der folgenden JSON-Modi festlegen, der das Format der generierten JSON-Datei steuert:

- **RAW** Gibt die Abfrageergebnisse als entschachtelte JSON-Darstellung zurück. Dieser Modus ist zwar ausführlicher, aber die syntaktische Analyse kann einfacher sein.
- **AUTO** Gibt die Abfrageergebnisse als verschachtelte JSON-Elemente zurück, basierend auf Abfrage-Joins.
- **EXPLICIT** Ermöglicht es Ihnen, festzulegen, wie Spaltendaten dargestellt werden. Sie können Spalten als einfache Werte, Objekte oder verschachtelte Objekte angeben, um einheitliche oder heterogene Arrays zu erzeugen.

Siehe auch

- „FOR JSON RAW“ auf Seite 716
- „FOR JSON AUTO“ auf Seite 716
- „FOR JSON EXPLICIT“ auf Seite 717
- „SELECT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ARRAY-Konstruktor [zusammengesetzt]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

FOR JSON RAW

Wenn Sie FOR JSON RAW in einer Abfrage angeben, wird jede Zeile als entschachtelte JSON-Darstellung zurückgegeben.

Syntax

FOR JSON RAW

Verwendung

Diese Klausel ist die empfohlene Methode zum Abrufen von Abfrageergebnissen als JSON-Objekte, da sie am leichtesten syntaktisch zu analysieren und zu verstehen ist.

Beispiel

Die folgende Abfrage verwendet FOR JSON RAW, um Mitarbeiterinformationen aus der Tabelle "Employees" zurückzugeben:

```
SELECT
    emp.EmployeeID,
    so.CustomerID,
    so.Region
FROM Employees AS emp KEY JOIN SalesOrders AS so WHERE emp.EmployeeID <= 195
ORDER BY 1
FOR JSON RAW;
```

Im Gegensatz zu den Ergebnissen, die bei Verwendung von FOR JSON AUTO zurückgegeben werden und bei denen eine Verschachtelung der Ergebnisse erfolgt, wird bei FOR JSON RAW eine entschachtelte Ergebnismenge zurückgegeben:

```
[
  { "EmployeeID" : 129, "CustomerID" : 107, "Region" : "Eastern" },
  { "EmployeeID" : 129, "CustomerID" : 119, "Region" : "Western" },
  . . .
  { "EmployeeID" : 129, "CustomerID" : 131, "Region" : "Eastern" },
  { "EmployeeID" : 195, "CustomerID" : 176, "Region" : "Eastern" }
]
```

Siehe auch

- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

FOR JSON AUTO

Wenn Sie FOR JSON AUTO in einer Abfrage angeben, gibt die Abfrage eine verschachtelte Hierarchie von JSON-Objekten zurück, basierend auf Abfrage-Joins.

Syntax

FOR JSON AUTO

Verwendung

Verwenden Sie die FOR JSON AUTO-Klausel in einer Abfrage, wenn die Ergebnismenge die hierarchischen Beziehungen zwischen den JSON-Objekten zeigen soll.

Beispiel

Das folgende Beispiel gibt ein JSON-Array von **emp**-Objekten zurück, von denen jedes einen EmployeeID-Wert und ein **so**-Objekt enthält. Das **so**-Objekt ist ein Array von Objekten, die aus einem CustomerID-Wert und einer Region bestehen.

```
SELECT
    emp.EmployeeID,
    so.CustomerID,
    so.Region
FROM Employees AS emp KEY JOIN SalesOrders AS so WHERE emp.EmployeeID <= 195
ORDER BY 1
FOR JSON AUTO;
```

Anders als bei FOR JSON RAW wird mit FOR JSON AUTO eine verschachtelte Hierarchie von Daten zurückgegeben, wobei ein **emp**- oder Employee-Objekt aus einem **so**- oder SalesOrders-Objekt besteht, das ein Array von CustomerID-Daten enthält:

```
[
  {
    "emp": {
      "EmployeeID" : 129,
      "so": [
        { "CustomerID" : 107 , "Region" : "Eastern" },
        ...
        { "CustomerID" : 131 , "Region" : "Eastern" }
      ]
    },
    "emp": {
      "EmployeeID" : 195,
      "so": [
        { "CustomerID" : 109 , "Region" : "Eastern" },
        ...
        { "CustomerID" : 176 , "Region" : "Eastern" }
      ]
    }
  ]
}
```

Siehe auch

- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ARRAY-Konstruktor [zusammengesetzt]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

FOR JSON EXPLICIT

Wenn Sie FOR JSON EXPLICIT in einer Abfrage verwenden, können Sie Spalten als einfache Werte, Objekte oder verschachtelte Objekte angeben, um einheitliche oder heterogene Arrays zu erzeugen.

Syntax

FOR JSON EXPLICIT

Verwendung

FOR JSON EXPLICIT verwendet einen Spaltenalias, um eine detaillierte Formatspezifikation bereitzustellen. Wenn kein Alias vorhanden ist, wird die angegebene Spalte als Wert ausgegeben. Ein

Alias muss vorhanden sein, wenn Werte (oder Objekte) innerhalb einer verschachtelten Struktur ausgedrückt werden sollen.

Sie müssen die ersten beiden Spalten in der Auswahlliste mit **tag** und **parent** benennen. Eine Vereinigung von mehreren Abfragen kann eine verschachtelte JSON-Ausgabe zurückgeben, indem die Beziehung zwischen Tag und übergeordnetem Element in jeder Abfrage angegeben wird.

Das Format für die Aliasdirektive ist `[encapsulating_object!tag!name!qualifier]`. Dabei gilt:

- **!** grenzt Direktivenkriterien ab.
- **encapsulating_object** gibt ein verkapselndes Objekt (Array-Objekt) für das Auswahllistenelement aus.
- **tag** definiert einen Bezeichner für die Spalte, die in nachfolgenden Abfragen verwendet wird. Außerdem werden Verschachtelungskriterien (relativ zum übergeordneten Objekt) festgelegt.
- **name** weist einen Namen für das (Name/Wert-Paar)-Objekt zu.
- **qualifier** kann entweder **element** (Standardwert) oder **hide** sein, um das Element in der Ergebnismenge zu verschleiern.

Beispiel

Die folgende Abfrage verwendet FOR JSON EXPLICIT, um Mitarbeiterinformationen aus der Tabelle "Employees" zurückzugeben:

```
SELECT
  1                AS tag,
  NULL             AS parent,
  emp.EmployeeID   AS [!1!EmployeeID],
  so.CustomerID    AS [!1!CustomerID],
  so.Region        AS [!1!Region]
FROM Employees AS emp KEY JOIN SalesOrders AS so WHERE emp.EmployeeID <= 195
ORDER BY 3
FOR JSON EXPLICIT;
```

Das Ergebnis ist identisch mit dem des FOR JSON RAW-Beispiels:

```
[
  { "EmployeeID" : 129, "CustomerID" : 107, "Region" : "Eastern" },
  { "EmployeeID" : 129, "CustomerID" : 119, "Region" : "Western" },
  ...
  { "EmployeeID" : 129, "CustomerID" : 131, "Region" : "Eastern" },
  { "EmployeeID" : 195, "CustomerID" : 176, "Region" : "Eastern" }
]
```

Im folgenden Beispiel wird eine Ergebnismenge zurückgegeben, die dem Ergebnis des FOR JSON AUTO-Beispiels ähnlich ist:

```
SELECT
  1                AS tag,
  NULL             AS parent,
  emp.EmployeeID   AS [emp!1!EmployeeID],
  null             AS [so!2!CustomerID],
  null             AS [!2!Region]
FROM Employees as emp where emp.EmployeeID <= 195
```



```

UNION ALL
SELECT
    2,
    1,
    emp.EmployeeID,
    so.CustomerID,
    so.Region
FROM Employees as emp KEY JOIN SalesOrders as so where emp.EmployeeID <= 195
ORDER BY 3, 1
FOR JSON EXPLICIT;

```

Die obige Abfrage gibt das folgende Ergebnis zurück:

```

[
  {
    "emp": [ { "EmployeeID": 102 } ],
    "emp": [ { "EmployeeID": 105 } ],
    "emp": [
      { "EmployeeID": 129,
        "so": [
          { "CustomerID": 101, "Region": "Eastern" },
          ...
          { "CustomerID": 205, "Region": "Eastern" }
        ]
      }
    ]
  },
  {
    "emp": [ { "EmployeeID": 148 } ],
    "emp": [ { "EmployeeID": 160 } ],
    "emp": [ { "EmployeeID": 184 } ],
    "emp": [ { "EmployeeID": 191 } ],
    "emp": [
      { "EmployeeID": 195,
        "so": [
          { "CustomerID": 101, "Region": "Eastern" },
          ...
          { "CustomerID": 209, "Region": "Western" }
        ]
      }
    ]
  }
]

```

Außer der Reihenfolge der Arrays und der Einbeziehung von Mitarbeitern ohne Bestellungen unterscheidet sich das obige Format von den FOR JSON AUTO-Ergebnissen nur insofern, als **emp** ein Array von Strukturen ist. In FOR JSON AUTO ist klar, dass **emp** nur ein einzelnes Objekt hat. FOR JSON EXPLICIT verwendet eine Array-Verkapselung, die Aggregation unterstützt.

Im folgenden Beispiel wird die **emp**-Verkapselung entfernt und "Region" als Wert zurückgegeben. Dieses Beispiel zeigt, wie der FOR JSON EXPLICIT-Modus eine granulare Formatierungssteuerung ermöglicht, um ein Ergebnis zu erzeugen, das zwischen denen der Modi RAW und AUTO liegt.

```

SELECT
    1 AS tag,
    NULL AS parent,
    emp.EmployeeID AS [!1!EmployeeID],           // remove "emp"
    encapsulation
    null AS [so!2!id],                           // change "CustomerID"
    to just "id"
    null AS [!2!]                                // stipulate that region
    should be emitted as a value
FROM Employees AS emp WHERE emp.EmployeeID <= 195
UNION ALL
SELECT

```

```
2,
1,
emp.EmployeeID,
so.CustomerID,
so.Region
FROM Employees as emp KEY JOIN SalesOrders AS so WHERE emp.EmployeeID <= 195
ORDER BY 3, 1
FOR JSON EXPLICIT;
```

Im Abfrageergebnis ist **so** nicht mehr ein Array von Objekten, sondern ein zweidimensionales Array:

```
[
  { "EmployeeID":102},{ "EmployeeID":105},{ "EmployeeID":129,
    "so":[
      [{"id":101}, "Eastern"],
      .
      [{"id":205}, "Eastern"]
    ]
  },
  { "EmployeeID":148},
  { "EmployeeID":160},
  { "EmployeeID":184},
  { "EmployeeID":191},
  { "EmployeeID":195,
    "so":[
      [{"id":101}, "Eastern"],
      .
      [{"id":209}, "Western"]
    ]
  }
]
```

Das folgende Beispiel ist ähnlich wie die Verwendung von FOR JSON RAW, aber EmployeeID, CustomerID und Region werden als Werte und nicht Name/Wert-Paare ausgegeben:

```
SELECT
  1 AS tag,
  NULL AS parent,
  emp.EmployeeID, // no alias directives
  so.CustomerID,
  so.Region
FROM Employees AS emp KEY JOIN SalesOrders AS so WHERE emp.EmployeeID <= 195
ORDER BY 3
FOR JSON EXPLICIT;
```

Diese Abfrage gibt das folgende Ergebnis zurück, wobei ein zweidimensionales Array aus EmployeeID, CustomerID und Region erzeugt wird:

```
[
  [129,107,"Eastern"],
  .
  [195,176,"Eastern"]
]
```

Siehe auch

- „ARRAY-Konstruktor [zusammengesetzt]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Datenimport und -export

Der Ausdruck **Massenvorgänge** wird verwendet, um den Prozess des Imports und Exports von Daten zu beschreiben. Massenvorgänge sind nicht Teil von üblichen Endbenutzeranwendungen und erfordern spezielle Privilegien. Massenvorgänge können sich auf Parallelität und Transaktionslogs auswirken und sollten durchgeführt werden, wenn keine Benutzer mit der Datenbank verbunden sind.

Die folgenden Situationen sind typisch für den Import und Export von Daten.

- Datenbestand in eine neue Datenbank importieren
- Neue Kopien einer Datenbank, gegebenenfalls mit einer veränderten Struktur, erstellen
- Daten aus einer Datenbank exportieren, um sie in anderen Anwendungen bearbeiten zu können, z.B. einer Tabellenkalkulation
- Auszüge aus einer Datenbank für die Replikation oder die Synchronisation erstellen
- Eine beschädigte Datenbank reparieren
- Eine Datenbank neu aufbauen, um ihre Performance zu verbessern
- Erwerb einer neueren Version der Datenbanksoftware und Durchführung von Software-Upgrades

Performance-Aspekte von Massenvorgängen

Die Performance von Massenvorgängen hängt von verschiedenen Faktoren ab, etwa ob der Vorgang im Verhältnis zum Datenbankserver intern oder extern ist.

Interne Massenvorgänge

Interne Massenvorgänge, auch *serverseitige* Massenvorgänge genannt, sind Import- und Exportvorgänge, die vom Datenbankserver unter Verwendung der Anweisungen LOAD TABLE und UNLOAD durchgeführt werden.

Wenn Sie interne Massenvorgänge durchführen, können Sie von ASCII-Textdateien oder von Adaptive Server Enterprise BCP-Dateien laden bzw. entladen. Diese Dateien können sich auf demselben Computer wie der Datenbankserver oder auf einem Clientcomputer befinden. Der angegebene Pfad zur Datei, die geschrieben oder gelesen wird, ist relativ zum Datenbankserver. Interne Massenvorgänge stellen die schnellste Methode dar, um Daten in die Datenbank zu importieren oder zu exportieren..

Externe Massenvorgänge

Externe Massenvorgänge, auch *clientseitige* Massenvorgänge genannt, sind Import- und Exportvorgänge, die von einem Client, wie z.B. Interactive SQL, unter Verwendung von INPUT- und OUTPUT-Anweisungen durchgeführt werden. Wenn der Client eine INPUT-Anweisung ausgibt, wird eine INSERT-Anweisung im Transaktionslog für jede Zeile protokolliert, die während der Verarbeitung der in der INPUT-Anweisung angegebenen Datei gelesen wird. Dadurch ist ein clientseitiges Laden deutlich

langsamer als ein serverseitiges Laden. Überdies werden INSERT-Trigger während eines INPUT-Vorgangs ausgelöst.

Die OUTPUT-Anweisung ermöglicht es Ihnen, die Ergebnismenge einer SELECT-Anweisung in vielen verschiedenen Formaten zu schreiben.

Bei externen Massenvorgängen ist der angegebene Pfad der Datei, die gelesen oder geschrieben wird, relativ zum Computer, auf dem die Clientanwendung läuft.

Siehe auch

- „LOAD TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „UNLOAD-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „INPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „OUTPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Performance-Tipps für den Datenimport“ auf Seite 723
- „Datenbankserveroption -b“ [[SQL Anywhere Server - Datenbankadministration](#)]

Die Datenwiederherstellung bei Massenvorgängen

Sie können den Datenbankserver im Massenvorgangsmodus (Serveroption -b) ausführen. Wenn Sie diese Option verwenden, führt der Datenbankserver bestimmte wichtige Funktionen nicht aus. Insbesondere sind das folgende Funktionen:

Funktion	Auswirkungen
Ein Transaktionslog führen	Änderungen werden nicht aufgezeichnet. Jedes COMMIT bewirkt einen Checkpoint.
Datensätze sperren	Es gibt keine gravierenden Auswirkungen.

Andererseits kann es auch sinnvoll sein, sicherzustellen, dass Daten eines Massenimports für den Fall einer Wiederherstellung weiterhin zur Verfügung stehen. Sie erreichen dies, indem Sie die ursprünglichen Datenquellen und ihre ursprüngliche Position unverändert lassen. Sie können auch einige der Protokollierungsoptionen der LOAD TABLE-Anweisung verwenden, die ein Aufzeichnen von massenimportierten Daten im Transaktionslog ermöglichen.

Vorsicht

Sie sollten die Datenbank sichern, bevor und nachdem Sie Vorgänge im Massenvorgangsmodus durchführen, da in diesem Modus die Datenbank nicht gegen Datenträgerausfall gesichert ist.

Siehe auch

- „Datenbankserveroption -b“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „LOAD TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Datenimport

Der Datenimport ist eine Verwaltungsaufgabe, bei der Daten als ein Massenvorgang in Ihre Datenbank eingelesen werden. Verwenden Sie SQL Anywhere, um Folgendes durchzuführen:

- Ganze Tabellen oder Teile von Tabellen aus Textdateien importieren
- Daten aus einer Variablen importieren
- Mehrere Tabellen nacheinander importieren, indem die Importprozedur mit einem Skript automatisiert wird
- Daten in Tabellen einfügen oder hinzufügen
- Daten in Tabellen ersetzen
- Eine Tabelle vor oder während des Importierens erstellen
- Daten von einer Datei auf einem Clientcomputer laden
- Dateien zwischen SQL Anywhere und Adaptive Server Enterprise mithilfe der BCP FORMAT-Klausel übertragen

Wenn Sie eine Datenbank komplett neu erstellen wollen, sollten Sie die Daten mithilfe von LOAD TABLE laden, um die beste Performance zu erreichen.

Siehe auch

- „LOAD TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „UNLOAD-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „INPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „OUTPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Performance-Tipps für den Datenimport“ auf Seite 723
- „Performance-Aspekte von Massenvorgängen“ auf Seite 721
- „Datenbankserveroption -b“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Tabellenstrukturen für den Datenimport“ auf Seite 741
- „Zugriff auf Daten auf Clientcomputern“ auf Seite 760
- „Neuaufbau von Datenbanken“ auf Seite 763

Performance-Tipps für den Datenimport

Der Import großer Datenmengen kann zeitaufwändig sein. Um Zeit zu sparen, können Sie Folgendes durchführen:

- Speichern Sie die Datendateien auf einem anderen physischen Laufwerk als die Datenbankdatei. Damit können Sie die Bewegungen des Lesekopfes der Festplatte reduzieren.
- Erhöhen Sie die Größe der Datenbank. Die ALTER DBSPACE-Anweisung ermöglicht die großvolumige Erweiterung einer Datenbank, bevor der Platz wirklich benötigt wird, anstatt in kleinem

Umfang, wenn der Bedarf eintritt. Das steigert auch die Performance beim Laden großer Datenmengen und führt dazu, dass die Daten im Dateisystem näher beieinander liegen.

- Verwenden Sie temporäre Tabellen, um Daten zu laden. Lokale bzw. globale temporäre Tabellen sind sinnvoll, wenn Sie eine Datenmenge wiederholt laden wollen oder Tabellen mit unterschiedlichen Strukturen zusammenführen müssen.
- Starten Sie den Datenbankserver ohne die Option -b (Massenvorgangsmodus), wenn Sie die LOAD TABLE-Anweisung verwenden.
- Führen Sie Interactive SQL oder die Clientanwendung auf dem Computer aus, auf dem auch der Datenbankserver läuft, wenn Sie die INPUT- oder die OUTPUT-Anweisung verwenden. Wenn Daten über das Netzwerk importiert werden, erhöht dies das Netzwerk-Kommunikationsaufkommen. Sie sollten neue Daten möglichst dann importieren, wenn der Datenbankserver nicht stark ausgelastet ist.

Siehe auch

- „LOAD TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „INPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „OUTPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Datenbankserveroption -b“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „ALTER DBSPACE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Importieren von Daten mit dem Import-Assistenten

Verwenden Sie den **Import-Assistenten** von Interactive SQL, um eine Quelle, das Format und eine Zieltabelle für die Daten auszuwählen. Sie können Daten aus Textdateien, aus Dateien im FIXED-Format oder aus Formdateien in eine bestehende oder eine neue Tabelle importieren.

Voraussetzungen

Wenn Sie Daten in eine vorhandene Tabelle importieren, müssen Sie Eigentümer der Tabelle sein, SELECT- und INSERT-Privilegien für die Tabelle haben oder die Systemprivilegien SELECT ANY TABLE und INSERT ANY TABLE haben.

Wenn Sie Daten in eine neue Tabelle importieren, benötigen Sie das CREATE TABLE-Systemprivileg, das CREATE ANY TABLE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg.

Kontext und Bemerkungen

Sie können den **Import-Assistenten** auch verwenden, um Daten zwischen folgenden Objekten zu übertragen:

- Datenbanken unterschiedlichen Typs, wie eine SQL Anywhere-Datenbank und eine UltraLite-Datenbank.
- Datenbanken unterschiedlicher Versionen (sofern Sie einen ODBC-Treiber für jede Datenbank haben), wie eine Datenbank von SQL Anywhere 16 und eine SQL Anywhere-Datenbank der Version 11.

Verwenden Sie den Interactive SQL **Import-Assistenten** in folgenden Situationen:

- Sie wollen gleichzeitig mit dem Import Ihrer Daten eine Tabelle erstellen.
- Sie ziehen es vor, eine mausgesteuerte Oberfläche zum Importieren von Daten in einem Nicht-Textformat zu verwenden.

Aufgabe

1. Klicken Sie in Interactive SQL auf **Daten » Importieren**.
2. Befolgen Sie die Anweisungen im **Import-Assistenten**.

Ergebnisse

Die Daten werden in die angegebene Datenbank importiert.

Beispiel

Gehen Sie folgendermaßen vor, um Daten aus einer Datei in die SQL Anywhere-Beispieldatenbank zu importieren:

1. Erstellen und speichern Sie eine Textdatei namens *newProducts.csv* mit den folgenden Werten:

```
ID,Name,Description,Size,Color,Quantity,UnitPrice
701,Shorts,Spandex Cycling,Small,Black,500,23.00
702,Shorts,Spandex Cycling,Medium,Black,500,23.00
703,Shorts,Spandex Cycling,Large,Black,200,25.00
```

2. Klicken Sie in Interactive SQL auf **Daten » Importieren**.
3. Klicken Sie auf **In einer Textdatei** und anschließend auf **Weiter**.
4. Klicken Sie auf **Durchsuchen** und navigieren Sie zu dem Ordner mit **newProducts.csv**.

Die Datei befindet sich auf dem Clientcomputer, nicht auf dem Computer des Datenbankservers.

5. Klicken Sie auf **newProducts.csv** und auf **Öffnen**.
6. Klicken Sie auf **In einer bestehenden Tabelle**.
7. Klicken Sie auf **Products** und dann auf **Weiter**.
8. Klicken Sie in der Liste **Feldtrennzeichen** auf **Komma(,)**.
9. Klicken Sie in der Liste **Weitere Optionen** auf **Erste Zeile enthält Spaltennamen**.
10. Klicken Sie auf **Weiter**.
11. Klicken Sie auf **Importieren**.
12. Klicken Sie auf **Schließen**.

Die vom Assistenten erstellten SQL-Anweisungen werden in der Verlaufsliste gespeichert, wenn der Import abgeschlossen ist.

Sie können die generierte SQL INPUT-Anweisung anzeigen, indem Sie im Menü **SQL** auf **Vorherige SQL-Anweisung** klicken.

Die INPUT-Anweisung, die vom **Import-Assistenten** generiert wurde, erscheint im Fensterausschnitt **SQL-Anweisungen**:

```
-- Vom Import-Assistenten generiert
INPUT INTO "GROUPO"."Products" FROM 'C:\\LocalTemp\\newSwimwear.csv'
  format text escapes on escape character '\\\\'
  delimited by ',' encoding 'Cp1252' skip 1
```

Gehen Sie folgendermaßen vor, um Daten aus der SQL Anywhere-Beispieldatenbank in eine UltraLite-Datenbank zu importieren:

1. Stellen Sie eine Verbindung zu einer UltraLite-Datenbank her, beispielsweise *C:\\Users\\Public\\Documents\\SQL Anywhere 16\\Samples\\UltraLite\\CustDB\\custdb.udb*.
2. Klicken Sie in Interactive SQL auf **Daten » Importieren**.
3. Klicken Sie auf **In eine Datenbank**. Klicken Sie auf **Weiter**.
4. Klicken Sie in der Liste **Datenbanktyp** auf **SQL Anywhere**.
5. Klicken Sie in der Dropdown-Liste **Aktion** auf **Mit einer ODBC-Datenquelle verbinden**.
6. Klicken Sie auf **ODBC-Datenquellenname** und geben Sie dann im Feld darunter **SQL Anywhere 16 Demo** ein.
7. Klicken Sie auf **Weiter**.
8. Klicken Sie in der Liste **Tabellenname** auf **Customers**. Klicken Sie auf **Weiter**.
9. Klicken Sie auf **In einer neuen Tabelle**.
10. In das Feld **Tabellenname** geben Sie **SQLAnyCustomers** ein .
11. Klicken Sie auf **Importieren**.
12. Klicken Sie auf **Schließen**.
13. Klicken Sie zum Anzeigen der generierten SQL-Anweisung auf **SQL » Vorherige SQL-Anweisung**.

Die INPUT-Anweisung, die vom **Import-Assistenten** generiert wurde, erscheint im Fensterausschnitt **SQL-Anweisungen**.

```
-- Vom Import-Assistenten generiert
INPUT USING 'dsn=SQL Anywhere 16 Demo;CON=\\'
  FROM "GROUPO.Customers" INTO "SQLAnyCustomers"
  CREATE TABLE ON
```

Daten mit der INPUT-Anweisung importieren

Verwenden Sie die INPUT-Anweisung, um Daten in unterschiedlichen Dateiformaten in vorhandene oder neue Tabellen zu importieren. Wenn Sie die ODBC-Treiber für die Datenbanken installiert haben,

verwenden Sie die USING-Klausel, um Daten aus verschiedenen Datenbankentypen und aus verschiedenen Versionen von SQL Anywhere-Datenbanken zu importieren.

Mit der INPUT-Anweisung können Sie Daten in TEXT- und FIXED-Formaten importieren. Zum Importieren von Daten in einem anderen Dateiformat verwenden Sie die USING-Klausel mit einer ODBC-Datenquelle.

Sie können das Standard-Eingabeformat verwenden oder das Dateiformat bei jeder INPUT-Anweisung festlegen. Da die INPUT-Anweisung eine Interactive SQL-Anweisung ist, können Sie sie nicht in zusammengesetzten Anweisungen (wie z.B. einer IF-Anweisung) oder in gespeicherten Prozeduren verwenden.

Verwenden Sie die INPUT-Anweisung, wenn Sie Daten aus einer Datei oder aus einer anderen Datenbank importieren wollen.

Hinweise zu materialisierten Ansichten

Bei Sofortansichten wird ein Fehler zurückgegeben, wenn Sie versuchen, einen Massenimport von Daten in eine Basistabelle durchzuführen. Sie müssen die Daten in der Ansicht erst kürzen und können erst dann den Massenimport vornehmen.

Bei manuellen Ansichten können Sie Daten mit einem Massenimport in eine Basistabelle laden. Die Daten in der Ansicht bleiben allerdings bis zur nächsten Aktualisierung veraltet.

Denken Sie daher daran, eine Kürzung in abhängigen materialisierten Ansichten vorzunehmen, bevor Sie einen Massenimport wie z.B. eine INPUT-Anweisung in einer Tabelle versuchen. Nachdem Sie die Daten geladen haben, aktualisieren Sie die Ansicht.

Hinweise zu Textindizes

Bei Sofort-Textindizes kann das Aktualisieren eines Textindexes nach dem Massenimport wie etwa einer INPUT-Anweisung in der Basistabelle eine bestimmte Zeit in Anspruch nehmen, auch wenn die Aktualisierung automatisch erfolgt. Bei manuellen Indizes kann sogar die Aktualisierung eine Zeitlang dauern.

Denken Sie daran, abhängige Textindizes zu löschen, bevor Sie einen Massenimport wie eine INPUT-Anweisung für eine Tabelle vornehmen. Nachdem Sie die Daten geladen haben, erstellen Sie den Textindex neu.

Auswirkungen auf die Datenbank

Wenn Sie die INPUT-Anweisung verwenden, werden Änderungen im Transaktionslog aufgezeichnet. Im Fall eines Datenträgerausfalls steht eine detaillierte Aufzeichnung der Änderungen zur Verfügung. Beim Import großer Datenmengen mit dieser Methode sind allerdings Performance-Auswirkungen zu berücksichtigen, weil alle Zeilen in das Transaktionslog geschrieben werden.

Im Vergleich dazu speichert die LOAD TABLE-Anweisung nicht jede Zeile im Transaktionslog und kann daher schneller als die INPUT-Anweisung sein. Die INPUT-Anweisung unterstützt mehrere Datenbanken und Dateiformate.

Siehe auch

- „INPUT-Anweisung [Interactive SQL]“ [SQL Anywhere Server - SQL-Referenzhandbuch]
- „TRUNCATE-Anweisung“ [SQL Anywhere Server - SQL-Referenzhandbuch]
- „REFRESH MATERIALIZED VIEW-Anweisung“ [SQL Anywhere Server - SQL-Referenzhandbuch]
- „CREATE TEXT INDEX-Anweisung“ [SQL Anywhere Server - SQL-Referenzhandbuch]
- „DROP TEXT INDEX-Anweisung“ [SQL Anywhere Server - SQL-Referenzhandbuch]

Importieren von Daten mit der INPUT-Anweisung

Sie können Daten aus einer Textdatei oder aus einer kommagetrennten CSV-Datei mit Interactive SQL in eine Datenbank importieren.

Voraussetzungen

Sie müssen Eigentümer der Tabelle sein oder die folgenden Privilegien haben:

- INSERT-Privileg für die Tabelle oder INSERT ANY TABLE-Systemprivileg
- SELECT-Privileg für die Tabelle oder SELECT ANY TABLE-Systemprivileg

Kontext und Bemerkungen

Da die INPUT-Anweisung eine Interactive SQL-Anweisung ist, können Sie sie nicht in zusammengesetzten Anweisungen (wie z.B. einer IF-Anweisung) oder in gespeicherten Prozeduren verwenden.

Aufgabe

1. Erstellen Sie eine Textdatei namens *newSwimwear.csv* mit den folgenden Werten und speichern Sie sie im Verzeichnis *C:\LocalTemp*:

```
ID,Name,Description,Size,Color,Quantity,UnitPrice
800,Swimsuit,Lycra,Small,Blue,10,81.00
801,Swimsuit,Lycra,Medium,Blue,10,81.00
802,Swimsuit,Lycra,Large,Blue,7,85.00
```

2. Öffnen Sie Interactive SQL und stellen Sie eine Verbindung zur SQL Anywhere-Beispieldatenbank her.
3. Geben Sie eine INPUT-Anweisung im Fensterausschnitt **SQL-Anweisungen** ein.

```
INPUT INTO Products
FROM C:\LocalTemp\newSwimwear.csv
FORMAT TEXT
SKIP 1;
```

In dieser Anweisung ist der Name der Zieltabelle "Products" und *newSwimwear.csv* ist der Name der Datendatei. Die erste Zeile der Datei mit den Spaltennamen wird übersprungen. Die Datei befindet sich auf dem Clientcomputer.

4. Führen Sie die Anweisung aus.

Wenn der Import erfolgreich verläuft, wird auf der Registerkarte **Meldungen** die Dauer des Importvorgangs angezeigt. Wenn das Importieren fehlschlägt, werden Sie in einer Meldung über den Grund dafür informiert.

Ergebnisse

Die Daten werden in die angegebene Datenbank importiert.

Beispiel

Gehen Sie folgendermaßen vor, um Daten aus einer Excel CSV-Datei mit der INPUT-Anweisung einzugeben.

1. Speichern Sie in Excel die Daten aus Ihrer Excel-Datei in einer CSV-Datei. Benennen Sie die Datei z. B. *newSales.csv*.
2. Stellen Sie in Interactive SQL eine Verbindung mit einer SQL Anywhere-Datenbank her, beispielsweise mit der Beispieldatenbank.
3. Erstellen Sie eine Tabelle mit dem Namen **imported_sales** und fügen Sie die erforderlichen Spalten hinzu.
4. Führen Sie eine INPUT-Anweisung unter Verwendung der SKIP-Klausel aus, um die Spaltennamen zu überspringen, die Excel in die erste Zeile in der CSV-Datei schreibt.

```
INPUT INTO imported_sales FROM 'C:\\LocalTemp\\newSales.csv' SKIP 1;
```

Siehe auch

- „INPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Datenimport mit der LOAD TABLE-Anweisung

Verwenden Sie die LOAD TABLE-Anweisung, um Daten, die sich auf einem Datenbankserver oder einem Clientcomputer befinden, in eine vorhandene Tabelle im Text/ASCII-Format zu importieren.

Sie können die LOAD TABLE-Anweisung auch verwenden, um Daten aus einer Spalte von einer anderen Tabelle oder aus einem Wertausdruck zu importieren (z.B. aus den Ergebnissen einer Funktion oder Systemprozedur).

Die LOAD TABLE-Anweisung fügt Zeilen einer Tabelle hinzu, ohne welche zu ersetzen.

Das Laden mit der LOAD TABLE-Anweisung (ohne die Optionen WITH ROW LOGGING und WITH CONTENT LOGGING) ist bedeutend schneller als die Verwendung der INPUT-Anweisung.

Es werden keine Trigger bei Daten ausgelöst, die mit der LOAD TABLE-Anweisung geladen werden.

Hinweise zu materialisierten Ansichten

Bei Sofortansichten wird ein Fehler zurückgegeben, wenn Sie versuchen, einen Massenimport von Daten in eine Basistabelle durchzuführen. Sie müssen die Daten in der Ansicht erst kürzen und können erst dann den Massenimport vornehmen.

Bei manuellen Ansichten können Sie einen Massenimport von Daten in eine Basistabelle durchführen. Allerdings werden die Daten in der Ansicht veraltet, bis die nächste Aktualisierung erfolgt.

Denken Sie daher daran, eine Kürzung in abhängigen materialisierten Ansichten vorzunehmen, bevor Sie einen Massenimport wie z.B. eine LOAD TABLE-Anweisung in einer Tabelle versuchen. Nachdem Sie die Daten geladen haben, aktualisieren Sie die Ansicht.

Hinweise zu Textindizes

Bei Sofort-Textindizes kann das Aktualisieren eines Textindexes nach dem Massenimport wie etwa einer LOAD TABLE-Anweisung in der Basistabelle eine bestimmte Zeit in Anspruch nehmen, auch wenn die Aktualisierung automatisch erfolgt. Bei manuellen Indizes kann sogar die Aktualisierung eine Zeitlang dauern.

Denken Sie daran, abhängige Textindizes zu löschen, bevor Sie einen Massenimport wie eine LOAD TABLE-Anweisung für eine Tabelle vornehmen. Nachdem Sie die Daten geladen haben, erstellen Sie den Textindex neu.

Hinweise zu Datenbank-Wiederherstellung und Synchronisation

Wenn Daten von einer Datei geladen werden (z.B. `LOAD TABLE table-name FROM filename;`), wird standardmäßig nur die LOAD TABLE-Anweisung im Transaktionslog protokolliert, und nicht die tatsächlichen Datenzeilen, die geladen werden. Dies stellt ein Problem dar, wenn ein Wiederherstellen der Datenbank unter Verwendung des Transaktionslogs versucht wird, falls die ursprüngliche Datei geändert, verschoben oder gelöscht wurde. Es bedeutet auch, dass Datenbanken, die an einer Synchronisation oder Replikation beteiligt sind, die neuen Daten nicht erhalten.

Um die Schwierigkeiten bei der Wiederherstellung und Synchronisation zu beseitigen, gibt es bei der LOAD TABLE-Anweisung zwei Protokollierungsoptionen: `WITH ROW LOGGING`, wobei INSERT-Anweisungen im Transaktionslog für jede geladene Zeile erstellt werden, und `WITH CONTENT LOGGING`, wobei die geladenen Zeilen in Abschnitte gruppiert und die Abschnitte im Transaktionslog protokolliert werden. Diese Optionen ermöglichen eine Wiederholung eines Ladevorgangs, selbst wenn die Quelle der geladenen Daten nicht mehr verfügbar ist.

Hinweise zur Datenbankspiegelung

Wenn Ihre Datenbank an einer Spiegelung beteiligt ist, sollten Sie die LOAD TABLE-Anweisung nur mit Vorsicht verwenden. Wenn Sie beispielsweise Daten von einer Datei laden, bedenken Sie, ob die Datei zum Laden auf den Spiegelserver verfügbar ist oder ob sich die Daten in der Quelle, von der Sie laden, bis zu dem Zeitpunkt ändern werden, an dem die Spiegeldatenbank die Daten lädt. Wenn eines dieser Risiken besteht, sollten Sie vielleicht `WITH ROW LOGGING` oder `WITH CONTENT LOGGING` als Protokollierungsstufe in der LOAD TABLE-Anweisung angeben. Dadurch sind die Daten, die in die Spiegeldatenbank geladen werden, identisch mit den Daten, die in die gespiegelte Datenbank geladen wurden.

Siehe auch

- „CREATE TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DROP TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Zugriff auf Daten auf Clientcomputern“ auf Seite 760
- „Datenbankspiegelung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „INPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „LOAD TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „TRUNCATE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „REFRESH MATERIALIZED VIEW-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Importieren von Daten mit der INSERT-Anweisung

Verwenden Sie die INSERT-Anweisung, um der Datenbank Zeilen hinzuzufügen. Da die Importdaten für Ihre Zieltabelle in die INSERT-Anweisung aufgenommen werden, wird dies als interaktive Eingabe angesehen. Sie können die INSERT-Anweisung auch mit Fern Datenzugriff verwenden, um Daten aus einer anderen Datenbank und nicht aus einer Datei zu importieren.

Verwenden Sie die INSERT-Anweisung unter folgenden Umständen für den Datenimport:

- Sie wollen geringe Datenmengen in eine einzige Tabelle importieren.
- Sie sind im Hinblick auf die Dateiformate flexibel.
- Sie wollen entfernte Daten aus einer externen Datenbank und nicht aus einer Datei importieren.

Die INSERT-Anweisung bietet eine ON EXISTING-Klausel, um eine Aktion anzugeben, die ausgeführt wird, wenn eine einzufügende Zeile in der Zieltabelle gefunden wird. Wenn es jedoch 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.

Hinweise zu materialisierten Ansichten

Bei Sofortansichten wird ein Fehler zurückgegeben, wenn Sie versuchen, einen Massenimport von Daten in eine Basistabelle durchzuführen. Sie müssen die Daten in der Ansicht erst kürzen und können erst dann den Massenimport vornehmen.

Bei manuellen Ansichten können Sie einen Massenimport von Daten in eine Basistabelle durchführen. Allerdings werden die Daten in der Ansicht veraltet, bis die nächste Aktualisierung erfolgt.

Denken Sie daher daran, eine Kürzung in abhängigen materialisierten Ansichten vorzunehmen, bevor Sie einen Massenimport wie z.B. eine INSERT-Anweisung in einer Tabelle versuchen. Nachdem Sie die Daten geladen haben, aktualisieren Sie die Ansicht.

Hinweise zu Textindizes

Bei Sofort-Textindizes kann das Aktualisieren eines Textindexes nach dem Massenimport wie etwa einer INSERT-Anweisung in der Basistabelle eine bestimmte Zeit in Anspruch nehmen, auch wenn die

Aktualisierung automatisch erfolgt. Bei manuellen Indizes kann sogar die Aktualisierung eine Zeitlang dauern.

Denken Sie daran, abhängige Textindizes zu löschen, bevor Sie einen Massenimport wie eine INSERT-Anweisung für eine Tabelle vornehmen. Nachdem Sie die Daten geladen haben, erstellen Sie den Textindex neu.

Auswirkungen auf die Datenbank

Wenn Sie die INSERT-Anweisung verwenden, werden Änderungen im Transaktionslog aufgezeichnet. Im Fall eines Datenträgerausfalls, der die Datenbankdatei betrifft, können Sie anhand des Transaktionslogs Informationen über die von Ihnen durchgeführten Änderungen beziehen.

Siehe auch

- „Das Transaktionslog“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „MERGE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „INSERT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „TRUNCATE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DROP TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „LOAD TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „REFRESH MATERIALIZED VIEW-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „INPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Importieren von Daten mit der MERGE-Anweisung

Verwenden Sie die MERGE-Anweisung, um einen Aktualisierungsvorgang durchzuführen und große Mengen von Tabellendaten zu aktualisieren. Wenn Sie Daten zusammenführen, können Sie die Aktionen angeben, die durchgeführt werden sollen, wenn Zeilen aus den Quelldaten mit Zeilen in den Zieldaten übereinstimmen bzw. nicht übereinstimmen.

Zusammenführungsverhalten definieren

Das Folgende Syntax ist eine verkürzte Version der MERGE-Anweisung.

```
MERGE INTO target-object  
USING source-object  
ON merge-search-condition  
{ WHEN MATCHED | WHEN NOT MATCHED } [...]
```

Wenn die Datenbank einen Zusammenführungsprozess durchführt, vergleicht sie Zeilen im Quellobjekt (*source-object*) mit Zeilen im Zielobjekt (*target-object*), um Zeilen zu finden, die miteinander übereinstimmen bzw. nicht übereinstimmen, abhängig von der in der ON-Klausel enthaltenen Definition. 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.

Das Quellobjekt *source-object* kann eine Basistabelle, eine Ansicht, eine materialisierte Ansicht, eine abgeleitete Tabelle oder die Ergebnisse einer Prozedur sein. Das Zielobjekt *target-object* kann jedes dieser Objekte außer einer materialisierten Ansichten und Prozeduren sein.

Der ANSI SQL/2008-Standard lässt nicht zu, dass während eines Zusammenführungsvorgangs Zeilen im *target-object* mit mehr als einer Zeile im *source-object* aktualisiert werden.

Wenn eine Zeile im *source-object* als übereinstimmend oder nicht-übereinstimmend angesehen wird, wird sie anhand der übereinstimmenden bzw. nicht-übereinstimmenden WHEN-Klauseln (WHEN MATCHED oder WHEN NOT MATCHED) evaluiert. Eine WHEN MATCHED-Klausel legt die Aktion fest, die auf der Zeile im *target-object* durchgeführt wird (WHEN MATCHED ... UPDATE beispielsweise gibt eine Aktualisierung der Zeile im *target-object* an). Eine WHEN NOT MATCHED-Klausel definiert eine Aktion, die auf dem *target-object* unter Verwendung von nicht übereinstimmenden Zeilen des *source-object* durchzuführen ist.

Sie können beliebig viele WHEN-Klauseln angeben, die in der von Ihnen angegebenen Reihenfolge abgearbeitet werden. Sie können auch die AND-Klausel innerhalb der WHEN-Klausel verwenden, um Aktionen für eine Teilmenge der Zeilen anzugeben. Beispiel: Die folgenden WHEN-Klauseln legen verschiedene auszuführende Aktionen fest, abhängig vom Wert der Quantity-Spalte bei übereinstimmenden Zeilen:

```
WHEN MATCHED AND myTargetTable.Quantity<=500 THEN SKIP
WHEN MATCHED AND myTargetTable.Quantity>500 THEN UPDATE SET
myTargetTable.Quantity=500
```

Verzweigungen in einem Zusammenführungsvorgang

Das Gruppieren von übereinstimmenden und nicht-übereinstimmenden Zeilen wird als **Verzweigung** bezeichnet, und die einzelnen Gruppen werden **Zweige** genannt. Eine **Verzweigung** entspricht einer einzelnen WHEN MATCHED- oder WHEN NOT MATCHED-Klausel. Beispiel: Ein Zweig enthält die Menge von nicht-übereinstimmenden Zeilen aus dem *source-object*, die eingefügt werden muss. Die Ausführung der Zweigaktionen beginnt erst, nachdem alle Verzweigungsaktionen abgeschlossen wurden (alle Zeilen im *source-object* wurden ausgewertet). Der Datenbankserver beginnt mit der Ausführung der Zweigaktionen entsprechend der Reihenfolge, in der die WHEN-Klauseln angegeben wurden.

Wenn eine nicht übereinstimmende Zeile aus dem *source-object* oder ein Paar übereinstimmender Zeilen aus dem *source-object* und dem *target-object* in eine Verzweigung gesetzt werden, erfolgt keine weitere Auswertung gegenüber nachfolgenden Verzweigungen. Dadurch ist die Reihenfolge, in der Sie WHEN-Klauseln angeben, entscheidend.

Eine Zeile im *source-object*, die als Übereinstimmung oder Nicht-Übereinstimmung angesehen wird, die aber zu keiner Verzweigung gehört (d.h. sie erfüllt keine der WHEN-Klauseln), wird ignoriert. Das kann eintreten, wenn die WHEN-Klauseln AND-Klauseln enthalten und die Zeile keine der AND-Klauselbedingungen erfüllt. In diesem Fall wird die Zeile ignoriert, da für sie keine Aktion festgelegt ist.

Im Transaktionslog werden Aktionen, die Daten ändern, als einzelne INSERT-, UPDATE- und DELETE-Anweisungen protokolliert.

Für die Zieltabelle definierte Trigger

Trigger werden üblicherweise ausgelöst, wenn die einzelnen INSERT-, UPDATE- und DELETE-Anweisungen während des Zusammenführungsvorgangs ausgeführt werden. Beispiel: Bei der

Prozessverarbeitung eines Zweigs, für den eine UPDATE-Aktion festgelegt ist, führt der Datenbankserver Folgendes durch:

1. Er löst alle BEFORE UPDATE-Trigger aus.
2. Er führt die UPDATE-Anweisung auf der Kandidatenmenge von Zeilen durch, während er etwaige UPDATE-Trigger auf Zeilenebene auslöst.
3. Er löst die AFTER UPDATE-Trigger aus.

Trigger für die Zieltabelle *target-table* können während eines Zusammenführungsvorgangs Konflikte bewirken, wenn dieser sich auf Zeilen auswirkt, die für eine andere Verzweigung aktualisiert werden. Beispiel: Angenommen, eine Aktion wird in der Zeile A ausgeführt, wonach ein Trigger auslöst, der die Zeile B löscht. Für B ist allerdings eine Aktion definiert, die noch nicht durchgeführt wurde. Wenn eine Aktion nicht auf einer Zeile durchgeführt werden kann, schlägt der Zusammenführungsvorgang fehl, alle Änderungen werden zurückgesetzt und ein Fehler wird gemeldet.

Ein Trigger, für den mehr als eine Trigger-Aktion festgelegt ist, wird behandelt, als ob er einmal für jede der Trigger-Aktionen im selben Hauptteil angegeben wurde (was äquivalent mit der Definition von separaten Trigger, jeweils mit einer einzelnen Trigger-Aktion, ist).

Hinweise zu materialisierten Sofortansichten

Die Performance des Datenbankservers kann beeinträchtigt werden, wenn die MERGE-Anweisung eine große Anzahl von Zeilen aktualisiert. Um zahlreiche Zeilen zu aktualisieren, sollten Sie in Betracht ziehen, die Daten in abhängigen sofortigen materialisierten Ansichten zu kürzen, bevor Sie die MERGE-Anweisung für die Tabelle ausführen. Nachdem Sie die MERGE-Anweisung ausgeführt haben, führen Sie eine REFRESH MATERIALIZED VIEW-Anweisung aus.

Hinweise zu Textindizes

Die Performance des Datenbankservers kann beeinträchtigt werden, wenn die MERGE-Anweisung eine große Anzahl von Zeilen aktualisiert. Ziehen Sie daher in Betracht, Textindizes zu löschen, bevor Sie die MERGE-Anweisung für eine Tabelle ausführen. Nachdem Sie die MERGE-Anweisung ausgeführt haben, erstellen Sie den Textindex neu.

Siehe auch

- „MERGE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „REFRESH MATERIALIZED VIEW-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „TRUNCATE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DROP TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Beispiel 1

Angenommen, Sie haben ein kleines Unternehmen, das Jacken und Pullover vertreibt. Die Preise der Stoffe für die Jacken sind um 5% gestiegen, und Sie wollen Ihre Preise entsprechend anpassen. Unter Verwendung der folgenden CREATE TABLE-Anweisung erstellen Sie eine kleine Tabelle namens "myProducts", um aktuelle Preisinformationen für die von Ihnen verkauften Jacken und Pullover aufzunehmen. Die nachfolgenden INSERT-Anweisungen füllen "myProducts" mit Daten an. Für dieses Beispiel müssen Sie das CREATE TABLE-Privileg haben.


```

CREATE TABLE myProducts (
  product_id    NUMERIC(10),
  product_name  CHAR(20),
  product_size  CHAR(20),
  product_price NUMERIC(14,2));
INSERT INTO myProducts VALUES (1, 'Jacket', 'Small', 29.99);
INSERT INTO myProducts VALUES (2, 'Jacket', 'Medium', 29.99);
INSERT INTO myProducts VALUES (3, 'Jacket', 'Large', 39.99);
INSERT INTO myProducts VALUES (4, 'Sweater', 'Small', 18.99);
INSERT INTO myProducts VALUES (5, 'Sweater', 'Medium', 18.99);
INSERT INTO myProducts VALUES (6, 'Sweater', 'Large', 19.99);
SELECT * FROM myProducts;

```

product_id	product_name	product_size	product_price
1	Jacket	Small	29.99
2	Jacket	Medium	29.99
3	Jacket	Large	39.99
4	Sweater	Small	18.99
5	Sweater	Medium	18.99
6	Sweater	Large	19.99

Nun verwenden Sie die folgende Anweisung, um die Tabelle "myPrice" zu erstellen, um Informationen über die Preisänderungen bei Jacken aufzunehmen. Eine SELECT-Anweisung wird am Ende hinzugefügt, damit Sie den Inhalt der myPrice-Tabelle sehen können, bevor der Zusammenführungsvorgang durchgeführt wird.

```

CREATE TABLE myPrices (
  product_id    NUMERIC(10),
  product_name  CHAR(20),
  product_size  CHAR(20),
  product_price NUMERIC(14,2),
  new_price     NUMERIC(14,2));
INSERT INTO myPrices (product_id) VALUES (1);
INSERT INTO myPrices (product_id) VALUES (2);
INSERT INTO myPrices (product_id) VALUES (3);
INSERT INTO myPrices (product_id) VALUES (4);
INSERT INTO myPrices (product_id) VALUES (5);
INSERT INTO myPrices (product_id) VALUES (6);
SELECT * FROM myPrices;

```

product_id	product_name	product_size	product_price	new_price
1	(NULL)	(NULL)	(NULL)	(NULL)
2	(NULL)	(NULL)	(NULL)	(NULL)
3	(NULL)	(NULL)	(NULL)	(NULL)

product_id	product_name	product_size	product_price	new_price
4	(NULL)	(NULL)	(NULL)	(NULL)
5	(NULL)	(NULL)	(NULL)	(NULL)
6	(NULL)	(NULL)	(NULL)	(NULL)

Verwenden Sie die folgende MERGE-Anweisung, um Daten aus der myProducts-Tabelle in der myPrices-Tabelle zusammenzuführen. Beachten Sie, dass das *source-object* eine abgeleitete Tabelle ist, die gefiltert wurde, um nur jene Zeilen zu enthalten, bei denen product_name "Jacket" ist. Beachten Sie außerdem, dass die ON-Klausel angibt, dass Zeilen im *target-object* und im *source-object* übereinstimmen, wenn die Werte in ihren product_id-Spalten übereinstimmen.

```

MERGE INTO myPrices p
USING ( SELECT
        product_id,
        product_name,
        product_size,
        product_price
      FROM myProducts
      WHERE product_name='Jacket') pp
ON (p.product_id = pp.product_id)
WHEN MATCHED THEN
  UPDATE SET
    p.product_id=pp.product_id,
    p.product_name=pp.product_name,
    p.product_size=pp.product_size,
    p.product_price=pp.product_price,
    p.new_price=pp.product_price * 1.05;
SELECT * FROM myPrices;

```

product_id	product_name	product_size	product_price	new_price
1	Jacket	Small	29.99	31.49
2	Jacket	Medium	29.99	31.49
3	Jacket	Large	39.99	41.99
4	(NULL)	(NULL)	(NULL)	(NULL)
5	(NULL)	(NULL)	(NULL)	(NULL)
6	(NULL)	(NULL)	(NULL)	(NULL)

Die Spaltenwerte für product_id 4, 5 und 6 bleiben NULL, weil diese Produkte nicht zu einer der Zeilen in der Tabelle myProducts passen, deren Produkte (product_name= 'Jacket ') waren.

Beispiel 2

Im folgenden Beispiel werden Zeilen von den Tabellen mySourceTable und myTargetTable mithilfe der Primärschlüsselwerte von myTargetTable zur Suche nach übereinstimmenden Zeilen zusammengeführt.

Die Zeile wird als übereinstimmend angesehen, wenn eine Zeile in mySourceTable denselben Wert wie die Primärschlüsselspalte von myTargetTable hat.

```
MERGE INTO myTargetTable
  USING mySourceTable ON PRIMARY KEY
  WHEN NOT MATCHED THEN INSERT
  WHEN MATCHED THEN UPDATE;
```

Die WHEN NOT MATCHED THEN INSERT-Klausel gibt an, dass Zeilen, die in mySourceTable gefunden werden und in myTargetTable nicht zu finden sind, der myTargetTable hinzugefügt werden müssen. Die WHEN MATCHED THEN UPDATE-Klausel gibt an, dass übereinstimmende Zeilen von myTargetTable mit den Werten in mySourceTable aktualisiert werden.

Die nachstehende Syntax ist der obenstehenden Syntax gleichwertig. Sie setzt voraus, dass myTargetTable die Spalten (I1, I2, .. In) hat und der Primärschlüssel für die Spalten (I1, I2) definiert ist. Die mySourceTable hat die Spalten (U1, U2, .. Un).

```
MERGE INTO myTargetTable ( I1, I2, .. In )
  USING mySourceTable ON myTargetTable.I1 = mySourceTable.U1
  AND myTargetTable.I2 = mySourceTable.U2
  WHEN NOT MATCHED
    THEN INSERT ( I1, I2, .. In )
      VALUES ( mySourceTable.U1, mySourceTable.U2, ..., mySourceTable.Un )
  WHEN MATCHED
    THEN UPDATE SET
      myTargetTable.I1 = mySourceTable.U1,
      myTargetTable.I2 = mySourceTable.U2,
      ...
      myTargetTable.In = mySourceTable.Un;
```

Die RAISERROR-Aktion verwenden

Eine der Aktionen, die Sie für eine Übereinstimmungs- oder Nicht-Übereinstimmungsaktion angeben können, ist RAISERROR. RAISERROR ermöglicht es Ihnen, den Zusammenführungsvorgang fehlschlagen zu lassen, wenn die Bedingung einer WHEN-Klausel erfüllt wird.

Wenn Sie RAISERROR angeben, gibt der Datenbankservers 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.

Die Angabe eines benutzerdefinierten SQLCODE ist sinnvoll, wenn Sie später versuchen, die spezifischen Umstände zu ermitteln, die zu dem Fehler geführt haben.

Der benutzerdefinierte SQLCODE muss eine positive Ganzzahl größer als 17.000 sein und kann entweder als Zahl oder als Variable angegeben werden.

Die folgenden Anweisungen zeigen auf einfache Weise, wie sich das Anpassen eines benutzerdefinierten SQLCODE auf die Rückgabe auswirkt. Für dieses Beispiel müssen Sie das CREATE TABLE-Privileg haben.

Erstellen Sie folgendermaßen die Tabelle "targetTable":

```
CREATE TABLE targetTable( c1 int );
INSERT INTO targetTable VALUES( 1 );
```

Die folgende Anweisung gibt einen Fehler mit SQLSTATE = '23510' und SQLCODE = -1254 zurück:

```
MERGE INTO targetTable
  USING (SELECT 1 c1 ) AS sourceData
  ON targetTable.c1 = sourceData.c1
  WHEN MATCHED THEN RAISERROR;
SELECT sqlstate, sqlcode;
```

Die folgende Anweisung gibt einen Fehler mit SQLSTATE = '23510' und SQLCODE = -17001 zurück:

```
MERGE INTO targetTable
  USING (SELECT 1 c1 ) AS sourceData
  ON targetTable.c1 = sourceData.c1
  WHEN MATCHED THEN RAISERROR 17001
  WHEN NOT MATCHED THEN RAISERROR 17002;
SELECT sqlstate, sqlcode;
```

Die folgende Anweisung gibt einen Fehler mit SQLSTATE = '23510' und SQLCODE = -17002 zurück:

```
MERGE INTO targetTable
  USING (SELECT 2 c1 ) AS sourceData
  ON targetTable.c1 = sourceData.c1
  WHEN MATCHED THEN RAISERROR 17001
  WHEN NOT MATCHED THEN RAISERROR 17002;
SELECT sqlstate, sqlcode;
```

Tipps zum Importieren von Daten mit Proxy-Tabellen

Eine Proxy-Tabelle ist eine lokale Tabelle mit Metadaten, die verwendet wird, um auf eine Tabelle auf einem entfernten Datenbankserver so zuzugreifen, als handele es sich um eine lokale Tabelle. Damit können Sie Daten direkt importieren.

Verwenden Sie unter folgenden Umständen Proxy-Tabellen zum Datenimport:

- Sie haben Zugriff auf Ferndaten.
- Sie wollen Daten direkt aus einer anderen Datenbank importieren.

Auswirkungen auf die Datenbank

Wenn Sie Proxy-Tabellen für den Import verwenden, werden Änderungen im Transaktionslog aufgezeichnet. Im Fall eines Datenträgerausfalls, der die Datenbankdatei betrifft, können Sie anhand des Transaktionslogs Informationen über die von Ihnen durchgeführten Änderungen beziehen.

Verwendung von Proxy-Tabellen

Erstellen Sie eine Proxy-Tabelle und verwenden Sie eine INSERT-Anweisung mit einer SELECT-Klausel, um Daten aus der entfernten Datenbank in eine permanente Tabelle in Ihrer Datenbank einzufügen.

Siehe auch

- „Ferndatenzugriff“ auf Seite 785
- „INSERT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Konvertierungsfehler beim Import

Es kann vorkommen, dass die aus externen Quellen eingelesenen Daten fehlerhaft sind. Es gibt z.B. möglicherweise ungültige Datumsangaben und Nummern. Verwenden Sie die Datenbankoption `conversion_error`, um Konvertierungsfehler zu ignorieren und ungültige Werte in NULL-Werte zu konvertieren.

Siehe auch

- „`conversion_error`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „`SET OPTION`-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Importieren einer Tabelle (Interactive SQL-Importassistent)

Sie können Interactive SQL verwenden, um Daten aus einer Textdatei, aus einer anderen Tabelle in einer Datenbank oder aus einer Formdatei in eine Tabelle in Ihrer Datenbank zu importieren.

Voraussetzungen

Sie benötigen das `CREATE TABLE`-Privileg, um eine Tabelle zu erstellen, deren Eigentümer Sie sind, bzw. das `CREATE ANY TABLE`-Systemprivileg oder das `CREATE ANY OBJECT`-Systemprivileg, um eine Tabelle zu erstellen, deren Eigentümer andere Benutzer sind.

Aufgabe

1. Klicken Sie in Interactive SQL auf **Daten » Importieren**.
2. Klicken Sie auf **In einer Textdatei** und anschließend auf **Weiter**.
3. Im Feld **Dateiname** klicken Sie auf **Durchsuchen**, um die Datei hinzuzufügen.
4. Klicken Sie auf **In einer neuen Tabelle** und füllen Sie den **Tabellennamen** aus.
5. Klicken Sie auf **Weiter**.
6. Bei der Textdatei geben Sie an, wie sie gelesen wird, und klicken Sie auf **Weiter**.
7. Führen Sie Änderungen an den Spaltennamen und Datentypen durch und klicken Sie auf **Importieren**.
8. Klicken Sie auf **Schließen**.

Ergebnisse

Die Daten werden in die angegebene Tabelle importiert.

Importvorgang von Tabellen (SQL)

Sie können SQL verwenden, um Daten aus einer Textdatei, aus einer anderen Tabelle in einer Datenbank oder aus einer Formdatei in eine Tabelle in Ihrer Datenbank zu importieren.

Voraussetzungen

Sie benötigen das CREATE TABLE-Privileg, um eine Tabelle zu erstellen, deren Eigentümer Sie sind, bzw. das CREATE ANY TABLE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg, um eine Tabelle zu erstellen, deren Eigentümer andere Benutzer sind.

Welche Privilegien für das Importieren (Laden) von Daten erforderlich sind, hängt von den Einstellungen der Datenbankoption -gl ab sowie von der Datenquelle, aus der Sie die Daten importieren. Weitere Hinweise zu den für das Laden der Daten erforderlichen Privilegien finden Sie unter der LOAD TABLE-Anweisung.

Aufgabe

1. Verwenden Sie die Anweisung CREATE TABLE, um die Zieltabelle zu erstellen. Beispiel:

```
CREATE TABLE Departments (  
  DepartmentID      integer NOT NULL,  
  DepartmentName    char(40) NOT NULL,  
  DepartmentHeadID  integer NULL,  
  CONSTRAINT DepartmentsKey PRIMARY KEY (DepartmentID) );
```

2. Führen Sie die LOAD TABLE-Anweisung aus. Beispiel:

```
LOAD TABLE Departments  
FROM 'C:\\ServerTemp\\Departments.csv';
```

3. Um die nachgestellten Leerzeichen in Ihren Werten zu behalten, benutzen Sie die Anweisung STRIP OFF in Ihrer LOAD TABLE-Anweisung. Die Standardeinstellung (STRIP RTRIM) entfernt nachgestellte Leerzeichen aus Werten, bevor sie eingefügt werden.

Die Anweisung LOAD TABLE ersetzt die bestehenden Zeilen in der Tabelle nicht, sondern fügt den Inhalt der Datei den vorhandenen Zeilen der Tabelle hinzu. Sie können die Anweisung TRUNCATE TABLE verwenden, um alle Zeilen aus einer Tabelle zu entfernen.

Die FROM-Klausel gibt eine Datei auf dem Computer des Datenbankservers an.

Weder die Anweisung TRUNCATE TABLE noch die Anweisung LOAD TABLE lösen Trigger aus oder führen Aktionen zur referenziellen Integrität durch, wie z.B. kaskadierendes Löschen.

Ergebnisse

Die Daten werden in die angegebene Tabelle importiert.

Siehe auch

- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „LOAD TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „TRUNCATE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Tabellenstrukturen für den Datenimport

Die Struktur der Quelldaten braucht nicht mit der Struktur der Zieltabelle übereinzustimmen. Beispiel: Die Spalten-Datentypen können verschieden oder in unterschiedlicher Reihenfolge sein, bzw. es kann zusätzliche Werte in den zu importierenden Daten geben, die nicht zu Spalten in der Zieltabelle passen.

Tabelle oder Daten neu strukturieren

Wenn Sie wissen, dass die Struktur der zu importierenden Daten nicht der Struktur der Zieltabelle entspricht, haben Sie folgende Möglichkeiten:

- Sie stellen eine Liste mit Spaltennamen bereit, die in die Anweisung `LOAD TABLE` geladen wird.
- Sie ordnen die Importdaten mithilfe einer Variante der `INSERT`-Anweisung und einer globalen temporären Tabelle neu an, damit sie in die Tabelle passen.
- Sie verwenden die `INPUT`-Anweisung, um eine bestimmte Menge oder Reihenfolge von Spalten anzugeben.

Spalten mit NULL zulassen

Wenn die zu importierende Datei Daten für eine Teilmenge der Spalten in einer Tabelle enthält oder die Spalten in einer anderen Reihenfolge erscheinen, können Sie auch die `DEFAULTS`-Option der `LOAD TABLE`-Anweisung verwenden, um die Leerstellen zu füllen und nicht zueinander passende Tabellenstrukturen zusammenzuführen.

- Wenn `DEFAULTS` auf `OFF` gesetzt ist, erhalten alle Spalten, die in der Spaltenliste nicht vorhanden sind, `NULL`. 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 `DEFAULTS` auf `ON` gesetzt ist und die Spalte einen Standardwert hat, wird dieser Wert benutzt.

Sie können beispielsweise einen Standardwert für die Spalte "City" in der Tabelle "Customers" definieren und dann neue Zeilen in die Tabelle "Customers" aus einer Datei namens *newCustomers.csv* in `C:\ServerTemp` auf dem Computer des Datenbankservers laden, indem Sie eine `LOAD TABLE`-Anweisung wie die folgende verwenden:

```
ALTER TABLE Customers
ALTER City DEFAULT 'Waterloo';

LOAD TABLE Customers ( Surname, GivenName, Street, State, Phone )
FROM 'C:\ServerTemp\newCustomers.csv'
DEFAULTS ON;
```

Da für die Spalte "City" kein Wert angegeben wird, wird der Standardwert eingesetzt. Falls `DEFAULTS OFF` angegeben wurde, würde eine leere Zeichenfolge in die Spalte "City" eingesetzt.

Zusammenführen unterschiedlicher Tabellenstrukturen

Verwenden Sie eine Variation der `INSERT`-Anweisung und eine globale temporäre Tabelle, um die Importdaten so anzuordnen, dass sie in die Tabelle passen.

Voraussetzungen

Wenn Sie eine globale temporäre Tabelle erstellen möchten, müssen Sie eines der folgenden Systemprivilegien haben:

- CREATE TABLE
- CREATE ANY TABLE
- CREATE ANY OBJECT

Welche Privilegien für das Importieren (Laden) von Daten erforderlich sind, hängt von den Einstellungen der Datenbankoption -gl ab sowie von der Datenquelle, aus der Sie die Daten importieren. Weitere Hinweise zu den für das Laden der Daten erforderlichen Privilegien finden Sie unter der LOAD TABLE-Anweisung.

Um die INSERT-Anweisung verwenden zu können, müssen Sie Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- INSERT ANY TABLE-Systemprivileg
- INSERT-Privileg für die Tabelle

Wenn die ON EXISTING UPDATE-Klausel angegeben wird, müssen Sie außerdem das UPDATE ANY TABLE-Systemprivileg oder das UPDATE-Privileg für die Tabelle haben.

Aufgabe

1. Erstellen Sie im Fensterausschnitt **SQL-Anweisungen** eine globale temporäre Tabelle, deren Struktur zur Struktur der Eingabedatei passt.

Sie können die Anweisung CREATE TABLE verwenden, um die globale temporäre Tabelle zu erstellen.

2. Benutzen Sie die Anweisung LOAD TABLE, um Ihre Daten in die globale temporäre Tabelle einzulesen.

Wenn Sie eine Datenbankverbindung schließen, verschwinden die Daten in der globalen temporären Tabelle. Die Tabellendefinition bleibt jedoch erhalten. Sie können sie das nächste Mal verwenden, wenn Sie Verbindung zur Datenbank herstellen.

3. Mit der INSERT-Anweisung und einer SELECT-Klausel extrahieren und fassen Sie die Daten aus der temporären Tabelle zusammen und kopieren die Daten in eine oder mehrere permanente Tabellen der Datenbank.

Ergebnisse

Die Daten werden in eine permanente Datenbanktabelle geladen.

Beispiel

Im Folgenden finden Sie ein Beispiel für die oben angegebenen Schritte.

```
CREATE GLOBAL TEMPORARY TABLE TempProducts  
(
```



```

        ID                integer NOT NULL,
        Name              char(15) NOT NULL,
        Description        char(30) NOT NULL,
        Size              char(18) NOT NULL,
        Color             char(18) NOT NULL,
        Quantity          integer NOT NULL,
        UnitPrice          numeric(15,2) NOT NULL,
        CONSTRAINT ProductsKey PRIMARY KEY (ID)
    )
    ON COMMIT PRESERVE ROWS;

LOAD TABLE TempProducts
FROM 'C:\\ServerTemp\\newProducts.csv'
SKIP 1;

INSERT INTO Products WITH AUTO NAME
    (SELECT Name, Description, ID, Size, Color, Quantity,
        UnitPrice * 1.25 AS UnitPrice
    FROM TempProducts);

```

Der Preis der Artikel in der globalen temporären Tabelle werden um 25 % nach oben korrigiert, bevor die Zeilen in die Produkttabelle eingefügt werden.

Siehe auch

- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „LOAD TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „INSERT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Datenexport

Der Datenexport ist eine Verwaltungsaufgabe, bei der Sie Daten aus Ihrer Datenbank herausschreiben. Der Datenexport ist dann sinnvoll, wenn Sie große Teile Ihrer Datenbank in anderen Systemen verwenden oder Teile der Datenbank nach bestimmten Kriterien extrahieren wollen. Verwenden Sie SQL Anywhere, um Folgendes durchzuführen:

- Einzelne Tabellen, Abfrageergebnisse oder Tabellenschemata exportieren
- Skripten erstellen, die das Exportieren automatisieren, wodurch Sie mehrere Tabellen nacheinander exportieren können
- Viele verschiedene Dateiformate exportieren
- Sie möchten Daten in eine Datei auf einem Clientcomputer exportieren.
- Dateien zwischen SQL Anywhere und Adaptive Server Enterprise mithilfe der BCP FORMAT-Klausel übertragen

Bevor Sie Daten exportieren, ermitteln Sie, welche Ressourcen Ihnen zur Verfügung stehen und welchen Typ von Informationen Sie aus Ihrer Datenbank exportieren wollen.

Wenn Sie eine komplette Datenbank exportieren, sollten Sie die Datenbank aus Gründen der Performance entladen statt die Daten zu exportieren.

Export-Einschränkungen

Wenn Daten aus einer SQL Anywhere-Datenbank in eine Excel-Datenbank mit dem Microsoft Excel ODBC-Treiber exportiert werden, können folgende Datentypänderungen auftreten:

- Wenn Sie Daten exportieren, die den CHAR-, LONG VARCHAR-, NCHAR-, NVARCHAR- oder LONG NVARCHAR-Datentyp haben, werden diese als VARCHAR-Datentyp gespeichert (der ähnlichste vom Excel-Treiber unterstützte Datentyp).

Der Microsoft Excel ODBC-Treiber unterstützt Textspaltenbreiten bis zu 255 Zeichen.

- Daten, die als MONEY- und SMALLMONEY-Datentyp gespeichert sind, werden mit dem CURRENCY-Datentyp exportiert. Ansonsten werden numerische Daten als Zahlen exportiert.

Siehe auch

- „UNLOAD-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „OUTPUT-Anweisung [Interactive SQL]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Performance-Aspekte von Massenvorgängen“ auf Seite 721
- „Konfigurieren Sie die Behandlung von NULL-Werten in Interactive SQL“ auf Seite 755
- „Zugriff auf Daten auf Clientcomputern“ auf Seite 760
- „Neuaufbau von Datenbanken“ auf Seite 763

Exportieren von Daten mit dem Exportassistenten

Verwenden Sie den **Export-Assistenten** in Interactive SQL, um Abfrageergebnisse in einem bestimmten Format in eine Datei oder Datenbank zu exportieren.

Voraussetzungen

Sie müssen Eigentümer der Tabelle sein, die Sie abfragen, das SELECT-Privileg für die Tabelle haben oder das SELECT ANY TABLE-Systemprivileg haben.

Aufgabe

1. Führen Sie eine Abfrage aus.
2. Klicken Sie in Interactive SQL auf **Daten » Exportieren**.
3. Befolgen Sie die Anweisungen im **Export-Assistenten**.

Ergebnisse

Die Abfrageergebnisse werden in die angegebene Datei oder Datenbank exportiert.

Beispiel

1. Führen Sie die folgende Abfrage aus, während Sie mit der Beispieldatenbank verbunden sind. Sie müssen das SELECT-Privileg für die Tabelle "Employees" oder das SELECT ANY TABLE-Systemprivileg haben.

```
SELECT * FROM Employees WHERE State = 'GA';
```

2. Die Ergebnismenge enthält eine Liste aller Mitarbeiter, die in Georgia leben.
3. Klicken Sie auf **Daten » Exportieren**.
4. Klicken Sie auf **In einer Datenbank** und anschließend auf **Weiter**.
5. Klicken Sie in der Liste **Datenbanktyp** auf **UltraLite**.
6. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
7. Im Feld **Kennwort** geben Sie **sql** ein.
8. Im Feld **Datenbankdatei** geben Sie *C:\Users\Public\Documents\SQL Anywhere 16\Samples\UltraLite\CustDB\custdb.udb* ein.
9. Klicken Sie auf **Weiter**.
10. Klicken Sie auf **Neue Tabelle erstellen**.
11. In das Feld **Tabellenname** geben Sie **GAEmployees** ein .
12. Klicken Sie auf **Exportieren**.
13. Klicken Sie auf **Schließen**.
14. Klicken Sie auf **SQL » Vorherige SQL-Anweisung**.

Die OUTPUT USING-Anweisung, die vom **Export-Assistenten** erstellt und verwendet wurde, wird im Fensterausschnitt **SQL-Anweisungen** eingeblendet:

```
-- Vom Export-Assistenten generiert
OUTPUT USING 'driver=UltraLite 16;UID=DBA;PWD=***;
DBF=C:\\Users\\Public\\Documents\\SQL Anywhere 16\\Samples\\Ultralite\\
\\custdb\\custdb.udb'
      INTO "GAEmployees"
      CREATE TABLE ON
```

Tipps zum Exportieren von Daten mit der OUTPUT-Anweisung

Verwenden Sie die OUTPUT-Anweisung, um Abfrageergebnisse, Tabellen oder Ansichten aus Ihrer Datenbank zu exportieren.

Die OUTPUT-Anweisung ist nützlich, wenn Kompatibilität gewünscht wird, da sie die Ergebnismenge einer SELECT-Anweisung in mehrere verschiedene Dateiformate ausschreiben kann. Sie können ein Standard-Ausgabeformat verwenden oder das Dateiformat bei jeder OUTPUT-Anweisung festlegen. Interactive SQL kann eine SQL-Skriptdatei mit mehreren OUTPUT-Anweisungen ausführen.

Das Standard-Ausgabeformat für Interactive SQL wird im Fenster **Optionen** von Interactive SQL auf der Registerkarte **Import/Export** festgelegt. (Klicken Sie in Interactive SQL auf **Extras » Optionen**, um darauf zuzugreifen.)

Verwenden Sie die Interactive SQL-Anweisung OUTPUT, wenn Sie folgende Aufgaben durchführen wollen:

- Eine Tabelle bzw. Ansicht in einem Nicht-Textformat teilweise oder vollständig exportieren
- Den Exportvorgang durch eine SQL-Skriptdatei automatisieren

Auswirkungen auf die Datenbank

Wenn Sie die Wahl zwischen den Anweisungen OUTPUT, UNLOAD und UNLOAD TABLE haben, wählen Sie aus Gründen der Performance die Anweisung UNLOAD TABLE.

Der Export großer Datenmengen mit der OUTPUT-Anweisung kann die Performance signifikant verschlechtern. Führen Sie die OUTPUT-Anweisung auf demselben Computer aus, auf dem auch der Server läuft, um den Transport großer Datenvolumina über das Netzwerk zu vermeiden.

Siehe auch

- „OUTPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Exportieren von Daten in eine Excel-Datei

In Interactive SQL können Sie Daten aus Ihrer Datenbank mit der OUTPUT-Anweisung in eine Excel-Datei exportieren.

Voraussetzungen

Sie müssen das SELECT-Privileg für die Tabelle oder das SELECT ANY TABLE-Systemprivileg haben.

Wenn der Excel-ODBC-Treiber ein 32-Bit-Treiber ist, müssen Sie eine 32-Bit Version von Interactive SQL verwenden, um einen Architekturübereinstimmungsfehler zu vermeiden.

Aufgabe

1. Stellen Sie in Interactive SQL eine Verbindung mit einer SQL Anywhere-Datenbank her.
2. Führen Sie eine OUTPUT-Anweisung unter Verwendung der READONLY-Klausel aus. Beispiel:

```
SELECT * FROM SalesOrders;  
OUTPUT USING 'Driver=Microsoft Excel Driver  
(*.xls);DBQ=sales.xls;READONLY=0'  
INTO "newSalesData";
```

Eine neue Excel-Datei mit dem Namen *sales.xls* wird erstellt. Sie enthält die Tabelle mit dem Namen "newSalesData".

Ergebnisse

Die Daten werden in die angegebene Excel-Datei exportiert.

Beispiel

Das folgende Beispiel exportiert Daten aus der Employees-Tabelle in der Beispieldatenbank in eine neue Tabelle in einer Datenbank namens *demo2*. Die für die Verbindung mit der zweiten Datenbank

verwendete Benutzer-ID muss das CREATE TABLE-Systemprivileg, das CREATE ANY TABLE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben.

```
SELECT * FROM Employees;
OUTPUT USING 'DRIVER=SQL Anywhere 16;UID=DBA;PWD=sql;DBN=demo2;CON='''''
INTO "newEmployees"
CREATE TABLE ON;
```

Siehe auch

- „OUTPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Exportieren von Daten in eine CSV-Datei

In Interactive SQL können Sie Daten aus Ihrer Datenbank mit der OUTPUT-Anweisung in eine CSV-exportieren.

Voraussetzungen

Sie müssen das SELECT ANY TABLE-Systemprivileg oder das SELECT-Privileg für die Tabelle haben.

Aufgabe

1. Stellen Sie in Interactive SQL eine Verbindung mit einer SQL Anywhere-Datenbank her.
2. Führen Sie die OUTPUT-Anweisung mit den Klauseln FORMAT TEXT, QUOTE "" und WITH COLUMN NAMES aus, um ein Format mit Kommatrennzeichen mit den Spaltennamen in der ersten Zeile der Datei zu erstellen. Zeichenfolgenwerte werden zwischen Anführungszeichen gesetzt.
Beispiel:

```
SELECT * FROM SalesOrders;
OUTPUT TO 'C:\\LocalTemp\\newSales.csv'
FORMAT TEXT
QUOTE ''
WITH COLUMN NAMES;
```

Ergebnisse

Die Daten werden in die angegebene CSV-Datei exportiert.

Beispiel

Das folgende Beispiel exportiert die Daten aus der Tabelle "Employees" in der SQL Anywhere-Beispieldatenbank in die *Employees.csv*-Datei namens *C:\\LocalTemp*.

```
SELECT * FROM Employees;
OUTPUT TO C:\\LocalTemp\\Employees.csv
FORMAT TEXT;
```

Siehe auch

- „OUTPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Tipps zum Exportieren von Daten mit der UNLOAD TABLE-Anweisung

Mit der UNLOAD TABLE-Anweisung können Sie Daten effizient nur in Text-Formaten exportieren. Die UNLOAD TABLE-Anweisung exportiert jeweils eine Zeile pro Dateizeile, wobei die Werte mit einem Komma voneinander getrennt werden. Um anschließend das Einlesen zu beschleunigen, werden die Daten in der Reihenfolge der Primärschlüssel exportiert.

Verwenden Sie die Anweisung UNLOAD TABLE in folgenden Situationen:

- Sie möchten ganze Tabellen im Textformat exportieren.
- Die Performance der Datenbank ist Ihnen wichtig.
- Sie möchten Daten in eine Datei auf einem Clientcomputer exportieren.

Für die UNLOAD TABLE-Anweisung benötigen Sie die erforderlichen Privilegien. Das SELECT ANY TABLE-Systemprivileg reicht beispielsweise in der Regel aus, es sei denn, die Datenbankserveroption -gl ist auf NONE gesetzt.

Weitere Hinweise darüber, wer die UNLOAD TABLE-Anweisung verwenden kann, finden Sie unter [„Datenbankserveroption -gl“ \[SQL Anywhere Server - Datenbankadministration\]](#).

Auswirkungen auf die Datenbank

Wenn Sie die Wahl zwischen den Anweisungen OUTPUT, UNLOAD und UNLOAD TABLE haben, wählen Sie aus Gründen der Performance die Anweisung UNLOAD TABLE.

Die UNLOAD TABLE-Anweisung setzt eine Exklusivsperrung auf die gesamte Tabelle, während Sie sie entladen.

Beispiel

In der SQL Anywhere-Beispieldatenbank können Sie die Tabelle "Employees" in eine Textdatei namens *Employees.csv* entladen, indem Sie die folgende Anweisung ausführen:

```
UNLOAD TABLE Employees TO 'C:\\ServerTemp\\Employees.csv';
```

Mit dieser Form der UNLOAD TABLE-Anweisung ist der Pfad relativ zum Computer des Datenbankservers.

Siehe auch

- „Zugriff auf Daten auf Clientcomputern“ auf Seite 760
- „UNLOAD-Anweisung“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- „OUTPUT-Anweisung [Interactive SQL]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)

Tipps zum Exportieren von Daten mit der UNLOAD-Anweisung

Die UNLOAD-Anweisung ist der OUTPUT-Anweisung ähnlich, weil beide Abfrageergebnisse in eine Datei exportieren. Die UNLOAD-Anweisung exportiert Daten allerdings schneller in einem Textformat. Die UNLOAD-Anweisung exportiert jeweils eine Zeile pro Dateizeile, wobei die Werte mit einem Komma voneinander getrennt werden.

Verwenden Sie die UNLOAD-Anweisung in folgenden Fällen zum Entladen von Daten:

- Sie wollen Abfrageergebnisse mit geringster Auswirkung auf die Performance exportieren.
- Sie wollen Ausgabedaten im Textformat speichern.
- Sie wollen eine Exportanweisung in eine Anwendung einbetten.
- Sie möchten Daten in eine Datei auf einem Clientcomputer exportieren.

Um die UNLOAD-Anweisung mit SELECT verwenden zu können, müssen Sie die erforderlichen Privilegien haben. Das SELECT ANY TABLE-Systemprivileg reicht beispielsweise in der Regel aus, es sei denn, die Datenbankserveroption -gl ist auf NONE gesetzt. Sie benötigen zumindest die Berechtigungen zum Ausführen der SELECT-Anweisung für die Tabelle, die in der UNLOAD-Anweisung angegeben ist.

Weitere Hinweise darüber, wer die UNLOAD-Anweisung verwenden kann, finden Sie unter [„Datenbankserveroption -gl“ \[SQL Anywhere Server - Datenbankadministration\]](#).

Auswirkungen auf die Datenbank

Wenn Sie die Wahl zwischen den Anweisungen OUTPUT, UNLOAD und UNLOAD TABLE haben, wählen Sie aus Gründen der Performance die Anweisung UNLOAD TABLE.

Die UNLOAD-Anweisung mit SELECT wird in der aktuellen Isolationsstufe ausgeführt.

Beispiel

In der SQL Anywhere-Beispieldatenbank können Sie eine Teilmenge der Tabelle "Employees" in eine Textdatei namens *GAEmployees.csv* entladen, indem Sie die folgende Anweisung ausführen:

```
UNLOAD
SELECT * FROM Employees
WHERE State = 'GA'
TO 'C:\\ServerTemp\\GAEmployees.csv'
QUOTE ' ';
```

Mit dieser Form der UNLOAD TABLE-Anweisung ist der Pfad relativ zum Computer des Datenbankservers.

Siehe auch

- „Zugriff auf Daten auf Clientcomputern“ auf Seite 760
- „UNLOAD-Anweisung“ [SQL Anywhere Server - SQL-Referenzhandbuch]
- „OUTPUT-Anweisung [Interactive SQL]“ [SQL Anywhere Server - SQL-Referenzhandbuch]

Tipps zum Exportieren von Daten mit dem Dienstprogramm Entladen (dbunload)

Verwenden Sie das Dienstprogramm Entladen (dbunload), um eine, mehrere oder alle Datenbanktabellen zu exportieren. Sie können Tabellendaten und Tabellenschemas exportieren. Um Ihre Datenbanktabellen neu zu ordnen, können Sie auch das Dienstprogramm Entladen (dbunload) verwenden, um die erforderlichen SQL-Skriptdateien zu erstellen und, falls erforderlich, zu ändern. Diese Dateien können benutzt werden, um identische Tabellen in unterschiedlichen Datenbanken zu erstellen. Sie können Tabellen nur mit der Struktur, nur mit Daten oder mit Struktur und Daten entladen. Sie können auch die Option -ac verwenden, um direkt in eine vorhandene Datenbank zu entladen.

Verwenden Sie dbunload in den folgenden Fällen:

- Sie müssen die Datenbank neu aufbauen oder extrahieren.
- Sie möchten Daten im Textformat exportieren.
- Sie müssen große Datenmengen rasch verarbeiten.
- Sie sind bei den Anforderungen an Dateiformate flexibel.

Hinweis

Das Dienstprogramm Entladen (dbunload) ist mit dem **Assistenten zum Entladen einer Datenbank** von Sybase Central funktional gleichwertig. Sie können beide verwenden und erhalten dabei dieselben Ergebnisse.

Siehe auch

- „Dienstprogramm zum Entladen (dbunload)“ [SQL Anywhere Server - Datenbankadministration]

Tipps zum Exportieren von Daten mit dem Assistenten zum Entladen einer Datenbank

Verwenden Sie den **Assistenten zum Entladen einer Datenbank**, um eine vorhandene Datenbank in eine neue Datenbank zu entladen.

Wenn Sie den **Assistenten zum Entladen einer Datenbank** benutzen, können Sie entweder alle Objekte der Datenbank oder eine Teilmenge der Tabellen in der Datenbank entladen. Es werden nur Tabellen der Benutzer, die im Fenster **Eigentümerfilter konfigurieren** ausgewählt wurden, im **Assistenten zum Entladen einer Datenbank** angezeigt. Um Tabellen anzuzeigen, die Eigentum eines bestimmten Datenbankbenutzers sind, rechtsklicken Sie auf die zu entladende Datenbank, klicken Sie auf

Eigentümerfilter konfigurieren und wählen Sie im daraufhin geöffneten Fenster den gewünschten Benutzer aus.

Sie können auch den **Assistenten zum Entladen einer Datenbank** verwenden, um eine komplette Datenbank im kommagetrennten Textformat zu entladen und die erforderlichen SQL-Skriptdateien zu erstellen, mit denen die Datenbank komplett neu aufgebaut wird. Dies ist hilfreich, wenn Sie SQL Remote-Extraktionen erstellen oder neue Kopien Ihrer Datenbank mit derselben oder einer leicht geänderten Struktur aufbauen möchten. Der **Assistent zum Entladen einer Datenbank** ist für den Export von SQL Anywhere-Dateien bestimmt, die in SQL Anywhere wieder verwendet werden sollen.

Der **Assistent zum Entladen der Datenbank** bietet auch die Möglichkeit, die Daten in eine vorhandene oder in eine neue Datenbank, und nicht in eine Reload-Datei, zu laden.

Hinweis

Das Dienstprogramm Entladen (dbunload) ist mit dem **Assistenten zum Entladen einer Datenbank** funktional gleichwertig. Sie können beide verwenden und erhalten dabei dieselben Ergebnisse.

Siehe auch

- „Dienstprogramm zum Entladen (dbunload)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Entladen einer Datenbankdatei oder einer laufenden Datenbank

Sie können eine gestoppte oder laufende Datenbank in Sybase Central mithilfe des **Assistenten zum Entladen einer Datenbank** entladen.

Voraussetzungen

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.

Folgendes gilt beim Entladen einer Datei auf einem Clientcomputer:

- Sie müssen das WRITE CLIENT FILE-Privileg haben:
- Sie müssen Schreibberechtigung für das Verzeichnis haben, in dem sich die Datei befindet.
- Die Datenbankoption allow_write_client_file muss aktiviert sein.
- Die Sicherheitsfunktion write_client_file muss aktiviert sein.

Kontext und Bemerkungen

Hinweis

Wenn Sie nur Tabellen entladen, werden die Benutzer-IDs, die Eigentümer der Tabellen sind, nicht entladen. Sie müssen diese Benutzer-IDs in der neuen Datenbank erstellen, bevor Sie die Tabellen neu laden.

Aufgabe

1. Klicken Sie auf **Extras » SQL Anywhere 16 » Datenbank entladen**.
2. Befolgen Sie die Anweisungen im **Assistenten zum Entladen einer Datenbank**.

Ergebnisse

Die angegebene Datenbank wird entladen.

Siehe auch

- „UNLOAD-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Exportieren von Daten mit dem Fenster Daten entladen

Sie können in Sybase Central das Fenster **Daten entladen** verwenden, um Tabellen zu entladen.

Voraussetzungen

Sie müssen Eigentümer der Tabelle sein, das SELECT-Privileg für die Tabelle haben oder das SELECT ANY TABLE-Systemprivileg haben.

Kontext und Bemerkungen

Sie können das Fenster **Daten entladen** in Sybase Central verwenden, um eine oder mehrere Tabellen in einer Datenbank zu entladen. Auch der **Assistent zum Entladen einer Datenbank** sowie das Dienstprogramm Entladen (dbunload) stellen diese Funktionalität zur Verfügung, aber dieses Dialogfeld ermöglicht es Ihnen, Tabellen in einem Schritt zu entladen, anstatt den **Assistenten zum Entladen der Datenbank** in seiner vollen Länge auszuführen.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf **Tabellen**.
3. Rechtsklicken Sie auf die Tabelle, aus der Sie Daten exportieren möchten, und klicken Sie **Daten entladen**.
4. Machen Sie Ihre Eingaben im Fenster **Daten entladen**. Klicken Sie auf **OK**.

Ergebnisse

Die Daten werden in der angegebenen Datei gespeichert.

Exportieren von Abfrageergebnissen mit der OUTPUT-Anweisung

Sie können Abfrageergebnisse in Interactive SQL mit der OUTPUT-Anweisung exportieren.

Voraussetzungen

Sie müssen Eigentümer der Tabelle sein, die Sie abfragen, das SELECT-Privileg für die Tabelle haben oder das SELECT ANY TABLE-Systemprivileg haben.

Kontext und Bemerkungen

Sie können die Klauseln APPEND und VERBOSE kombinieren, um Ergebnisse und Meldungen am Ende einer bestehenden Datei einzufügen. Geben Sie beispielsweise **OUTPUT TO 'filename' APPEND VERBOSE** ein.

Die OUTPUT-Anweisung mit den Klauseln APPEND und VERBOSE entspricht den Operatoren >#, >>#, >& und >>& früherer Versionen von Interactive SQL. Sie können diese Operatoren benutzen, um Daten umzuleiten. Die Interactive SQL-Anweisungen ermöglichen aber präzisere Ausgabeanweisungen und machen den Programmcode leichter lesbar.

Verwenden Sie die BCP FORMAT-Klausel, um Dateien zwischen SQL Anywhere und Adaptive Server Enterprise zu importieren und zu exportieren.

Aufgabe

1. Geben Sie Ihre Abfrage im Fenster **SQL-Anweisungen** von Interactive SQL ein.
2. Am Ende der Abfrage geben Sie folgende Information ein: **OUTPUT TO 'filename'**.

Um beispielsweise die gesamte Tabelle "Employees" in die Datei *Employees.csv* zu exportieren, geben Sie folgende Abfrage ein:

```
SELECT * FROM Employees;
OUTPUT TO 'C:\\LocalTemp\\Employees.csv';
```

3. Um Abfrageergebnisse zu exportieren und die Ergebnisse an eine andere Datei anzuhängen, verwenden Sie die APPEND-Klausel. Beispiel:

```
SELECT * FROM Employees;
OUTPUT TO 'C:\\LocalTemp\\Employees.csv'
APPEND;
```

Um Abfrageergebnisse zu exportieren und Meldungen einzubeziehen, verwenden Sie die VERBOSE-Klausel. Beispiel:

```
SELECT * FROM Employees;  
OUTPUT TO 'C:\\LocalTemp\\Employees.csv'  
VERBOSE;
```

4. Klicken Sie auf **SQL » Ausführen**.

Wenn der Export erfolgreich verläuft, werden auf der Registerkarte **Meldungen** folgende Informationen angezeigt: Dauer des Datenexports, Dateiname und Pfad der exportierten Daten, Anzahl der geschriebenen Zeilen. Wenn das Exportieren fehlschlägt, werden Sie ebenfalls in einer Meldung darüber informiert.

Ergebnisse

Die Abfrageergebnisse werden an den angegebenen Speicherort exportiert.

Siehe auch

- „OUTPUT-Anweisung [Interactive SQL]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Adaptive Server Enterprise-Kompatibilität“ auf Seite 784

Exportieren von Abfrageergebnissen mit der UNLOAD-Anweisung

Sie können Abfrageergebnisse in Interactive SQL mit der UNLOAD-Anweisung exportieren.

Voraussetzungen

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.

Folgendes gilt beim Entladen einer Datei auf einem Clientcomputer:

- Sie müssen das WRITE CLIENT FILE-Privileg haben:
- Sie müssen Schreibberechtigung für das Verzeichnis haben, in dem sich die Datei befindet.
- Die Datenbankoption allow_write_client_file muss aktiviert sein.
- Die Sicherheitsfunktion write_client_file muss aktiviert sein.

Kontext und Bemerkungen

Verwenden Sie die BCP FORMAT-Klausel, um Dateien zwischen SQL Anywhere und Adaptive Server Enterprise zu importieren und zu exportieren.

Aufgabe

- Führen Sie im Fensterausschnitt **SQL-Anweisungen** die UNLOAD-Anweisung aus: Beispiel:

```
UNLOAD
SELECT * FROM Employees
TO 'C:\\ServerTemp\\Employees.csv';
```

Wenn der Export erfolgreich verläuft, werden auf der Registerkarte **Meldungen** folgende Informationen angezeigt: Dauer des Datenexports, Dateiname und Pfad der exportierten Daten, Anzahl der geschriebenen Zeilen. Wenn das Exportieren fehlschlägt, werden Sie ebenfalls in einer Meldung darüber informiert.

Mit dieser Form der UNLOAD TABLE-Anweisung ist der Pfad relativ zum Computer des Datenbankservers.

Ergebnisse

Die Abfrageergebnisse werden an den angegebenen Speicherort exportiert.

Siehe auch

- „UNLOAD-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Adaptive Server Enterprise-Kompatibilität“ auf Seite 784

Konfigurieren Sie die Behandlung von NULL-Werten in Interactive SQL

Sie können den Fensterausschnitt **Ergebnisse** von Interactive SQL konfigurieren, um anzugeben, wie NULL-Werte dargestellt werden, wenn Sie die OUTPUT-Anweisung verwenden.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

- Klicken Sie in Interactive SQL auf **Extras » Optionen**.
- Klicken Sie auf **SQL Anywhere**.
- Klicken Sie auf die Registerkarte **Ergebnisse**.
- Im Feld **Nullwerte anzeigen als** geben Sie den Wert ein, den Sie für NULL-Werte verwenden möchten.

5. Klicken Sie auf **OK**.

Ergebnisse

Der Wert, der anstelle von NULL erscheint, wird geändert.

Siehe auch

- „SET OPTION-Anweisung [Interactive SQL]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „output_nulls-Option [Interactive SQL]“ [*SQL Anywhere Server - Datenbankadministration*]
- „IFNULL-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Exportieren von Datenbanken (Sybase Central)

Sie können den **Assistenten zum Entladen einer Datenbank** in Sybase Central verwenden, um Daten aus einer Datenbank in eine Reload-Datei, eine neue Datenbank oder eine vorhandene Datenbank zu entladen.

Voraussetzungen

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.

Folgendes gilt beim Entladen einer Datei auf einem Clientcomputer:

- Sie müssen das WRITE CLIENT FILE-Privileg haben:
- Sie müssen Schreibberechtigung für das Verzeichnis haben, in dem sich die Datei befindet.
- Die Datenbankoption allow_write_client_file muss aktiviert sein.
- Die Sicherheitsfunktion write_client_file muss aktiviert sein.

Aufgabe

1. Klicken Sie auf **Extras » SQL Anywhere 16 » Datenbank entladen**.
2. Befolgen Sie die Anweisungen im **Assistenten zum Entladen einer Datenbank**.

Ergebnisse

Die Daten werden an den angegebenen Speicherort entladen.

Siehe auch

- „UNLOAD-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Datenbankserveroption -gl“ [*SQL Anywhere Server - Datenbankadministration*]
- „Exportieren von Datenbanken (Befehlszeile)“ auf Seite 757

Exportieren von Datenbanken (Befehlszeile)

Sie können das Dienstprogramm Entladen (dbunload) verwenden, um Daten aus einer Datenbank in eine Reload-Datei, eine neue Datenbank oder eine vorhandene Datenbank zu entladen.

Voraussetzungen

Für einen Entladevorgang ohne Neuladen müssen Sie das SELECT ANY TABLE-Systemprivileg haben. Für einen Entladevorgang mit Neuladen müssen Sie die Systemprivilegien SELECT ANY TABLE und SERVER OPERATOR haben.

Aufgabe

- Führen Sie das Dienstprogramm Entladen (dbunload) aus und verwenden Sie die Option -c, um die Verbindungsparameter anzugeben.

Option	Aktion
Gesamte Datenbank entladen	So entladen Sie die gesamte Datenbank in das Verzeichnis <i>C:\ServerTemp\DataFiles</i> auf dem Serversystem: <code>dbunload -c "DBN=demo;UID=DBA;PWD=sql" C:\ServerTemp\DataFiles</code>
Exportieren Sie nur die Daten.	Verwenden Sie die Optionen -d und -ss. Beispiel: <code>dbunload -c "DBN=demo;UID=DBA;PWD=sql" -d -ss C:\ServerTemp\DataFiles</code>
Exportieren Sie nur das Schema.	Verwenden Sie die Option -n. Beispiel: <code>dbunload -c "DBN=demo;UID=DBA;PWD=sql" -n</code>

Die erforderlichen Anweisungen zum Neuerstellen des Schemas bzw. Neuladen der Tabelle werden in *reload.sql* im aktuellen lokalen Verzeichnis des Clientcomputers geschrieben.

Ergebnisse

Die Daten werden an den angegebenen Speicherort entladen.

Siehe auch

- „Datenbankserveroption -gl“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Dienstprogramm zum Entladen (dbunload)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Exportieren von Datenbanken (Sybase Central)“ auf Seite 756
- „UNLOAD-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Exportieren von Tabellen (SQL)

Sie können eine Tabelle durch Ausführen einer UNLOAD TABLE-Anweisung aus Interactive SQL exportieren.

Voraussetzungen

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.

Folgendes gilt beim Entladen einer Datei auf einem Clientcomputer:

- Wenn die Option -gl auf DBA gesetzt ist, müssen Sie das SELECT ANY TABLE-Systemprivileg haben.
- Sie müssen Schreibberechtigung für das Verzeichnis haben, in dem sich die Datei befindet.
- Die Datenbankoption allow_write_client_file muss aktiviert sein.
- Die Sicherheitsfunktion write_client_file muss aktiviert sein.

Kontext und Bemerkungen

Sie können eine Tabelle auch exportieren, indem Sie alle Daten in einer Tabelle auswählen und die Abfrageergebnisse exportieren.

Aufgabe

- Führen Sie die UNLOAD TABLE-Anweisung aus. Beispiel:

```
UNLOAD TABLE Departments  
TO 'C:\\ServerTemp\\Departments.csv';
```

Mit dieser Anweisung wird die Tabelle "Departments" aus der SQL Anywhere-Beispieldatenbank in die Datei *Departments.csv* in einem Arbeitsverzeichnis auf dem Computer des Datenbankservers, nicht des Clientcomputers entladen. Da der Dateipfad in einem SQL-Literal angegeben ist, werden die

Backslash-Zeichen verdoppelt, damit sie nicht irrtümlich in Escapesequenzen wie '\n' oder '\x' übersetzt werden.

Jede Zeile der Tabelle wird in der Exportdatei in eine Zeile gesetzt, und Spaltennamen werden nicht exportiert. Die Spalten werden durch ein Komma begrenzt. Das Begrenzungszeichen kann mit der Klausel DELIMITED BY geändert werden. Die Felder haben keine feste Breite. Nur die Zeichen in den Einträgen werden exportiert, nicht die volle Breite der Spalte.

Ergebnisse

Die Daten werden in die angegebene Datei exportiert.

Siehe auch

- „Exportieren von Daten mit dem Exportassistenten“ auf Seite 744
- „Dienstprogramm zum Entladen (dbunload)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankserveroption -gl“ [*SQL Anywhere Server - Datenbankadministration*]
- „UNLOAD-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Exportieren von Tabellen (Befehlszeile)

Sie können eine Tabelle exportieren, indem Sie das Dienstprogramm Entladen (dbunload) in der Befehlszeile ausführen.

Voraussetzungen

Für einen Entladevorgang ohne Neuladen müssen Sie das SELECT ANY TABLE-Systemprivileg haben. Für einen Entladevorgang mit Neuladen müssen Sie die Systemprivilegien SELECT ANY TABLE und SERVER OPERATOR haben.

Kontext und Bemerkungen

Sie können mehr als eine Tabelle entladen, indem Sie die Tabellennamen mit einem Komma trennen.

Aufgabe

- Führen Sie den folgenden Befehl aus:

```
dbunload -c "DBN=demo;UID=DBA;PWD=sql" -t Employees C:\ServerTemp
\DataFiles
```

Dabei gilt: -c gibt die Verbindungsparameter für die Datenbank an, und -t den Namen der Tabelle(n), die Sie exportieren wollen. Mit diesem dbunload-Befehl werden die Daten aus der SQL Anywhere-Beispieldatenbank (die hier auf dem Standard-Datenbankserver läuft) in eine Gruppe von Dateien im Verzeichnis C:\ServerTemp\DataFiles auf dem Servercomputer entladen. Eine SQL-Skriptdatei, mit der die Tabellen aus den Datendateien neu aufgebaut werden können, wird mit dem Standardnamen reload.sql im aktuellen Verzeichnis des Clients erstellt.

Ergebnisse

Die Daten werden an den angegebenen Speicherort exportiert.

Siehe auch

- „Datenbankserveroption -gl“ [\[SQL Anywhere Server - Datenbankadministration\]](#)
- „UNLOAD-Anweisung“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- „Dienstprogramm zum Entladen (dbunload)“ [\[SQL Anywhere Server - Datenbankadministration\]](#)

Zugriff auf Daten auf Clientcomputern

SQL Anywhere ermöglicht es Ihnen, Daten mithilfe von SQL-Anweisungen und Funktionen aus einer Datei auf einem Clientcomputer zu laden bzw. in eine solche Datei zu entladen, ohne Dateien auf den oder von dem Datenbankserver-Computer kopieren zu müssen. Um das zu tun, initiiert der Datenbankserver die Übertragung, indem eine Datei-Verarbeitungsroutine "Befehlsabfolge-Kommunikationsprotokoll (CmdSeq)" verwendet wird. Die Datei-Verarbeitungsroutine "CmdSeq" wird aufgerufen, nachdem der Datenbankserver eine Anforderung von der Clientanwendung erhalten hat, die eine Datenübertragung vom oder an den Clientcomputer verlangt, und bevor die Antwort gesendet wird. Die Datei-Verarbeitungsroutine unterstützt zu jedem beliebigen Zeitpunkt die gleichzeitige und miteinander verwobene Übertragung von mehreren Dateien vom Client. Der Datenbankserver kann beispielsweise die simultane Übertragung mehrerer Dateien initiieren, wenn dies die von der Clientanwendung ausgeführte Anweisung erfordert.

Die Verwendung der Datei-Verarbeitungsroutine "CmdSeq" für die Übertragung von Clientdaten bedeutet, dass Anwendungen keinen neuen speziellen Code des Anwenders benötigen und unmittelbar die Funktionen der unten angeführten SQL-Komponenten nutzen können:

- **READ_CLIENT_FILE-Funktion** Die Funktion `READ_CLIENT_FILE` liest die Daten aus der angegebenen Datei auf dem Clientcomputer und gibt einen LONG BINARY-Wert zurück, der den Inhalt der Datei darstellt. Diese Funktion kann überall im SQL-Code verwendet werden, wo ein BLOB benutzt werden kann. Die von der Funktion `READ_CLIENT_FILE` zurückgegebenen Daten werden soweit wie möglich nicht im Speicher materialisiert, außer die Anweisung bewirkt explizit eine Materialisierung. Die `LOAD TABLE`-Anweisung beispielsweise überträgt Daten von der Clientdatei, ohne sie zu materialisieren. Die Zuweisung des von der Funktion `READ_CLIENT_FILE` zurückgegebenen Werts an eine Verbindungsvariable bewirkt, dass der Datenbankserver den Inhalt der Clientdatei abrufen und materialisiert.
- **WRITE_CLIENT_FILE-Funktion** Die Funktion `WRITE_CLIENT_FILE` schreibt Daten in die angegebene Datei auf dem Clientcomputer.
- **READ CLIENT FILE-Systemprivileg** Mit dem `READ CLIENT FILE`-Systemprivileg können Sie aus einer Datei auf einem Clientcomputer lesen.
- **WRITE CLIENT FILE-Systemprivileg** Mit dem `WRITE CLIENT FILE`-Systemprivileg können Sie in eine Datei auf einem Clientcomputer schreiben.
- **LOAD TABLE ... USING CLIENT FILE-Klausel** Mit der `USING CLIENT FILE`-Klausel können Sie eine Tabelle mithilfe von Daten aus einer Datei auf dem Clientcomputer laden. Beispiel: `LOAD TABLE ... USING CLIENT FILE 'my-file.txt';` lädt die Datei namens *my-file.txt* vom Clientcomputer.

- **LOAD TABLE ... USING VALUE-Klausel** Die USING VALUE-Klausel ermöglicht es Ihnen, einen BLOB-Ausdruck als Wert anzugeben. Der BLOB-Ausdruck kann die Funktion `READ_CLIENT_FILE` verwenden, um einen BLOB aus einer Datei auf einem Clientcomputer zu laden. Beispiel: `LOAD TABLE ... USING VALUE READ_CLIENT_FILE('my-file')`, wobei *my-file* eine Datei auf dem Clientcomputer ist.
- **UNLOAD TABLE ... INTO CLIENT FILE-Klausel** Die INTO CLIENT FILE-Klausel ermöglicht es Ihnen, eine Datei auf dem Clientcomputer anzugeben, in die Daten entladen werden.
- **UNLOAD TABLE ... INTO VARIABLE-Klausel** Die INTO VARIABLE-Klausel ermöglicht es Ihnen, eine Variable anzugeben, in die Daten entladen werden.
- **Sicherheitsfunktionen `read_client_file` und `write_client_file`** Die Sicherheitsfunktionen "`read_client_file`" und "`write_client_file`" steuern die Verwendung von Anweisungen, die bewirken, dass von einer Clientdatei gelesen oder in sie geschrieben wird.

Eine Callback-Funktion muss registriert werden, damit es möglich ist, aus einer Clientdatei zu lesen oder aus einer Prozedur, Funktion oder anderen indirekten Anweisungen in sie zu schreiben. Die Callback-Funktion wird aufgerufen, um zu bestätigen, dass die Anwendung die Clientübertragung zulässt, die sie nicht direkt angefordert hat.

Siehe auch

- „Schlüssel für gesicherte Funktionen erstellen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption -sf“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „UNLOAD-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „LOAD TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`READ_CLIENT_FILE`-Funktion“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`WRITE_CLIENT_FILE`-Funktion [Zeichenfolge]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Erweiterte Verbindungsattribute für `SQLSetConnectAttr`“ [[SQL Anywhere Server - Programmierung](#)] (ODBC)
- „`db_register_a_callback`-Funktion“ [[SQL Anywhere Server - Programmierung](#)] (ESQL)
- „JDBC-Callbacks“ [[SQL Anywhere Server - Programmierung](#)] (JDBC)

Clientseitige Datensicherheit

SQL Anywhere bietet die Möglichkeit sicherzustellen, dass die Übertragung von Clientdateien nicht die unautorisierte Übertragung von Daten zulässt, die sich auf dem Clientcomputer befinden, der häufig an einem anderen Standort als das Datenbankserversystem ist.

Dafür verfolgt der Datenbankserver den Ursprung von jeder ausgeführten Anweisung und ermittelt, ob die empfangene Anweisung direkt von der Clientanwendung stammt. Wenn er die Übertragung einer Datei vom Clientcomputer initialisiert, erfasst der Datenbankserver Informationen zum Ursprung der Anweisung. Die Datei-Verarbeitungsroutine "`CmdSeq`" lässt dann die Übertragung von Dateien bei Anweisungen zu, die direkt von der Clientanwendung gesendet werden. Wenn die Anweisung nicht direkt von der Clientanwendung gesendet wurde, muss die Anwendung ein Verifizierungs-Callback registrieren. Wenn kein Callback registriert wird, wird die Übertragung abgelehnt und die Anweisung schlägt mit einer Fehlermeldung fehl.

Außerdem ist die Übertragung von Clientdaten erst wieder zulässig, nachdem eine Verbindung erfolgreich hergestellt wurde. Diese Einschränkung verhindert einen unautorisierten Zugriff mit Verbindungszeichenfolgen oder Login-Prozeduren.

Um sich gegen Versuche von Benutzern zu schützen, Zugriff auf ein System als scheinbar autorisierte Benutzer zu erhalten, ist es empfehlenswert, die zu übertragenden Daten zu verschlüsseln.

SQL Anywhere bietet auch die folgenden Sicherheitsmechanismen, um den Zugriff auf verschiedenen Ebenen zu steuern:

- **Sicherheit auf der Serverebene** Die Sicherheitsfunktionen "read_client_file" und "write_client_file" ermöglichen es Ihnen, alle clientseitigen Übertragungen auf einer serverweiten Basis zu unterbinden.
- **Sicherheit auf der Anwendungs- und DBA-Ebene** Die Datenbankoptionen "allow_read_client_file" und "allow_write_client_file" ermöglichen eine Zugriffssteuerung auf der Datenbank-, Benutzer- oder Verbindungsebene. Eine Anwendung könnte beispielsweise diese Datenbankoption nach der Verbindungsherstellung auf OFF setzen, um zu verhindern, dass sie selbst für irgendwelche clientseitigen Übertragungen verwendet wird.
- **Sicherheit auf der Benutzerebene** Die Systemprivilegien READ CLIENT FILE und WRITE CLIENT FILE bieten eine Zugriffssteuerung auf Benutzerebene für das Lesen bzw. Schreiben von Daten auf einem Clientcomputer.

Siehe auch

- „Datenbankserveroption -sf“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „allow_read_client_file-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

Wiederherstellung beim Laden von clientseitigen Daten

Falls Sie eine LOAD TABLE-Anweisung von Ihrem Transaktionslog wiederherstellen müssen, stehen Dateien auf dem Clientcomputer, die Sie zum Laden von Daten verwendet haben, SQL Anywhere wahrscheinlich nicht mehr zur Verfügung, oder sie haben sich geändert, wodurch die ursprünglichen Daten nicht mehr zugänglich sind. Um so eine Situation zu verhindern, stellen Sie sicher, dass die Protokollierung nicht deaktiviert ist. Anschließend geben Sie entweder die WITH ROW LOGGING- oder die WITH CONTENT LOGGING-Klausel an, wenn Sie die Daten laden. Diese Klauseln bewirken, dass die von Ihnen geladenen Daten im Transaktionslog aufgezeichnet werden, das später im Fall einer Wiederherstellung wiedergegeben werden kann.

WITH ROW LOGGING bewirkt, dass jede eingefügte Zeile als eine INSERT-Anweisung im Transaktionslog aufgezeichnet wird. WITH CONTENT LOGGING bewirkt, dass die eingefügten Daten im Transaktionslog als Abschnitte protokolliert werden, die der Datenbankserver während der Wiederherstellung abarbeitet. Beide Methoden gewährleisten, dass die clientseitigen Daten für ein Laden während der Wiederherstellung zur Verfügung stehen. Sie können jedoch WITH CONTENT LOGGING nicht verwenden, wenn Sie Daten in eine Datenbank laden, die an einer Synchronisation beteiligt ist.

Wenn Sie eine der folgenden LOAD TABLE-Anweisungen angeben, ohne eine Protokollierungsstufe festzulegen, ist WITH CONTENT LOGGING das Standardverhalten:

- LOAD TABLE ... USING CLIENT FILE *client-filename-expression*
- LOAD TABLE ... USING VALUE *value-expression*
- LOAD TABLE ... USING COLUMN *column-expression*

Neuaufbau von Datenbanken

Der Neuaufbau einer Datenbank ist ein spezieller Typ des Importierens und Exportierens, bei dem die gesamte Datenbank entladen und wieder eingelesen wird. Mithilfe der Tools für Neuerstellung (Entladen/Laden) und Extraktion können Sie Datenbanken neu aufbauen, neue Datenbanken aus Teilen einer vorhandenen erstellen sowie unbenutzte freie Seiten entfernen.

Sie können die Datenbank über Sybase Central oder mit dem Dienstprogramm Entladen (dbunload) neu aufbauen.

Hinweis

Es ist wichtig, vor dem Neuaufbau einer Datenbank eine Datensicherung durchzuführen, besonders, wenn die Originaldatenbank durch die neu aufgebaute Datenbank ersetzt wird.

Beim Importieren und Exportieren werden die Daten in Ihre Datenbank eingelesen oder aus Ihrer Datenbank ausgelesen. Beim Importieren werden Daten in die Datenbank eingelesen. Beim Exportieren werden Daten aus der Datenbank ausgelesen. Die Informationen kommen oft von einer anderen Datenbank, die nicht aus SQL Anywhere stammen, oder gehen zu einer solchen Datenbank.

Wenn Sie die Verschlüsselungsoptionen -ek, -ep oder -et angeben, müssen die LOAD TABLE-Anweisungen in der Datei *reload.sql* den Chiffrierschlüssel einbeziehen. Die Hartkodierung des Schlüssels stellt ein Sicherheitsrisiko dar, deshalb wird der Chiffrierschlüssel in der Datei *reload.sql* über einen Parameter festgelegt. Wenn Sie die Datei *reload.sql* mit Interactive SQL ausführen, müssen Sie den Chiffrierschlüssel als Parameter angeben. Wenn Sie den Schlüssel nicht in der READ-Anweisung angeben, werden Sie von Interactive SQL aufgefordert, den Schlüssel bereitzustellen.

Beim Laden und Entladen werden Daten und Schemata aus einer SQL Anywhere-Datenbank herausgeholt und dann wieder in eine SQL Anywhere-Datenbank gesetzt. Die Prozedur zum Entladen erzeugt Datendateien und die Datei *reload.sql*, die Tabellendefinitionen für eine exakte Neuerstellung der Tabellen enthält. Durch das Ausführen des Skripts *reload.sql* werden die Tabellen neu erstellt und Daten darin eingelesen.

Der Neuaufbau einer Datenbank kann viel Zeit und Plattenspeicher in Anspruch nehmen. Während des Entladens und des Ladens ist die Datenbank darüber hinaus zeitweilig nicht verfügbar. Aus diesen Gründen wird davon abgeraten, den Neuaufbau einer Datenbank in einer Produktionsumgebung vorzunehmen, wenn nicht stichhaltige Gründe dafür vorhanden sind.

Von einer SQL Anywhere-Datenbank in eine andere

Beim Neuaufbau werden in der Regel Daten aus einer SQL Anywhere-Datenbank kopiert und danach neu in eine andere SQL Anywhere-Datenbank geladen. Entladen und Neuladen stehen miteinander in Verbindung, da normalerweise immer beide Aufgaben durchgeführt werden, nicht aber nur eine der beiden.

Neuaufbau versus Exportieren

Der Neuaufbau unterscheidet sich vom Export, weil nicht nur die Daten exportiert und importiert werden, sondern auch die Tabellendefinitionen und das Schema. Der Entladevorgang während des Neuaufbaus produziert Datendateien im Text-Format und eine Datei namens *reload.sql*, die Tabellendefinitionen und andere Schemainformationen enthält. Sie können das Skript *reload.sql* ausführen, um die Tabellen neu zu erstellen und die Daten in die Tabellen zu laden.

Sie können die Datenbank auch extrahieren (eine neue Datenbank aus einer alten erstellen), wenn Sie SQL Remote oder MobiLink verwenden wollen.

Replizierende Datenbanken neu aufbauen

Wie beim Neuaufbau einer Datenbank vorgegangen wird, hängt davon ab, ob sie in ein Replikationssystem eingebunden ist oder nicht. Wenn die Datenbank in ein Replikationssystem eingebunden ist, müssen Sie die Offsets des Transaktionslogs während des Vorgangs beibehalten, da der Nachrichtenagent diese Informationen benötigt. Wenn die Datenbank nicht in ein Replikationssystem eingebunden ist, wird der Vorgang einfacher.

Siehe auch

- „Minimieren der Ausfallzeit beim Neuaufbau einer Datenbank“ auf Seite 773
- „An Replikations- oder Synchronisationssystemen beteiligte Datenbanken neu aufbauen (Befehlszeile)“ auf Seite 767
- „Nicht an Replikations- oder Synchronisationssystemen beteiligte Datenbanken neu aufbauen“ auf Seite 766
- „Datenbankkollationen ändern“ [*SQL Anywhere Server - Datenbankadministration*]
- „Materialisierten Ansichten manuell aktualisieren“ auf Seite 67
- „Interactive SQL-Dienstprogramm (dbisql)“ [*SQL Anywhere Server - Datenbankadministration*]
- Vergleich von externem Entladen und Neuladen [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbankextraktion“ auf Seite 774
- „Sicherung und Datenwiederherstellung“ [*SQL Anywhere Server - Datenbankadministration*]

Gründe für den Neuaufbau von Datenbanken

Es gibt mehrere Gründe, Ihre Datenbank neu aufzubauen. Folgende Situationen bieten Anlass zum Neuaufbau einer Datenbank:

- **Upgrade des Datenbankdateiformats** Wenn Sie das Upgrade-Dienstprogramm benutzen, werden einige neue Funktionen automatisch zur Verfügung gestellt. Andere Funktionen erfordern jedoch ein Upgrade des Datenbank-Dateiformats, das durch Entladen und Neuladen der Datenbank durchgeführt wird.

Neue Versionen des SQL Anywhere-Datenbankservers können ohne Upgrade der Datenbank benutzt werden. Um Funktionen der neuen Version zu benutzen, die Zugriff auf neue Systemtabellen oder Datenbankoptionen erfordern, müssen Sie das Dienstprogramm zum Upgrade verwenden, um ein Upgrade Ihrer Datenbank durchzuführen. Durch das Dienstprogramm zum Upgrade werden keine Daten entladen oder neu geladen.

Um die neue Version von SQL Anywhere zu verwenden, die auf den Änderungen des Datenbank-Dateiformats beruht, müssen Sie Ihre Datenbank entladen und dann neu laden. Sie sollten Ihre Datenbank sichern, bevor Sie einen Neuaufbau durchführen.

Hinweis

Falls Sie ein Upgrade von Version 9 oder früher durchführen, müssen Sie die Datenbankdatei neu aufbauen. Falls Sie ein Upgrade von 10.0.0 oder später durchführen, können Sie das Upgrade-Dienstprogramm benutzen oder die Datenbank neu aufbauen.

- **Plattenspeicher freigeben** Datenbanken werden nicht kleiner, wenn Sie Daten löschen. Alle leeren Seiten werden einfach als frei markiert, damit sie wieder benutzt werden können. Sie werden aus der Datenbank erst entfernt, wenn Sie sie neu aufbauen. Mit dem Neuaufbau einer Datenbank können Sie Plattenspeicher frei machen, wenn Sie große Datenmengen aus der Datenbank entfernt haben und nicht erwarten, dass die leeren Seiten wieder gefüllt werden.
- **Datenbankperformance verbessern** Durch den Neuaufbau von Datenbanken kann die Performance verbessert werden. Da die Datenbank in der Reihenfolge der Primärschlüssel entladen und geladen werden kann, wird der Zugriff auf die Daten schneller, da miteinander in Beziehung stehende Zeilen nun auf derselben Seite oder auf benachbarten Seiten angezeigt werden können.

Hinweis

Wenn Sie feststellen, dass die Performance aufgrund der starken Fragmentierung einer Tabelle schlecht ist, können Sie die Tabelle neu organisieren.

Siehe auch

- „Upgrade auf SQL Anywhere 16“ [[SQL Anywhere 16 - Änderungen und Upgrades](#)]
- „Upgrade und Neuaufbau in einem Datenbankspiegelungssystem“ [[SQL Anywhere 16 - Änderungen und Upgrades](#)]
- „REORGANIZE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Dienstprogramm zum Upgrade (dbupgrad)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Dienstprogramm zum Entladen (dbunload)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Tipps zum Neuaufbau von Datenbanken mit dem Dienstprogramm Entladen (dbunload)

Sie können das Dienstprogramm Entladen (dbunload) benutzen, um eine komplette Datenbank in ein Textformat mit Kommatrennzeichen zu entladen und die erforderlichen SQL-Skriptdateien zu erstellen, mit denen die Datenbank komplett wieder aufgebaut werden kann. Beispiel: Sie verwenden diese Dateien, um SQL Remote-Extraktionen zu erstellen oder neue Kopien Ihrer Datenbank mit derselben oder einer leicht geänderten Struktur aufzubauen.

Verwenden Sie das Dienstprogramm Entladen (dbunload) in folgenden Situationen:

- Sie wollen Ihre Datenbank neu aufbauen oder Ihre Datenbank extrahieren.
- Sie möchten Daten im Textformat exportieren.

- Sie müssen große Datenmengen rasch verarbeiten.
- Sie sind bei den Anforderungen an Dateiformate flexibel.

Hinweis

Das Dienstprogramm Entladen (dbunload) und der **Assistent zum Entladen einer Datenbank** sind funktionell gleichwertig. Sie können beide verwenden und erhalten dabei dieselben Ergebnisse. Sie können eine Datenbank auch entladen, indem Sie die Interactive SQL-Anweisung OUTPUT oder die SQL-Anweisung UNLOAD ausführen.

Siehe auch

- „Nicht an Replikations- oder Synchronisationssystemen beteiligte Datenbanken neu aufbauen“ auf Seite 766
- „An Replikations- oder Synchronisationssystemen beteiligte Datenbanken neu aufbauen (Befehlszeile)“ auf Seite 767

Nicht an Replikations- oder Synchronisationssystemen beteiligte Datenbanken neu aufbauen

Mit dem Dienstprogramm Entladen (dbunload) können Sie eine Datenbank entladen und eine neue Datenbank daraus aufbauen sowie eine vorhandene Datenbank damit aktualisieren oder dadurch ersetzen.

Voraussetzungen

Die folgende Prozedur sollten Sie nur verwenden, wenn Ihre Datenbank nicht in ein Synchronisations- oder Replikationssystem einbezogen ist.

Sie müssen die Systemprivilegien SELECT ANY TABLE und SERVER OPERATOR haben.

Kontext und Bemerkungen

Die Optionen -an und -ar gelten nur für Verbindungen zu einem Personal Server oder zu einem Netzwerkserver über gemeinsam genutzten Speicher. Die Optionen -ar und -an sollten außerdem schneller ausgeführt werden als der **Assistent zum Entladen einer Datenbank** in Sybase Central, die Option -ac ist jedoch langsamer als der **Assistent zum Entladen einer Datenbank**.

Sie können andere dbunload-Optionen angeben, um eine laufende oder nicht laufende Datenbank sowie Datenbankparameter anzugeben.

Aufgabe

1. Führen Sie das Dienstprogramm Entladen (dbunload) mit einer der folgenden Optionen aus:

Gewünschte Aktion	Zu verwendende Option	Beispiel
Eine Datenbank neu aufbauen	-an	<code>dbunload -c "DBF=demo.db;UID=DBA;PWD=sql" -an DemoBackup.db</code>
Neu in eine vorhandene Datenbank laden	-ac	<code>dbunload -c "DBF=demo.db;UID=DBA;PWD=sql" -ac "UID=DBA;PWD=sql;DBF=mynewdemo.db"</code>
Eine vorhandene Datenbank ersetzen	-ar	<code>dbunload -c "DBF=demo.db;UID=DBA;PWD=sql" -ar</code>

Wenn Sie eine dieser Optionen verwenden, wird keine Zwischenkopie der Daten auf der Festplatte angelegt. Sie brauchen daher in der Befehlszeile kein Entladeverzeichnis anzugeben. Dies bietet größere Sicherheit für Ihre Daten.

- Fahren Sie die Datenbank herunter und speichern Sie das Transaktionslog, bevor Sie die neu geladene Datenbank verwenden.

Ergebnisse

Die Datenbank wird entladen und am angegebenen Speicherort wieder geladen.

Siehe auch

- „Dienstprogramm zum Entladen (dbunload)“ [[SQL Anywhere Server - Datenbankadministration](#)]

An Replikations- oder Synchronisationssystemen beteiligte Datenbanken neu aufbauen (Befehlszeile)

Sie können eine Datenbank, die an einem Synchronisations- oder Replikationssystem teilnimmt, mit der Option -ar von dbunload neu aufbauen. Dabei wird die Datenbank entladen und neu geladen, sodass sie mit der Synchronisation und Replikation nicht in Konflikt gerät.

Voraussetzungen

Sie müssen die Systemprivilegien `SELECT ANY TABLE` und `SERVER OPERATOR` haben.

Alle Subskriptionen müssen synchronisiert werden, bevor eine Datenbank neu aufgebaut wird, die an einer MobiLink-Synchronisation teilnimmt.

Kontext und Bemerkungen

Dieser Abschnitt bezieht sich auf SQL Anywhere MobiLink-Clients (Clients, die `dbmlsync` benutzen) und auf SQL Remote.

Die Synchronisation oder Replikation bezieht sich auf die sogenannten "Offsets" (Ausgangspunkte) im Transaktionslog. Wenn Sie eine Datenbank neu aufbauen, unterscheiden sich die Offsets im alten Transaktionslog von denen im neuen Transaktionslog, wodurch das alte Transaktionslog nicht mehr zur Verfügung steht. Aus diesem Grund ist für Datenbanken in einem Synchronisations- oder Replikationssystem das Sichern besonders wichtig.

Hinweis

Sie können andere `dbunload`-Optionen angeben, um eine laufende oder nicht laufende Datenbank sowie Datenbankparameter anzugeben.

Aufgabe

1. Fahren Sie die Datenbank herunter.
2. Führen Sie eine vollständige Offline-Sicherung durch, indem Sie Kopien der Datenbank- und Transaktionslogdateien an einem sicheren Standort speichern.
3. Führen Sie den folgenden `dbunload`-Befehl aus, um die Datenbank neu aufzusetzen:

```
dbunload -c connection-string -ar directory
```

Die *connection-string* ist eine Verbindung mit entsprechenden Privilegien und *directory* ist das Verzeichnis, das in Ihrer Replikationsumgebung für alte Transaktionslogs verwendet wird. Es kann keine weiteren Verbindungen zur Datenbank geben.

Die Option `-ar` gilt nur für Verbindungen zu einem Personal Server oder zu einem Netzwerkserver über gemeinsam genutzten Speicher.

4. Fahren Sie die neue Datenbank herunter und führen Sie dann die Gültigkeitsprüfungen durch, die Sie normalerweise nach dem Wiederherstellen einer Datenbank durchführen würden.
5. Starten Sie die Datenbank mit allen Optionen, die Sie für die Produktionsumgebung normalerweise benutzen. Geben Sie den Zugriff der Benutzer auf die neu geladene Datenbank frei.

Ergebnisse

Die Datenbank wird neu geladen und gestartet.

Siehe auch

- „Dienstprogramm zum Entladen (dbunload)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbanken validieren (Sybase Central)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Neuaufbau von Datenbanken“ auf Seite 763
- „MobiLink-Upgrades“ [[SQL Anywhere 16 - Änderungen und Upgrades](#)]
- „SQL Remote-Upgrades“ [[SQL Anywhere 16 - Änderungen und Upgrades](#)]

An Replikations- oder Synchronisationssystemen beteiligte Datenbanken neu aufbauen (manuell)

Sie können einen manuellen Neuaufbau einer in ein Replikations- oder Synchronisationssystem einbezogenen Datenbank vornehmen (dbunload).

Voraussetzungen

Sie müssen die Systemprivilegien `SELECT ANY TABLE` und `SERVER OPERATOR` haben, um die Datenbank neu aufbauen zu können.

Alle Subskriptionen müssen synchronisiert werden, bevor eine Datenbank neu aufgebaut wird, die an einer MobiLink-Synchronisation teilnimmt.

Kontext und Bemerkungen

Dieser Abschnitt bezieht sich auf SQL Anywhere MobiLink-Clients (Clients, die `dbmlsync` benutzen) und auf SQL Remote.

Die Synchronisation oder Replikation bezieht sich auf die sogenannten "Offsets" (Ausgangspunkte) im Transaktionslog. Wenn Sie eine Datenbank neu aufbauen, unterscheiden sich die Offsets im alten Transaktionslog von denen im neuen Transaktionslog, wodurch das alte Transaktionslog nicht mehr zur Verfügung steht. Aus diesem Grund ist für Datenbanken in einem Synchronisations- oder Replikationssystem das Sichern besonders wichtig.

Aufgabe

1. Fahren Sie die Datenbank herunter.
2. Führen Sie eine vollständige Offline-Sicherung durch, indem Sie Kopien der Datenbank- und Transaktionslogdateien an einem sicheren Standort speichern.
3. Führen Sie das `dbtran`-Dienstprogramm aus, um das Start-Offset und das End-Offset der aktuellen Transaktionslogdatei der Datenbank anzuzeigen.

Notieren Sie das End-Offset zur Verwendung in Schritt 8.

4. Benennen Sie die aktuelle Transaktionslog-Datei um, damit sie während des Entladungsprozesses nicht geändert wird, und setzen Sie diese Datei in das `dbremote-Offline-Logverzeichnis`.
5. Bauen Sie die Datenbank neu auf.

6. Fahren Sie die neue Datenbank herunter.
7. Löschen Sie die aktuelle Transaktionslogdatei für die neue Datenbank.
8. Führen Sie "dblog" mit der neuen Datenbank aus und verwenden Sie das End-Offset, das Sie in Schritt 3 notiert haben, als Option -z. Setzen Sie das relative Offset auf null.

```
dblog -x 0 -z 0000698242 -ir -is database-name.db
```
9. Wenn Sie den Nachrichtenagent ausführen, übergeben Sie ihm den Standort des Original-Offline-Verzeichnisses auf der Befehlszeile.
10. Starten Sie die Datenbank. Geben Sie den Zugriff der Benutzer auf die neu geladene Datenbank frei.

Ergebnisse

Die Datenbank wird neu geladen und gestartet.

Siehe auch

- „Datenbanken validieren (Sybase Central)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Neuaufbau von Datenbanken“ auf Seite 763
- „Transaktionslog-Dienstprogramm (dblog)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Dienstprogramm zum Entladen (dbunload)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Dienstprogramm zur Logkonvertierung (dbtran)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Transaktionslog-Dienstprogramm (dblog)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Tipps zum Neuaufbau von Datenbanken mit der UNLOAD TABLE-Anweisung

Mit der UNLOAD TABLE-Anweisung können Sie Daten in einer bestimmten Zeichenkodierung effizient exportieren. Sie sollten die Anweisung UNLOAD TABLE zum Neuaufbau von Datenbanken verwenden, wenn Sie Daten im Textformat exportieren wollen.

Auswirkungen auf die Datenbank

Die UNLOAD TABLE-Anweisung setzt eine Exklusivsperrung auf die gesamte Tabelle.

Siehe auch

- „UNLOAD-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Exportieren von Tabellendaten

Das Dienstprogramm Entladen (dbunload) ermöglicht Ihnen, nur die Tabellendaten zu entladen.

Voraussetzungen

Sie müssen Eigentümer der abgefragten Tabelle sein, das SELECT-Privileg für die Tabelle haben oder das SELECT ANY TABLE-Systemprivileg haben.

Kontext und Bemerkungen

Die erforderlichen Anweisungen zum Neuerstellen des Schemas und zum Neuladen der angegebenen Tabellen werden in *reload.sql* im aktuellen lokalen Verzeichnis geschrieben.

Sie können mehr als eine Tabelle entladen, indem Sie die Tabellennamen mit einem Komma trennen.

Aufgabe

- Führen Sie den dbunload-Befehl aus, indem Sie Folgendes angeben: Verbindungsparameter unter Verwendung der Option -c, die Tabelle(n), für die Sie Daten exportieren wollen, unter Verwendung der Option -t, wenn Sie Spaltenstatistiken unterdrücken möchten, die Option -ss und, wenn Sie nur Daten entladen wollen, die Option -d.

Um beispielsweise die Daten aus der Tabelle "Employees" zu exportieren, verwenden Sie folgenden Befehl:

```
dbunload -c "DBN=demo;UID=DBA;PWD=sql" -ss -d -t Employees C:\ServerTemp
\DataFiles
```

Die Datei *reload.sql* wird in das aktuelle Verzeichnis des Clients geschrieben und enthält die LOAD TABLE-Anweisung, die zum Neuladen der Daten für die Tabelle Employees erforderlich ist. Die Datendateien werden in das Serververzeichnis *C:\ServerTemp\DataFiles* geschrieben.

Ergebnisse

Die Tabellendaten werden in das angegebene Verzeichnis exportiert.

Siehe auch

- „Dienstprogramm zum Entladen (dbunload)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Exportieren eines Tabellenschemas

Das Dienstprogramm Entladen (dbunload) bietet Optionen, mit denen Sie nur das Tabellenschema entladen können.

Voraussetzungen

Sie müssen Eigentümer der Tabelle sein, das SELECT-Privileg für die Tabelle haben oder das SELECT ANY TABLE-Systemprivileg haben.

Kontext und Bemerkungen

Die erforderlichen Anweisungen zum Neuerstellen des Schemas und zum Neuladen der angegebenen Tabellen werden in *reload.sql* im aktuellen Verzeichnis des Clients geschrieben.

Sie können die Daten oder das Schema aus mehr als einer Tabelle entladen, indem Sie die Tabellennamen mit einem Komma trennen.

Aufgabe

- Führen Sie den `dbunload`-Befehl aus, indem Sie Folgendes angeben: Verbindungsparameter unter Verwendung der Option `-c`; die Tabelle(n), für die Sie Daten exportieren wollen, unter Verwendung der Option `-t`; und, wenn Sie nur das Schema entladen wollen, die Option `-n`.

Um beispielsweise nur das Schema der `Employees`-Tabelle zu exportieren, führen Sie den folgenden Befehl aus:

```
dbunload -c "DBN=demo;UID=DBA;PWD=sql" -n -t Employees
```

Ergebnisse

Das Tabellenschema wird exportiert.

Siehe auch

- „Dienstprogramm zum Entladen (`dbunload`)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Neuladen einer Datenbank

Sie können Datenbanken über die Befehlszeile neu laden. Das Neuladen umfasst das Erstellen einer leeren Datenbankdatei und das Verwenden der Datei `reload.sql`, um das Schema zu erstellen und dann alle Daten, die von einer anderen SQL Anywhere-Datenbank entladen wurden, in die neu erstellten Tabellen einzufügen.

Voraussetzungen

Die Datei `reload.sql` muss vorhanden sein.

Aufgabe

1. Führen Sie das Dienstprogramm "dbinit" aus, um eine neue, leere Datenbankdatei zu erstellen.
2. Mit der neuen Datenbank verbinden.
3. Führen Sie das `reload.sql`-Skript aus.

Ergebnisse

Die Datenbank wird neu geladen.

Beispiel

Mit dem folgenden Befehl wird eine Datei namens `mynewdemo.db` erstellt.

```
dbinit -dba DBA,sql mynewdemo.db
```

Mit dem folgenden Befehl laden Sie das `reload.sql`-Skript in das aktuelle Verzeichnis und führen es aus.

```
dbisql -c "DBF=mynewdemo;UID=DBA;PWD=sql" reload.sql
```

Siehe auch

- „Dienstprogramm Initialisierung (dbinit)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Interactive SQL-Dienstprogramm (dbisql)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Neuaufbau von Datenbanken“ auf Seite 763

Minimieren der Ausfallzeit beim Neuaufbau einer Datenbank

Sie können die Ausfallzeit beim Neuaufbau einer Datenbank minimieren, indem Sie die Dienstprogramme für die Sicherung und die Logkonvertierung verwenden.

Voraussetzungen

Es wird empfohlen, vor dem Neuaufbau einer Datenbank Sicherungskopien von Ihren Datenbankdateien zu erstellen.

Stellen Sie sicher, dass keine anderen geplanten Sicherungen das Transaktionslog umbenennen können. Wenn das Log umbenannt wird, müssen die Transaktionen aus den umbenannten Transaktionslogs in der richtigen Reihenfolge auf die neu erstellte Datenbank angewendet werden.

Sie müssen das BACKUP DATABASE-Systemprivileg haben.

Aufgabe

1. Erstellen Sie mit dbbackup -r eine Sicherungskopie der Datenbank und des Transaktionslogs und benennen Sie das Transaktionslog um.
2. Bauen Sie die gesicherte Datenbank auf einem anderen Computer neu auf.
3. Führen Sie dbbackup -r noch einmal auf dem Produktionsserver aus, um das Transaktionslog umzubenennen.
4. Führen Sie das Dienstprogramm dbtran auf dem Transaktionslog aus und übernehmen Sie die Transaktionen in die neu aufgebaute Datenbank.
5. Fahren Sie den Produktionsserver herunter und erstellen Sie Kopien von Datenbank und Transaktionslog.
6. Kopieren Sie die neu aufgebaute Datenbank auf den Produktionsserver.
7. Führen Sie dbtran mit dem Transaktionslog aus Schritt 5 aus.
8. Starten Sie die neu aufgebaute Datenbank auf einem Personal Server (dbeng16), um sicherzustellen, dass sich Benutzer nicht verbinden können.
9. Übernehmen Sie die Transaktionen aus Schritt 8.
10. Fahren Sie den Datenbankserver herunter und starten Sie die Datenbank auf einem Netzwerkserver (dbsrv16), damit sich die Benutzer verbinden können.

Ergebnisse

Die Ausfallzeit wird während des Neuaufbaus einer Datenbank minimiert.

Siehe auch

- „Sicherungsdienstprogramm (dbbackup)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Dienstprogramm zur Logkonvertierung (dbtran)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Datenbankextraktion

Die Datenbankextraktion wird von SQL Remote benutzt. Durch die Extraktion wird eine entfernte SQL Anywhere-Datenbank aus einer konsolidierten SQL Anywhere-Datenbank erstellt.

Sie können den **Assistenten "Datenbank extrahieren"** in Sybase Central oder das Extraktionsdienstprogramm verwenden, um Datenbanken zu extrahieren. Das Extraktionsdienstprogramm (dbxtract) ist die empfohlene Methode zum Erstellen von entfernten Datenbanken aus einer konsolidierten Datenbank für die Verwendung in der SQL Remote-Replikation.

Siehe auch

- „Extraktionsdienstprogramm (dbxtract)“ [[SQL Remote](#)]
- „Extraktion von entfernten Datenbanken“ [[SQL Remote](#)]
- „Entfernte MobiLink-Datenbanken durch Anpassen eines Prototyps bereitstellen“ [[MobiLink - Clientadministration](#)]

Datenbankmigration nach SQL Anywhere

Verwenden Sie die Systemprozeduren sa_migrate oder den **Assistenten zum Migrieren einer Datenbank**, um Tabellen aus den folgenden Quellen zu importieren:

- SQL Anywhere
- UltraLite
- Sybase Adaptive Server Enterprise
- IBM DB2
- Microsoft SQL Server
- Microsoft Access
- Oracle
- MySQL
- Advantage Database Server
- Generische ODBC-Treiber, die eine Verbindung zu einem Fremdserver herstellen

Bevor Sie Daten mit dem **Assistenten zum Migrieren einer Datenbank** oder mit Systemprozeduren der sa_migrate-Gruppe migrieren können, müssen Sie zuerst eine **Zieldatenbank** erstellen. Die Zieldatenbank ist die Datenbank, in die Daten migriert werden.

Siehe auch

- „Datenbankerstellung“ [[SQL Anywhere Server - Datenbankadministration](#)]

Verwenden des Assistenten zum Migrieren einer Datenbank

Mit dem **Assistenten zum Migrieren einer Datenbank** können Sie in Sybase Central einen Fremdserver erstellen, um sich mit der entfernten Datenbank zu verbinden, und ein externes Login einrichten (falls erforderlich), um den aktuellen Benutzer mit der entfernten Datenbank zu verbinden.

Voraussetzungen

Sie müssen bereits einen Fremdserver erstellt haben. Sie müssen bereits einen Benutzer haben, der Eigentümer der Tabellen in der Zieldatenbank ist.

Sie benötigen entweder sowohl das CREATE PROXY TABLE-Systemprivileg als auch das CREATE TABLE-Systemprivileg oder alle folgenden Systemprivilegien:

- CREATE ANY TABLE
- ALTER ANY TABLE
- DROP ANY TABLE
- INSERT ANY TABLE
- SELECT ANY TABLE
- CREATE ANY INDEX

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie auf **Extras » SQL Anywhere 16 » Datenbank migrieren**.
3. Klicken Sie auf **Weiter**.
4. Wählen Sie die Zieldatenbank und klicken Sie auf **Weiter**.
5. Wählen Sie den Fremdserver zum Verbinden mit der entfernten Datenbank und klicken Sie auf **Weiter**.

Sie können auch ein externes Login für den Fremdserver erstellen. Standardmäßig benutzt SQL Anywhere die Benutzer-ID und das Kennwort des aktuellen Benutzers, wenn er für diesen Benutzer eine Verbindung mit einem Fremdserver herstellt. Wenn allerdings der Fremdserver keinen Benutzer mit derselben Benutzer-ID und Kennwort wie der aktuelle Benutzer hat, müssen Sie ein externes Login erstellen. Das externe Login ordnet dem aktuellen Benutzer eine Alternativ-Benutzer-ID und Kennwort zu, damit er sich mit dem Fremdserver verbinden kann.

6. Wählen Sie die zu migrierenden Tabellen und klicken Sie auf **Weiter**.

Sie können keine Systemtabellen migrieren, daher sind sie nicht in der Liste enthalten.

7. Wählen Sie den Benutzer, dem die Tabellen in der Zieldatenbank gehören sollen, und klicken Sie auf **Weiter**.
8. Wählen Sie, ob Sie die Daten bzw. Fremdschlüssel der entfernten Tabelle migrieren wollen und ob die für den Migrationsprozess erstellten Proxy-Tabellen erhalten bleiben sollen, und klicken Sie dann auf **Weiter**.
9. Klicken Sie auf **Fertig stellen**.

Ergebnisse

Die angegebenen Tabellen werden migriert.

Siehe auch

- „CREATE SERVER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Fremdserver erstellen (Sybase Central)“ auf Seite 789
- „Benutzer erstellen (Sybase Central)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Externe Logins erstellen (Sybase Central)“ auf Seite 804

Verwenden der Systemprozeduren sa_migrate

Verwenden Sie die Systemprozeduren sa_migrate, um entfernte Daten zu migrieren. Verwenden Sie die erweiterte Methode, um Tabellen oder Fremdschlüssel-Zuordnungen zu entfernen.

Siehe auch

- „sa_migrate-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_migrate_create_fks-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_migrate_create_remote_fks_list-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_migrate_create_remote_table_list-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_migrate_create_tables-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_migrate_data-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_migrate_drop_proxy_tables-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Alle Tabellen mit der sa_migrate-Systemprozedur migrieren

Sie können alle Tabellen mit der sa_migrate-Systemprozedur migrieren.

Voraussetzungen

Sie müssen die folgenden Systemprivilegien haben:

- CREATE TABLE oder CREATE ANY TABLE (wenn Sie nicht der Basistabelleneigentümer sind)
- SELECT ANY TABLE (wenn Sie nicht der Basistabelleneigentümer sind)

- INSERT ANY TABLE (wenn Sie nicht der Basistabelleneigentümer sind)
- ALTER ANY TABLE (wenn Sie nicht der Basistabelleneigentümer sind)
- CREATE ANY INDEX (wenn Sie nicht der Basistabelleneigentümer sind)
- DROP ANY TABLE (wenn Sie nicht der Basistabelleneigentümer sind)

Sie müssen bereits einen Benutzer haben, der Eigentümer der migrierten Tabellen in der Zieldatenbank ist.

Wenn Sie ein externes Login erstellen möchten, benötigen Sie das **MANAGE ANY USER**-Systemprivileg.

Kontext und Bemerkungen

Tabellen, die in der entfernten Datenbank den gleichen Namen haben, aber unterschiedlichen Eigentümern gehören, werden in der Zieldatenbank alle einem Eigentümer zugeordnet. Aus diesen Gründen sollten Sie Tabellen, die einem einzelnen Eigentümer zugeordnet sind, gleichzeitig migrieren.

Wenn Sie nicht möchten, dass alle migrierten Tabellen dem selben Benutzer der Zieldatenbank gehören, müssen Sie die Prozedur `sa_migrate` für jeden einzelnen Benutzer der Zieldatenbank unter Angabe der Argumente *local-table-owner* und *owner-name* wiederholen.

Aufgabe

1. Verbinden Sie sich aus Interactive SQL mit der Zieldatenbank.
2. Erstellen Sie einen Fremdserver, um sich mit der entfernten Datenbank zu verbinden.
3. (Optional) Erstellen Sie ein externes Login für die Verbindung zur entfernten Datenbank. Dies ist nur dann erforderlich, wenn der Benutzer für Ziel- und entfernte Datenbank unterschiedliche Kennwörter hat oder wenn Sie sich nicht mit der Benutzer-ID an der entfernten Datenbank anmelden möchten, die Sie für die Zieldatenbank verwenden.
4. Führen Sie im Fensterausschnitt **SQL-Anweisungen** die `sa_migrate`-Systemprozedur aus. Die Angabe `NULL` für die Parameter Tabellename und Eigentümername lässt alle Tabellen in der Datenbank migrieren, auch die Systemtabellen.

Beispiel:

```
CALL sa_migrate( 'local_user1', 'rmt_server1', NULL, 'remote_user1',  
NULL, 1, 1, 1 );
```

Ergebnisse

Diese Prozedur ruft wiederum einige andere Prozeduren auf und migriert alle die dem Benutzer `remote_user1` gehörenden entfernten Tabellen gemäß den angegebenen Kriterien.

Siehe auch

- „Benutzer erstellen (Sybase Central)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sa_migrate-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Fremdserver“ auf Seite 787
- „Externe Logins“ auf Seite 804

Migrieren einzelner Tabellen mit den Systemprozeduren für die Datenbankmigration

Sie können einzelne Tabellen mit den Systemprozeduren für die Datenbankmigration migrieren.

Voraussetzungen

Sie müssen die folgenden Systemprivilegien haben:

- CREATE ANY TABLE
- CREATE ANY INDEX
- INSERT ANY TABLE
- SELECT ANY TABLE
- ALTER ANY TABLE
- DROP ANY TABLE

Sie müssen bereits einen Fremdserver erstellt haben. Sie müssen bereits einen Benutzer haben, der Eigentümer der Tabellen in der Zieldatenbank ist.

Wenn Sie ein externes Login erstellen möchten, benötigen Sie das `MANAGE ANY USER`-Systemprivileg.

Kontext und Bemerkungen

Geben Sie für die Parameter *table-name* und *owner-name* nicht NULL an. Das hätte zur Folge, dass alle Tabellen, inklusive Systemtabellen, in die Datenbank migriert würden. Auch werden Tabellen, die in der entfernten Datenbank den gleichen Namen haben, aber unterschiedlichen Eigentümern gehören, in der Zieldatenbank alle einem Eigentümer zugeordnet. Es wird empfohlen, dass Sie nur einem Eigentümer zugeordnete Tabellen auf einmal migrieren.

Aufgabe

1. Erstellen Sie eine Zieldatenbank.
2. Verbinden Sie sich aus Interactive SQL mit der Zieldatenbank.
3. (Optional) Erstellen Sie ein externes Login für die Verbindung zur entfernten Datenbank. Ein externes Login ist nur dann erforderlich, wenn der Benutzer für Ziel- und entfernte Datenbank unterschiedliche

Kennwörter hat oder wenn Sie sich nicht mit der Benutzer-ID an der entfernten Datenbank anmelden möchten, die Sie für die Zieldatenbank verwenden.

4. Führen Sie die Systemprozedur `sa_migrate_create_remote_table_list` aus. Beispiel:

```
CALL sa_migrate_create_remote_table_list( 'rmt_server1',  
    NULL, 'remote_user1', 'mydb' );
```

Für Adaptive Server Enterprise und Microsoft SQL Server Datenbanken müssen Sie einen Datenbanknamen angeben.

Mit dieser Prozedur wird in die Tabelle `dbo.migrate_remote_table_list` eine Liste der zu migrierenden entfernten Tabellen eingetragen. Sie können Zeilen aus dieser Tabelle löschen und damit die entsprechenden entfernten Tabellen von der Migration ausschließen.

5. Führen Sie die Systemprozedur `sa_migrate_create_tables` aus. Beispiel:

```
CALL sa_migrate_create_tables( 'local_user1' );
```

Diese Prozedur nimmt die Liste der entfernten Tabellen aus der `dbo.migrate_remote_table_list` und erstellt eine Proxy-Tabelle und eine Basistabelle für jede aufgelistete entfernte Tabelle. Diese Prozedur erstellt auch alle Primärschlüsselindizes für die migrierten Tabellen.

6. Für eine Migration der Daten der entfernten Tabellen in die Basistabellen der Zieldatenbank müssen Sie die `sa_migrate_data`-Systemprozedur ausführen. Beispiel:

```
CALL sa_migrate_data( 'local_user1' );
```

Diese Prozedur migriert die Daten aus jeder entfernten Tabelle in die Basistabelle, die von der Prozedur `sa_migrate_create_tables` erstellt wurde.

Sollten Sie die Migration der Fremdschlüssel von der entfernten Datenbank nicht benötigen, können Sie mit Schritt 10 fortfahren.

7. Führen Sie die Systemprozedur `sa_migrate_create_remote_fks_list` aus. Beispiel:

```
CALL sa_migrate_create_remote_fks_list( 'rmt_server1' );
```

Diese Prozedur füllt die Tabelle `dbo.migrate_remote_fks_list` mit der Liste der Fremdschlüssel, die den in `dbo.migrate_remote_table_list` aufgelisteten entfernten Tabellen zugeordnet sind.

Sie können Fremdschlüsselzuordnungen entfernen, die in den lokalen Basistabellen nicht neu erstellt werden sollen.

8. Führen Sie die Systemprozedur `sa_migrate_create_fks` aus. Beispiel:

```
CALL sa_migrate_create_fks( 'local_user1' );
```

Diese Prozedur erstellt die in `dbo.migrate_remote_fks_list` definierten Fremdschlüsselzuordnungen in den Basistabellen.

9. Zum Löschen der für Migrationszwecke erstellten Proxy-Tabellen führen Sie die `sa_migrate_drop_proxy_tables`-Systemprozedur aus. Beispiel:

```
CALL sa_migrate_drop_proxy_tables( 'local_user1' );
```

Ergebnisse

Diese Prozedur löscht alle Proxy-Tabellen, die für die Migrationsprozedur erstellt wurden, und schließt dann den Migrationsprozess ab.

Siehe auch

- „sa_migrate_create_remote_table_list-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_migrate_create_tables-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_migrate_data-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_migrate_create_remote_fks_list-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_migrate_create_fks-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa_migrate_drop_proxy_tables-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Fremdserver“ auf Seite 787
- „Externe Logins“ auf Seite 804

SQL-Skriptdateien

SQL-Skriptdateien sind Textdateien, die SQL-Anweisungen enthalten, und dann von Nutzen, wenn Sie dieselben SQL-Anweisungen wiederholt ausführen. Skriptdateien können manuell oder automatisch mithilfe von Datenbank-Dienstprogrammen erstellt werden. Das Dienstprogramm Entladen (dbunload) erstellt z.B. eine Skriptdatei mit den SQL-Anweisungen, die zur Wiederherstellung einer Datenbank erforderlich sind.

SQL-Skriptdateien erstellen

Sie können SQL-Skriptdateien mit einem beliebigen Texteditor erstellen, aber Interactive SQL ist für die Erstellung von SQL-Skriptdateien empfehlenswert. Sie können in die SQL-Anweisungen auch Kommentarzeilen einbeziehen.

Hinweis

Sie können eine SQL-Skriptdatei in Interactive SQL aus Ihren Favoriten in den Fensterausschnitt **SQL-Anweisungen** laden.

Siehe auch

- „Kommentare“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „READ-Anweisung [Interactive SQL]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Interactive SQL anpassen“ [*SQL Anywhere Server - Datenbankadministration*]
- „SQL-Skriptdateien, SQL-Anweisungen und Verbindungen zu den Favoriten hinzufügen“ [*SQL Anywhere Server - Datenbankadministration*]

SQL-Skriptdateien ohne Laden ausführen

Sie können in Interactive SQL eine SQL-Skriptdatei ausführen, ohne sie in den Fensterausschnitt **SQL-Anweisungen** zu laden.

Voraussetzungen

Achten Sie darauf, dass Interactive SQL als Standardeditor für *.sql*-Dateien eingerichtet ist.

Klicken Sie in Interactive SQL auf **Extras » Optionen » Allgemein** und anschließend auf **Interactive SQL als Standardeditor für .SQL-Dateien und Plandateien einrichten**.

Welche Privilegien erforderlich sind, hängt davon ab, welche Anweisungen ausgeführt werden.

Kontext und Bemerkungen

Die Menüoption **Skript ausführen** entspricht der READ-Anweisung.

Aufgabe

1. Klicken Sie in Interactive SQL auf **Datei » Skript ausführen**.
2. Ermitteln Sie die Position der Datei und klicken Sie auf **Öffnen**.

Ergebnisse

Der Inhalt der angegebenen Datei wird umgehend ausgeführt. Ein Fenster **Status** wird eingeblendet, in dem der Verarbeitungsfortschritt angezeigt wird.

Siehe auch

- „[READ-Anweisung \[Interactive SQL\]](#)“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „[Interactive SQL anpassen](#)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „[SQL-Skriptdateien, SQL-Anweisungen und Verbindungen zu den Favoriten hinzufügen](#)“ [[SQL Anywhere Server - Datenbankadministration](#)]

SQL-Skriptdateien mit der Interactive SQL-Anweisung READ ausführen

Sie können eine SQL-Skriptdatei ebenfalls ausführen, ohne sie in den Fensterausschnitt **SQL-Anweisungen** zu laden, indem Sie die Interactive SQL-Anweisung READ verwenden.

Voraussetzungen

Welche Privilegien erforderlich sind, hängt davon ab, welche Anweisungen ausgeführt werden.

Aufgabe

- Führen Sie im Fensterausschnitt **SQL-Anweisungen** eine Anweisung wie im folgenden Beispiel aus:

```
READ 'C:\\LocalTemp\\Dateiname.sql';
```

Dabei gilt: *C:\\LocalTemp\\filename.sql* gibt den Pfad, den Namen und die Namensweiterung der Datei an. Apostrophe wie im Beispiel sind nur dann erforderlich, wenn der Pfad Leerstellen enthält. Wenn Sie Apostrophe verwenden, wird das Backslash-Zeichen verdoppelt, damit es nicht irrtümlich in Escapesequenzen wie `'\\n'` oder `'\\x'` übersetzt wird.

Ergebnisse

Die SQL-Skriptdatei wird ausgeführt.

Siehe auch

- „[READ-Anweisung \[Interactive SQL\]](#)“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „[Interactive SQL anpassen](#)“ [*SQL Anywhere Server - Datenbankadministration*]
- „[SQL-Skriptdateien, SQL-Anweisungen und Verbindungen zu den Favoriten hinzufügen](#)“ [*SQL Anywhere Server - Datenbankadministration*]

SQL-Skriptdateien im Batchmodus ausführen (Befehlszeile)

Sie können eine SQL-Skriptdatei als Befehlszeilenoption für Interactive SQL eingeben.

Voraussetzungen

Welche Privilegien erforderlich sind, hängt davon ab, welche Anweisungen ausgeführt werden.

Aufgabe

- Führen Sie das Dienstprogramm "dbisql" aus und geben Sie eine SQL-Skriptdatei als Befehlszeilenoption an.

Ergebnisse

Die SQL-Skriptdatei wird ausgeführt.

Beispiel

Mit dem folgenden Befehl wird die SQL-Skriptdatei *myscript.sql* in der SQL Anywhere-Beispieldatenbank ausgeführt.

```
dbisql -c "DSN=SQL Anywhere 16 Demo" myscript.sql
```

Siehe auch

- „[Interactive SQL-Dienstprogramm \(dbisql\)](#)“ [*SQL Anywhere Server - Datenbankadministration*]

Laden eines SQL-Skripts aus einer Datei in den Fensterausschnitt SQL-Anweisungen

Sie können in Interactive SQL eine SQL-Skriptdatei in den Fensterausschnitt **SQL-Anweisungen** laden und sie direkt von dort aus ausführen.

Voraussetzungen

Achten Sie darauf, dass Interactive SQL als Standardeditor für *.sql* Dateien ausgeführt wird.

Klicken Sie in Interactive SQL auf **Extras » Optionen » Allgemein** und anschließend auf **Interactive SQL als Standardeditor für .SQL-Dateien und Plandateien einrichten**.

Welche Privilegien erforderlich sind, hängt davon ab, welche Anweisungen ausgeführt werden.

Aufgabe

1. Klicken Sie auf **Datei » Öffnen**.
2. Ermitteln Sie die Position der Datei und klicken Sie auf **Öffnen**.

Ergebnisse

Die Anweisungen werden im Fensterausschnitt **SQL-Anweisungen** angezeigt, wo Sie sie lesen, bearbeiten oder ausführen.

Datenbankausgabe in eine Datei schreiben

In Interactive SQL bleiben die Ergebnismengendaten (falls vorhanden) einer Anweisung nur so lange auf der Registerkarte **Ergebnisse** im Fensterausschnitt **Ergebnisse** erhalten, bis die nächste Anweisung ausgeführt wird. Wenn Sie die Daten dauerhaft aufzeichnen möchten, können Sie die Ausgabe jeder Anweisung in einer separaten Datei speichern.

Voraussetzungen

Welche Privilegien erforderlich sind, hängt davon ab, welche Anweisungen ausgeführt werden.

Aufgabe

- Wenn *statement1* und *statement2* zwei unterschiedliche SELECT-Anweisungen sind, können die Ergebnisse ihrer Ausführung folgendermaßen in *file1* bzw. *file2* ausgegeben werden:

```
statement1; OUTPUT TO file1;statement2; OUTPUT TO file2;
```

Ergebnisse

Die Ausgabe jeder SQL-Anweisung ist in einer getrennten Datei enthalten.

Beispiel

Mit den folgenden Anweisungen speichern Sie das Ergebnis einer Datei namens *Employees.csv* im Verzeichnis *C:\LocalTemp*:

```
SELECT * FROM Employees;  
OUTPUT TO 'C:\LocalTemp\Employees.csv';
```

Siehe auch

- „SELECT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „OUTPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Tipps zum Exportieren von Daten mit der UNLOAD-Anweisung“ auf Seite 749

Adaptive Server Enterprise-Kompatibilität

Sie können Dateien zwischen SQL Anywhere und Adaptive Server Enterprise mit der BCP FORMAT-Klausel importieren und exportieren. Wenn Sie BLOB-Daten aus SQL Anywhere für Adaptive Server Enterprise benutzen, verwenden Sie die BCP FORMAT-Klausel mit der UNLOAD TABLE-Anweisung.

Bei der Verwendung des BCP OUT-Befehls zum Exportieren von Dateien aus Adaptive Server Enterprise und zum Importieren von Daten in SQL Anywhere müssen die Daten im Text/ASCII-Format und durch Kommas getrennt sein. Sie können die Option -c für den BCP OUT-Befehl verwenden, um Daten im Text/ASCII-Format zu exportieren. Mit der Option -t können Sie das Trennzeichen ändern, das standardmäßig ein Tabulator ist. Wenn Sie das Trennzeichen nicht ändern, müssen Sie in der TABLE LOAD-Anweisung beim Importieren der Daten in die SQL Anywhere-Datenbank **DELIMITED BY '\x09'** angeben.

Siehe auch

- „LOAD TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „UNLOAD-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Ferndatenzugriff

Mit dem SQL Anywhere-Ferndatenzugriff können Sie auf Daten in anderen Datenquellen zugreifen. Sie können diese Funktion verwenden, um Daten in eine SQL Anywhere-Datenbank zu migrieren. Außerdem können Sie diese Funktion verwenden, um Daten aus mehreren Datenbanken abzufragen.

Mit dem Ferndatenzugriff können Sie folgendes durchführen:

- Verschieben von Daten über SQL Anywhere von einem Standort an einen anderen mit Einfüge- und Auswahlabfragen.
- Zugriff auf Daten in relationalen Datenbanken wie Sybase ASE, Oracle Database und IBM DB2.
- Zugriff auf Daten in Excel-Tabellen, Microsoft Access-Datenbanken, FoxPro-Datenbanken und Textdateien.
- Zugriff auf jede Datenquelle, die die ODBC-Schnittstelle unterstützt.
- Joins zwischen lokalen und entfernten Daten ausführen, auch wenn die Performance niedriger ist, als wenn sich alle Daten in einer einzigen SQL Anywhere-Datenbank befinden.
- Verknüpfung von Tabellen in getrennten SQL Anywhere-Datenbanken durch Joins durchführen. Die Performance-Einschränkungen sind hier die gleichen wie bei anderen entfernten Datenquellen.
- Verwendung von SQL Anywhere-Funktionen mit Datenquellen, die diese Funktionalität normalerweise nicht haben. Sie könnten z.B. eine Java-Funktion mit Daten einsetzen, die in einer Oracle Database gespeichert sind, oder eine Unterabfrage in Tabellenkalkulationen ausführen. SQL Anywhere ergänzt die Funktionen, die von einer externen Datenquelle nicht unterstützt werden, indem die Daten nach der Abfrage entsprechend verarbeitet werden.
- Direkter Zugriff auf Fremdserver über die FORWARD TO-Anweisung.
- Ausführen von Remote Procedure Calls in anderen Servern.

SQL Anywhere ermöglicht den Zugriff auf folgende externe Datenquellen:

- SQL Anywhere
- Adaptive Server Enterprise
- Advantage Database Server
- IBM DB2
- Microsoft Access
- Microsoft SQL Server
- Oracle MySQL
- Oracle Database
- SAP HANA
- SAP Sybase IQ
- UltraLite
- Andere ODBC-Datenquellen

Hinweis

Sie können keinen Fremdserver für eine UltraLite-Datenbank unter Mac OS X erstellen.

Weitere Hinweise zur Verfügbarkeit von Plattformen finden Sie unter <http://www.sybase.com/detail?id=1002288>.

Zuordnungen entfernter Tabellen

SQL Anywhere präsentiert einer Clientanwendung die Tabellen so, als wären die Daten der Tabellen in der Datenbank gespeichert, mit der sich die Anwendung verbindet. Intern wird bei einer Abfrage auf entfernte Tabellen der Speicherort ermittelt und auf den entfernten Standort zugegriffen, um die Daten abzurufen.

Um auf Daten in einer entfernten Tabelle zuzugreifen, müssen Sie Folgendes einrichten.

1. Sie müssen den Fremdserver definieren, auf dem sich die entfernten Daten befinden. Dies umfasst die Serverklasse und den Standort des Fremdservers. Dafür wird die CREATE SERVER-Anweisung verwendet.
2. Sie müssen die Anmeldeinformationen für den Fremdserver definieren, wenn die erforderlichen Anmeldeinformationen für den Zugriff auf die Datenbank auf dem Fremdserver nicht mit denjenigen der Datenbank übereinstimmen, mit der Sie verbunden sind. Dafür wird die CREATE EXTERNLOGIN-Anweisung verwendet.
3. Sie müssen eine Proxy-Tabellendefinition erstellen. Damit wird die Zuordnung einer lokalen Proxy-Tabelle zu einer entfernten Tabelle festgelegt. Zu dieser Definition gehört auch der Server, auf dem sich die entfernte Tabelle befindet sowie Datenbankname, Eigentümername und Spaltennamen der entfernten Tabelle. Dafür wird die CREATE EXISTING TABLE-Anweisung verwendet. Die CREATE TABLE-Anweisung kann außerdem verwendet werden, um neue Tabellen auf dem Fremdserver zu erstellen.

Um die Definitionen der Fremdserver, die externen Logins und die Proxy-Tabellenzuordnungen zu verwalten, können Sie entweder Sybase Central aufrufen oder ein Werkzeug wie Interactive SQL benutzen und eigene SQL-Anweisungen eingeben.

Vorsicht

Einige Fremdserver, wie etwa Microsoft Access, Microsoft SQL Server und Sybase Adaptive Server Enterprise, lassen Cursor bei COMMIT und ROLLBACK nicht geöffnet. Bei diesen Fremdservern können Sie die Registerkarte **Daten** in Sybase Central nicht verwenden, um den Inhalt einer Proxy-Tabelle anzuzeigen oder zu ändern. Sie können Interactive SQL jedoch benutzen, um die Daten in diesen Proxy-Tabellen anzuzeigen und zu bearbeiten, solange "Autocommit" ausgeschaltet ist (dies ist das Standardverhalten in Interactive SQL). Bei anderen RDBMS wie Oracle Database, IBM DB2 und SQL Anywhere gibt es diese Einschränkung nicht.

Fremdserver

Bevor Sie entfernte Objekte einer lokalen Proxy-Tabelle zuordnen können, müssen Sie den Fremdserver definieren, auf dem sich das entfernte Objekt befindet. Wenn Sie einen Fremdserver definieren, muss die Serverklasse ausgewählt werden.

Die **Serverklasse** legt die Zugriffsmethode fest, die für die Interaktion mit dem Fremdserver verwendet wird. Für die unterschiedlichen Typen von Fremdservern gelten unterschiedliche Zugriffsmethoden. Die Serverklasse gibt SQL Anywhere detaillierte Informationen über die Serverfunktionen. SQL Anywhere kann daher seine Interaktionen mit dem Fremdserver entsprechend anpassen.

Die Serverklassen sind:

- **SAODBC** für SQL Anywhere
- **ULODBC** für UltraLite.

Hinweis

Sie können keinen Fremdserver für eine UltraLite-Datenbank unter Mac OS X erstellen.

- **ADSODBC** für Advantage Database Server.
- **ASEODBC** für Sybase Adaptive Server Enterprise (Version 10 und höher)
- **DB2ODBC** für IBM DB2.
- **HANAODBC** für SAP HANA.
- **IQODBC** für SAP Sybase IQ.
- **MSACCESSODBC** für Microsoft Access.
- **MSSODBC** für Microsoft SQLServer.
- **MYSQLODBC** für Oracle MySQL.
- **ODBC** für alle anderen ODBC-Datenquellen.
- **ORAODBC** für Oracle-Datenbankserver (Version 8.0 und höher).

Hinweis

Wenn Sie beim Ferndatenzugriff einen ODBC-Treiber verwenden, der Unicode nicht unterstützt, wird keine Zeichensatzkonvertierung bei Daten durchgeführt, die von diesem ODBC-Treiber stammen.

Wenn Sie einen Fremdserver definieren, wird in der ISYSSERVER-Systemtabelle für den Fremdserver ein Eintrag eingefügt.

Siehe auch

- [„Serverklassen für den Ferndatenzugriff“ auf Seite 824](#)

Fremdserver erstellen (SQL)

Verwenden Sie die CREATE SERVER-Anweisung, um Fremdserverdefinitionen einzurichten.

Voraussetzungen

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Kontext und Bemerkungen

Der Zugriff auf Fremdserver erfolgt über einen ODBC-Treiber. Eine Fremdserverdefinition ist für jede Datenbank erforderlich.

Eine Verbindungszeichenfolge wird zur Kennzeichnung einer Datenquelle benutzt. Auf Unix-Plattformen muss der ODBC-Treiber auch in der Verbindungszeichenfolge referenziert werden.

Fremdserver erstellen

- Verwenden Sie die CREATE SERVER-Anweisung, um einen Server für den Ferndatenzugriff zu erstellen, der eine Verknüpfung mit einem Fremdserver herstellt.

Zum Beispiel definiert die folgende Anweisung den Fremdserver RemoteASE. Der SQL Anywhere-Datenbankserver stellt mit der ODBC-Verbindungszeichenfolge, die in der USING-Klausel angegeben ist, eine Verbindung mit einer Adaptive Server Enterprise-Datenbankserver her.

```
CREATE SERVER RemoteASE
CLASS 'ASEODBC'
USING 'DRIVER=SYBASE ASE ODBC
Driver;Server=TestASE;Port=5000;Database=testdb;UID=username;PWD=password'
;
```

Im Folgenden sehen Sie eine Analyse der Komponenten der CREATE SERVER-Anweisung.

- **SERVER** Diese Klausel wird benutzt, um den Fremdserver zu benennen. In diesem Beispiel ist RemoteASE der Fremdservername.
- **CLASS** Diese Klausel wird verwendet, um anzuzeigen, wie der SQL Anywhere-Datenbankserver mit dem Fremdserver kommunizieren soll. In diesem Beispiel zeigt ASEODBC an, dass der Fremdserver der Adaptive Server Enterprise (ASE) ist und die Verbindung mithilfe des ASE ODBC-Treibers hergestellt wird.
- **USING** Diese Klausel gibt die ODBC-Verbindungszeichenfolge für den Fremdserver an. In diesem Beispiel wird der ASE ODBC-Treiber angegeben.

Ergebnisse

Mit der CREATE SERVER-Anweisung wird ein Eintrag in der SYSERVER-Systemtabelle erstellt.

Nächste Schritte

Erstellen Sie die Informationen für das externe Login, wenn erforderlich.

Beispiel

Die folgende Anweisung definiert den Fremdserver RemoteSA. Der SQL Anywhere-Datenbankserver stellt mit dem ODBC-Datenquellennamen (DSN), der in der USING-Klausel angegeben ist, eine Verbindung mit einem SQL Anywhere-Datenbankserver her.

```
CREATE SERVER RemoteSA
CLASS 'SAODBC'
USING 'SQL Anywhere 16 CustDB';
```

Die folgende Anweisung definiert den Fremdserver RemoteLinuxSA. Der SQL Anywhere-Datenbankserver stellt mit dem ODBC-Datenquellennamen (DSN), der in der USING-Klausel angegeben ist, eine Verbindung mit einem SQL Anywhere-Datenbankserver her. Auf Unix-Plattformen muss der ODBC-Treiber auch in der Verbindungszeichenfolge angegeben werden.

```
CREATE SERVER RemoteLinuxSA
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 16;DSN=my_sa_dsn';
```

Die folgende Anweisung definiert den Fremdserver RemoteLinuxASE. Der SQL Anywhere-Datenbankserver stellt mit dem ODBC-Datenquellennamen (DSN), der in der USING-Klausel angegeben ist, eine Verbindung mit einem Adaptive Server Enterprise-Datenbankserver (ASE) her. Auf Unix-Plattformen muss der ODBC-Treiber auch in der Verbindungszeichenfolge angegeben werden.

```
CREATE SERVER RemoteLinuxASE
CLASS 'ASEODBC'
USING '/opt/sybase/ase_odbc_1500/DataAccess/ODBC/lib/
libsybdrvodb.so;DSN=my_ase_dsn';
```

Die folgende Anweisung definiert den Fremdserver RemoteAccessDB. Der SQL Anywhere-Datenbankserver stellt mit der ODBC DSN MyAccessDataSource, die in der USING-Klausel angegeben ist, eine Verbindung mit einem Microsoft Access-Datenbankserver her.

```
CREATE SERVER RemoteAccessDB
CLASS 'MSACCESSODBC'
USING 'MyAccessDataSource';
```

Siehe auch

- „Externe Logins erstellen (Sybase Central)“ auf Seite 804
- „Erstellen von Proxy-Tabellen (SQL)“ auf Seite 807
- „CREATE SERVER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE EXTERNLOGIN-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Fremdserver erstellen (Sybase Central)

Administratoren können Sybase Central verwenden, um Definitionen für Fremdserver zu erstellen.

Voraussetzungen

Sie müssen die Systemprivilegien MANAGE ANY USER und SERVER OPERATOR haben.

Fremdserver erstellen

1. Stellen Sie eine Verbindung zur Host-Datenbank mithilfe des SQL Anywhere 16-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Fremdserver**.
3. Klicken Sie auf **Datei » Neu » Fremdserver**.
4. Geben Sie in das Feld **Wie lautet der Name des neuen Fremdservers?** einen Namen für den Fremdserver ein und klicken Sie dann auf **Weiter**.
5. Wählen Sie einen Fremdservertyp aus und klicken Sie auf **Weiter**.
6. Wählen Sie einen Verbindungstyp aus und geben Sie in das Feld **Wie lauten die Verbindungsinformationen?** die Verbindungsinformationen ein:
 - Für eine ODBC-basierte Verbindung geben Sie einen Datenquellennamen ein oder geben den ODBC Driver-Parameter sowie andere Verbindungsparameter an.
 - Geben Sie für eine JDBC-basierte Verbindung eine URL im folgenden Format ein: *computer-name:port-number*.

Die Datenzugriffsmethode (JDBC oder ODBC) ist die von SQL Anywhere benutzte Methode für den Zugriff auf die entfernte Datenbank. Dies entspricht nicht der Methode, die von Sybase Central zur Verbindung mit Ihrer Datenbank verwendet wird.

Der JDBC-basierte Fremdserverzugang wird vom aktuellen Release nicht unterstützt.

7. Klicken Sie auf **Weiter**.
8. Geben Sie an, ob der Fremdserver schreibgeschützt sein soll und klicken Sie auf **Weiter**.
9. Klicken Sie auf **Externes Login für den aktuellen Benutzer erstellen** und füllen Sie die erforderlichen Felder aus.

Standardmäßig benutzt SQL Anywhere die Benutzer-ID und das Kennwort des aktuellen Benutzers, wenn er für diesen Benutzer eine Verbindung mit einem Fremdserver herstellt. Wenn allerdings der Fremdserver keinen Benutzer mit derselben Benutzer-ID und Kennwort wie der aktuelle Benutzer hat, müssen Sie ein externes Login erstellen. Das externe Login ordnet dem aktuellen Benutzer eine Alternativ-Benutzer-ID und Kennwort zu, damit er sich mit dem Fremdserver verbinden kann.

10. Klicken Sie auf **Verbindung testen**, um die Fremdserver-Verbindung zu testen.
11. Klicken Sie auf **Fertig stellen**.

Ergebnisse

Ein Fremdserver wird mit den angegebenen Definitionen erstellt.

Nächste Schritte

Erstellen Sie die Informationen für das externe Login, wenn erforderlich.

Siehe auch

- „Externe Logins erstellen (Sybase Central)“ auf Seite 804
- „Proxy-Tabellen erstellen (Sybase Central)“ auf Seite 807
- „CREATE SERVER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE EXTERNLOGIN-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Fremdserver entfernen (SQL)

Administratoren können Fremdserver mit der DROP SERVER-Anweisung entfernen.

Voraussetzungen

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Kontext und Bemerkungen

Alle für den Fremdserver definierten Proxy-Tabellen müssen entfernt werden, bevor Sie den Fremdserver entfernen. Die folgende Abfrage kann verwendet werden, um zu ermitteln, welche Proxy-Tabellen für den Fremdserver *server-name* definiert sind.

```
SELECT st.table_name, sp.remote_location, sp.existing_obj
FROM sysproxytab sp
JOIN sysserver ss ON ss.srvid = sp.srvid
JOIN systab st ON sp.table_object_id = st.object_id
WHERE ss.srvname = 'server-name';
```

Fremdserver entfernen

1. Stellen Sie eine Verbindung mit der Host-Datenbank her.
2. Führen Sie eine DROP SERVER-Anweisung aus.

```
DROP SERVER server-name;
```

Ergebnisse

Der Fremdserver wird entfernt.

Beispiel

Die folgende Anweisung entfernt den Fremdserver RemoteASE.

```
DROP SERVER RemoteASE;
```

Siehe auch

- „Fremdserver entfernen (Sybase Central)“ auf Seite 792
- „DROP SERVER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Fremdserver entfernen (Sybase Central)

Administratoren können Fremdserver in Sybase Central entfernen.

Voraussetzungen

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Kontext und Bemerkungen

Alle für den Fremdserver definierten Proxy-Tabellen müssen entfernt werden, bevor Sie den Fremdserver entfernen. Sybase Central bestimmt automatisch, welche Proxy-Tabellen für einen Fremdserver definiert sind und entfernt sie zuerst.

Fremdserver entfernen

1. Stellen Sie eine Verbindung zur Host-Datenbank mithilfe des SQL Anywhere 16-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Fremdserver**.
3. Wählen Sie den Fremdserver und klicken Sie dann auf **Bearbeiten » Löschen**.

Ergebnisse

Der Fremdserver wird entfernt.

Siehe auch

- „Fremdserver entfernen (SQL)“ auf Seite 791
- „DROP SERVER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Ändern von Fremdservern (SQL)

Administratoren können die Eigenschaften eines Fremdservers in Interactive SQL ändern.

Voraussetzungen

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Kontext und Bemerkungen

Die Anweisung ALTER SERVER kann auch benutzt werden, um die bekannten Fähigkeiten eines Servers zu ändern.

Ändern der Eigenschaften eines Fremdservers

Änderungen am Fremdserver werden erst nach der nächsten Verbindungsherstellung zum Fremdserver wirksam.

1. Stellen Sie eine Verbindung mit der Host-Datenbank her.

2. Führen Sie eine ALTER SERVER-Anweisung aus.

Ergebnisse

Die Eigenschaften des Fremdservers werden geändert.

Beispiel

Die folgende Anweisung ändert die Serverklasse des Servers RemoteASE auf ASEODBC:

```
ALTER SERVER RemoteASE  
CLASS 'ASEODBC';
```

Siehe auch

- „ALTER SERVER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Ändern von Fremdservern (Sybase Central)

Sie können die Eigenschaften eines Fremdservers in Sybase Central ändern.

Voraussetzungen

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Kontext und Bemerkungen

Änderungen am Fremdserver werden erst nach der nächsten Verbindungsherstellung zum Fremdserver wirksam.

Ändern der Eigenschaften eines Fremdservers

1. Stellen Sie eine Verbindung zur Host-Datenbank mithilfe des SQL Anywhere 16-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Fremdserver**.
3. Wählen Sie den Fremdserver aus und klicken Sie dann auf **Datei » Eigenschaften**.
4. Ändern Sie die Fremdserver-Einstellungen und klicken Sie dann auf **OK**.

Ergebnisse

Die Eigenschaften des Fremdservers werden geändert.

Siehe auch

- „ALTER SERVER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Tabellen auf einem Fremdserver auflisten (SQL)

Mithilfe einer Systemprozedur können Sie eine eingeschränkte oder eine umfassende Liste aller Tabellen auf einem Fremdserver anzeigen.

Voraussetzungen

Keiner

Auflisten der Tabellen auf einem Fremdserver

- Rufen Sie die `sp_remote_tables`-Systemprozedur auf, um eine Liste der Tabellen auf einem Fremdserver zurückzugeben.

Wenn Sie `@table_name` oder `@table_owner` angeben, wird die Liste auf die damit übereinstimmenden Tabellen eingeschränkt.

Ergebnisse

Eine Liste aller Tabellen bzw. eine eingeschränkte Liste der Tabellen wird zurückgegeben.

Beispiel

Um eine Liste aller Tabellen in einer Datenbank auf dem Fremdserver RemoteSA mit dem Eigentümer GROUPO zu erhalten, führen Sie die folgende Anweisung aus:

```
CALL sp_remote_tables('RemoteSA', null, 'GROUPO');
```

Um eine Liste aller Tabellen in der Datenbank "Production" auf einem Adaptive Server Enterprise-Server mit dem Namen RemoteASE und dem Eigentümer Fred abzurufen, müssen Sie die folgende Anweisung ausführen:

```
CALL sp_remote_tables('RemoteASE', null, 'Fred', 'Production');
```

Um eine Liste aller Microsoft Excel-Arbeitsblätter abzurufen, die auf einem Fremdserver mit dem Namen Excel verfügbar sind, müssen Sie die folgende Anweisung ausführen:

```
CALL sp_remote_tables('Excel');
```

Siehe auch

- „`sp_remote_tables`-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Funktionalität von Fremdservern

Die Systemprozedur "sp_servercaps" zeigt Informationen über die Funktionalität eines Fremdservers. SQL Anywhere verwendet diese Informationen, um zu ermitteln, wie viel von einer SQL-Anweisung an einen Fremdserver weitergeleitet werden kann.

Sie können auch Informationen über die Funktionalitäten von Fremdservern anzeigen, indem Sie die Systemansichten SYSCAPABILITY und SYSCAPABILITYNAME abfragen. Die Systemansichten sind leer, bis SQL Anywhere erstmals eine Verbindung zum Fremdserver hergestellt hat.

Wenn Sie die Systemprozedur `sp_servercaps` verwenden, muss der angegebene *server-name* der *server-name* sein, der in der `CREATE SERVER`-Anweisung verwendet wurde.

Führen Sie die gespeicherte Prozedur "`sp_servercaps`" folgendermaßen aus:

```
CALL sp_servercaps('server-name');
```

Siehe auch

- „`sp_servercaps`-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`SYSCAPABILITY`-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`SYSCAPABILITYNAME`-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „`CREATE SERVER`-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Verzeichniszugriffsserver

Ein **Verzeichniszugriffsserver** ist ein Fremdserver, der Ihnen Zugriff auf die lokale Dateistruktur des Computers gibt, auf dem der Datenbankserver läuft. Sobald Sie mit dem Verzeichniszugriffsserver verbunden sind, verwenden Sie Proxy-Tabellen zum Zugriff auf die Unterverzeichnisse des Computers. Datenbankbenutzer müssen über ein externes Login für den Verzeichniszugriffsserver verfügen.

Wenn ein Verzeichniszugriffsserver einmal erstellt wurde, können Sie ihn nicht mehr ändern. Falls es erforderlich wird, einen Verzeichniszugriffsserver zu ändern, müssen Sie ihn löschen und dann mit den gewünschten Einstellungen neu erstellen. Sie müssen zuerst alle Proxy-Tabellen löschen, die den Verzeichniszugriffsserver referenzieren und sie dann nach der Neuerstellung des Verzeichniszugriffsservers neu erstellen.

Es folgt eine Beschreibung des Formats der Proxy-Tabelle.

- **permissions VARCHAR(10)** Eine Berechtigungszeichenfolge im Posix-Stil, wie z.B. "drwxrwxrwx".
- **size BIGINT** Die Größe der Datei in Byte.
- **access_date_time TIMESTAMP** Datum und Uhrzeit des letzten Zugriffs auf die Datei (z.B. 2010-02-08 11:00:24.000).
- **modified_date_time TIMESTAMP** Datum und Uhrzeit der letzten Änderung der Datei (z.B. 2009-07-28 10:50:11.000).
- **create_date_time TIMESTAMP** Datum und Uhrzeit der Erstellung der Datei (z.B. 2008-12-18 10:32:26.000).
- **owner VARCHAR(20)** Die Benutzer-ID des Erstellers der Datei (z.B. "root" unter Linux). Für Windows ist dieser Wert immer "0".
- **file_name VARCHAR(260)** Der Name der Datei, einschließlich ein relativer Pfad (z.B. bin \perl.exe).

- **contents LONG BINARY** Der Inhalt der Datei, wenn diese Spalte explizit in der Ergebnismenge referenziert wird.

Erstellen von Verzeichniszugriffsservern (Sybase Central)

Administratoren können in Sybase Central den **Assistenten zum Erstellen von Verzeichniszugriffsservern** verwenden, um Verzeichniszugriffsserver zu erstellen.

Voraussetzungen

Sie müssen die Systemprivilegien **MANAGE ANY USER** und **SERVER OPERATOR** haben.

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.

Kontext und Bemerkungen

Beim Erstellen eines Verzeichniszugriffsservers können Sie die Anzahl der Unterverzeichnisse festlegen, auf die zugegriffen werden kann, und bestimmen, ob der Verzeichniszugriffsserver vorhandene Dateien ändern kann.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Hostdatenbank mithilfe des **SQL Anywhere 16-**Plug-Ins her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf **Verzeichniszugriffsserver** und klicken Sie auf **Neu » Verzeichniszugriffsserver**.
3. Befolgen Sie die Anweisungen im **Assistenten zum Erstellen von Verzeichniszugriffsservern**.
4. Rechtsklicken Sie in Sybase Central im linken Fensterausschnitt auf **Tabellen** und klicken Sie auf **Neu » Tabelle**.
5. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen einer Proxy-Tabelle**.

Ergebnisse

Ein Verzeichniszugriffsserver wird erstellt und konfiguriert.

Siehe auch

- „CREATE SERVER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE EXTERNLOGIN-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE EXISTING TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Erstellen von Verzeichniszugriffsservern (SQL)

Administratoren können in Interactive SQL die CREATE SERVER-Anweisung verwenden, um Verzeichniszugriffsserver zu erstellen.

Voraussetzungen

Sie müssen die Systemprivilegien SERVER OPERATOR und MANAGE ANY USER haben.

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.

Aufgabe

1. Verwenden Sie die CREATE SERVER-Anweisung, um einen Fremdserver zu erstellen.

```
CREATE SERVER my_dir_server
CLASS 'DIRECTORY'
USING 'ROOT=c:\Program Files\SUBDIRS=3';
```

2. Verwenden Sie die CREATE EXTERNLOGIN-Anweisung, um ein externes Login zu erstellen.

```
CREATE EXTERNLOGIN DBA TO my_dir_server;
```

3. Verwenden Sie die CREATE EXISTING TABLE-Anweisung, um eine Proxy-Tabelle für das Verzeichnis zu erstellen.

```
CREATE EXISTING TABLE my_program_files AT 'my_dir_server;;;';
```

In diesem Beispiel ist "my_program_files" der Name der Proxy-Tabelle und "my_dir_server" ist der Name des Verzeichniszugriffsservers.

4. Zeigen Sie Zeilen in der Proxy-Tabelle an.

```
SELECT * FROM my_program_files ORDER BY file_name;
```

5. Mithilfe der sp_remote_tables-Systemprozedur können Sie alle Unterverzeichnisse anzeigen, die sich unter c:\mydir auf dem Computer befinden, auf dem der Datenbankserver ausgeführt wird:

```
CALL sp_remote_tables( 'my_dir_server' );
```

Ergebnisse

Der Verzeichniszugriffsserver wird erstellt und konfiguriert.

Siehe auch

- „CREATE SERVER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE EXTERNLOGIN-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE EXISTING TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Beispiel: Dynamische Verzeichniszugriffsserver (SQL)

In diesem Beispiel können Administratoren dynamische Verzeichniszugriffsserver mit der CREATE SERVER-Anweisung mit Variablen für den Verzeichniszugriffsserver und die Unterverzeichnisebene erstellen.

Voraussetzungen

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Kontext und Bemerkungen

Nehmen wir an, Sie sind DBA und verfügen über eine Datenbank, die manchmal auf Computer A gestartet wird, mit dem Datenbankserver namens server1, und zu anderen Zeiten auf Computer B gestartet wird, mit dem Server namens server2. Angenommen, Sie wollen einen Verzeichniszugriffsserver einrichten, der sowohl auf das lokale Laufwerk *c:\temp* auf Computer A als auch auf das Netzwerkserver-Laufwerk *d:\temp* auf Computer B zeigt. Außerdem wollen Sie eine Gruppe von Proxy-Tabellen einrichten, aus der alle Benutzer die Liste ihrer eigenen privaten Verzeichnisse abrufen können. Durch die Verwendung von Variablen in der USING-Klausel einer CREATE SERVER-Anweisung und in der AT-Klausel einer CREATE EXISTING TABLE-Anweisung können Sie diesen Anforderungen gerecht werden, indem Sie einen einzigen Verzeichniszugriffsserver und eine einzige Proxy-Tabelle erstellen, und zwar folgendermaßen:

Aufgabe

1. In diesem Beispiel wird der Name des Servers, mit dem die Verbindung hergestellt werden soll, mit *server1* angenommen und es wird vorausgesetzt, dass die folgenden Verzeichnisse bereits existieren.

```
c:\temp\dba
c:\temp\updater
c:\temp\browser
```

Erstellen Sie den Verzeichniszugriffsserver mit Variablen für das Stammverzeichnis des Verzeichniszugriffsservers und die Unterverzeichnisebene.

```
CREATE SERVER dir
CLASS 'DIRECTORY'
USING 'root={@directory};subdirs={@subdirs}';
```

2. Erstellen Sie explizite externe Logins für jeden Benutzer, der berechtigt ist, den Verzeichniszugriffsserver zu verwenden.

```
CREATE EXTERNLOGIN "DBA" TO dir;
CREATE EXTERNLOGIN "UPDATER" TO dir;
CREATE EXTERNLOGIN "BROWSER" TO dir;
```

3. Erstellen Sie Variablen, die verwendet werden, um den Verzeichniszugriffsservers und die damit verbundenen Proxy-Tabellen dynamisch zu konfigurieren.

```
CREATE VARIABLE @directory LONG VARCHAR;
SET @directory = 'c:\\temp';
```



```
CREATE VARIABLE @subdirs VARCHAR(10);
SET @subdirs = '7';
```

```
CREATE VARIABLE @curuser VARCHAR(128);
SET @curuser = 'updater';
```

```
CREATE VARIABLE @server VARCHAR(128);
SET @server = 'dir';
```

4. Erstellen Sie eine Proxy-Tabelle, die auf `@directory\@curuser` auf dem Verzeichniszugriffsserver `@server` verweist.

```
CREATE EXISTING TABLE dbo.userdir AT '{@server};;;{@curuser}';
```

5. Die Variablen werden nicht mehr benötigt und können daher mithilfe der folgenden Anweisungen gelöscht werden:

```
DROP VARIABLE @server;
DROP VARIABLE @curuser;
DROP VARIABLE @subdirs;
DROP VARIABLE @directory;
```

6. Erstellen Sie die -Prozedur, die die Benutzer verwenden, um den Inhalt ihrer jeweiligen Benutzerverzeichnisse zu sehen.

```
CREATE OR REPLACE PROCEDURE dbo.listmydir()
SQL SECURITY INVOKER
BEGIN
    DECLARE @directory LONG VARCHAR;
    DECLARE @subdirs VARCHAR(10);
    DECLARE @server VARCHAR(128);
    DECLARE @curuser VARCHAR(128);

    -- for this example we always use the "dir" remote directory access
    server
    SET @server = 'dir';

    -- the root directory is based on the name of the server the user is
    connected to
    SET @directory = if property('name') = 'server1' then 'c:\\temp'
                    else 'd:\\temp' endif;

    -- the subdir limit is based on the connected user
    SET @curuser = user_name();

    -- all users get a subdir limit of 7 except "browser" who gets a
    limit of 1
    SET @subdirs = convert( varchar(10), if @curuser = 'browser' then 1
    else 7 endif);

    -- with all the variables set above, the proxy table dbo.userdir
    -- now points to @directory\@curuser and has a subdir limit of
    @subdirs
    SELECT * FROM dbo.userdir;

    DROP REMOTE CONNECTION TO dir CLOSE CURRENT;
END;
```

Der letzte Schritt in der Prozedur schließt die Verbindung zum Fremdserver, sodass der Benutzer die entfernten Tabellen auf dem Verzeichniszugriffsserver nicht auflisten kann (z.B. mithilfe der `sp_remote_tables`-Systemprozedur).

7. Legen Sie die erforderlichen Berechtigungen für die allgemeine Verwendung der gespeicherten Prozedur fest.

```
GRANT SELECT ON dbo.userdir TO PUBLIC;  
GRANT EXECUTE ON dbo.listmydir TO PUBLIC;
```

8. Trennen Sie die Verbindung zum Datenbankserver und stellen Sie eine neue Verbindung als UPDATER (Kennwort "update") oder BROWSER (Kennwort "browse") her. Führen Sie die folgende Abfrage aus.

```
CALL dbo.listmydir()
```

Ergebnisse

Der dynamische Verzeichniszugriffsserver wird erstellt und konfiguriert.

Siehe auch

- „CREATE SERVER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE EXTERNLOGIN-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE EXISTING TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DROP REMOTE CONNECTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Abfragen in Verzeichniszugriff-Proxy-Tabellen

Um die Performance zu steigern, vermeiden Sie die Spalte "contents" bei Abfragen, die zu einem Table-Scan führen. Verwenden Sie, soweit möglich, den Dateinamen, um den Inhalt einer Verzeichniszugriff-Proxy-Tabelle abzurufen. Die Verwendung des Dateinamens als Prädikat verbessert die Performance, weil der Verzeichniszugriffsserver nur die angegebene Datei liest. Wenn der Dateiname unbekannt ist, führen Sie zunächst eine Abfrage zum Abrufen der Dateiliste aus und anschließend eine Abfrage für jede Datei in der Liste zum Abrufen des jeweiligen Inhalts.

Beispiel 1

Die folgende Abfrage läuft möglicherweise langsam (je nach Anzahl und Größe der Dateien im Verzeichnis), weil der Verzeichniszugriffsserver den Inhalt aller Dateien im Verzeichnis lesen muss, um diejenigen zu finden, die mit dem Prädikat übereinstimmen:

```
SELECT contents FROM DirAccessProxyTable WHERE file_name LIKE 'something%';
```

Beispiel 2

Die folgende Abfrage gibt den Inhalt der einzelnen Datei zurück, ohne dass ein Verzeichnis-Scan ausgeführt wird:

```
SELECT contents FROM DirAccessProxyTable WHERE file_name = 'something';
```

Beispiel 3

Die folgende Abfrage läuft möglicherweise langsam (je nach Anzahl und Größe der Dateien im Verzeichnis), weil der Verzeichniszugriffsserver aufgrund des disjunkten OR einen Table-Scan ausführen muss:

```
SELECT contents FROM DirAccessProxyTable WHERE file_name = 'something' OR  
size = 10;
```

Beispiel 4

Als Alternative zum Einfügen des Dateinamens als Literalkonstante in der Abfrage können Sie den Dateinamenwert in eine Variable einsetzen und die Variable in der Abfrage verwenden:

```
DECLARE @filename LONG VARCHAR;  
SET @filename = 'something';  
SELECT contents FROM DirAccessProxyTable WHERE file_name = @filename;
```

Konsistenz der Trennzeichen

Beim Abfragen von Verzeichniszugriff-Proxy-Tabellen müssen Sie darauf achten, dass Pfadnamen-Trennzeichen konsistent verwendet werden. Am besten verwenden Sie das native Trennzeichen der jeweiligen Plattform: unter Windows \ und unter Unix /. Obwohl der Server unter Windows auch / als Trennzeichen erkennt, werden Dateinamen beim Ferndatenzugriff immer mit einem konsistenten Trennzeichen zurückgegeben. Daher gibt eine Abfrage mit inkonsistenten Trennzeichen keine Zeilen zurück.

Beispiel

Die folgende Abfrage gibt keine Zeilen zurück:

```
SELECT contents FROM DirAccessProxyTable WHERE filename = 'some/dir\thing';
```

Proxy-Tabellen des Verzeichniszugriffsservers löschen (Sybase Central)

Administratoren können mit Sybase Central Proxy-Tabellen löschen, die einem Verzeichniszugriffsserver zugeordnet sind.

Voraussetzungen

Sie müssen entweder Eigentümer sein oder das DROP ANY TABLE-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Kontext und Bemerkungen

Bevor Sie einen Verzeichniszugriffsserver löschen können, müssen Sie alle dem Verzeichniszugriffsserver zugeordneten Proxy-Tabellen löschen.

Löschen einer Proxy-Tabelle

1. Stellen Sie eine Verbindung zur Host-Datenbank mithilfe des SQL Anywhere 16-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Verzeichniszugriffsserver**. Danach klicken Sie auf den Verzeichniszugriffsserver mit der Proxy-Tabelle, die Sie löschen möchten.

3. Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **Proxy-Tabellen**.
4. Wählen Sie die Proxy-Tabelle aus und klicken Sie dann auf **Bearbeiten » Löschen**.
5. Klicken Sie auf **Ja**.

Ergebnisse

Die Proxy-Tabelle wird gelöscht.

Nächste Schritte

Nachdem alle dem Verzeichniszugriffsserver zugeordneten Proxy-Tabellen gelöscht wurden, können Sie den Verzeichniszugriffsserver löschen.

Siehe auch

- „Löschen von Verzeichniszugriffsservern (Sybase Central)“ auf Seite 802
- „DROP SERVER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DROP TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Löschen von Verzeichniszugriffsservern (Sybase Central)

Administratoren können Sybase Central verwenden, um Verzeichniszugriffsserver zu löschen.

Voraussetzungen

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Alle für den Verzeichniszugriffsserver definierten Proxy-Tabellen müssen entfernt werden, bevor Sie den Verzeichniszugriffsserver entfernen. Die folgende Abfrage kann verwendet werden, um zu ermitteln, welche Proxy-Tabellen für den Verzeichniszugriffsserver *server-name* definiert sind.

```
SELECT st.table_name, sp.remote_location, sp.existing_obj
FROM sysproxycat sp
JOIN sysserver ss ON ss.srvid = sp.srvid
JOIN systab st ON sp.table_object_id = st.object_id
WHERE ss.srvname = 'server-name';
```

Kontext und Bemerkungen

Sie können einen bestehenden Verzeichniszugriffsserver nicht ändern: Sie müssen den bestehenden Verzeichniszugriffsserver mit der Anweisung DROP SERVER löschen und dann einen neuen erstellen.

Verzeichniszugriffsserver löschen

1. Stellen Sie eine Verbindung zur Host-Datenbank mithilfe des SQL Anywhere 16-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Verzeichniszugriffsserver**.
3. Wählen Sie den Verzeichniszugriffsserver aus und klicken Sie dann auf **Bearbeiten » Löschen**.

Ergebnisse

Der Verzeichniszugriffsserver wird gelöscht.

Siehe auch

- „Löschen von Proxy-Tabellen (Sybase Central)“ auf Seite 809

Löschen von Verzeichniszugriffsservern (SQL)

Administratoren können Interactive SQL verwenden, um Verzeichniszugriffsserver zu löschen.

Voraussetzungen

Sie müssen das SERVER OPERATOR-Systemprivileg haben.

Alle für den Verzeichniszugriffsserver definierten Proxy-Tabellen müssen entfernt werden, bevor Sie den Verzeichniszugriffsserver entfernen. Die folgende Abfrage kann verwendet werden, um zu ermitteln, welche Proxy-Tabellen für den Verzeichniszugriffsserver *server-name* definiert sind.

```
SELECT st.table_name, sp.remote_location, sp.existing_obj
FROM sysproxycat sp
JOIN sysserver ss ON ss.srvname = sp.srvname
JOIN systab st ON sp.table_object_id = st.object_id
WHERE ss.srvname = 'server-name';
```

Kontext und Bemerkungen

Sie können einen bestehenden Verzeichniszugriffsserver nicht ändern: Sie müssen den bestehenden Verzeichniszugriffsserver mit der Anweisung DROP SERVER löschen und dann einen neuen erstellen.

Verzeichniszugriffsserver löschen

1. Stellen Sie eine Verbindung mit der Host-Datenbank her.
2. Führen Sie eine DROP TABLE-Anweisung für jede Proxy-Tabelle aus, die dem Verzeichniszugriffsserver zugeordnet ist.

```
DROP TABLE my_program_files;
```

3. Führen Sie eine DROP SERVER-Anweisung für den Verzeichniszugriffsserver aus.

```
DROP SERVER my_dir_server;
```

Ergebnisse

Der Verzeichniszugriffsserver wird gelöscht.

Siehe auch

- „Löschen von Proxy-Tabellen (Sybase Central)“ auf Seite 809

Externe Logins

Standardmäßig benutzt SQL Anywhere die Namen und Kennwörter des Clients, wenn für diese Clients eine Verbindung mit einem Fremdserver hergestellt wird. Dieser Standardwert kann aber durch die Einrichtung externer Logins außer Kraft gesetzt werden. Externe Logins sind alternative Login-Namen und Kennwörter, die bei der Kommunikation mit einem Fremdserver verwendet werden.

Siehe auch

- „Integrierte Windows-Logins“ [[SQL Anywhere Server - Datenbankadministration](#)]

Externe Logins erstellen (Sybase Central)

Verwenden Sie Sybase Central, um ein externes Login zu erstellen, das für die Kommunikation mit einem Fremdserver verwendet wird.

Voraussetzungen

Sie müssen das **MANAGE ANY USER**-Systemprivileg haben.

Ein externes Login erstellen

1. Stellen Sie eine Verbindung zur Host-Datenbank mithilfe des SQL Anywhere 16-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Fremdserver**.
3. Wählen Sie den Fremdserver aus und klicken Sie im rechten Fensterausschnitt auf die Registerkarte **Externe Logins**.
4. Klicken Sie im Menü **Datei** auf **Neu » Externes Login**.
5. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen von externen Logins**.

Ergebnisse

Das externe Login wird erstellt.

Siehe auch

- „CREATE EXTERNLOGIN-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Externe Logins löschen (Sybase Central)

Sie können Sybase Central verwenden, um nicht mehr benötigte externe Logins zu löschen.

Voraussetzungen

Sie müssen das **MANAGE ANY USER**-Systemprivileg haben.

Externe Logins löschen

1. Stellen Sie eine Verbindung zur Host-Datenbank mithilfe des SQL Anywhere 16-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Fremdserver**.
3. Wählen Sie den Fremdserver aus und klicken Sie im rechten Fensterausschnitt auf die Registerkarte **Externe Logins**.
4. Wählen Sie das externe Login aus und klicken Sie dann auf **Bearbeiten » Löschen**.
5. Klicken Sie auf **Ja**.

Ergebnisse

Das externe Login wird gelöscht.

Siehe auch

- „DROP EXTERNLOGIN-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Proxy-Tabellen

Die Transparenz des Standorts von entfernten Daten wird durch lokale **Proxy-Tabellen** ermöglicht, die dem entfernten Objekt zugeordnet werden. Sie können eine Proxy-Tabelle für den Zugriff auf beliebige Objekte (wie etwa Tabellen, Ansichten oder materialisierte Ansichten) benutzen, die von der entfernten Datenbank als Kandidaten für eine Proxy-Tabelle exportiert werden. Mit einer der folgenden Anweisungen können Sie eine Proxy-Tabelle erstellen:

- Wenn die Tabelle am entfernten Standort bereits existiert, benutzen Sie die Anweisung CREATE EXISTING TABLE. Diese Anweisung definiert die Proxy-Tabelle für eine bestehende Tabelle auf dem entfernten Server.
- Wenn die Tabelle am entfernten Standort noch nicht existiert, benutzen Sie die Anweisung CREATE TABLE. Diese Anweisung erstellt eine neue Tabelle auf dem Fremdserver und definiert die Proxy-Tabelle für diese Tabelle.

Hinweis

Sie können Daten in einer Proxy-Tabelle nicht ändern, wenn Sie sich an einem Savepoint befinden.

Wenn ein Trigger auf einer Proxy-Tabelle ausgelöst wird, sind die Berechtigungen die des Benutzers, der das Auslösen des Triggers bewirkt hat, und nicht die des Eigentümers der Proxy-Tabelle.

Siehe auch

- „Savepoints innerhalb von Transaktionen“ auf Seite 884

Standorte von Proxy-Tabellen

Das AT-Schlüsselwort wird mit den Anweisungen CREATE TABLE und CREATE EXISTING TABLE verwendet, um den Standort eines vorhandenen Objekts festzulegen. Diese Standortangabe hat vier Bestandteile, die durch Punkt oder Semikolon voneinander getrennt werden. Mit dem Semikolon-Trennzeichen können Dateinamen oder Erweiterungen in den Datenbank- und Eigentümerfeldern verwendet werden.

Die Syntax für die AT-Klausel lautet wie folgt:

```
... AT 'server.database.owner.table-name'
```

- **Datenbankserver** Dies ist der Name, unter dem der Server in der aktuellen Datenbank gemäß der Anweisung CREATE SERVER bekannt ist. Dieses Feld muss für alle entfernten Datenquellen angegeben werden.
- **Datenbank** Was in diesem Feld eingegeben wird, hängt von der Datenquelle ab. In einigen Fällen ist dieses Feld ohne Bedeutung und muss leer gelassen werden. Das Trennzeichen ist allerdings immer zu setzen.

Wenn es sich bei der Datenquelle um Adaptive Server Enterprise handelt, wird durch *database* jene Datenbank angegeben, in der sich die Tabelle befindet, z.B. master oder pubs2.

Wenn es sich bei der Datenquelle um SQL Anywhere handelt, wird dieses Feld nicht benutzt und bleibt leer.

Wenn es sich bei der Datenquelle um Excel, Lotus Notes oder Access handelt, müssen Sie den Namen der Datei angeben, die die Tabelle enthält. Wenn der Dateiname einen Punkt enthält, verwenden Sie das Semikolon als Trennzeichen.

- **Eigentümer** Wenn die Datenbank die Definition von Eigentümern unterstützt, wird in dieses Feld der Name des Eigentümers eingetragen. Dieses Feld ist nur erforderlich, wenn mehrere Eigentümer Tabellen mit demselben Namen haben.
- **Tabellenname** In diesem Feld wird der Name der Tabelle festgelegt. Bei einer Excel-Tabelle ist dies der Name des Arbeitsblatts in der Arbeitsmappe. Bei fehlender Angabe eines *table-name* in diesem Feld wird angenommen, dass der Name der entfernten Tabelle mit dem Namen der entsprechenden lokalen Proxy-Tabelle identisch ist.

Beispiele

Die folgenden Beispiele veranschaulichen die Verwendung von Standortangaben:

- SQL Anywhere:

```
'RemoteSA..GROUPO.Employees'
```

- Adaptive Server Enterprise:

```
'RemoteASE.pubs2.dbo.publishers'
```

- Excel:


```
'RemoteExcel;d:\pcdb\quarter3.xls;sheet1$'
```

- Access:

```
'RemoteAccessDB;\\server1\production\inventory.mdb;parts'
```

Proxy-Tabellen erstellen (Sybase Central)

Sie können Proxy-Tabellen mit Sybase Central erstellen.

Voraussetzungen

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.

Kontext und Bemerkungen

Sybase Central unterstützt nicht das Erstellen von Proxy-Tabellen für Systemtabellen. Diese können jedoch mithilfe der CREATE EXISTING TABLE-Anweisung erstellt werden.

Erstellen einer Proxy-Tabelle

1. Stellen Sie eine Verbindung zur Host-Datenbank mithilfe des SQL Anywhere 16-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Fremdserver**.
3. Wählen Sie den Fremdserver aus und klicken Sie im rechten Fensterausschnitt auf die Registerkarte **Proxy-Tabellen**.
4. Klicken Sie im Menü **Datei** auf **Neu » Proxy-Tabelle**.
5. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen einer Proxy-Tabelle**.

Ergebnisse

Die Proxy-Tabelle wird erstellt.

Siehe auch

- „CREATE EXISTING TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Erstellen von Proxy-Tabellen (SQL)

Sie können Proxy-Tabellen in Interactive SQL erstellen, indem Sie die Anweisung CREATE TABLE oder CREATE EXISTING TABLE ausführen.

Voraussetzungen

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.

Kontext und Bemerkungen

Die CREATE TABLE-Anweisung erstellt eine neue Tabelle auf dem Fremdservers und legt die Proxy-Tabelle für diese Tabelle fest, wenn Sie die AT-Klausel verwenden. Spalten werden mit den Datentypen von SQL Anywhere definiert. SQL Anywhere konvertiert die Daten automatisch in die nativen Datentypen des Fremdservers.

Wenn Sie die Anweisung CREATE TABLE verwenden, um eine lokale und eine entfernte Tabelle zu erstellen, und danach die Anweisung DROP TABLE benutzen, um die Proxy-Tabelle zu löschen, wird die entfernte Tabelle ebenfalls gelöscht. Sie können die Anweisung DROP TABLE jedoch benutzen, um eine Proxy-Tabelle zu löschen, die mit der Anweisung CREATE EXISTING TABLE erstellt wurde. In einem solchen Fall wird die entfernte Tabelle nicht gelöscht.

Die Anweisung CREATE EXISTING TABLE erstellt eine Proxy-Tabelle, die einer bestehenden Tabelle auf dem Fremdservers zugeordnet ist. SQL Anywhere leitet die Spaltenattribute und Indexinformationen vom Objekt am entfernten Standort ab.

Erstellen einer Proxy-Tabelle mit der CREATE EXISTING TABLE-Anweisung

1. Stellen Sie eine Verbindung mit der Host-Datenbank her.
2. Führen Sie eine CREATE EXISTING TABLE-Anweisung aus.

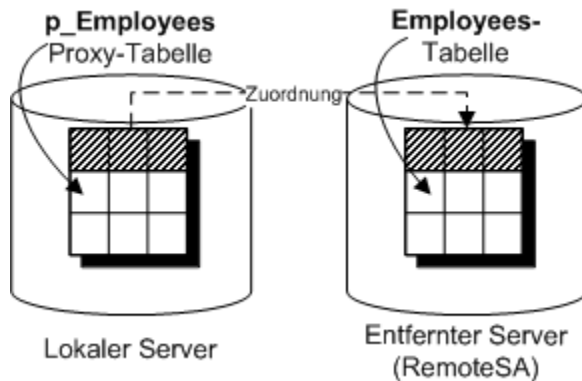
Ergebnisse

Die Proxy-Tabelle wird erstellt.

Beispiel

Um eine Proxy-Tabelle namens "p_Employees" auf dem aktuellen Server zu erstellen, die einer entfernten Tabelle namens "Employees" auf dem Server "RemoteSA" zugeordnet wird, verwenden Sie folgende Syntax:

```
CREATE EXISTING TABLE p_Employees  
AT 'RemoteSA..GROUPO.Employees';
```



Die folgende Anweisung ordnet der Microsoft Access-Datei *mydbfile.mdb* die Proxy-Tabelle "a1" zu. In diesem Beispiel wird für die AT-Klausel ein Semikolon (;) als Trennzeichen verwendet. Der für Microsoft Access definierte Server heißt "access".

```
CREATE EXISTING TABLE a1
AT 'access;d:\mydbfile.mdb;a1';
```

Die folgende Anweisung erstellt eine Tabelle namens "Employees" auf dem Fremdservers "RemoteSA" und eine Proxy-Tabelle namens "Members", die der entfernten Tabelle zugeordnet wird.

```
CREATE TABLE Members
( membership_id INTEGER NOT NULL,
  member_name CHAR( 30 ) NOT NULL,
  office_held CHAR( 20 ) NULL )
AT 'RemoteSA..GROUPO.Employees';
```

Siehe auch

- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE EXISTING TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Löschen von Proxy-Tabellen (Sybase Central)

Administratoren können mit Sybase Central Proxy-Tabellen löschen, die einem Fremdservers zugeordnet sind.

Voraussetzungen

Sie müssen entweder Eigentümer sein oder das DROP ANY TABLE-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Kontext und Bemerkungen

Bevor Sie einen Fremdservers löschen können, müssen Sie alle dem Fremdservers zugeordneten Proxy-Tabellen löschen.

Löschen einer Proxy-Tabelle

1. Stellen Sie eine Verbindung zur Host-Datenbank mithilfe des SQL Anywhere 16-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Fremdserver**.
3. Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **Proxy-Tabellen**.
4. Wählen Sie die Proxy-Tabelle aus und klicken Sie dann auf **Bearbeiten » Löschen**.
5. Klicken Sie auf **Ja**.

Ergebnisse

Die Proxy-Tabelle wird gelöscht.

Nächste Schritte

Nachdem alle dem Fremdserver zugeordneten Proxy-Tabellen gelöscht wurden, können Sie Fremdserver löschen.

Siehe auch

- „Fremdserver entfernen (Sybase Central)“ auf Seite 792
- „DROP SERVER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „DROP TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Spalten in einer entfernten Tabelle auflisten

Bevor Sie die Anweisung CREATE EXISTING TABLE ausführen, sollten Sie eine Liste der Spalten erstellen, die in einer entfernten Tabelle zur Verfügung stehen. Die Systemprozedur sp_remote_columns gibt eine Liste der Spalten in einer entfernten Tabelle und eine Beschreibung ihrer Datentypen aus. Die Systemprozedur "sp_remote_columns" hat folgende Syntax:

```
CALL sp_remote_columns( @server_name, @table_name [, @table_owner [,  
@table_qualifier ] ] )
```

Wenn ein Tabellename, Eigentümer oder Datenbankname angegeben werden, wird die Liste auf die passenden Spalten reduziert.

Folgende Anweisung listet z.B. die Spalten in der Tabelle sysobjects der Produktdatenbank auf dem Adaptive Server Enterprise-Server asetest auf:

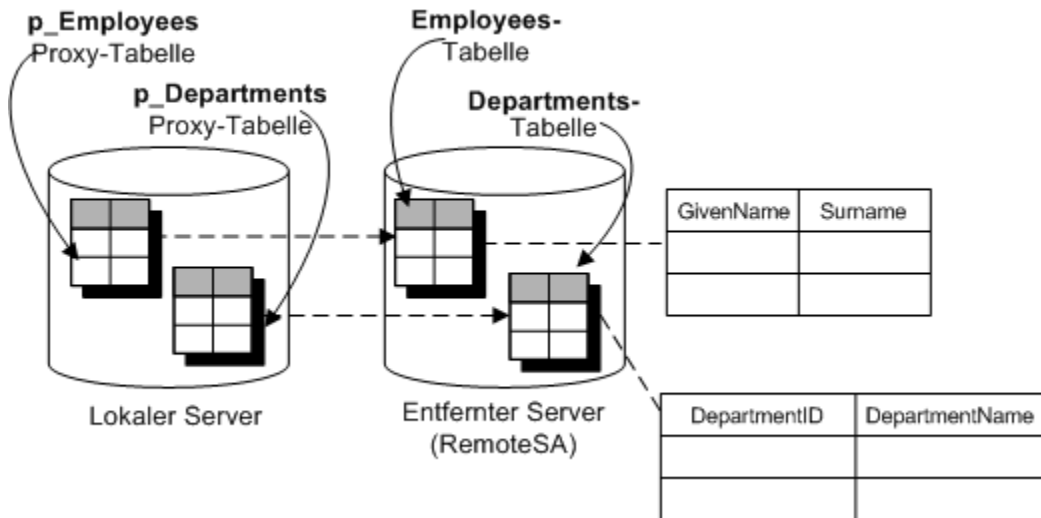
```
CALL sp_remote_columns('asetest', 'sysobjects', null, 'production');
```

Siehe auch

- „sp_remote_columns-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Joins zwischen entfernten Tabellen

Die folgende Abbildung zeigt Proxy-Tabellen auf einem lokalen Datenbankserver, die den entfernten Tabellen "Employees" und "Departments" der SQL Anywhere-Beispieldatenbank auf dem Fremdserver "RemoteSA" zugeordnet sind.



Sie können Joins zwischen Tabellen in unterschiedlichen SQL Anywhere-Datenbanken benutzen. Im folgenden, einfachen Beispiel wird zur Verdeutlichung des Prinzips nur eine Datenbank benutzt.

Beispiel

Führen Sie einen Join zwischen zwei entfernten Tabellen aus:

1. Erstellen Sie eine neue Datenbank namens *empty.db*.

Diese Datenbank enthält keine Daten. Sie wird lediglich benutzt, um die entfernten Objekte zu definieren und um auf die SQL Anywhere-Beispieldatenbank zuzugreifen.

2. Starten Sie den Datenbankserver, der auf dem *empty.db* ausgeführt wird. Dazu können Sie den folgenden Befehl ausführen:

```
dbsrv16 empty
```

3. Stellen Sie in Interactive SQL als DBA-Benutzer eine Verbindung zu *empty.db* her.
4. Erstellen Sie in der neuen Datenbank einen Fremdserver namens "RemoteSA". Dessen Serverklasse ist SAODBC und die Verbindungszeichenfolge bezieht sich auf die ODBC-Datenquelle von SQL Anywhere 16 Demo:

```
CREATE SERVER RemoteSA
CLASS 'SAODBC'
USING 'SQL Anywhere 16 Demo';
```

5. In diesem Beispiel wird in der entfernten Datenbank dieselbe Benutzer-ID und das gleiche Kennwort wie in der lokalen Datenbank verwendet, sodass keine externen Logins erforderlich sind.

In einigen Fällen müssen Sie eine Benutzer-ID und ein Kennwort angeben, wenn Sie eine Verbindung zur Datenbank auf dem Fremdserver herstellen. In der neuen Datenbank können Sie ein externes Login zum Fremdserver erstellen. Um das Beispiel zu vereinfachen, wird "DBA" hier sowohl als lokaler Login-Name als auch als ID des entfernten Benutzers verwendet:

```
CREATE EXTERNLOGIN DBA
TO RemoteSA
REMOTE LOGIN DBA
IDENTIFIED BY sql;
```

6. Definieren Sie die Proxy-Tabelle "p_Employees":

```
CREATE EXISTING TABLE p_Employees
AT 'RemoteSA..GROUPO.Employees';
```

7. Definieren Sie die Proxy-Tabelle "p_Departments":

```
CREATE EXISTING TABLE p_Departments
AT 'RemoteSA..GROUPO.Departments';
```

8. Benutzen Sie die Proxy-Tabellen in der SELECT-Anweisung, um den Join auszuführen.

```
SELECT GivenName, Surname, DepartmentName
FROM p_Employees JOIN p_Departments
ON p_Employees.DepartmentID = p_Departments.DepartmentID
ORDER BY Surname;
```

Joins zwischen Tabellen aus mehreren lokalen Datenbanken

Ein SQL Anywhere-Server kann mehrere lokale Datenbanken gleichzeitig ausführen. Durch die Definition von Tabellen in anderen lokalen SQL Anywhere-Datenbanken als entfernte Tabellen können Sie Joins über Datenbanken hinweg vornehmen.

Weitere Hinweise über das Angeben mehrerer Datenbanken finden Sie unter „[USING-Klausel in der CREATE SERVER-Anweisung](#)“ auf Seite 825.

Beispiel

Angenommen, Sie benutzen Datenbank "db1", und Sie möchten auf Daten zugreifen, die sich in Tabellen der Datenbank "db2" befinden. Dazu müssen Sie Definitionen für Proxy-Tabellen einrichten, welche auf die Tabellen in der Datenbank "db2" verweisen. Auf einem SQL Anywhere-Server namens "RemoteSA" könnten beispielsweise die drei Datenbanken "db1", "db2" und "db3" verfügbar sein.

1. Wenn Sie ODBC verwenden, erstellen Sie für jede Datenbank, auf die Sie zugreifen, einen Namen für die ODBC-Datenquelle.
2. Verbinden Sie sich mit der Datenbank, aus der Sie den Join herstellen werden. Verbinden Sie sich z.B. mit db1.

3. Führen Sie die Anweisung CREATE SERVER für jede weitere lokale Datenbank aus, auf die Sie zugreifen. Damit stellen Sie eine **Loopback**-Verbindung zu Ihrem SQL Anywhere-Server her.

```
CREATE SERVER remote_db2
CLASS 'SAODBC'
USING 'RemoteSA_db2';
CREATE SERVER remote_db3
CLASS 'SAODBC'
USING 'RemoteSA_db3';
```

4. Erstellen Sie mithilfe von CREATE EXISTING TABLE-Anweisungen Proxy-Tabellen-Definitionen für die Tabellen in den anderen Datenbanken, auf die Sie zugreifen wollen.

```
CREATE EXISTING TABLE Employees
AT 'remote_db2...Employees';
```

Native Anweisungen und Fremdserver

Benutzen Sie die Anweisung FORWARD TO, um Anweisungen in der nativen Syntax an einen Fremdserver zu senden. Diese Anweisung kann auf zwei Arten eingesetzt werden:

- Anweisung an einen Fremdserver senden.
- Versetzt SQL Anywhere in den Durchreichmodus, um eine Serie von Anweisungen an einen Fremdserver zu senden.

Die Anweisung FORWARD TO kann benutzt werden, um zu prüfen, ob ein Server richtig konfiguriert ist. Wenn Sie eine Anweisung an den Fremdserver senden und SQL Anywhere keine Fehlermeldung zurückgibt, ist der Fremdserver richtig konfiguriert.

Die Anweisung FORWARD TO kann nicht innerhalb von Prozeduren oder Batches verwendet werden.

Wenn eine Verbindung mit einem angegebenen Server nicht hergestellt werden kann, wird der Grund dafür in einer Meldung ausgegeben. Wenn die Verbindung hergestellt wird, werden Ergebnisse in eine Form umgewandelt, die vom Clientprogramm erkannt werden kann.

Beispiel 1

Die folgende Anweisung prüft die Systemanbindung des Servers "RemoteASE", indem eine Auswahlabfrage der Versionsdaten vorgenommen wird:

```
FORWARD TO RemoteASE {SELECT @@version};
```

Beispiel 2

Die folgenden Anweisungen zeigen eine Durchreichsitzung mit dem Server namens "RemoteASE":

```
FORWARD TO RemoteASE;
SELECT * FROM titles;
SELECT * FROM authors;
FORWARD TO;
```

Siehe auch

- „FORWARD TO-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Entfernte Prozeduraufrufe

SQL Anywhere-Benutzer können Prozeduraufrufe an die folgenden Fremdserver ausgeben:

- SQL Anywhere
- Adaptive Server Enterprise
- Oracle-Datenbank
- IBM DB2

SQL Anywhere unterstützt das Abrufen von Ergebnismengen von entfernten Prozeduren und auch das Abrufen von mehreren Ergebnismengen. Entfernte Funktionen können außerdem benutzt werden, um Rückgabewerte von entfernten Prozeduren und Funktionen abzurufen. Entfernte Prozeduren können in der FROM-Klausel einer SELECT-Anweisung benutzt werden.

Datentypen für entfernte Prozeduren

Die folgenden Datentypen sind für RPC-Parameter und Rückgabewerte zulässig:

- [UNSIGNED] SMALLINT
- [UNSIGNED] INTEGER
- [UNSIGNED] BIGINT
- [UNSIGNED] TINYINT
- TIME
- DATE
- TIMESTAMP
- REAL
- DOUBLE
- CHAR
- BIT
- Die Datentypen LONG VARCHAR, LONG NVARCHAR und LONG BINARY sind für IN-Parameter zulässig, nicht jedoch für OUT- oder INOUT-Parameter oder RETURNS-Werte.
- Die Datentypen NUMERIC und DECIMAL sind für IN-Parameter zulässig, nicht jedoch für OUT- oder INOUT-Parameter oder RETURNS-Werte.

Erstellen von entfernten Prozeduren (SQL)

Sie können entfernte Prozeduren und Funktionen in Interactive SQL erstellen.

Voraussetzungen

Sie müssen das CREATE PROCEDURE-Systemprivileg haben, um Prozeduren und Funktionen erstellen zu können, deren Eigentümer Sie sind. Sie müssen das CREATE ANY PROCEDURE-Privileg oder das CREATE ANY OBJECT-Privileg haben, um Prozeduren und Funktionen erstellen zu können, deren

Eigentümer andere Benutzer sind. Wenn Sie externe Prozeduren und Funktionen erstellen möchten, müssen Sie außerdem das CREATE EXTERNAL REFERENCE-Systemprivileg haben.

Kontext und Bemerkungen

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.

Entfernte Prozedur erstellen

1. Stellen Sie eine Verbindung mit der Host-Datenbank her.
2. Führen Sie eine Anweisung aus, um die Prozedur oder Funktion zu definieren.

Beispiel:

```
CREATE PROCEDURE RemoteProc()
AT 'bostonase.master.dbo.sp_proc';

CREATE FUNCTION RemoteFunc()
RETURNS INTEGER
AT 'bostonasa..dbo.sp_func';
```

Die Syntax ist ähnlich einer lokalen Prozedurdefinition. Die Speicherort-Zeichenfolge legt den Pfad der Prozedur fest.

Ergebnisse

Die entfernte Prozedur oder Funktion wird erstellt.

Beispiel

Das Beispiel gibt einen Parameter an, wenn eine entfernte Prozedur aufgerufen wird:

```
CREATE PROCEDURE RemoteUser ( IN username CHAR( 30 ) )
AT 'bostonase.master.dbo.sp_helpuser';
CALL RemoteUser( 'joe' );
```

Dieses Beispiel erstellt eine Schnittstelle zu einer Funktion auf dem Fremdservers RemoteSA:

```
CREATE FUNCTION proxy_maxorder()
RETURNS INTEGER
AT 'RemoteSA;;DBA;maxorder';
```

Siehe auch

- „CREATE FUNCTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE PROCEDURE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Erstellen von entfernten Prozeduren (Sybase Central)

Administratoren können Sybase Central verwenden, um eine entfernte Prozedur zu erstellen.

Voraussetzungen

Sie müssen das CREATE PROCEDURE-Systemprivileg haben, um Prozeduren und Funktionen erstellen zu können, deren Eigentümer Sie sind. Sie müssen das CREATE ANY PROCEDURE-Privileg oder das CREATE ANY OBJECT-Privileg haben, um Prozeduren und Funktionen erstellen zu können, deren Eigentümer andere Benutzer sind. Wenn Sie externe Prozeduren und Funktionen erstellen möchten, müssen Sie außerdem das CREATE EXTERNAL REFERENCE-Systemprivileg haben.

Entfernte Prozedur erstellen

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.

1. Stellen Sie eine Verbindung zur Host-Datenbank mithilfe des SQL Anywhere 16-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Fremdserver**.
3. Wählen Sie den Fremdserver aus und klicken Sie im rechten Fensterausschnitt auf die Registerkarte **Entfernte Prozeduren**.
4. Klicken Sie im Menü **Datei** auf **Neu » Entfernte Prozedur**.
5. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen von entfernten Prozeduren**.

Ergebnisse

Die entfernte Prozedur wird erstellt.

Siehe auch

- „CREATE FUNCTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE PROCEDURE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Entfernte Prozeduren löschen (SQL)

Administratoren können entfernte Prozeduren mit SQL-Anweisungen löschen.

Voraussetzungen

Keine Privilegien erforderlich.

Entfernte Prozeduren löschen

- Führen Sie eine Anweisung aus, um die Prozedur oder Funktion zu löschen.

```
DROP PROCEDURE RemoteProc;
```

```
DROP FUNCTION RemoteFunc;
```

Ergebnisse

Die entfernte Prozedur oder Funktion wird gelöscht.

Siehe auch

- „DROP FUNCTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DROP PROCEDURE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Entfernte Prozeduren löschen (Sybase Central)

Administratoren können entfernte Prozeduren und Funktionen in Sybase Central löschen.

Voraussetzungen

Keine Privilegien erforderlich.

Löschen einer entfernten Prozedur

1. Stellen Sie eine Verbindung zur Host-Datenbank mithilfe des SQL Anywhere 16-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Fremdserver**.
3. Wählen Sie den Fremdserver aus und klicken Sie im rechten Fensterausschnitt auf die Registerkarte **Entfernte Prozeduren**.
4. Wählen Sie die entfernte Prozedur oder Funktion aus und klicken Sie dann auf **Bearbeiten » Löschen**.
5. Klicken Sie auf **Ja**.

Ergebnisse

Die entfernte Prozedur wird gelöscht.

Siehe auch

- „DROP FUNCTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DROP PROCEDURE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Transaktionsverwaltung und entfernte Daten

Transaktionen stellen eine Möglichkeit dar, SQL-Anweisungen zu gruppieren, sodass sie als Einheit verarbeitet werden. Danach wird entweder der komplette Vorgang in der Datenbank festgeschrieben oder gar nichts.

In fast allen Aspekten ist die Transaktionsverwaltung mit entfernten Tabellen die gleiche wie bei lokalen Tabellen in SQL Anywhere, mit einigen Ausnahmen.

Siehe auch

- „Transaktionen und Isolationsstufen“ auf Seite 881

Entfernte Transaktionsverwaltung

Die Methode zur Verwaltung von Transaktionen unter Einbeziehung von Fremdservern benutzt ein Zwei-Phasen-Commit-Protokoll. SQL Server Anywhere implementiert eine Strategie, die in den meisten Fällen die Integrität der Transaktionen gewährleistet. Wenn mehr als ein Fremdserver in eine Transaktion einbezogen werden, besteht trotzdem die Möglichkeit, dass eine Arbeitseinheit in einem unbestimmten Zustand verbleibt. Obwohl das Zwei-Phasen-Commit-Protokoll benutzt wird, gibt es keinen integrierten Wiederherstellungsprozess.

Die allgemeine Logik für die Verwaltung einer Benutzertransaktion ist wie folgt:

1. SQL Anywhere stellt den an einen Fremdserver gesendeten Aufgaben die Mitteilung BEGIN TRANSACTION voran.
2. Wenn die Transaktion zum Festschreiben bereit ist, sendet SQL Anywhere die Mitteilung PREPARE TRANSACTION an jeden Fremdserver, der an der Transaktion teilgenommen hat. Damit wird sichergestellt, dass der Fremdserver bereit ist, die Transaktion festzuschreiben.
3. Wenn eine PREPARE TRANSACTION-Anforderung fehlschlägt, werden alle Fremdserver angewiesen, die aktuelle Transaktion zurückzusetzen.

Wenn alle PREPARE TRANSACTION-Anforderungen erfolgreich sind, sendet der Server eine COMMIT TRANSACTION-Anforderung an jeden Fremdserver, der an der Transaktion beteiligt ist.

Jede Anweisung mit der Einleitung BEGIN TRANSACTION kann eine Transaktion beginnen. Andere Anweisungen werden am Fremdserver als einzelne, entfernte Arbeitseinheit ausgeführt.

Einschränkungen bei der Transaktionsverwaltung

Folgende Einschränkungen gelten für die Transaktionsverwaltung:

- Savepoints werden nicht an Fremdserver weitergegeben.
- Wenn verschachtelte BEGIN TRANSACTION- und COMMIT TRANSACTION-Anweisungen in eine Transaktion eingeschlossen werden, die mit Fremdservern arbeitet, wird nur der äußerste Mantel von Transaktionen verarbeitet. Der innere Kern, der die BEGIN TRANSACTION- und COMMIT TRANSACTION-Anweisungen enthält, wird nicht an Fremdserver übermittelt.

Interne Vorgänge

In diesem Abschnitt werden die zugrunde liegenden Schritte beschrieben, die SQL Anywhere für Clientanwendungen auf Fremdservern durchführt.

Interne Vorgänge bei Abfragen

Die folgenden Verarbeitungsschritte werden bei allen Abfragen, sowohl lokalen als auch entfernten, ausgeführt:

Syntaktische Analyse der Abfragen

Wenn eine Anweisung von einem Client eintrifft, wird sie vom Datenbankserver syntaktisch analysiert. Der Datenbankserver meldet einen Fehler, wenn es sich nicht um eine gültige SQL-Anweisung für SQL Anywhere handelt.

Abfragenormalisierung

Referenzierte Objekte in der Abfrage werden geprüft und Datentypen werden auf Kompatibilität untersucht.

Nehmen wir als Beispiel folgende Abfrage:

```
SELECT *  
FROM t1  
WHERE c1 = 10;
```

Im Stadium der Abfragenormalisierung soll überprüft werden, dass die Tabelle t1 mit der Spalte c1 in den Systemtabellen existiert. Es wird auch geprüft, ob der Datentyp der Spalte c1 mit dem Wert 10 kompatibel ist. Wenn zum Beispiel der Datentyp der Spalte TIMESTAMP ist, wird diese Anweisung zurückgewiesen.

Vorverarbeitung der Abfrage

Während der Vorverarbeitung wird die Abfrage optimiert. In dieser Phase kann die Darstellung einer Abfrage geändert werden, sodass die SQL-Anweisung, die SQL Anywhere für die Weitergabe an einen Fremdserver generiert, syntaktisch von der ursprünglichen Anweisung abweicht, auch wenn sie semantisch gleichwertig ist.

Die Vorverarbeitung führt eine Ansichtserweiterung durch, damit eine Abfrage in Tabellen ausgeführt werden kann, die von der Ansicht referenziert werden. Die Ausdrücke können neu geordnet und Unterabfragen können umformuliert werden, um den Wirkungsgrad der Verarbeitung zu erhöhen. Es könnte zum Beispiel vorkommen, dass bestimmte Unterabfragen in Joins umgewandelt werden.

Serverfunktionalität

Welche Verfahrensschritte ausgeführt werden, hängt von der Art der SQL-Anweisung und der Funktionalität der jeweiligen Fremdserver ab.

In SQL Anywhere verfügt jeder Fremdserver über eine Reihe von Funktionalitäten, die für ihn definiert wurden. Diese Funktionen werden in der ISYSCAPABILITIES-Systemtabelle gespeichert und während der ersten Verbindung zu einem Fremdserver initialisiert.

Die generische Serverklasse ODBC verlässt sich beim Ermitteln dieser Funktionen strikt auf die Informationen, die vom ODBC-Treiber zurückgegeben werden. Andere Serverklassen wie DB2ODBC haben detailliertere Kenntnisse der Funktionalitäten eines Fremdservers und benutzen dieses Wissen, um die vom Server zurückgegebenen Ergebnisse zu ergänzen.

Wenn ein Server zu ISYSCAPABILITIES hinzugefügt wurde, werden die Informationen über die Funktionalitäten nur noch aus der Systemtabelle abgerufen.

Da ein Fremdserver möglicherweise nicht alle Funktionen einer bestimmten SQL-Anweisung unterstützt, schlüsselt SQL Anywhere die Anweisung in einfachere Komponenten auf, damit die Abfrage an den Fremdserver übermittelt werden kann. SQL-Funktionen, die nicht an einen Fremdserver weitergegeben wurden, müssen von SQL Anywhere selbst ausgewertet werden.

Eine Abfrage kann beispielsweise eine ORDER BY-Anweisung enthalten. Wenn ein Fremdserver ORDER BY nicht verarbeiten kann, wird die Anweisung ohne die Klausel an den Fremdserver geschickt und SQL Anywhere führt die Funktion ORDER BY in der zurückgegebenen Ergebnismenge durch, bevor sie an den Benutzer weitergegeben wird. Der Benutzer kann daher alle von SQL Anywhere unterstützten SQL-Anweisungen einsetzen.

Komplettes Passthrough der Anweisung

Um eine möglichst hohe Effizienz zu gewährleisten, reicht SQL Anywhere so viele Anweisungen wie möglich an den Fremdserver durch. Häufig ist dies die vollständige Anweisung, die ursprünglich an SQL Anywhere übergeben wurde.

SQL Anywhere gibt unter folgenden Bedingungen die komplette Anweisung weiter:

- Alle Tabellen der Anweisung befinden sich auf demselben Fremdserver.
- Der Fremdserver kann alle Syntaxbestandteile der Anweisung verarbeiten.

In seltenen Fällen kann es wirkungsvoller sein, SQL Anywhere die Verarbeitung von Anweisungsteilen zu überlassen, statt dies vom Fremdserver durchführen zu lassen. SQL Anywhere hat z.B. einen besseren Algorithmus für die Datensortierung. In diesem Fall können Sie die Funktionalitäten eines Fremdservers mit einer ALTER SERVER Anweisung verändern.

Siehe auch

- „ALTER SERVER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Partielles Passthrough der Anweisung

Wenn eine Anweisung Bezugnahmen auf mehrere Server enthält oder SQL-Funktionen benutzt, die von einem Fremdserver nicht unterstützt werden, wird die Abfrage in einfachere Teile zerlegt.

SELECT

SELECT-Anweisungen werden aufgeteilt, indem Teile entfernt werden, die nicht weitergegeben werden können und daher von SQL Anywhere verarbeitet werden. Nehmen wir z.B. an, dass ein Fremdserver die ATAN2-Funktion in der folgenden Anweisung nicht verarbeiten kann:

```
SELECT a,b,c
WHERE ATAN2( b, 10 ) > 3
AND c = 10;
```

Die an den Fremdserver gesendete Anweisung würde wie folgt umgewandelt werden:

```
SELECT a,b,c WHERE c = 10;
```

SQL Anywhere wendet dann WHERE ATAN2(b, 10) > 3 lokal auf die Zwischenergebnismenge an.

Joins

Wenn zwei Tabellen verknüpft werden, wird eine Tabelle als äußere Tabelle definiert. Die äußere Tabelle wird basierend auf den WHERE-Bedingungen durchsucht, die auf sie passen. Für jede gefundene qualifizierte Zeile wird die andere Tabelle, die innere Tabelle, auf eine Zeile durchsucht, die der Join-Bedingung entspricht.

Derselbe Algorithmus wird verwendet, wenn auf entfernte Tabellen Bezug genommen wird. Da der Aufwand für das Durchsuchen einer entfernten Tabelle in der Regel höher ist als bei einer lokalen Tabelle (wegen des erforderlichen Datenverkehrs im Netzwerk), sollte alles getan werden, dass im Join die entfernte Tabelle als äußere Tabelle definiert wird.

UPDATE und DELETE

Wenn eine qualifizierende Zeile gefunden wird und SQL Anywhere eine UPDATE- oder DELETE-Anweisung nicht komplett an einen Fremdserver weitergeben kann, muss er die Anweisung in einen Tabellensuchvorgang umwandeln, der so viel von der ursprünglichen WHERE-Klausel wie möglich enthält. Darauf folgt eine positionsgesteuerte UPDATE- oder DELETE-Anweisung, die WHERE CURRENT OF *cursor-name* angibt.

Wenn z.B. die Funktion ATAN2 von einem Fremdserver nicht unterstützt wird, gestaltet sich der Vorgang folgendermaßen:

```
UPDATE t1
SET a = atan2( b, 10 )
WHERE b > 5;
```

Diese Anweisung wird wie folgt umgewandelt:

```
SELECT a,b
FROM t1
WHERE b > 5;
```

Wenn eine Zeile gefunden wurde, berechnet SQL Anywhere den neuen Wert von "a" und führt Folgendes aus:

```
UPDATE t1
SET a = 'new value'
WHERE CURRENT OF CURSOR;
```

Wenn "a" bereits einen Wert hat, der dem neuen Wert entspricht, ist ein positionsbasiertes UPDATE nicht erforderlich und wird daher nicht an den Fremdserver geschickt.

Um eine UPDATE- oder DELETE-Anweisung auszuführen, für die das Durchsuchen einer Tabelle erforderlich ist, muss die entfernte Datenquelle ein positionsbasiertes UPDATE oder DELETE (WHERE CURRENT OF *cursor-name*) unterstützen. Einige Datenquellen unterstützen diese Funktion nicht.

Temporäre Tabellen können nicht aktualisiert werden

Wenn eine temporäre Tabelle erforderlich ist, kann keine UPDATE oder DELETE-Anweisung durchgeführt werden. Dies kommt in Abfragen mit ORDER BY und einigen Abfragen mit Unterabfragen vor.

Fehlerbehandlung beim Ferndatenzugriff

In diesem Abschnitt finden Sie Hinweise zur Fehlerbehandlung beim Zugriff auf Fremdserver.

Für den Ferndatenzugriff nicht unterstützte Funktionen

Die im Folgenden beschriebenen Funktionen von SQL Anywhere werden bei Ferndaten nicht unterstützt:

- ALTER TABLE-Anweisung in entfernten Tabellen.
- Trigger, die für Proxy-Tabellen definiert wurden.
- SQL Remote.
- Fremdschlüssel, die entfernte Tabellen referenzieren.
- Die Funktionen READTEXT, WRITETEXT und TEXTPTR.
- Positionsbasierte UPDATE- und DELETE-Anweisungen.
- UPDATE- und DELETE-Anweisungen, für die eine temporäre Zwischentabelle erforderlich ist.
- Rückwärts-Abrollen in Cursors, die für Ferndaten geöffnet wurden. Die Abrufanweisungen müssen NEXT oder RELATIVE 1 sein.
- Aufrufe von Funktionen, die einen Ausdruck enthalten, der eine Proxy-Tabelle referenziert.
- Wenn eine Spalte in einer entfernten Tabelle einen Namen hat, der als Schlüsselwort auf dem Fremdserver verwendet wird, können Sie auf Daten in dieser Spalte nicht zugreifen. Sie können eine CREATE EXISTING TABLE-Anweisung ausführen und die Definition importieren, diese Spalte jedoch nicht auswählen.

Groß-/Kleinschreibung

Die Einstellung für die Berücksichtigung von Groß- und Kleinschreibung in der SQL Anywhere-Datenbank muss den Einstellungen entsprechen, die auf Fremdservern gelten.

Die SQL Anywhere-Datenbanken werden standardmäßig ohne Berücksichtigung von Groß- und Kleinschreibung erstellt. In dieser Konfiguration kann es zu unvorhersehbaren Ergebnissen kommen, wenn Sie eine Auswahlabfrage in einer Datenbank vornehmen, in der die Groß- und Kleinschreibung berücksichtigt wird. Unterschiedliche Ergebnisse können ausgegeben werden, je nachdem, ob ORDER

BY oder der Vergleich von Zeichenfolgen an einen Fremdserver übergeben oder vom lokalen SQL Anywhere-Server verarbeitet werden.

Systemanbindungstests

Führen Sie folgende Maßnahmen durch, um sicherzustellen, dass Sie die Verbindung zu einem Fremdserver herstellen können:

- Vergewissern Sie sich, dass Sie sich über einen Client wie z.B. Interactive SQL mit einem Fremdserver verbinden können, bevor Sie SQL Anywhere konfigurieren.
- Führen Sie eine einfache Durchreichen-Anweisung in einem Fremdserver aus, um die Systemanbindung und die Konfiguration des entfernten Logins zu testen. Beispiel:

```
FORWARD TO RemoteSA {SELECT @@version};
```

- Aktivieren Sie die Fernprotokollierung, um das Zusammenspiel mit Fremdservern zu verfolgen. Beispiel:

```
SET OPTION cis_option = 7;
```

Wenn Sie die Fernprotokollierung eingeschaltet haben, werden die Protokollierungsinformationen im Fenster "Datenbankservermeldungen" angezeigt. Sie können diese Ausgabe in eine Datei schreiben, indem Sie die Serveroption -o verwenden, wenn Sie den Datenbankserver starten.

Siehe auch

- „cis_option-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenbankserveroption -o“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „PASSTHROUGH-Anweisung [SQL Remote]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Selbstblockierende Abfragen

Sie brauchen genügend Threads zur Unterstützung der einzelnen Aufgaben, die von einer Abfrage abgewickelt werden. Wenn nicht genügend Aufgaben bereit stehen, kann eine Abfrage sich selbst blockieren.

Siehe auch

- „Transaktion blockieren und Deadlock“ auf Seite 901

Verbindungen für Ferndatenzugriff über ODBC

Wenn Sie auf entfernte Datenbanken mittels ODBC zugreifen, wird der Verbindung zum Fremdserver ein Name zugeteilt. Sie können den Namen zum Beenden der Verbindung verwenden, um eine Fernanforderung abzuberechnen.

Die Verbindungen werden ASACIS_*conn-name* benannt, wobei *conn-name* die Verbindungs-ID der lokalen Verbindung darstellt. Die Verbindungs-ID können Sie der gespeicherten Prozedur "sa_conn_info" entnehmen.

Siehe auch

- „sa_conn_info-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Serverklassen für den Ferndatenzugriff

Die Serverklasse, die Sie in der CREATE SERVER-Anweisung angeben, bestimmt das Verhalten einer entfernten Verbindung. Die Serverklassen geben SQL Anywhere detaillierte Informationen über die Serverfunktionen. SQL Anywhere formatiert SQL-Anweisungen unter Berücksichtigung der Möglichkeiten der jeweiligen Serverklasse.

Alle Serverklassen sind ODBC-basiert. Jede Serverklasse verfügt über eine Gruppe von eindeutigen Eigenschaften, die Sie kennen müssen, um den Server für den Fernzugriff richtig zu konfigurieren. Beachten Sie die Informationen, die für die Serverklasse allgemein gelten, als auch die Informationen, die für die einzelne Serverklasse spezifisch sind.

Die Serverklassen sind:

- SAODBC
- ULODBC
- ADSODBC
- ASEODBC
- DB2ODBC
- HANAODBC
- IQODBC
- MIRROR
- MSACCESSODBC
- MSSODBC
- MYSQLODBC
- ODBC
- ORAODBC

Hinweis

Wenn Sie beim Ferndatenzugriff einen ODBC-Treiber verwenden, der Unicode nicht unterstützt, wird keine Zeichensatzkonvertierung bei Daten durchgeführt, die von diesem ODBC-Treiber stammen.

Definitionen für externe ODBC-Server

Der einfachste Weg zur Definition eines ODBC-basierten Fremdservers ist der Aufbau auf einer ODBC-Datenquelle. Dazu können Sie eine Datenquelle im ODBC-Datenquellenadministrator erstellen.

Wenn Sie die Datenquelle definiert haben, muss die USING-Klausel in der CREATE SERVER-Anweisung den Namen der ODBC-Datenquelle (DSN) referenzieren.

Um z. B. einen IBM DB2-Server namens "mydb2" zu erstellen, dessen Datenquellenname ebenfalls "mydb2" ist, benutzen Sie folgenden Befehl:

```
CREATE SERVER mydb2
CLASS 'DB2ODBC'
USING 'mydb2';
```

Der verwendete Treiber muss mit dem Bitwert des Datenbankservers übereinstimmen.

Unter Windows müssen Sie auch einen Datenquellennamen (System-DSN) mit einem Bitwert definieren, der mit dem Datenbankserver übereinstimmt. Verwenden Sie beispielsweise den 32-Bit ODBC-Datenquellenadministrator, um einen DSN für ein 32-Bit-System zu erstellen. Eine Benutzer-DSN hat keinen Bitwert.

Verbindungszeichenfolgen anstelle von Datenquellen verwenden

Eine Alternative, bei der die Verwendung von Datenquellennamen vermieden werden kann, besteht im Angeben einer Verbindungszeichenfolge in der USING-Klausel der CREATE SERVER-Anweisung. Dazu müssen Sie die Verbindungsparameter für den ODBC-Treiber kennen, den Sie verwenden. Eine Verbindung mit einem SQL Anywhere-Datenbankserver kann z. B. folgendermaßen eingerichtet werden:

```
CREATE SERVER TestSA
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 16;HOST=myhost;Server=TestSA;DBN=sample';
```

Dies definiert eine Verbindung zu einem Datenbankserver namens TestSA, der auf dem Computer "myhost" läuft, und der Beispieldatenbank "sample" unter Verwendung des TCP-IP-Protokolls.

Siehe auch

Hinweise zu bestimmten ODBC-Serverklassen:

- „SAODBC-Serverklasse“ auf Seite 826
- „Serverklasse ULODBC“ auf Seite 827
- „Serverklasse ADSODBC“ auf Seite 828
- „Serverklasse ASEODBC“ auf Seite 829
- „Serverklasse DB2ODBC“ auf Seite 832
- „Serverklasse HANAODBC“ auf Seite 834
- „IQODBC-Serverklasse“ auf Seite 836
- „MSACCESSODBC-Serverklasse“ auf Seite 836
- „Serverklasse MSSODBC“ auf Seite 838
- „MYSQLODBC-Serverklasse“ auf Seite 840
- „Serverklasse ODBC“ auf Seite 842
- „Serverklasse ORAODBC“ auf Seite 845
- „ODBC-Datenquellen“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE SERVER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

USING-Klausel in der CREATE SERVER-Anweisung

Sie müssen eine eigene CREATE SERVER-Anweisung für jede entfernte SQL Anywhere-Datenbank ausführen, auf die Sie zugreifen wollen. Wenn z. B. ein SQL Anywhere-Server namens TestSA auf dem

Computer Banana läuft und 3 Datenbanken (db1, db2, db3) hat, konfigurieren Sie die Fremdserver folgendermaßen:

```
CREATE SERVER TestSadb1
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 16;HOST=Banana;Server=TestSA;DBN=db1';

CREATE SERVER TestSadb2
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 16;HOST=Banana;Server=TestSA;DBN=db2';

CREATE SERVER TestSadb3
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 16;HOST=Banana;Server=TestSA;DBN=db3';
```

Wenn Sie keinen Wert für einen Datenbanknamen eingeben, benutzt die Verbindung mit dem Fremdserver die Standarddatenbank auf dem entfernten SQL Anywhere-Server.

Siehe auch

- „CREATE SERVER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Verbindungsparameter“ [[SQL Anywhere Server - Datenbankadministration](#)]

SAODBC-Serverklasse

Ein Fremdserver mit der SAODBC-Serverklasse ist ein SQL Anywhere-Datenbankserver. Für die Konfiguration einer SQL Anywhere-Datenquelle gibt es keine besonderen Anforderungen.

Um auf SQL Anywhere-Datenbankserver zuzugreifen, die mehrere Datenbanken unterstützen, erstellen Sie einen ODBC-Datenquellennamen, der eine Verbindung zu jeder Datenbank definiert. Führen Sie die CREATE SERVER-Anweisung für jeden dieser ODBC-Datenquellennamen aus.

Beispiel

Geben Sie eine Verbindungszeichenfolge in der USING-Klausel der CREATE SERVER-Anweisung an, um eine Verbindung mit einer SQL Anywhere-Datenbank herzustellen.

```
CREATE SERVER TestSA
CLASS 'SAODBC'
USING 'DRIVER=SQL Anywhere 16;HOST=myhost;Server=TestSA;DBN=sample';
```

MIRROR-Serverklasse

Ein Server mit der MIRROR-Serverklasse ist ein SQL Anywhere-Datenbankserver. Die MIRROR-Serverklasse stellt eine Verbindung mit einem entfernten SQL Anywhere-Server über ODBC her. Beim Erstellen des Fremdservers enthält jedoch die USING-Klausel einen Spiegelservernamen aus der SYS.SYSMIRRORSERVER-Katalogtabelle. Die Schicht für den Ferndatenzugriff verwendet diesen Spiegelservernamen, um die Verbindungszeichenfolge für den entfernten SQL Anywhere-Server zu erstellen.

Hinweise

Wenn Sie eine Proxy-Tabelle abfragen, die einer Tabelle auf einem Spiegelserver mit Ferndatenzugriff zugeordnet ist, sucht die Schicht für den Ferndatenzugriff in den Katalogtabellen SYS.SYSMIRRORSERVER und SYS.SYSMIRRORSERVEROPTION, um zu ermitteln, welche Verbindungszeichenfolge verwendet werden soll, um eine Verbindung mit dem SA-Server herzustellen, auf den vom Spiegelserver für den Ferndatenzugriff verwiesen wird.

Beispiel

Um einen Spiegelserver für den Ferndatenzugriff einzurichten, damit Sie sich mit dem Server MyMirrorServer verbinden können, führen Sie eine Anweisung ähnlich der folgenden aus:

```
CREATE SERVER remote_server_name
CLASS 'MIRROR'
USING 'MirrorServer=MyMirrorServer';
```

Hinweis

Im Gegensatz zu anderen Serverklassen für den Ferndatenzugriff werden Verbindungen mit Spiegelservern für den Ferndatenzugriff automatisch wiederaufgenommen, wenn die Verbindung mit dem Fremdrechner unterbrochen wird.

Serverklasse ULODBC

Ein Datenbankserver mit der ULODBC-Serverklasse ist ein UltraLite-Datenbankserver. Erstellen Sie einen ODBC-Datenquellennamen, der eine Verbindung zur UltraLite-Datenbank definiert. Führen Sie eine CREATE SERVER-Anweisung für den ODBC-Datenquellennamen aus.

Es gibt eine Eins-zu-eins-Zuordnung zwischen den UltraLite- und den SQL Anywhere-Datentypen, weil UltraLite eine Teilmenge der in SQL Anywhere verfügbaren Datentypen unterstützt.

Hinweis

Sie können keinen Fremdserver für eine UltraLite-Datenbank unter Mac OS X erstellen.

Siehe auch

- „UltraLite, SQLDatentypen“ [[UltraLite - Datenbankverwaltung](#)]

Beispiel

Geben Sie eine Verbindungszeichenfolge in der USING-Klausel der CREATE SERVER-Anweisung an, um eine Verbindung mit einer UltraLite-Datenbank herzustellen.

```
CREATE SERVER TestUL
CLASS 'ULODBC'
USING 'DRIVER=UltraLite 16;UID=DBA;PWD=sql;DBF=custdb.udb'
```

Serverklasse ADSODBC

Wenn Sie eine CREATE TABLE-Anweisung ausführen, konvertiert SQL Anywhere automatisch die Datentypen in die entsprechenden Advantage Database Server-Datentypen, wobei die folgenden Datentypkonvertierungen verwendet werden.

SQL Anywhere-Datentyp	ADS-Standarddatentyp
BIT	Logical
VARBIT(<i>n</i>)	Binary(<i>n</i>)
LONG VARBIT	Binary(2G)
TINYINT	Integer
SMALLINT	Integer
INTEGER	Integer
BIGINT	Numeric(32)
UNSIGNED TINYINT	Numeric(11)
UNSIGNED SMALLINT	Numeric(11)
UNSIGNED INTEGER	Numeric(11)
UNSIGNED BIGINT	Numeric(32)
CHAR(<i>n</i>)	Character(<i>n</i>)
VARCHAR(<i>n</i>)	VarChar(<i>n</i>)
LONG VARCHAR	VarChar(65000)
NCHAR(<i>n</i>)	NChar(<i>n</i>)
NVARCHAR(<i>n</i>)	NVarChar(<i>n</i>)
LONG NVARCHAR	NVarChar(32500)
BINARY(<i>n</i>)	Binary(<i>n</i>)
VARBINARY(<i>n</i>)	Binary(<i>n</i>)
LONG BINARY	Binary(2G)
DECIMAL(<i>precision</i> , <i>scale</i>)	Numeric(<i>precision</i> +3)

SQL Anywhere-Datentyp	ADS-Standarddatentyp
NUMERIC(<i>precision, scale</i>)	Numeric(<i>precision</i> +3)
SMALLMONEY	Money
MONEY	Money
REAL	Double
DOUBLE	Double
FLOAT(<i>n</i>)	Double
DATE	Date
TIME	Time
TIMESTAMP	TimeStamp
TIMESTAMP WITH TIMEZONE	Char(254)
XML	Binary(2G)
ST_GEOMETRY	Binary(2G)
UNIQUEIDENTIFIER	Binary(2G)

Serverklasse ASEODBC

Ein Server mit der ASEODBC-Serverklasse ist ein Adaptive Server Enterprise-Datenbankserver (Version 10 und höher). SQL Anywhere erfordert die Installation des Adaptive Server Enterprise ODBC-Treibers und der Open Client Connectivity-Bibliotheken, um eine Verbindung mit einem entfernten Adaptive Server Enterprise-Datenbankserver mit der ASEODBC-Klasse herzustellen.

Hinweise

- Open Client muss mindestens Version 11.1.1, EBF 7886 oder später sein. Installieren Sie Open Client und prüfen Sie die Systemanbindung an den Adaptive Server Enterprise-Server, bevor Sie ODBC installieren und SQL Anywhere konfigurieren. Der Sybase ODBC-Treiber muss mindestens Version 11.1.1, EBF 7911 sein.
- Die lokale Einstellung der `quoted_identifier`-Option steuert die Verwendung von Bezeichnern in Anführungszeichen für Adaptive Server Enterprise. Wenn Sie z.B. die `quoted_identifier`-Option lokal auf "Off" festlegen, werden Bezeichner in Anführungszeichen für Adaptive Server Enterprise deaktiviert.

- Konfigurieren Sie eine Benutzerdatenquelle im **Konfigurationsmanager** mit folgenden Attributen:

- **Registerkarte "Allgemein"** Geben Sie einen Wert für den **Datenquellennamen** ein. Dieser Wert wird in der USING-Klausel der CREATE SERVER-Anweisung verwendet.

Der Servername muss zum Namen des Servers in der Sybase-Interface-Datei passen.

- **Registerkarte "Erweitert"** Klicken Sie auf die Optionen **Anwendung verwendet Threads** und **Bezeichner in Anführungszeichen aktivieren**.

- **Registerkarte "Verbindung"** Setzen Sie das Feld "Zeichensatz" auf den Zeichensatz von SQL Anywhere.

Setzen Sie das Feld "Sprache" auf die Sprache, in der die Fehlermeldungen erscheinen sollen.

- **Registerkarte "Performance"** Setzen Sie die **Prepare-Methode** auf **2-Full**.

Stellen Sie die **Abruf-Arraygröße** so groß wie möglich ein, um die optimale Performance zu erzielen. Damit erhöhen sich die Speicheranforderungen, da der Wert die Anzahl von Zeilen festlegt, die in den Cache gelesen werden müssen. Adaptive Server Enterprise empfiehlt einen Wert von 100.

Setzen Sie die **Select-Methode** auf **0-Cursor**.

Setzen Sie die **Paketgröße** auf den größtmöglichen Wert. Adaptive Server Enterprise empfiehlt einen Wert von -1.

Setzen Sie den **Verbindungs-cache** auf 1.

Datentypkonvertierungen: ODBC und Adaptive Server Enterprise

Wenn Sie eine CREATE TABLE-Anweisung ausführen, konvertiert SQL Anywhere automatisch die Datentypen in die entsprechenden Adaptive Server Enterprise-Datentypen. In der folgenden Tabelle wird die Datentypkonvertierung von SQL Anywhere zu Adaptive Server Enterprise beschrieben:

SQL Anywhere-Datentyp	Adaptive Server Enterprise-Standarddatentyp
BIT	bit
VARBIT(<i>n</i>)	if (<i>n</i> <= 255) varbinary(<i>n</i>) else image
LONG VARBIT	image
TINYINT	tinyint
SMALLINT	smallint
INT, INTEGER	int
BIGINT	numeric(20,0)

SQL Anywhere-Datentyp	Adaptive Server Enterprise-Standarddatentyp
UNSIGNED TINYINT	tinyint
UNSIGNED SMALLINT	int
UNSIGNED INTEGER	numeric(11,0)
UNSIGNED BIGINT	numeric(20,0)
CHAR(<i>n</i>)	if (n <= 255) char(<i>n</i>) else text
VARCHAR(<i>n</i>)	if (n <= 255) varchar(<i>n</i>) else text
LONG VARCHAR	text
NCHAR(<i>n</i>)	if (n <= 255) nchar(<i>n</i>) else ntext
NVARCHAR(<i>n</i>)	if (n <= 255) nvarchar(<i>n</i>) else ntext
LONG NVARCHAR	ntext
BINARY(<i>n</i>)	if (n <= 255) binary(<i>n</i>) else image
VARBINARY(<i>n</i>)	if (n <= 255) varbinary(<i>n</i>) else image
LONG BINARY	image
DECIMAL(<i>prec</i> , <i>scale</i>)	decimal(<i>prec</i> , <i>scale</i>)
NUMERIC(<i>prec</i> , <i>scale</i>)	numeric(<i>prec</i> , <i>scale</i>)
SMALLMONEY	numeric(10,4)
MONEY	numeric(19,4)
REAL	real
DOUBLE	float
FLOAT(<i>n</i>)	float(<i>n</i>)
DATE	datetime
TIME	datetime
SMALLDATETIME	smalldatetime
TIMESTAMP	datetime

SQL Anywhere-Datentyp	Adaptive Server Enterprise-Standarddatentyp
TIMESTAMP WITH TIMEZONE	varchar(254)
XML	text
ST_GEOMETRY	image
UNIQUEIDENTIFIER	binary(16)

Beispiel

Geben Sie eine Verbindungszeichenfolge in der USING-Klausel der CREATE SERVER-Anweisung an, um eine Verbindung mit einer Adaptive Server Enterprise-Datenbank herzustellen.

```
CREATE SERVER TestASE
CLASS 'ASEODBC'
USING 'DRIVER=SYBASE ASE ODBC
Driver;Server=TestASE;Port=5000;Database=testdb;UID=username;PWD=password'
```

Serverklasse DB2ODBC

Ein Server der DB2ODBC-Serverklasse ist ein IBM DB2-Datenbankserver.

Hinweise

- Sybase zertifiziert die Nutzung der IBM DB2 Connect Version 5 mit dem Fix Pack WR09044. Konfigurieren und testen Sie Ihre ODBC-Konfiguration unter Zuhilfenahme der Anleitungen für diese Software. SQL Anywhere stellt keine besonderen Anforderungen an die Konfiguration von IBM DB2-Datenquellen.
- Nachstehend finden Sie ein Beispiel für die CREATE EXISTING TABLE-Anweisung auf einem IBM DB2-Server mit einer ODBC-Datenquelle namens mydb2:

```
CREATE EXISTING TABLE ibmcol
AT 'mydb2..sysibm.syscolumns';
```

Datentypkonvertierungen: IBM DB2

Wenn Sie eine CREATE TABLE-Anweisung ausführen, konvertiert SQL Anywhere automatisch die Datentypen in die entsprechenden IBM DB2-Datentypen. In der folgenden Tabelle finden Sie die Entsprechungen für SQL Anywhere- und IBM DB2-Datentypen.

SQL Anywhere-Datentyp	IBM DB2-Standarddatentyp
BIT	smallint
VARBIT(<i>n</i>)	if (n <= 4000) varchar(<i>n</i>) for bit data else long varchar for bit data
LONG VARBIT	long varchar für Bit-Daten

SQL Anywhere-Datentyp	IBM DB2-Standarddatentyp
TINYINT	smallint
SMALLINT	smallint
INTEGER	int
BIGINT	decimal(20,0)
UNSIGNED TINYINT	int
UNSIGNED SMALLINT	int
UNSIGNED INTEGER	decimal(11,0)
UNSIGNED BIGINT	decimal(20,0)
CHAR(<i>n</i>)	if (<i>n</i> < 255) char(<i>n</i>) else if (<i>n</i> <= 4000) varchar(<i>n</i>) else long varchar
VARCHAR(<i>n</i>)	if (<i>n</i> <= 4000) varchar(<i>n</i>) else long varchar
LONG VARCHAR	long varchar
NCHAR(<i>n</i>)	Nicht unterstützt
NVARCHAR(<i>n</i>)	Nicht unterstützt
LONG NVARCHAR	Nicht unterstützt
BINARY(<i>n</i>)	if (<i>n</i> <= 4000) varchar(<i>n</i>) for bit data else long varchar for bit data
VARBINARY(<i>n</i>)	if (<i>n</i> <= 4000) varchar(<i>n</i>) for bit data else long varchar for bit data
LONG BINARY	long varchar für Bit-Daten
DECIMAL(<i>prec</i> , <i>scale</i>)	decimal(<i>prec</i> , <i>scale</i>)
NUMERIC(<i>prec</i> , <i>scale</i>)	decimal(<i>prec</i> , <i>scale</i>)
SMALLMONEY	decimal(10,4)
MONEY	decimal(19,4)
REAL	real
DOUBLE	float
FLOAT(<i>n</i>)	float(<i>n</i>)

SQL Anywhere-Datentyp	IBM DB2-Standarddatentyp
DATE	date
TIME	time
TIMESTAMP	timestamp
TIMESTAMP WITH TIMEZONE	varchar(254)
XML	long varchar für Bit-Daten
ST_GEOMETRY	long varchar für Bit-Daten
UNIQUEIDENTIFIER	varchar(16) for bit data

Serverklasse HANAODBC

Ein Fremdserver mit der HANAODBC-Serverklasse ist ein SAP HANA-Datenbankserver.

Hinweise

- Nachstehend finden Sie ein Beispiel für die CREATE EXISTING TABLE-Anweisung auf einem SAP HANA-Datenbankserver mit einer ODBC-Datenquelle namens mySAPHANA:

```
CREATE EXISTING TABLE hanatable
AT 'mySAPHANA..dbo.hanatable';
```

Datentypkonvertierungen: SAP HANA

Wenn Sie eine CREATE TABLE-Anweisung ausführen, konvertiert SQL Anywhere automatisch die Datentypen in die entsprechenden SAP HANA-Datentypen. In der folgenden Tabelle finden Sie die Entsprechungen für SQL Anywhere- und SAP HANA-Datentypen.

SQL Anywhere-Datentyp	SAP HANA-Standarddatentyp
BIT	TINYINT
VARBIT(<i>n</i>)	if (<i>n</i> <= 5000) VARBINARY(<i>n</i>) else BLOB
LONG VARBIT	BLOB
TINYINT	TINYINT
SMALLINT	SMALLINT
INTEGER	INTEGER
BIGINT	BIGINT

SQL Anywhere-Datentyp	SAP HANA-Standarddatentyp
UNSIGNED TINYINT	TINYINT
UNSIGNED SMALLINT	INTEGER
UNSIGNED INTEGER	BIGINT
UNSIGNED BIGINT	DECIMAL (20,0)
CHAR(<i>n</i>)	if (n <= 5000) VARCHAR(<i>n</i>) else CLOB
VARCHAR(<i>n</i>)	if (n <= 5000) VARCHAR(<i>n</i>) else CLOB
LONG VARCHAR	CLOB
NCHAR(<i>n</i>)	if (n <= 5000) NVARCHAR(<i>n</i>) else NCLOB
NVARCHAR(<i>n</i>)	if (n <= 5000) NVARCHAR(<i>n</i>) else NCLOB
LONG NVARCHAR	NCLOB
BINARY(<i>n</i>)	if (n <= 5000) VARBINARY(<i>n</i>) else BLOB
VARBINARY(<i>n</i>)	if (n <= 5000) VARBINARY(<i>n</i>) else BLOB
LONG BINARY	BLOB
DECIMAL(<i>precision</i> , <i>scale</i>)	DECIMAL(<i>precision</i> , <i>scale</i>)
NUMERIC(<i>precision</i> , <i>scale</i>)	DECIMAL(<i>precision</i> , <i>scale</i>)
SMALLMONEY	DECIMAL (13,4)
MONEY	DECIMAL (19,4)
REAL	REAL
DOUBLE	FLOAT
FLOAT(<i>n</i>)	FLOAT
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP
TIMESTAMP WITH TIMEZONE	VARCHAR(254)

SQL Anywhere-Datentyp	SAP HANA-Standarddatentyp
XML	BLOB
ST_GEOMETRY	BLOB
UNIQUEIDENTIFIER	VARBINARY(16)

IQODBC-Serverklasse

Ein Fremdserver mit der IQODBC-Serverklasse ist ein SAP Sybase IQ-Datenbankserver. Für die Konfiguration einer SAP Sybase IQ-Datenquelle gibt es keine besonderen Anforderungen.

Um auf SAP Sybase IQ-Datenbankserver zuzugreifen, die mehrere Datenbanken unterstützen, erstellen Sie einen ODBC-Datenquellennamen, der eine Verbindung zu jeder Datenbank definiert. Führen Sie die CREATE SERVER-Anweisung für jeden dieser ODBC-Datenquellennamen aus.

Siehe auch

- [„USING-Klausel in der CREATE SERVER-Anweisung“ auf Seite 825](#)

MSACCESSODBC-Serverklasse

Access-Datenbanken werden in einer *.mdb*-Datei gespeichert. Mit dem ODBC-Manager erstellen Sie eine ODBC-Datenquelle und ordnen sie einer dieser Dateien zu. Eine neue *.mdb*-Datei kann mit dem ODBC-Manager erstellt werden. Diese Datenbankdatei wird als Standard verwendet, wenn beim Erstellen der Tabelle über SQL Anywhere keine andere angegeben wurde.

Wenn Sie eine ODBC-Datenquelle namens access haben, können Sie mit einer der folgenden Anweisungen auf Daten zugreifen:

- ```
CREATE TABLE tabl (a int, b char(10))
AT 'access...tabl';
```
- ```
CREATE TABLE tabl (a int, b char(10))
AT 'access;d:\pcdb\data.mdb;tabl';
```
- ```
CREATE EXISTING TABLE tabl
AT 'access;d:\pcdb\data.mdb;tabl';
```

Access unterstützt die Eigentümernamensqualifikation nicht, und daher muss dieses Feld leer gelassen werden.

### Datentypkonvertierungen: Microsoft Access

| SQL Anywhere-Datentyp | Microsoft Access-Standarddatentyp |
|-----------------------|-----------------------------------|
| BIT                   | TINYINT                           |

| SQL Anywhere-Datentyp                      | Microsoft Access-Standarddatentyp            |
|--------------------------------------------|----------------------------------------------|
| VARBIT( <i>n</i> )                         | if (n <= 4000) BINARY( <i>n</i> ) else IMAGE |
| LONG VARBIT                                | IMAGE                                        |
| TINYINT                                    | TINYINT                                      |
| SMALLINT                                   | SMALLINT                                     |
| INTEGER                                    | INTEGER                                      |
| BIGINT                                     | DECIMAL(19,0)                                |
| UNSIGNED TINYINT                           | TINYINT                                      |
| UNSIGNED SMALLINT                          | INTEGER                                      |
| UNSIGNED INTEGER                           | DECIMAL(11,0)                                |
| UNSIGNED BIGINT                            | DECIMAL(20,0)                                |
| CHAR( <i>n</i> )                           | if (n < 255) CHARACTER( <i>n</i> ) else TEXT |
| VARCHAR( <i>n</i> )                        | if (n < 255) CHARACTER( <i>n</i> ) else TEXT |
| LONG VARCHAR                               | TEXT                                         |
| NCHAR( <i>n</i> )                          | Not supported                                |
| NVARCHAR( <i>n</i> )                       | Not supported                                |
| LONG NVARCHAR                              | Not supported                                |
| BINARY( <i>n</i> )                         | if (n <= 4000) BINARY( <i>n</i> ) else IMAGE |
| VARBINARY( <i>n</i> )                      | if (n <= 4000) BINARY( <i>n</i> ) else IMAGE |
| LONG BINARY                                | IMAGE                                        |
| DECIMAL( <i>precision</i> , <i>scale</i> ) | DECIMAL( <i>precision</i> , <i>scale</i> )   |
| NUMERIC( <i>precision</i> , <i>scale</i> ) | DECIMAL( <i>precision</i> , <i>scale</i> )   |
| SMALLMONEY                                 | MONEY                                        |
| MONEY                                      | MONEY                                        |
| REAL                                       | REAL                                         |

| SQL Anywhere-Datentyp   | Microsoft Access-Standarddatentyp |
|-------------------------|-----------------------------------|
| DOUBLE                  | FLOAT                             |
| FLOAT( <i>n</i> )       | FLOAT                             |
| DATE                    | DATETIME                          |
| TIME                    | DATETIME                          |
| TIMESTAMP               | DATETIME                          |
| TIMESTAMP WITH TIMEZONE | CHARACTER(254)                    |
| XML                     | XML                               |
| ST_GEOMETRY             | IMAGE                             |
| UNIQUEIDENTIFIER        | BINARY(16)                        |

## Serverklasse MSSODBC

Die MSSODBC-Serverklasse wird verwendet, um auf einen Microsoft SQL Server über einen seiner ODBC-Treiber zuzugreifen.

### Hinweise

- Folgende Versionen von Microsoft SQL Server ODBC-Treibern werden benutzt:
  - Microsoft SQL Server ODBC-Treiber Version 06.01.7601
  - Microsoft SQL Server ODBC-Treiber Version 10.00.1600
- Im Folgenden finden Sie ein Beispiel für Microsoft SQL Server:

```
CREATE SERVER mysqlserver
CLASS 'MSSODBC'
USING 'DSN=MSSODBC_cli';

CREATE EXISTING TABLE accounts
AT 'mysqlserver.master.dbo.accounts';
```

- Die lokale Einstellung der `quoted_identifier`-Option steuert die Verwendung von Bezeichnern in Anführungszeichen bei Microsoft SQL Server. Wenn Sie z.B. die `quoted_identifier`-Option lokal auf "Off" festlegen, werden die Bezeichner in Anführungszeichen für Microsoft SQL Server deaktiviert.

### Datentypkonvertierungen: Microsoft SQL Server

Wenn Sie eine `CREATE TABLE`-Anweisung ausführen, konvertiert SQL Anywhere automatisch die Datentypen in die entsprechenden Microsoft SQL Server-Datentypen, wobei die folgenden Datentypkonvertierungen verwendet werden.



| SQL Anywhere-Datentyp                      | Microsoft SQL Server-Standarddatentypen        |
|--------------------------------------------|------------------------------------------------|
| BIT                                        | bit                                            |
| VARBIT( <i>n</i> )                         | if (n <= 255) varbinary( <i>n</i> ) else image |
| LONG VARBIT                                | image                                          |
| TINYINT                                    | tinyint                                        |
| SMALLINT                                   | smallint                                       |
| INTEGER                                    | int                                            |
| BIGINT                                     | numeric(20,0)                                  |
| UNSIGNED TINYINT                           | tinyint                                        |
| UNSIGNED SMALLINT                          | int                                            |
| UNSIGNED INTEGER                           | numeric(11,0)                                  |
| UNSIGNED BIGINT                            | numeric(20,0)                                  |
| CHAR( <i>n</i> )                           | if (n <= 255) char( <i>n</i> ) else text       |
| VARCHAR( <i>n</i> )                        | if (n <= 255) varchar( <i>n</i> ) else text    |
| LONG VARCHAR                               | text                                           |
| NCHAR( <i>n</i> )                          | if (n <= 4000) nchar( <i>n</i> ) else ntext    |
| NVARCHAR( <i>n</i> )                       | if (n <= 4000) nvarchar( <i>n</i> ) else ntext |
| LONG NVARCHAR                              | ntext                                          |
| BINARY( <i>n</i> )                         | if (n <= 255) binary( <i>n</i> ) else image    |
| VARBINARY( <i>n</i> )                      | if (n <= 255) varbinary( <i>n</i> ) else image |
| LONG BINARY                                | image                                          |
| DECIMAL( <i>precision</i> , <i>scale</i> ) | decimal( <i>precision</i> , <i>scale</i> )     |
| NUMERIC( <i>precision</i> , <i>scale</i> ) | numeric( <i>precision</i> , <i>scale</i> )     |
| SMALLMONEY                                 | smallmoney                                     |
| MONEY                                      | money                                          |

| SQL Anywhere-Datentyp   | Microsoft SQL Server-Standarddatentypen |
|-------------------------|-----------------------------------------|
| REAL                    | real                                    |
| DOUBLE                  | float                                   |
| FLOAT( <i>n</i> )       | float( <i>n</i> )                       |
| DATE                    | datetime                                |
| TIME                    | datetime                                |
| SMALLDATETIME           | smalldatetime                           |
| DATETIME                | datetime                                |
| TIMESTAMP               | datetime                                |
| TIMESTAMP WITH TIMEZONE | varchar(254)                            |
| XML                     | xml                                     |
| ST_GEOMETRY             | image                                   |
| UNIQUEIDENTIFIER        | binary(16)                              |

## MYSQLODBC-Serverklasse

Wenn Sie eine CREATE TABLE-Anweisung ausführen, konvertiert SQL Anywhere automatisch die Datentypen in die entsprechenden MySQL-Datentypen, wobei die folgenden Datentypkonvertierungen verwendet werden.

| SQL Anywhere-Datentyp | MySQL-Standarddatentyp                                     |
|-----------------------|------------------------------------------------------------|
| BIT                   | bit(1)                                                     |
| VARBIT( <i>n</i> )    | if ( <i>n</i> <= 4000) varbinary( <i>n</i> ) else longblob |
| LONG VARBIT           | longblob                                                   |
| TINYINT               | tinyint unsigned                                           |
| SMALLINT              | smallint                                                   |
| INTEGER               | int                                                        |

| SQL Anywhere-Datentyp                      | MySQL-Standarddatentyp                                                                                                               |
|--------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| BIGINT                                     | bigint                                                                                                                               |
| UNSIGNED TINYINT                           | tinyint unsigned                                                                                                                     |
| UNSIGNED SMALLINT                          | int                                                                                                                                  |
| UNSIGNED INTEGER                           | bigint                                                                                                                               |
| UNSIGNED BIGINT                            | decimal(20,0)                                                                                                                        |
| CHAR( <i>n</i> )                           | if ( <i>n</i> < 255) char( <i>n</i> ) else<br>if ( <i>n</i> <= 4000) varchar( <i>n</i> ) else longtext                               |
| VARCHAR( <i>n</i> )                        | if ( <i>n</i> <= 4000) varchar( <i>n</i> ) else longtext                                                                             |
| LONG VARCHAR                               | longtext                                                                                                                             |
| NCHAR( <i>n</i> )                          | if ( <i>n</i> < 255) national character( <i>n</i> ) else if ( <i>n</i> <= 4000) national character varying( <i>n</i> ) else longtext |
| NVARCHAR( <i>n</i> )                       | if ( <i>n</i> <= 4000) national character varying( <i>n</i> ) else longtext                                                          |
| LONG NVARCHAR                              | longtext                                                                                                                             |
| BINARY( <i>n</i> )                         | if ( <i>n</i> <= 4000) varbinary( <i>n</i> ) else longblob                                                                           |
| VARBINARY( <i>n</i> )                      | if ( <i>n</i> <= 4000) varbinary( <i>n</i> ) else longblob                                                                           |
| LONG BINARY                                | longblob                                                                                                                             |
| DECIMAL( <i>precision</i> , <i>scale</i> ) | decimal( <i>precision</i> , <i>scale</i> )                                                                                           |
| NUMERIC( <i>precision</i> , <i>scale</i> ) | decimal( <i>precision</i> , <i>scale</i> )                                                                                           |

| SQL Anywhere-Datentyp   | MySQL-Standarddatentyp |
|-------------------------|------------------------|
| SMALLMONEY              | decimal(10,4)          |
| MONEY                   | decimal(19,4)          |
| REAL                    | real                   |
| DOUBLE                  | float                  |
| FLOAT( <i>n</i> )       | float( <i>n</i> )      |
| DATE                    | date                   |
| TIME                    | time                   |
| TIMESTAMP               | datetime               |
| TIMESTAMP WITH TIMEZONE | varchar(254)           |
| XML                     | longblob               |
| ST_GEOMETRY             | longblob               |
| UNIQUEIDENTIFIER        | varbinary(16)          |

### Beispiel

Geben Sie eine Verbindungszeichenfolge in der USING-Klausel der CREATE SERVER-Anweisung an, um eine Verbindung mit einer MySQL-Datenbank herzustellen.

```
CREATE SERVER TestMySQL
CLASS 'MySQLODBC'
USING 'DRIVER=MySQL ODBC 5.1
Driver;DATABASE=mydatabase;SERVER=MySQLHost;UID=me;PWD=secret'
```

## Serverklasse ODBC

ODBC-Datenquellen, die keine eigene Serverklasse haben, verwenden Serverklasse **ODBC**. Sie können jeden ODBC-Treiber verwenden. Sybase zertifiziert die folgenden ODBC-Datenquellen:

- „Microsoft Excel (Microsoft 3.51.171300)“
- „Microsoft Foxpro (Microsoft 3.51.171300)“
- „Lotus Notes SQL“

Die letzten Versionen der Microsoft ODBC-Treiber können über den Microsoft Data Access Components (MDAC)-Vertrieb im Microsoft Download Center bezogen werden. Die oben aufgelisteten Microsoft-Treiber sind Teil von MDAC 2.0.

## Microsoft Excel (Microsoft 3.51.171300)

In Excel wird jede Excel-Arbeitsmappe als Datenbank mit mehreren Tabellen angesehen. Die Tabellen sind Blättern dieser Arbeitsmappe zugeordnet. Wenn Sie einen ODBC-Datenquellennamen im ODBC-Treiber-Manager konfigurieren, geben Sie einen Standard-Arbeitsmappennamen ein, der dieser Datenquelle zugeordnet ist. Wenn Sie jedoch eine CREATE TABLE-Anweisung ausführen, können Sie den Standardwert aufheben und in der Standortzeichenfolge einen Arbeitsmappennamen angeben. So können Sie einen einzelnen ODBC DSN benutzen, um auf alle Excel-Arbeitsmappen zugreifen zu können.

Erstellen Sie einen Fremdservers namens excel, der eine Verbindung mit dem Microsoft Excel ODBC-Treiber herstellt.

```
CREATE SERVER excel
CLASS 'ODBC'
USING 'DRIVER=Microsoft Excel Driver (*.xls);DBQ=d:\
\work1.xls;READONLY=0;DriverID=790'
```

So erstellen Sie eine Arbeitsmappe namens *work1.xls* mit einer Tabelle namens "mywork":

```
CREATE TABLE mywork (a int, b char(20))
AT 'excel;d:\work1.xls;mywork';
```

Für eine zweite Tabelle lautet die Anweisung:

```
CREATE TABLE mywork2 (x float, y int)
AT 'excel;d:\work1.xls;mywork2';
```

Sie können vorhandene Arbeitsblätter mit CREATE EXISTING in SQL Anywhere importieren, jedoch unter der Voraussetzung, dass die erste Zeile des Arbeitsblattes Spaltennamen enthält.

```
CREATE EXISTING TABLE mywork
AT 'excel;d:\work1;mywork';
```

Wenn SQL Anywhere meldet, dass die Tabelle nicht gefunden werden konnte, müssen Sie eventuell ausdrücklich den Bereich von Spalten und Zeilen angeben, den Sie übernehmen wollen. Beispiel:

```
CREATE EXISTING TABLE mywork
AT 'excel;d:\work1;mywork$';
```

Wenn Sie das Zeichen \$ dem Tabellennamen hinzufügen, wird die komplette Arbeitsmappe gewählt.

Hinweis: In der Standortangabe, die von AT festgelegt wurde, wird anstelle eines Punkts ein Semikolon als Feldtrennzeichen verwendet. Dies liegt daran, dass in den Dateinamen Punkte enthalten sind. Excel unterstützt das Eigentümernamensfeld nicht, und daher muss dieses leer gelassen werden.

Löschvorgänge werden nicht unterstützt. Möglicherweise sind auch gewisse Aktualisierungen nicht möglich, da der Excel-Treiber positionierte Aktualisierungen nicht unterstützt.

### Beispiel

Die folgenden Anweisungen erstellen einen Datenbankserver namens TestExcel, der einen ODBC DSN für den Zugriff auf die Excel-Arbeitsmappe *LogFile.xlsx* verwendet, und importieren die Tabelle darin in SQL Anywhere.

```
CREATE SERVER TestExcel
CLASS 'ODBC'
USING 'DRIVER=Microsoft Excel Driver (*.xls);DBQ=c:\\temp\\
\LogFile.xlsx;READONLY=0;DriverID=790'

CREATE EXISTING TABLE MyWorkbook
AT 'TestExcel;c:\\temp\\LogFile.xlsx;logfile$';

SELECT * FROM MyWorkbook;
```

## Microsoft Foxpro (Microsoft 3.51.171300)

FoxPro-Tabellen können gemeinsam in einer einzigen FoxPro-Datenbankdatei (.dbc) oder jeweils in einer einzelnen .dbf-Datei gespeichert werden. Wenn Sie .dbf-Dateien verwenden, muss der Dateiname in die Standortangabe eingefügt werden, da sonst das Verzeichnis verwendet wird, von dem aus SQL Anywhere gestartet wurde.

```
CREATE TABLE fox1 (a int, b char(20))
AT 'foxpro;d:\\pcdb;fox1';
```

Diese Anweisung erstellt eine Datei namens *d:\\pcdb\\fox1.dbf*, wenn Sie die Option "Freies Tabellenverzeichnis" im ODBC-Treibermanager auswählen.

## Lotus Notes SQL

Sie finden diesen Treiber auf der Lotus NotesSQL Website unter <http://www.ibm.com/developerworks/lotus/products/notesdomino/notessql/>. In der Dokumentation, die mit diesem Treiber geliefert wird, finden Sie eine Erklärung zur Zuordnung von Notes-Daten zu relationalen Tabellen. Sie können Tabellen von SQL Anywhere einfach den Notes-Formularen zuordnen.

So wird SQL Anywhere für den Zugriff auf eine Lotus Notes-Adressendatei eingerichtet.

- Stellen Sie sicher, dass der Lotus Notes Programmordner in Ihrem Pfad enthalten ist (z.B. C:\\Programe (x86)\\IBM\\Lotus\\Notes).
- Erstellen Sie eine 32-Bit-ODBC-Datenquelle mit dem NotesSQL ODBC-Treiber. Verwenden Sie die Datenbank *names.nsf* für dieses Beispiel. Die Option **Map Special Characters** muss aktiviert sein. In diesem Beispiel lautet der **Datenquellenname** *my\_notes\_dsn*.
- Erstellen Sie mit Interactive SQL und einer Verbindung zu einem 32-Bit-Datenbankserver einen Server für den Ferndatenzugriff. Ein Beispiel:

```
CREATE SERVER NotesContacts
CLASS 'ODBC'
USING 'my_notes_dsn';
```

- Erstellen Sie ein externes Login für den Lotus Notes-Server Ein Beispiel:

```
CREATE EXTERNLOGIN "DBA" TO "NotesContacts"
REMOTE LOGIN 'John Doe/SYBASE' IDENTIFIED BY 'MyNotesPassword';
```

- Ordnen Sie einige Spalten des Formulars "Person" einer Tabelle von SQL Anywhere zu:

```
CREATE EXISTING TABLE PersonDetails
(DisplayName CHAR(254),
 DisplayMailAddress CHAR(254),
 JobTitle CHAR(254),
 CompanyName CHAR(254),
 Department CHAR(254),
 Location CHAR(254),
 OfficePhoneNumber CHAR(254))
AT 'NotesContacts...Person';
```

- Führen Sie eine Abfrage in der Tabelle aus:

```
SELECT * FROM PersonDetails
WHERE Location LIKE 'Waterloo%';
```

## Serverklasse ORAODBC

Ein Fremdserver mit der ORAODBC-Serverklasse ist eine Oracle Database der Version 8.0 oder höher.

### Hinweise

- Sybase zertifiziert die Verwendung des ODBC-Treibers Version 8.0.03 von Oracle. Konfigurieren und testen Sie Ihre ODBC-Konfiguration unter Zuhilfenahme der Anleitungen für diese Software.
- Nachstehend finden Sie ein Beispiel für die CREATE EXISTING TABLE-Anweisung auf einem Oracle-Datenbankserver namens myora:

```
CREATE EXISTING TABLE employees
AT 'myora.database.owner.employees';
```

### Datentypkonvertierungen: Oracle Database

Wenn Sie eine CREATE TABLE-Anweisung ausführen, konvertiert SQL Anywhere automatisch die Datentypen in die entsprechenden Oracle-Datentypen, wobei die folgenden Datentypkonvertierungen verwendet werden.

| SQL Anywhere-Datentyp | Datentyp in Oracle Database                            |
|-----------------------|--------------------------------------------------------|
| BIT                   | number(1,0)                                            |
| VARBIT( <i>n</i> )    | if ( <i>n</i> <= 255) raw( <i>n</i> ) else<br>long raw |
| LONG VARBIT           | long raw                                               |
| TINYINT               | number(3,0)                                            |
| SMALLINT              | number(5,0)                                            |
| INTEGER               | number(11,0)                                           |
| BIGINT                | number(20,0)                                           |

| SQL Anywhere-Datentyp                      | Datentyp in Oracle Database                            |
|--------------------------------------------|--------------------------------------------------------|
| UNSIGNED TINYINT                           | number(3,0)                                            |
| UNSIGNED SMALLINT                          | number(5,0)                                            |
| UNSIGNED INTEGER                           | number(11,0)                                           |
| UNSIGNED BIGINT                            | number(20,0)                                           |
| CHAR( <i>n</i> )                           | if ( <i>n</i> <= 255) char( <i>n</i> ) else long       |
| VARCHAR( <i>n</i> )                        | if ( <i>n</i> <= 2000) varchar( <i>n</i> ) else long   |
| LONG VARCHAR                               | long                                                   |
| NCHAR( <i>n</i> )                          | if ( <i>n</i> <= 255) nchar( <i>n</i> ) else nclob     |
| NVARCHAR( <i>n</i> )                       | if ( <i>n</i> <= 2000) nvarchar( <i>n</i> ) else nclob |
| LONG NVARCHAR                              | nclob                                                  |
| BINARY( <i>n</i> )                         | wenn ( <i>n</i> > 255) long raw, sonst raw( <i>n</i> ) |
| VARBINARY( <i>n</i> )                      | wenn ( <i>n</i> > 255) long raw, sonst raw( <i>n</i> ) |
| LONG BINARY                                | long raw                                               |
| DECIMAL( <i>precision</i> , <i>scale</i> ) | number( <i>precision</i> , <i>scale</i> )              |
| NUMERIC( <i>precision</i> , <i>scale</i> ) | number( <i>precision</i> , <i>scale</i> )              |
| SMALLMONEY                                 | numeric(13,4)                                          |
| MONEY                                      | number(19,4)                                           |
| REAL                                       | real                                                   |
| DOUBLE                                     | float                                                  |



| SQL Anywhere-Datentyp   | Datentyp in Oracle Database |
|-------------------------|-----------------------------|
| FLOAT( <i>n</i> )       | float                       |
| DATE                    | date                        |
| TIME                    | date                        |
| TIMESTAMP               | date                        |
| TIMESTAMP WITH TIMEZONE | varchar(254)                |
| XML                     | long raw                    |
| ST_GEOMETRY             | long raw                    |
| UNIQUEIDENTIFIER        | raw(16)                     |

### Beispiel

Geben Sie eine Verbindungszeichenfolge in der USING-Klausel der CREATE SERVER-Anweisung an, um eine Verbindung mit einer Oracle Database-Datenbank herzustellen.

```
CREATE SERVER TestOracle
CLASS 'ORAODBC'
USING 'DRIVER=Oracle ODBC Driver;DBQ=mydatabase;UID=username;PWD=password'
```



---

# Datenintegrität

Wenn Daten Integrität haben, sind die Daten gültig - korrekt und akkurat - und die relationale Struktur der Datenbank ist intakt. Integritätsregeln zur Erhaltung der referenziellen Integrität sichern die relationale Struktur der Datenbank. Diese Regeln halten die Konsistenz der Daten zwischen Tabellen aufrecht. Die Einbeziehung von Integritätsregeln zur Aufrechterhaltung der referenziellen Integrität in die Datenbank ist die beste Möglichkeit, um zu gewährleisten, dass die Daten konsistent bleiben.

Sie können mehrere Arten von Regeln zur referenziellen Integrität erzwingen. Sie können beispielsweise sicherstellen, dass individuelle Einträge korrekt sind, indem Sie Tabellen und Spalten Integritätsregeln und CHECK-Bedingungen auferlegen. Sie können auch die Spalteneigenschaften konfigurieren, indem Sie einen entsprechenden Datentyp wählen oder spezielle Standardwerte einstellen.

SQL Anywhere unterstützt gespeicherte Prozeduren, die Ihnen genaue Kontrolle darüber geben, wie Daten in die Datenbank gelangen. Sie können auch Trigger einrichten, d.h., benutzerdefinierte gespeicherte Prozeduren, die automatisch aufgerufen werden, wenn eine bestimmte Aktion, wie etwa die Aktualisierung einer bestimmten Spalte, ausgeführt wird.

## Siehe auch

- [„Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ auf Seite 81](#)

## Wie Ihre Daten ungültig werden können

Die Daten in Ihrer Datenbank können ungültig werden, wenn die richtigen Überprüfungen nicht durchgeführt werden. Sie können jedes dieser Beispiele mit den in diesem Abschnitt beschriebenen Maßnahmen verhindern.

### Falsche Angaben

- Ein Anwender gibt das Datum einer Verkaufstransaktion falsch ein.
- Das Gehalt eines Mitarbeiters wird zehn Mal zu klein, weil der Anwender eine Ziffer weggelassen hat.

### Redundante Daten

- Zwei verschiedene Mitarbeiter fügen in die Tabelle "Departments" der Unternehmensdatenbank dieselbe neue Abteilung ein (mit der DepartmentID 200).

### Fremdschlüsselbeziehungen ungültig

- Die durch "DepartmentID 300" gekennzeichnete Abteilung wird geschlossen und ein Mitarbeiterdatensatz wird versehentlich keiner neuen Abteilung zugeordnet.

## Integritätsregeln

Um die Validität der Daten in einer Datenbank sicherzustellen, müssen Sie Prüfungen erstellen, die gültige und ungültige Daten festlegen, und Regeln einrichten, die von den Daten befolgt werden müssen

(sogenannte Geschäftsregeln). Geschäftsregeln werden gewöhnlich mithilfe von Prüf-Integritätsregeln, benutzerdefinierten Datentypen und geeigneter Verwendung von Transaktionen implementiert.

Integritätsregeln, die in die Datenbank eingebaut werden, sind zuverlässiger als Integritätsregeln, die in Clientanwendungen enthalten sind oder als Anweisungen für den Datenbankbenutzer gegeben werden. Integritätsregeln in der Datenbank selbst werden Teil der Definition der Datenbank, und die Datenbank sorgt für ihre Einhaltung in allen Anwendungen. Wenn in der Datenbank eine Integritätsregel festgelegt wurde, wird diese Regel für alle nachfolgenden Interaktionen mit der Datenbank durchgesetzt.

Im Gegensatz dazu sind in Clientanwendungen integrierte Integritätsregeln immer dann anfällig, wenn die Software geändert wird, und es kann erforderlich sein, sie in mehrere Anwendungen oder an mehreren Stellen in einer Clientanwendung einzubeziehen.

## Wie sich der Inhalt Ihrer Datenbank ändert

Daten in Datenbanktabellen werden durch die Ausführung von SQL-Anweisungen von Clientanwendungen aus verändert. Im Grunde sind es nur wenige SQL-Anweisungen, die die Daten in der Datenbank tatsächlich verändern. Sie können Folgendes durchführen:

- Daten in einer Tabellenzeile mit der Anweisung UPDATE aktualisieren
- Eine vorhandene Tabellenzeile mit der Anweisung DELETE löschen
- Eine neue Tabellenzeile mit der Anweisung INSERT einfügen

## Werkzeuge zur Aufrechterhaltung der Datenintegrität

Zur Unterstützung der Datenintegrität können Sie Standardwerte, Daten-Integritätsregeln sowie Integritätsregeln verwenden, die die referenzielle Struktur der Datenbank aufrechterhalten.

### Standardwerte

Standardwerte können Spalten zugeordnet werden, damit bestimmte Arten der Dateneingabe zuverlässiger werden. Zum Beispiel:

- Eine Spalte kann den Standardwert CURRENT DATE zum Aufzeichnen des Transaktionsdatums für Aktionen von Benutzer- oder Clientanwendungen enthalten.
- Andere Typen von Standardwerten gestatten es, Spaltenwerte automatisch zu erhöhen, wobei keine spezifische Benutzeraktion außer der Eingabe einer neuen Zeile erforderlich ist. Mit dieser Funktion können Sie sicherstellen, dass Elemente (z.B. Bestellungen) eindeutige laufende Nummern erhalten.

### Primärschlüssel

Primärschlüssel garantieren, dass jede Zeile in einer Tabelle eindeutig gekennzeichnet werden kann.

## Tabellen- und Spalten-Integritätsregeln

Die folgenden Integritätsregeln halten die Struktur der Daten in der Datenbank aufrecht und definieren die Beziehung zwischen Tabellen in einer relationalen Datenbank:

- **Referenzielle Integritätsregeln** Die Datenintegrität wird auch mithilfe von Regeln für die referenzielle Integrität aufrechterhalten, die auch als **RI-Integritätsregeln** bezeichnet werden. RI-Integritätsregeln sind Datenregeln für Spalten und Tabellen, die steuern, um welche Daten es sich handeln kann. RI-Integritätsregeln definieren die Beziehung zwischen Tabellen in einer relationalen Datenbank.
- **NOT NULL-Integritätsregel** Eine NOT NULL-Integritätsregel verhindert, dass eine Spalte einen NULL-Eintrag enthält.
- **CHECK-Integritätsregel** Eine CHECK-Integritätsregel, die einer Spalte zugeordnet ist, kann sicherstellen, dass jedes Element in der Spalte einer bestimmten Bedingung entspricht. Sie können z.B. sicherstellen, dass Einträge in die Salary-Spalte einen bestimmten Bereich nicht überschreiten und vor Benutzerfehlern geschützt sind, wenn neue Werte eingegeben werden.

CHECK-Integritätsregeln können sich auf die relativen Werte in verschiedenen Spalten beziehen. Sie können z.B. sicherstellen, dass ein Rückgabe-Eintrag in einer Bibliotheksdatenbank nach einem Ausleihtag-Eintrag erfolgt.

Spalten-Integritätsregeln können von Domänen geerbt werden.

## Trigger für erweiterte Integritätsregeln

Ein **Trigger** ist eine in einer Datenbank gespeicherte Prozedur, die automatisch ausgeführt wird, wenn die Daten in einer angegebenen Tabelle geändert werden. Trigger sind leistungsfähige Instrumente für Datenbankadministratoren und Entwickler, um zu gewährleisten, dass die Daten verlässlich bleiben. Sie können Trigger auch verwenden, um die Datenintegrität aufrechtzuerhalten. Trigger ermöglichen höher entwickelte CHECK-Bedingungen.

### Siehe auch

- „Spalten-Standardwerte“ auf Seite 852
- „Primärschlüssel“ auf Seite 17
- „Entitätsintegrität und referenzielle Integrität“ auf Seite 868
- „Tabellen- und Spalten-Integritätsregeln“ auf Seite 859
- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ auf Seite 81

# SQL-Anweisungen für die Implementierung von Integritätsregeln

Die folgenden SQL-Anweisungen implementieren Integritätsregeln:

- **CREATE TABLE-Anweisung** Diese Anweisung implementiert Integritätsregeln während der Erstellung der Tabelle.

- **ALTER TABLE-Anweisung** Diese Anweisung fügt einer vorhandenen Tabelle Integritätsregeln hinzu oder ändert die Integritätsregeln für eine vorhandene Tabelle.
- **CREATE TRIGGER-Anweisung** Diese Anweisung erstellt Trigger, die komplexere Geschäftsregeln erzwingen.
- **CREATE DOMAIN-Anweisung** Diese Anweisung erstellt einen benutzerdefinierten Datentyp. Die Definition des Datentyps kann Integritätsregeln enthalten.

### Siehe auch

- „SQL-Anweisungen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Spalten-Standardwerte

Spalten-Standardwerte ordnen bestimmten Spalten automatisch einen bestimmten Wert zu, wenn eine neue Zeile in die Datenbank eingegeben wird. Der Standardwert erfordert keinerlei Aktion von der Clientanwendung, aber wenn die Clientanwendung einen Wert für die betreffende Spalte angibt, wird der Spalten-Standardwert überschrieben.

Spaltenstandardwerte können schnell und automatisch Spalten mit Daten füllen, wie z.B. mit dem Datum und der Uhrzeit, an dem bzw. zu der eine Zeile eingefügt wird, oder mit der Benutzerkennung der Person, die die Daten eingibt. Die Verwendung von Spalten-Standardwerten verbessert die Datenintegrität, erzwingt sie aber nicht. Clientanwendungen können die Standardwerte jederzeit überschreiben.

Wenn Standardwerte mithilfe von Variablen festgelegt werden, die mit @ beginnen, ist der Standardwert der Variablen der Wert zu dem Zeitpunkt, zu dem die DML- oder LOAD-Anweisung ausgeführt wird.

### Unterstützte Standardwerte

SQL unterstützt folgende Standardwerte:

- Eine in der Anweisung angegebene Zeichenfolge, und zwar in CREATE TABLE oder in ALTER TABLE.
- Eine in der Anweisung angegebene Zahl, und zwar in CREATE TABLE oder in ALTER TABLE.
- AUTOINCREMENT: Eine automatisch ansteigende Zahl, die um eins höher ist als der bisher höchste Wert in der Spalte.
- GLOBAL AUTOINCREMENT, womit die Eindeutigkeit von Primärschlüsseln über mehrere Datenbanken hinweg gewährleistet wird.
- Universally Unique Identifiers (UUIDs) die durch die Funktion NEWID generiert werden.
- CURRENT DATE, CURRENT TIME oder TIMESTAMP.
- Der CURRENT USER-Wert des Datenbankbenutzers.
- NULL.

- Einen Konstantenausdruck, sofern er keine Bezugnahme auf Datenbankobjekte enthält.

## Erstellen von Spalten-Standardwerten

Sie können die Anweisung CREATE TABLE verwenden, um Spalten-Standardwerte beim Erstellen einer Tabelle einzurichten, oder die Anweisung ALTER TABLE, um Spalten-Standardwerte später einzurichten.

### Beispiel

Die folgende Anweisung fügt einer vorhandenen Spalte mit dem Namen "ID" in der Tabelle "SalesOrders" einen Standardwert hinzu, sodass ihr Wert automatisch erhöht wird (es sei denn, die Clientanwendung gibt einen Wert an). Diese Spalte ist in der SQL Anywhere-Beispieldatenbank bereits auf AUTOINCREMENT gesetzt.

```
ALTER TABLE SalesOrders
ALTER ID DEFAULT AUTOINCREMENT;
```

### Siehe auch

- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Spalten-Standardwerte ändern

Sie können Spalten-Standardwerte mit derselben Form der ALTER TABLE-Anweisung ändern bzw. löschen wie beim Erstellen der Standardwerte.

Die folgende Anweisung ändert den Standardwert einer Spalte namens "OrderDate" von der aktuellen Einstellung auf CURRENT DATE:

```
ALTER TABLE SalesOrders
ALTER OrderDate DEFAULT CURRENT DATE;
```

Sie können Spalten-Standardwerte entfernen, indem Sie sie auf NULL modifizieren. Die folgende Anweisung entfernt den Standardwert aus der Spalte "OrderDate":

```
ALTER TABLE SalesOrders
ALTER OrderDate DEFAULT NULL;
```

## Arbeiten mit Spalten-Standardwerten

Sie können Standardwerte für Spalten in Sybase Central über die Registerkarte **Wert** des Fensters **Spalteneigenschaften** hinzufügen, ändern und löschen .

## Voraussetzungen

Sie müssen Eigentümer der Tabelle sein, zu der die Spalte gehört, oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Tabelle
- ALTER ANY TABLE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

## Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Tabellen**.
3. Doppelklicken Sie auf die Tabelle.
4. Klicken Sie auf die Registerkarte **Spalten**.
5. Rechtsklicken Sie auf die Spalte und klicken Sie auf **Eigenschaften**.
6. Klicken Sie auf die Registerkarte **Wert**.
7. Ändern Sie die Spalten-Standardwerte, wenn erforderlich.

## Ergebnisse

Die Eigenschaften der Spalte werden geändert.

## Siehe auch

- „ALTER TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

## Standardwerte für aktuelles Datum und aktuelle Uhrzeit

Bei Spalten mit dem Datentyp DATE, TIME oder TIMESTAMP können Sie CURRENT DATE, CURRENT TIME oder CURRENT TIMESTAMP als Standardwert verwenden. Der gewählte Standardwert muss mit dem Datentyp der Spalte kompatibel sein.

## Nützliche Beispiele für den Standardwert CURRENT DATE

Der Standardwert CURRENT DATE kann für folgende Aufzeichnungen nützlich sein:

- Datum von Telefonaten in einer Kontaktdatenbank.
- Datum von Bestellungen in einer Verkaufsdatenbank.
- Ausleihdatum in einer Bibliothek.



## CURRENT TIMESTAMP

Der Standardwert CURRENT TIMESTAMP ist dem Standardwert CURRENT DATE ähnlich, bietet jedoch eine größere Genauigkeit. Ein Benutzer einer Anwendung für Kontaktmanagement kann z.B. an einem Tag mehrere Kontakte mit dem gleichen Kunden haben: Der Standardwert CURRENT TIMESTAMP wäre hier nützlich, um die einzelnen Kontakte voneinander zu unterscheiden.

Da der Standardwert CURRENT TIMESTAMP Datum und Uhrzeit mit einer Genauigkeit bis einer Millionstel Sekunde aufzeichnen kann, können Sie ihn auch dann sinnvoll einsetzen, wenn die Ereignissequenz in der Datenbank wichtig ist.

## DEFAULT TIMESTAMP

DEFAULT TIMESTAMP bietet eine Möglichkeit, die Zeit der letzten Änderung einer Zeile in der Tabelle anzuzeigen. 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. Um einen Standardwert beim Einfügen bereitzustellen, die Spalte aber nicht bei jeder Zeilenaktualisierung zu aktualisieren, verwenden Sie DEFAULT CURRENT TIMESTAMP anstatt DEFAULT TIMESTAMP.

### Siehe auch

- „CREATE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SQL-Datentypen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Standardwerte für Benutzer-ID

Die Zuweisung von DEFAULT USER zu einer Spalte ist eine einfache und verlässliche Methode, um aufzuzeichnen, wer einen Eintrag in einer Datenbank vorgenommen hat. Diese Angabe kann erforderlich sein, wenn z.B. Handelsvertreter auf Provisionsbasis arbeiten.

Einen Standardwert "Benutzer-ID" in den Primärschlüssel einer Tabelle einzubeziehen, ist eine sinnvolle Technik bei nur gelegentlich verbundenen Benutzern, und diese Methode trägt dazu bei, Konflikte bei Datenaktualisierungen zu vermeiden. Diese Benutzer können auf einem portablen Computer eine Kopie der Tabellen führen, die für ihre Arbeit wichtig sind, darin Änderungen vornehmen, wenn sie nicht mit einer Mehrbenutzer-Datenbank verbunden sind, und später das Transaktionslog auf den Server übertragen.

Der spezielle Wert LAST USER gibt den Namen des Benutzers an, der die Zeile zuletzt geändert hat. 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

- „LAST USER-Spezialwert“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Der Standardwert "Autoincrement"

Der Standardwert AUTOINCREMENT ist bei numerischen Datenfeldern nützlich. Diese Funktion gibt jeder neuen Zeile einen eindeutigen Wert, der größer ist als alle anderen Werte in der Spalte. Sie können Autoincrement-Spalten benutzen, um Bestellnummern aufzuzeichnen, Kundendienstanrufe zu kennzeichnen oder für andere Einträge, bei denen eine Kennnummer erforderlich ist.

AUTOINCREMENT-Spalten sind typischerweise Primärschlüsselspalten oder Spalten, die auf eindeutige Werte beschränkt sind.

Sie können den zuletzt in eine AUTOINCREMENT-Spalte eingegebenen Wert mit der globalen Variable @@identity abrufen.

### AUTOINCREMENT und negative Zahlen

AUTOINCREMENT ist für die Verwendung mit positiven Ganzzahlen gedacht.

Der anfängliche AUTOINCREMENT-Wert wird beim Erstellen der Tabelle auf 0 gesetzt. Dieser Wert bleibt der höchste zugeordnete Wert, wenn ausdrücklich negative Werte in die Spalte eingefügt werden. Eine Einfügung ohne Wert führt dazu, dass AUTOINCREMENT den Wert 1 generiert, wodurch alle anderen generierten Werte positiv werden.

### AUTOINCREMENT und die IDENTITY-Spalte

Eine Spalte mit dem Autoincrement-Standardwert wird in Transact-SQL-Anwendungen als Identity-Spalte referenziert.

#### Siehe auch

- [Neuladen von Tabellen mit AUTOINCREMENT-Spalten \[SQL Anywhere 16 - Änderungen und Upgrades\]](#)
- [„CREATE TABLE-Anweisung“ \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- [„Der Standardwert GLOBAL AUTOINCREMENT“ auf Seite 856](#)
- [Wählen zwischen Sequenzen und AUTOINCREMENT-Werten auf Seite 950](#)
- [„GLOBAL AUTOINCREMENT-Spalten“ \[SQL Remote\]](#)
- [„sa\\_reset\\_identity-Systemprozedur“ \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- [„Globale Variable @@identity“ \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- [„Entitätsintegrität“ auf Seite 868](#)
- [„Die spezielle IDENTITY-Spalte“ auf Seite 662](#)

## Der Standardwert GLOBAL AUTOINCREMENT

Der Standardwert GLOBAL AUTOINCREMENT wird eingesetzt, wenn mehrere Datenbanken in einer SQL Remote-Replikation oder in einer MobiLink-Synchronisationsumgebung verwendet werden. Er gewährleistet, dass Primärschlüssel in mehreren Datenbanken eindeutig sind.

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 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 diese Option 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  $(n + 1)p$ , 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 als  $(n + 1)p$  ist, 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 vorhanden sein, wenn sie von einer anderen Datenbank über MobiLink-Synchronisation repliziert wurden.

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 Spaltentyp 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. Dies geschieht etwa, wenn die Spalte im Primärschlüssel der Tabelle enthalten ist.

NULL-Standardwerte werden auch generiert, wenn der Vorrat von Werten innerhalb der Partition aufgebraucht ist. In diesem Fall sollte der Datenbank ein neuer global\_database\_id-Wert zugewiesen werden, damit Standardwerte aus einer anderen Partition gewählt werden können. Der Versuch, NULL einzufügen, bewirkt einen Fehler, wenn die Spalte NULL nicht zulässt. Um festzustellen, ob der Vorrat von ungenutzten Werten zu Ende geht, und um diese Situation zu beheben, erstellen Sie ein Ereignis vom Typ GlobalAutoincrement.

GLOBAL AUTOINCREMENT-Spalten sind typischerweise Primärschlüsselspalten oder Spalten, die auf eindeutige Werte beschränkt sind.

Es ist zwar möglich, auch in anderen Fällen den Standardwert GLOBAL AUTOINCREMENT zu verwenden, aber dies kann sich negativ auf die Datenbank-Performance auswirken. Beispiel: Wenn der nächste Wert für jede Spalte als Ganzzahl mit Vorzeichen (64 Byte) gespeichert wird, kann die Verwendung großer "double"- oder "numeric"-Werte oder die Verwendung von Werten größer als  $2^{31} - 1$  zur Umwandlung in negative Werte führen.

Sie können den zuletzt in eine AUTOINCREMENT-Spalte eingegebenen Wert mit der globalen Variable @@identity abrufen.

### Siehe auch

- „GLOBAL AUTOINCREMENT“ [*MobiLink - Serveradministration*]
- „Entitätsintegrität“ auf Seite 868
- „Globale Variable @@identity“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Ereignisse“ [*SQL Anywhere Server - Datenbankadministration*]
- „GLOBAL AUTOINCREMENT-Spalten“ [*SQL Remote*]
- „CREATE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- Neuladen von Tabellen mit AUTOINCREMENT-Spalten [*SQL Anywhere 16 - Änderungen und Upgrades*]
- „Der Standardwert "Autoincrement"“ auf Seite 856
- Wählen zwischen Sequenzen und AUTOINCREMENT-Werten auf Seite 950
- „sa\_reset\_identity-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

## Der Standardwert NEWID

UUIDs (Universally Unique IDentifiers), auch bekannt als GUIDs (Globally Unique IDentifiers), können zur eindeutigen Kennzeichnung von Zeilen in einer Tabelle verwendet werden. Die Werte werden auf eine Weise generiert, dass ein auf einem bestimmten Computer produzierter Wert nicht dem Wert eines anderen Computers entspricht. Sie können deshalb als Schlüssel in Replikations- und Synchronisationsumgebungen verwendet werden.

Im Vergleich zur Verwendung von GLOBAL AUTOINCREMENT-Werten als Primärschlüssel, bergen die UUID-Werte gewisse Vor- und Nachteile. Zum Beispiel:

- UUIDs können einfacher eingerichtet werden als GLOBAL AUTOINCREMENT-Werte, da es nicht notwendig ist, jeder entfernten Datenbank eine eindeutige Datenbank-ID zuzuweisen. Zusätzlich kann die Anzahl der Datenbanken im System oder die Anzahl der Zeilen in den einzelnen Tabellen außer Acht gelassen werden. Das Extraktionsdienstprogramm (dbxtract) kann zur Zuweisung der Datenbank-ID verwendet werden. Davon ist GLOBAL AUTOINCREMENT normalerweise nicht betroffen, wenn der Datentyp BIGINT verwendet wird, muss aber für kleinere Datentypen in Betracht gezogen werden.
- UUID-Werte sind bedeutend länger als die, die für GLOBAL AUTOINCREMENT benötigt werden, und erfordern mehr Tabellenspeicherplatz in den Primär- und Fremdtabellen. Wenn UUIDs verwendet werden, sind auch die Indizes für diese Tabellen weniger effizient. Zusammenfassend betrachtet ist die Ausführung von GLOBAL AUTOINCREMENT möglicherweise besser.
- UUIDs haben keine implizite Sortierung. Wenn beispielsweise A und B UUID-Werte sind, bedeutet A > B nicht, dass A nach B generiert wurde, selbst wenn A und B auf dem selben Computer generiert wurden. Wenn Sie dieses Verhalten benötigen, könnten eine zusätzliche Spalte und ein Index notwendig sein.

### Siehe auch

- „NEWID-Funktion [Verschiedene]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „UNIQUEIDENTIFIER-Datentyp“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

## Der Standardwert NULL

Wenn bei Spalten, die NULL zulassen, NULL als Standardwert festgelegt wird, ist dies dasselbe, als wäre gar kein Standardwert definiert worden. Wenn der Client, der die Zeile einfügt, nicht ausdrücklich einen Wert zuordnet, erhält die Zeile automatisch NULL.

Sie können den Standardwert NULL verwenden, wenn Daten für bestimmte Spalten optional sind oder nicht immer zur Verfügung stehen.

### Siehe auch

- „NULL-Spezialwert“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Standardwerte mit Zeichenfolgen und Zahlen

Sie können eine bestimmte Zeichenfolge oder Zahl als Standardwert angeben, sofern die Spalte den Datentyp "string" oder "numeric" enthält. Sie müssen sicherstellen, dass der angegebene Standardwert in den Datentyp der Spalte konvertiert werden kann.

Standardzeichenfolgen und -zahlen sind nützlich, wenn es einen typischen Eintrag für die betreffende Spalte gibt. Wenn z.B. ein Unternehmen zwei Bürostandorte hat, die Zentrale in "Stadt\_1" und ein kleines Büro in "Stadt\_2", sollte für eine Standortspalte ein Standardwert auf "Stadt\_1" eingerichtet werden, damit die Datenerfassung vereinfacht wird.

## Standardkonstantenausdruck

Sie können einen Konstantenausdruck als Standardwert benutzen, sofern er keine Bezugnahme auf Datenbankobjekte enthält. Beispiel: Der folgende Ausdruck ermöglicht es, dass Spaltenstandardwerte das Datum fünfzehn Tage nach dem heutigen Datum enthalten:

```
... DEFAULT (DATEADD(day, 15, GETDATE()));
```

## Tabellen- und Spalten-Integritätsregeln

Zusammen mit der grundlegenden Tabellenstruktur (Anzahl, Namen und Datentyp der Spalten, Name und Speicherort der Tabelle) können die Anweisungen CREATE TABLE und ALTER TABLE mehrere verschiedene Tabellenattribute festlegen, die die Kontrolle der Datenintegrität ermöglichen. Mit Integritätsregeln können Sie die Werte einschränken, die in der Spalte vorkommen, oder die Beziehung zwischen Werten in verschiedenen Spalten beschränken. Integritätsregeln gelten entweder für die gesamte Tabelle oder für einzelne Spalten.

Dieser Abschnitt beschreibt, wie Integritätsregeln verwendet werden können, um die Korrektheit der Daten in der Tabelle sicherzustellen.

## CHECK-Integritätsregeln in Spalten

Eine CHECK-Bedingung wird eingesetzt, um sicherzustellen, dass die Werte in einer Spalte bestimmten Kriterien oder Regeln entsprechen. Diese Regeln oder Kriterien können erforderlich sein, um zu überprüfen, dass die Daten gültig sind, aber es kann sich auch um strengere Regeln handeln, die die Richtlinien und Prozeduren des Unternehmens widerspiegeln. CHECK-Bedingungen bei einzelnen Spaltenwerten sind nützlich, wenn nur ein reduzierter Bereich von Werten für die betreffende Spalte zulässig ist.

Sobald eine CHECK-Bedingung eingesetzt wird, werden zukünftige Werte entsprechend der Bedingung geprüft, bevor eine Zeile geändert wird. Wenn Sie einen Wert aktualisieren, der eine Prüf-Integritätsregel hat, werden die Integritätsregeln für diesen Wert sowie für den Rest der Zeile überprüft.

In CHECK-Integritätsregeln für Spalten sind keine Variablen zulässig. Eine Zeichenfolge beginnend mit @ innerhalb einer CHECK-Spaltenintegritätsregel wird durch den Namen der Spalte ersetzt, für die die Integritätsregel festgelegt ist.

Wenn der Datentyp der Spalte eine Domäne ist, übernimmt die Spalte alle für die Domäne festgelegten CHECK-Integritätsregeln.

### Hinweis

CHECK-Tests in der Spalte schlagen fehl, wenn die Bedingung den Wert FALSE zurückgibt. Wenn die Bedingung den Wert UNKNOWN zurückgibt, hat dies die gleiche Wirkung wie die Rückgabe des Werts TRUE, und der Wert wird erlaubt.

### Siehe auch

- „Von Domänen vererbte CHECK-Integritätsregeln für Spalten“ auf Seite 861
- „Suchbedingungen“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

### Beispiel 1

Sie können eine bestimmte Formatierungsanforderung erzwingen. Eine Tabelle enthält z.B. eine Spalte für Telefonnummern, und Sie möchten sicherstellen, dass alle Benutzer die Nummern auf dieselbe Weise eingeben. Bei nordamerikanischen Telefonnummern könnten Sie folgende Integritätsregel benutzen:

```
ALTER TABLE Customers
ALTER Phone
CHECK (Phone LIKE '(__) ____-____');
```

Wenn diese CHECK-Bedingung eingesetzt wird und Sie versuchen, etwa die Telefonnummer 9835 einzugeben, wird diese Änderung nicht erlaubt.

### Beispiel 2

Sie können dafür Sorge tragen, dass der Eintrag einem von einer begrenzten Anzahl von Werten entspricht. Wenn Sie z.B. sicherstellen wollen, dass in der Spalte "City" nur eine Stadt eingegeben werden kann, die zu einer bestimmten Gruppe von zulässigen Städten gehört (etwa die Städte, in denen das Unternehmen Büros hat), könnten Sie eine Integritätsregel wie die folgende einsetzen:

```
ALTER TABLE Customers
ALTER City
CHECK (City IN ('city_1', 'city_2', 'city_3'));
```

Standardmäßig berücksichtigen Zeichenfolgenvergleiche die Groß- und Kleinschreibung nicht, es sei denn, die Datenbank wurde ausdrücklich so erstellt, dass dies berücksichtigt wird.

### Beispiel 3

Sie können sicherstellen, dass ein Datum oder eine Zahl innerhalb eines bestimmten Bereiches liegen. Sie könnten z.B. festlegen, dass der StartDate-Wert für einen Mitarbeiter zwischen dem Gründungsdatum des Unternehmens und dem aktuellen Datum liegen muss. Um sicherzustellen, dass der StartDate-Wert zwischen diesen beiden Datumsangaben liegt, verwenden Sie folgende Integritätsregel:

```
ALTER TABLE Employees
ALTER StartDate
CHECK (StartDate BETWEEN '1983/06/27'
 AND CURRENT DATE);
```

Sie können verschiedene Datumsformate verwenden. Das in diesem Beispiel verwendete Format JJJJ/MM/TT hat den Vorteil, dass es immer erkannt wird, unabhängig von der aktuellen Einstellung der Optionen.

## CHECK-Integritätsregeln in Tabellen

Eine CHECK-Bedingung, die als Integritätsregel angewendet wird, gewährleistet normalerweise, dass zwei in einer Zeile eingegebene oder geänderte Werte eine ordnungsgemäße Beziehung zueinander haben.

Wenn Sie der Integritätsregel einen Namen geben, wird die Integritätsregel gesondert in den Systemtabellen gehalten, daher können Sie sie auch einzeln ersetzen und löschen. Da dies das flexiblere Verhalten ist, wird empfohlen, dass Sie, wann immer möglich, entweder eine CHECK-Integritätsregel benennen oder eine einzelne Spalten-Integritätsregel verwenden.

Sie können beispielsweise eine Integritätsregel für die Tabelle "Employees" hinzufügen, um sicherzustellen, dass "TerminationDate" immer nach "StartDate" liegt oder gleich mit diesem ist:

```
ALTER TABLE Employees
ADD CONSTRAINT valid_term_date
CHECK(TerminationDate >= StartDate);
```

Sie können Variable in CHECK-Integritätsregeln für Tabellen festlegen, aber ihr Name muss mit @ beginnen. Der verwendete Wert ist der Wert der Variablen zum Zeitpunkt, zu dem die DML oder LOAD-Anweisung ausgeführt wird.

### Siehe auch

- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Von Domänen vererbte CHECK-Integritätsregeln für Spalten

Sie können Domänen CHECK-Integritätsregeln zuordnen. Spalten, die mit solchen Domänen definiert wurden, erben die CHECK-Integritätsregeln. Eine speziell für die Spalte angegebene CHECK-

Integritätsregel überschreibt die Bedingung der Domäne. Die CHECK-Klausel in dieser Domänendefinition erfordert beispielsweise, dass nur positive Ganzzahlen in die Spalten eingegeben werden.

```
CREATE DATATYPE positive_integer INT
CHECK (@col > 0);
```

Jede Spalte, die die positive\_integer-Domäne verwendet, akzeptiert nur positive Ganzzahlen, es sei denn, für die Spalte selbst wurde ausdrücklich eine CHECK-Integritätsregel angegeben. Da alle Variablen mit dem Präfix @ durch den Spaltennamen ersetzt werden, wenn die CHECK-Integritätsregel ausgewertet wird, könnte ein beliebiger Variablenname mit @ an Stelle von @col verwendet werden.

Die Anweisung ALTER TABLE mit der Klausel DELETE CHECK löscht alle CHECK-Integritätsregeln aus der Tabellendefinition, einschließlich der von Domänen geerbten.

Etwaige Änderungen, die an einer Integritätsregel in einer Domänendefinition durchgeführt werden, nachdem eine Spalte für diese Domäne definiert wurde, werden nicht auf die Spalte angewendet. Die Spalte erhält die Integritätsregeln von der Domäne, wenn sie erstellt wird, aber danach gibt es keine Verbindung zwischen den beiden.

### Siehe auch

- „Domänen“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CHECK-Integritätsregeln in Spalten“ auf Seite 860

## Integritätsregeln verwalten

In Sybase Central können Sie Spalten-Integritätsregeln über die Registerkarte **Integritätsregeln** der Tabelle oder des Fensters **Spalteneigenschaften** hinzufügen, ändern und löschen.

### Voraussetzungen

Sie müssen Eigentümer der Tabelle sein, zu der die Spalte gehört, oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Tabelle
- ALTER ANY TABLE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plugins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Tabellen**.
3. Doppelklicken Sie auf die Tabelle, die Sie ändern wollen.
4. Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **Integritätsregeln** und ändern eine vorhandene Integritätsregel oder fügen eine neue Integritätsregel hinzu.



## Ergebnisse

Die Spalten-Integritätsregeln werden angezeigt.

## Nächste Schritte

Ändern Sie erforderlichenfalls die Integritätsregeln.

## Siehe auch

- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

# UNIQUE-Integritätsregeln hinzufügen

Sie können UNIQUE-Integritätsregeln für Spalten in Sybase Central erstellen und löschen.

## Voraussetzungen

Sie müssen Eigentümer der Tabelle sein oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Tabelle und das ALTER ANY INDEX-Systemprivileg, das COMMENT ANY OBJECT-Systemprivileg, das CREATE ANY INDEX-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg
- ALTER ANY TABLE-Systemprivileg und das ALTER ANY INDEX-Systemprivileg, das COMMENT ANY OBJECT-Systemprivileg, das CREATE ANY INDEX oder das CREATE ANY OBJECT-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

## Kontext und Bemerkungen

Spalten mit räumlichen Daten können nicht in eine UNIQUE-Integritätsregel einbezogen werden.

Bei einer Spalte gibt eine UNIQUE-Integritätsregel an, dass die Werte in der Spalte eindeutig sein müssen. Bei einer Tabelle kennzeichnet die UNIQUE-Integritätsregel eine oder mehrere Spalten, die Zeilen 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.

## Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Tabellen**.
3. Klicken Sie auf die Tabelle, die Sie ändern wollen.
4. Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **Integritätsregeln**.

5. Rechtsklicken Sie auf die Registerkarte **Integritätsregeln** und klicken Sie auf **Neu » Eindeutigkeits-Integritätsregel**.
6. Halten Sie sich an die Anleitungen des **Assistenten zum Erstellen einer Eindeutigkeits-Integritätsregel**.

### Ergebnisse

Eine UNIQUE-Integritätsregel wird erstellt.

## CHECK-Integritätsregeln ändern und löschen

Es gibt verschiedene Möglichkeiten, die vorhandenen CHECK-Integritätsregeln einer Tabelle zu ändern.

- Sie können eine neue CHECK-Integritätsregel zu einer Tabelle oder zu einer einzelnen Spalte hinzufügen.
- Sie können eine CHECK-Integritätsregel für eine Spalte löschen, indem Sie sie auf NULL setzen. Die folgende Anweisung zum Beispiel entfernt die CHECK-Integritätsregel aus der Spalte "Phone" in der Tabelle "Customers":

```
ALTER TABLE Customers
ALTER Phone CHECK NULL;
```

- Sie können eine CHECK-Integritätsregel für eine Spalte auf die gleiche Art und Weise ersetzen, wie Sie eine CHECK-Integritätsregel hinzufügen. Die folgende Anweisung zum Beispiel fügt die CHECK-Integritätsregel der Spalte "Phone" in der Tabelle "Customers" hinzu bzw. ersetzt sie:

```
ALTER TABLE Customers
ALTER Phone
CHECK (Phone LIKE '____-____-____');
```

- Sie können eine CHECK-Integritätsregel ändern, die für die Tabelle definiert wurde:
  - Sie können mit ALTER TABLE und einer Klausel "ADD *table-constraint*" eine neue CHECK-Integritätsregel hinzufügen.
  - Wenn Sie Integritätsregelnamen festgelegt haben, können Sie einzelne Integritätsregeln ändern.
  - Wenn Sie keine Integritätsregelnamen festgelegt haben, können Sie alle vorhandenen CHECK-Integritätsregeln (einschließlich CHECK-Integritätsregeln für Spalten und CHECK-Integritätsregeln, die von Domänen geerbt wurden) mithilfe von ALTER TABLE DELETE CHECK löschen und dann neue CHECK-Integritätsregeln hinzufügen.

So verwenden Sie die Anweisung ALTER TABLE mit der DELETE CHECK-Klausel:

```
ALTER TABLE table-name
DELETE CHECK;
```

In Sybase Central können Sie sowohl CHECK-Integritätsregeln für Tabellen als auch solche für Spalten hinzufügen, ändern und löschen.

Wenn eine Spalte aus einer Tabelle gelöscht wird, werden die CHECK-Integritätsregeln, die mit der in der Tabellen-Integritätsregel enthaltenen Spalte verbunden sind, nicht ebenfalls gelöscht. Wenn die Integritätsregeln nicht entfernt werden, wird bei jedem Versuch, Daten in der Tabelle einzufügen oder auch nur abzufragen, eine Fehlermeldung ausgegeben.

#### Hinweis

Tabellen-CHECK-Integritätsregeln schlagen fehl, wenn der Wert FALSE zurückgegeben wird. Wenn die Bedingung den Wert UNKNOWN zurückgibt, hat dies die gleiche Wirkung wie die Rückgabe des Werts TRUE, und der Wert wird erlaubt.

#### Siehe auch

- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Spalte '%1' nicht gefunden“ [[Fehlermeldungen](#)]
- „Integritätsregeln verwalten“ auf Seite 862

## Domänen

Eine **Domäne** ist ein benutzerdefinierter Datentyp, der zusammen mit anderen Attributen den Bereich der akzeptablen Werte einschränken oder Standardwerte vorgeben kann. Eine Domäne erweitert einen der integrierten Datentypen. Normalerweise ist der Bereich der erlaubten Werte durch eine Prüf-Integritätsregel eingeschränkt. Zusätzlich kann eine Domäne einen Standardwert vorgeben und NULL erlauben oder verbieten.

Das Festlegen Ihrer eigenen Domänen bietet viele Vorteile, wie z.B.:

- Verhinderung von verbreiteten Fehlern, wenn unpassende Werte eingegeben werden. Eine Integritätsregel in einer Domäne gewährleistet, dass alle Spalten und Variable, die für Werte in einem Bereich oder Format vorgesehen sind, nur solche Werte enthalten können. Ein Datentyp kann z.B. sicherstellen, dass alle Kreditkartennummern, die in die Datenbank eingegeben werden, die richtige Anzahl an Ziffern enthalten.
- Erleichterung des Verständnisses von Anwendungen und der Struktur einer Datenbank.
- Bedienungskomfort. Sie könnten z.B. verlangen, dass alle Identifizierungsmerkmale in Tabellen positive Ganzzahlen sind, die standardmäßig automatisch inkrementiert werden. Sie könnten dies erzwingen, indem Sie bei der Definition jeder neuen Tabelle die entsprechenden Integritätsregeln und Standardwerte eingeben. Der Arbeitsaufwand ist jedoch wesentlich geringer, wenn Sie eine neue Domäne definieren und dann einfach angeben, dass das Identifizierungsmerkmal nur Werte aus der angegebenen Domäne annehmen kann.

#### Siehe auch

- „Domänen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Erstellen von Domänen

Erstellen Sie eine benutzerdefinierte Domäne.

### Voraussetzungen

Sie müssen das CREATE DATATYPE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben.

### Kontext und Bemerkungen

Einige vordefinierte Domänen stehen in SQL Anywhere zur Verfügung, beispielsweise die Währungsdomäne MONEY.

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf **Domänen** und klicken Sie dann auf **Neu » Domäne**.
3. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen von Domänen**.

### Ergebnisse

Die neue Domäne wird erstellt.

### Siehe auch

- „CREATE DOMAIN-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Domänen auf Spalten anwenden

In Sybase Central können Sie eine Spalte so ändern, dass sie eine Domäne (benutzerdefinierter Datentyp) verwendet.

### Voraussetzungen

Sie müssen Eigentümer der Tabelle sein, zu der die Spalte gehört, oder eines der folgenden Privilegien haben:

- ALTER-Privileg für die Tabelle
- ALTER ANY TABLE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Tabellen**.
3. Klicken Sie auf die Tabelle.

4. Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **Spalten**.
5. Rechtsklicken Sie auf eine Spalte und klicken Sie auf **Eigenschaften**.
6. Klicken Sie auf die Registerkarte **Datentyp** und anschließend auf **Domäne**.
7. In der Liste **Domäne** wählen Sie eine Domäne aus.
8. Klicken Sie auf **OK**.

### Ergebnisse

Die Spalte verwendet die ausgewählte Domäne.

### Siehe auch

- „Domänen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Domänen löschen

Sie können benutzerdefinierte Datentypen (Domänen) in Sybase Central löschen.

### Voraussetzungen

Sie müssen das DROP DATATYPE-Systemprivileg oder das DROP ANY OBJECT-Systemprivileg haben.

Eine Domäne kann nicht gelöscht werden, wenn eine Variable oder Spalte in der Datenbank die Domäne nutzt. Löschen oder ändern Sie Spalten oder Variablen, die die Domäne verwenden, bevor Sie die Domäne löschen.

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Domänen**.
3. Rechtsklicken Sie im rechten Fensterausschnitt auf die Domäne und klicken Sie auf **Löschen**.
4. Klicken Sie auf **Ja**.

### Ergebnisse

Die Domäne wird gelöscht.

### Siehe auch

- „DROP DOMAIN-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Entitätsintegrität und referenzielle Integrität

Die relationale Struktur der Datenbank gestattet es dem Datenbankserver, Daten innerhalb der Datenbank zu identifizieren und sicherzustellen, dass alle Zeilen in allen Tabellen die Beziehungen, die im Datenbankschema beschrieben sind, zwischen den Tabellen berücksichtigen.

### Entitätsintegrität

Wenn ein Benutzer eine Zeile einfügt oder aktualisiert, sorgt der Datenbankserver dafür, dass der Primärschlüssel für die Tabelle gültig bleibt: Jede Zeile in der Tabelle ist eindeutig durch einen Primärschlüssel gekennzeichnet.

#### Beispiel 1

Die Tabelle "Employees" in der SQL Anywhere-Beispieldatenbank verwendet eine Mitarbeiterkennung als Primärschlüssel. Wenn Sie der Tabelle einen neuen Mitarbeiter hinzufügen, prüft der Datenbankserver, ob die neue Mitarbeiterkennung eindeutig und nicht NULL ist.

#### Beispiel 2

Die Tabelle "SalesOrderItems" in der SQL Anywhere-Beispieldatenbank verwendet zwei Spalten, um den Primärschlüssel zu definieren.

Diese Tabelle enthält Daten über bestellte Artikel. Eine Spalte enthält eine ID, die eine Bestellung angibt, aber jede Bestellung kann mehrere Artikel beinhalten, sodass diese Spalte selbst nicht der Primärschlüssel sein kann. Die zusätzliche Spalte "LineID" gibt an, welche Zeile dem Artikel entspricht. Die Spalten "ID" und "LineID" zusammen kennzeichnen einen Artikel eindeutig und bilden den Primärschlüssel.

## Wenn eine Clientanwendung die Entitätsintegrität verletzt

Entitätsintegrität erfordert, dass jeder Wert eines Primärschlüssels innerhalb der Tabelle eindeutig ist, und dass NULL nicht vorhanden ist. Wenn die Clientanwendung versucht, einen Primärschlüssel einzufügen bzw. zu aktualisieren und Daten liefert, die nicht eindeutig sind, wäre die Entitätsintegrität verletzt. Eine Verletzung der Entitätsintegrität führt dazu, dass der Datenbank keine neue Daten hinzugefügt werden können und die Clientanwendung stattdessen eine Fehlermeldung erhält.

Sie müssen entscheiden, wie eine Integritätsverletzung dem Benutzer zu präsentieren ist, und wie ihm die Möglichkeit gegeben wird, geeignete Maßnahmen zu ergreifen. Die geeignete Maßnahme besteht im Allgemeinen lediglich darin, den Benutzer um einen anderen, eindeutigen Wert für den Primärschlüssel zu bitten.

## Primärschlüssel erzwingen Entitätsintegrität

Nachdem für jede Tabelle ein Primärschlüssel angegeben wurde, sind keine weiteren Schritte von Seiten der Entwickler der Clientanwendung oder vom Datenbankadministrator zur Aufrechterhaltung der Entitätsintegrität erforderlich.

Der Primärschlüssel für eine Tabelle wird beim Erstellen vom Tabelleneigentümer festgelegt. Wenn die Struktur einer Tabelle zu einem späteren Zeitpunkt modifiziert wird, kann der Primärschlüssel neu festgelegt werden.

#### Siehe auch

- „Primärschlüssel“ auf Seite 17
- „CREATE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

## Referenzielle Integrität

Damit eine Fremdschlüsselbeziehung gültig ist, müssen die Einträge im Fremdschlüssel den Primärschlüsselwerten einer Zeile in der referenzierten Tabelle entsprechen. Gelegentlich kann eine andere eindeutige Spaltenkombination anstelle des Primärschlüssels referenziert werden.

Ein Fremdschlüssel ist eine Referenz auf einen Primärschlüssel oder eine UNIQUE-Regel, normalerweise in einer anderen Tabelle. Wenn dieser Primärschlüssel nicht existiert, wird der bezugslose Fremdschlüssel **Waise** genannt. SQL Anywhere stellt automatisch sicher, dass Ihre Datenbank keine Zeilen enthält, die die referenzielle Integrität verletzen. Dieser Prozess wird **Überprüfen der referenziellen Integrität** genannt. Der Datenbankserver überprüft die referenzielle Integrität, indem er Waisen zählt.

Bei Verwendung eines Fremdschlüssels mit mehreren Spalten können Sie mithilfe der MATCH-Klausel bestimmen, was als verwaiste Zeile zu behandeln ist und was als Verletzung der referenziellen Integrität. 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:

- **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.

#### Beispiel 1

Die SQL Anywhere-Beispieldatenbank enthält eine Mitarbeitertabelle ("Employees") und eine Abteilungstabelle ("Departments"). Der Primärschlüssel für die Tabelle "Employees" ist die Mitarbeiter-

ID, und der Primärschlüssel für die Tabelle "Departments" ist die Abteilungs-ID. In der Mitarbeitertabelle wird die Abteilungs-ID ein **Fremdschlüssel** für die Abteilungstabelle genannt, denn jede Abteilungs-ID in der Mitarbeitertabelle entspricht genau einer Abteilungs-ID in der Abteilungstabelle.

Die Fremdschlüssel-Beziehung ist eine Viele-zu-Eins-Beziehung. Mehrere Einträge in der Tabelle "Employees" haben dieselbe Abteilungs-ID, aber die Abteilungs-ID ist der Primärschlüssel für die Tabelle "Departments" und er ist somit eindeutig. Wenn ein Fremdschlüssel in der Lage wäre, sich auf eine Spalte in der Tabelle "Departments" zu beziehen, die mehrfach vorhandene Einträge oder NULL enthält, wäre es nicht möglich herauszufinden, welche Zeile in der Tabelle "Departments" die gewünschte Referenz ist. Dies lässt sich verhindern, indem die Fremdschlüsselspalte als NOT NULL festgelegt wird.

### Beispiel 2

Nehmen Sie an, die Datenbank enthielte ebenfalls eine Niederlassungstabelle mit den Standorten der Niederlassungen. Die Tabelle "Employees" kann einen Fremdschlüssel für die Niederlassungstabelle haben, der angibt, in welchem Ort sich die Niederlassung des Mitarbeiters befindet. Der Datenbankentwickler könnte sich entscheiden, bei der Einstellung eines Mitarbeiters keine Niederlassung anzugeben, weil der Mitarbeiter entweder noch keiner Niederlassung zugeordnet wurde oder nicht von einem Büro aus arbeitet. In diesem Fall ist der Fremdschlüssel optional und kann NULL zulassen.

### Beispiel 3

Führen Sie die folgende Anweisung aus, um einen zusammengesetzten Primärschlüssel zu erstellen.

```
CREATE TABLE pt(
 pk1 INT NOT NULL,
 pk2 INT NOT NULL,
 str VARCHAR(10)
 PRIMARY KEY (pk1, pk2));
```

Die folgenden Anweisungen erstellen einen Fremdschlüssel, der eine andere Spaltenreihenfolge aufweist als der Primärschlüssel, und eine andere Sortierung für die Fremdschlüsselspalten, die verwendet wird, um den Fremdschlüsselindex zu erstellen.

```
CREATE TABLE ft1(
 fpk INT PRIMARY KEY,
 ref1 INT,
 ref2 INT);

ALTER TABLE ft1 ADD FOREIGN KEY (ref2 ASC, ref1 DESC)
 REFERENCES pt (pk2, pk1) MATCH SIMPLE;
```

Führen Sie die folgenden Anweisungen aus, um einen Fremdschlüssel zu erstellen, der dieselbe Spaltenreihenfolge aufweist wie der Primärschlüssel, aber eine andere Sortierung für den Fremdschlüsselindex. Außerdem wird im Beispiel die MATCH FULL-Klausel verwendet, um anzugeben, dass verwaiste Zeilen resultieren, wenn beide Spalten NULL sind. Die UNIQUE-Klausel erzwingt eine Eins-zu-Eins-Relation zwischen der **pt**-Tabelle und der **ft2**-Tabelle für Spalten, die nicht NULL sind.

```
CREATE TABLE ft2(
 fpk INT PRIMARY KEY,
 ref1 INT,
 ref2 INT);

ALTER TABLE ft2 ADD FOREIGN KEY (ref1, ref2 DESC)
 REFERENCES pt (pk1, pk2) MATCH UNIQUE FULL;
```



## Referenzielle Zyklen

Eine referenzierende Tabelle und eine referenzierte Tabelle müssen nicht unterschiedlich sein. Eine Tabelle kann einen Fremdschlüssel enthalten, der sich selbst referenziert. Dies wird als selbstreferenzierende Tabelle bezeichnet. Eine selbstreferenzierende Tabelle ist ein Spezialfall eines referenziellen Zyklus.

### Beispiel 1

Die SQL Anywhere-Beispieldatenbank enthält eine Tabelle mit Mitarbeiterdaten und eine weitere mit Abteilungsdaten:

```
CREATE TABLE "GROUPO"."Employees" (
 "EmployeeID" int NOT NULL
 , "ManagerID" int NULL
 , "Surname" "person_name_t" NOT NULL
 , "GivenName" "person_name_t" NOT NULL
 , "DepartmentID" int NOT NULL
 , "Street" "street_t" NOT NULL
 , "City" "city_t" NOT NULL
 , "State" "state_t" NULL
 , "Country" "country_t" NULL
 , "PostalCode" "postal_code_t" NULL
 , "Phone" "phone_number_t" NULL
 , "Status" char(2) NULL
 , "SocialSecurityNumber" char(11) NOT NULL
 , "Salary" numeric(20,3) NOT NULL
 , "StartDate" date NOT NULL
 , "TerminationDate" date NULL
 , "BirthDate" date NULL
 , "BenefitHealthInsurance" bit NULL
 , "BenefitLifeInsurance" bit NULL
 , "BenefitDayCare" bit NULL
 , "Sex" char(2) NULL CONSTRAINT "Sexes"
 check(Sex in('F','M','NA'))
 , CONSTRAINT "EmployeesKey" PRIMARY KEY ("EmployeeID")
)

ALTER TABLE "GROUPO"."Employees"
 ADD CONSTRAINT "SSN" UNIQUE ("SocialSecurityNumber")

CREATE TABLE "GROUPO"."Departments" (
 "DepartmentID" int NOT NULL
 , "DepartmentName" char(40) NOT NULL
 , "DepartmentHeadID" int NULL
 , CONSTRAINT "DepartmentsKey" PRIMARY KEY ("DepartmentID")
 , CONSTRAINT "DepartmentRange" check(DepartmentID > 0 and DepartmentID <=
999)
)
```

Die Employees-Tabelle enthält den Primärschlüssel "EmployeeID" und den möglichen Schlüssel "SocialSecurityNumber". Die Departments-Tabelle enthält den Primärschlüssel "DepartmentID". Die Employees-Tabelle ist durch die Definition des Fremdschlüssels mit der Departments-Tabelle verknüpft:

```
ALTER TABLE "GROUPO"."Employees"
 ADD NOT NULL FOREIGN KEY "FK_DepartmentID_DepartmentID" ("DepartmentID")
 REFERENCES "GROUPO"."Departments" ("DepartmentID")
```

Um den Namen der Abteilung eines bestimmten Mitarbeiters finden zu können, muss der Name dieser Abteilung nicht in der Employees-Tabelle gespeichert werden. Stattdessen enthält die Employees-Tabelle

eine Spalte "DepartmentID" mit der Abteilungsnummer, die einem der DepartmentID-Werte in der Departments-Tabelle entspricht.

Die Employees-Tabelle referenziert die Departments-Tabelle durch die oben genannte Integritätsregel zur Erhaltung der referenziellen Integrität, mit der eine Viele-zu-Eins-Beziehung zwischen Employees- und Departments-Tabelle deklariert wird. Außerdem ist dies eine obligatorische Beziehung, weil die Fremdschlüsselspalte in der Employees-Tabelle, "DepartmentID", als NOT NULL deklariert ist. Dies ist jedoch nicht die einzige Beziehung zwischen Employees- und Departments-Tabelle, weil die Departments-Tabelle selbst einen Fremdschlüssel zur Employees-Tabelle für die Leiter der einzelnen Abteilungen enthält:

```
ALTER TABLE "GROUPO"."Departments"
 ADD FOREIGN KEY "FK_DepartmentHeadID_EmployeeID" ("DepartmentHeadID")
 REFERENCES "GROUPO"."Employees" ("EmployeeID")
 ON DELETE SET NULL
```

Dies stellt eine optionale Viele-zu-Eins-Beziehung zwischen der Departments-Tabelle und der Employees-Tabelle dar. Es handelt sich um eine Viele-zu-Eins-Beziehung, weil die Integritätsregel zur Erhaltung der referenziellen Integrität allein nicht verhindern kann, dass zwei oder mehr Abteilungen denselben Leiter haben. Daher bilden Employees- und Departments-Tabelle einen referenziellen Zyklus, wobei jede einen Fremdschlüssel zur anderen enthält.

## Fremdschlüssel erzwingen referenzielle Integrität

Wie bei Primärschlüsseln benutzen Sie die Anweisungen CREATE TABLE oder ALTER TABLE, um Fremdschlüssel zu erstellen. Wenn Sie einen Fremdschlüssel erstellt haben, können die Spalten in dem Schlüssel nur Werte enthalten, die in der Tabelle, die dem Fremdschlüssel zugeordnet sind, als Primärschlüsselwerte vorhanden sind.

## Verlust der referenziellen Integrität

Eine Datenbank kann die referenzielle Integrität verlieren, wenn eine der folgenden Situationen eintritt:

- Der Wert eines Primärschlüssels wird aktualisiert oder gelöscht. Alle Fremdschlüssel, die sich auf diesen Primärschlüssel beziehen, werden ungültig.
- Eine neue Zeile wird in die Fremdtabelle eingefügt, und es wird ein Wert für den Fremdschlüssel eingegeben, für den es keinen entsprechenden Primärschlüsselwert gibt. Die Datenbank würde damit ungültig.

SQL Anywhere bietet Schutz vor beiden Typen von Integritätsverlust.

## Wenn eine Clientanwendung die referenzielle Integrität verletzt

Wenn eine Clientanwendung einen Primärschlüsselwert in einer Tabelle aktualisiert oder löscht und der Primärschlüsselwert von einem Fremdschlüssel anderswo in der Datenbank referenziert wird, besteht die Gefahr, dass die referenzielle Integrität verletzt wird.

### Beispiel

Wenn der Server gestattet, dass der Primärschlüssel aktualisiert oder gelöscht wird, und keine Änderungen an den Fremdschlüsseln vorgenommen hat, die sich auf diesen bezogen haben, wäre die Fremdschlüsselreferenz ungültig. Jeder Versuch, die Fremdschlüsselreferenz zu benutzen, z.B. in einer SELECT-Anweisung mit einer KEY JOIN-Klausel, würde fehlschlagen, da in der referenzierten Tabelle kein korrespondierender Wert vorhanden ist.

Während die Verletzung der Entitätsintegrität für SQL Anywhere im Allgemeinen einfach zu handhaben ist, und zwar indem der Zugang zu den Daten verweigert und eine Fehlermeldung zurückgegeben wird, sind potenzielle Verletzungen der referenziellen Integrität komplizierter. Es stehen Ihnen mehrere Möglichkeiten zur Verfügung (sogenannte Aktionen zur referenziellen Integrität), um die referenzielle Integrität aufrechtzuerhalten.

## Aktionen zur referenziellen Integrität

Im einfachsten Fall wird die referenzielle Integrität beim Aktualisieren oder Löschen eines referenzierten Primärschlüssels dadurch bewahrt, dass Aktualisieren oder Löschen nicht erlaubt werden. Oft ist es jedoch auch möglich, für jeden Fremdschlüssel eine Maßnahme zu ergreifen, um die referenzielle Integrität aufrechtzuerhalten. Mit den Anweisungen CREATE TABLE und ALTER TABLE kann der Datenbankadministrator bzw. der Tabelleneigentümer angeben, welche Maßnahmen in Bezug auf Fremdschlüssel bei einer Verletzung ergriffen werden müssen, die sich auf einen geänderten Primärschlüssel beziehen.

### Hinweis

Aktionen zur referenziellen Integrität werden durch **physische**, nicht **logische**, Aktualisierungen des eindeutigen Werts ausgelöst. Beispielsweise wird selbst in einer Datenbank ohne Berücksichtigung von Groß- und Kleinschreibung durch das Aktualisieren des Primärschlüsselwerts von *SAMPLE-VALUE* auf *sample-value* eine Aktion zur referenziellen Integrität ausgelöst, auch wenn die beiden Werte logisch gleich sind.

Sie können jede der folgenden Aktionen zur referenziellen Integrität separat für Aktualisierungen und Löschungen des Primärschlüssels angeben:

- **RESTRICT** Erzeugt einen Fehler und verhindert die Änderung, wenn versucht wird, einen referenzierten Primärschlüssel zu ändern. Dies ist die Standardmaßnahme für referenzielle Integrität.
- **SET NULL** Setzt alle Fremdschlüssel, die sich auf den referenzierten Primärschlüssel beziehen, auf NULL.

- **SET DEFAULT** Setzt alle Fremdschlüssel, die sich auf den geänderten Primärschlüssel beziehen, auf den Standardwert für die betreffende Spalte (wie in der Tabellendefinition angegeben).
- **CASCADE** Im Zusammenhang mit ON UPDATE: Aktualisiert alle Fremdschlüssel, die sich auf den aktualisierten Primärschlüssel beziehen, auf den neuen Wert. Im Zusammenhang mit ON DELETE: Löscht alle Zeilen, die sich auf den gelöschten Primärschlüssel beziehen.

Systemtrigger implementieren Aktionen für die referenzielle Integrität. Der Trigger, der für die Primärtabelle definiert ist, wird mit den Privilegien des Eigentümers der Sekundärtabelle ausgeführt. Dieses Verhalten bedeutet, dass kaskadierte Vorgänge zwischen Tabellen mit unterschiedlichen Eigentümern stattfinden können, ohne dass zusätzliche Privilegien erteilt werden müssen.

## Referenzielle Integrität prüfen

Für Fremdschlüssel, die zur Einschränkung (RESTRICT) von Vorgängen definiert wurden, die die referenzielle Integrität verletzen würden, werden zum Zeitpunkt der Ausführung einer Anweisung Prüfungen durchgeführt. Wenn Sie die Klausel CHECK ON COMMIT angeben, werden nur dann Prüfungen ausgeführt, wenn die Transaktion festgeschrieben wird.

### Datenbankoption zur Steuerung des Zeitpunktes der Überprüfung verwenden

Die Einstellung der Datenbankoption "wait\_for\_commit" steuert das Verhalten, wenn ein Fremdschlüssel zur Einschränkung von Vorgängen definiert wird, die die referenzielle Integrität verletzen würden. Die CHECK ON COMMIT-Klausel kann diese Option aufheben.

Wird der Standardwert "wait\_for\_commit" auf "Off" gesetzt, werden Vorgänge, die zu einer inkonsistenten Datenbank führen würden, für die Ausführung gesperrt. Ein DELETE-Vorgang für eine Abteilung, die Mitarbeiter enthält, wäre z.B. nicht zulässig. Die folgende Anweisung liefert einen Fehler:

```
DELETE FROM Departments
WHERE DepartmentID = 200;
```

Wenn "wait\_for\_commit" auf "On" gesetzt wird, bleibt die referenzielle Integrität ungeprüft, bis ein COMMIT ausgeführt wird. Wenn die Datenbank inkonsistent ist, wird das Festschreiben nicht gestattet, und es wird ein Fehler ausgegeben. In diesem Modus könnte ein Datenbankbenutzer zwar eine Abteilung löschen, die Mitarbeiter enthält, aber die Änderungen der Datenbank können erst festgeschrieben werden, wenn eine der folgenden Bedingungen erfüllt ist:

- Die Mitarbeiter dieser Abteilung wurden gelöscht oder neu zugeordnet.
- Die Zeile für die DepartmentID wurde wieder in die Tabelle "Departments" eingefügt.
- Die Transaktion wurde zurückgesetzt, um die DELETE-Operation zu widerrufen.

## Integritätsprüfungen bei INSERT

SQL Anywhere führt Integritätsprüfungen während der Ausführung von INSERT-Anweisungen durch. Beispiel: Sie möchten eine Abteilung erstellen, geben jedoch einen bereits verwendeten Wert in "DepartmentID" ein:

```
INSERT
INTO Departments (DepartmentID, DepartmentName, DepartmentHeadID)
VALUES (200, 'Eastern Sales', 902);
```

Die INSERT-Anweisung wird zurückgewiesen, weil der Primärschlüssel für die Tabelle nicht mehr eindeutig wäre. Da es sich bei der Spalte "DepartmentID" um einen Primärschlüssel handelt, sind doppelte Werte in diesem Feld nicht zulässig.

### Werte einfügen, die Beziehungen verletzen

Folgende Anweisung fügt eine neue Zeile in die Tabelle "SalesOrders" ein. Sie enthält aber fälschlicherweise eine "SalesRepresentative ID", die in der Tabelle "Employees" nicht vorhanden ist.

```
INSERT
INTO SalesOrders (ID, CustomerID, OrderDate, SalesRepresentative)
VALUES (2700, 186, '2000-10-19', 284);
```

Zwischen den Tabellen "Employees" und "SalesOrders" gibt es eine "Eins-zu-viele-Beziehung", die auf der Spalte "SalesRepresentative" in der Tabelle "SalesOrders" und der Spalte "EmployeeID" in der Tabelle "Employees" basiert. Erst, nachdem ein Datensatz in der Primärtabelle (Employees) eingegeben wurde, kann ein entsprechender Datensatz in der Fremdtabelle (SalesOrders) eingefügt werden.

### Fremdschlüssel

Der Primärschlüssel für die Mitarbeitertabelle "Employees" ist die Mitarbeiter-ID. Die Verkäufer-ID in der Verkäufertabelle "SalesRepresentative" ist ein Fremdschlüssel für die Tabelle "Employees". Das heißt, dass jede Verkäufersnummer in der Auftrags-tabelle "SalesOrders" mit einer Mitarbeiter-ID für einen Mitarbeiter in der Tabelle "Employees" übereinstimmen muss.

Wenn Sie versuchen, einen Auftrag für den Verkäufer 284 hinzuzufügen, wird eine Fehlermeldung ausgegeben.

In der Mitarbeitertabelle "Employees" ist kein Mitarbeiter mit dieser ID enthalten. Dies verhindert, dass Sie Bestellungen ohne eine gültige Handelsvertreter-ID einfügen.

### Siehe auch

- „Beziehungen zwischen Tabellen“ [[SQL Anywhere 16 - Einführung](#)]

## Integritätsprüfungen bei DELETE oder UPDATE

Fremdschlüsselfehler können auch dann auftreten, wenn Sie einen Aktualisierungs- oder einen Löschvorgang ausführen. Beispiel: Sie möchten die Abteilung R&D aus der Abteilungstabelle "Departments" löschen. Das Feld "DepartmentID" ist der Primärschlüssel der Abteilungstabelle "Departments" und stellt die EINS-Seite der Eins-zu-Viele-Beziehung dar. Beim Feld "DepartmentID" der Tabelle "Employees" handelt es sich um den entsprechenden Fremdschlüssel und damit um die VIELE-Seite. Ein Datensatz auf der EINS-Seite einer solchen Beziehung kann erst dann gelöscht werden, wenn alle entsprechenden Einträge auf der VIELE-Seite ebenfalls gelöscht sind.

### Fehler der referenziellen Integrität bei DELETE

Angenommen, Sie versuchen, die Abteilung R&D (DepartmentID 100) in der Departments-Tabelle zu löschen. Es wird ein Fehler gemeldet, der darauf hinweist, dass andere Datensätze in der Datenbank

enthalten sind, die die Abteilung R&D referenzieren, und der Löschvorgang wird nicht ausgeführt. Um die Abteilung R&D zu löschen, müssen Sie zuerst die darin enthaltenen Mitarbeiter entfernen, und zwar folgendermaßen:

```
DELETE
FROM Employees
WHERE DepartmentID = 100;
```

Nachdem Sie alle Mitarbeiter, die zur Abteilung R&D gehören, gelöscht haben, können Sie nun die Abteilung R&D löschen.

```
DELETE
FROM Departments
WHERE DepartmentID = 100;
```

Machen Sie diese Änderungen in der Datenbank rückgängig, indem Sie eine ROLLBACK-Anweisung eingeben:

```
ROLLBACK;
```

### Fehler der referenziellen Integrität bei UPDATE

Jetzt möchten Sie zum Beispiel das Feld "DepartmentID" in der Tabelle "Employees" ändern. Das Feld "DepartmentID" ist der Fremdschlüssel der Tabelle "Employees" und stellt die VIELE-Seite der Eins-zu-Viele-Beziehung dar. Beim Feld "DepartmentID" der Tabelle "Departments" handelt es sich um den entsprechenden Primärschlüssel und damit um die EINS-Seite. Ein Datensatz auf der VIELE-Seite einer solchen Beziehung kann erst dann geändert werden, wenn er einen entsprechenden Eintrag auf der EINS-Seite besitzt. Das heißt, erst dann, wenn er über einen Primärschlüssel zum Referenzieren verfügt.

Folgende UPDATE-Anweisung verursacht zum Beispiel einen Integritätsfehler:

```
UPDATE Employees
SET DepartmentID = 600
WHERE DepartmentID = 100;
```

Es wird eine Fehlermeldung ausgegeben, weil in der Departments-Tabelle keine Abteilung mit dem DepartmentID-Wert 600 vorhanden ist.

Um den Wert im Feld "DepartmentID" in der Tabelle "Employees" ändern zu können, muss er mit einem vorhandenen Wert in der Tabelle "Departments" übereinstimmen. Zum Beispiel:

```
UPDATE Employees
SET DepartmentID = 300
WHERE DepartmentID = 100;
```

Diese Anweisung kann ausgeführt werden, da die "DepartmentID" 300 mit der bereits vorhandenen Abteilung "Finance" übereinstimmt.

Machen Sie diese Änderungen in der Datenbank rückgängig, indem Sie eine ROLLBACK-Anweisung eingeben:

```
ROLLBACK;
```

### Integrität zum Zeitpunkt der Festschreibung mit COMMIT prüfen

In den oben genannten Beispielen wurde die Integrität der Datenbank bei der Ausführung der einzelnen Anweisungen geprüft. Es wird kein Vorgang ausgeführt, der zu einer Inkonsistenz in der Datenbank führen würde.

Sie können die Datenbank auch so konfigurieren, dass die Integrität erst zum Zeitpunkt der Festschreibung mit COMMIT geprüft wird, indem Sie die `wait_for_commit`-Option verwenden. Wählen Sie diese Option, wenn Sie Änderungen vornehmen möchten, die bei der Eingabe von Änderungen zu temporären Dateninkonsistenzen führen können. Beispiel: Sie möchten die Abteilung R&D aus den Tabellen "Employees" und "Departments" löschen. Da diese Tabellen aufeinander verweisen und die Löschungen jeweils nur in einer Tabelle durchgeführt werden können, sind in den Tabellen während des Löschvorgangs Inkonsistenzen vorhanden. In diesem Fall kann die Datenbank erst einen COMMIT-Vorgang ausführen, wenn der Löschvorgang abgeschlossen ist. Aktivieren Sie die `wait_for_commit`-Option, um Dateninkonsistenzen zuzulassen, bis der COMMIT-Vorgang ausgeführt wird.

Sie können auch Fremdschlüssel so definieren, dass sie automatisch entsprechend den Änderungen der Primärschlüssel abgeändert werden. Wenn für den Fremdschlüssel aus "Employees" zu "Departments" zum Beispiel mit `ON DELETE CASCADE` eine kaskadierende Löschung festgelegt ist, werden bei einer Löschung der Abteilungskennung automatisch auch die entsprechenden Einträge in der Tabelle "Employees" gelöscht.

In den obigen Fällen besteht keine Möglichkeit, eine inkonsistente Datenbank auf Dauer festzuschreiben. SQL Anywhere unterstützt auch alternative Aktionen, wenn Änderungen die Datenbank inkonsistent machen würden.

#### Siehe auch

- „`wait_for_commit`-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Datenintegrität“ auf Seite 849

## Integritätsregeln in den Systemtabellen

Alle Informationen über Integritätsprüfungen und Regeln für eine Datenbank werden in Systemtabellen gespeichert: Verwenden Sie die zugehörigen Systemansichten wie folgt, um auf diese Informationen zuzugreifen:

| Systemansicht     | Beschreibung                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SYS.SYSCONSTRAINT | <p>Jede Zeile in der Systemansicht SYS.SYSCONSTRAINT beschreibt eine Integritätsregel in der Datenbank. Die derzeit unterstützten Integritätsregeln umfassen Tabellen- und Spaltenprüfungen, Primärschlüssel, Fremdschlüssel und Eindeutigkeits-Integritätsregeln.</p> <p>Bei Prüf-Integritätsregeln für Tabellen und Spalten befindet sich die CHECK-Bedingung in der SYS.ISYSCHECK-Systemtabelle.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| SYS.SYSCHECK      | <p>Jede Zeile in der Systemansicht SYS.SYSCHECK definiert eine Prüf-Integritätsregel, die in der Systemansicht SYS.SYSCONSTRAINT aufgeführt wird.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| SYS.SYSFKEY       | <p>Jede Zeile in der Systemansicht SYS.SYSFKEY beschreibt einen Fremdschlüssel einschließlich des Übereinstimmungstyps, der für den Schlüssel definiert wurde.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| SYS.SYSIDX        | <p>Jede Zeile in der Systemansicht SYS.SYSIDX definiert einen Index in der Datenbank.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| SYS.SYSTRIGGER    | <p>Jede Zeile in der Systemansicht SYS.SYSTRIGGER beschreibt einen Trigger in der Datenbank, einschließlich solcher Trigger, die automatisch für Fremdschlüssel-Integritätsregeln erstellt werden, die über eine referenzielle Trigger-Aktion verfügen (wie etwa ON DELETE CASCADE).</p> <p>In der Spalte "referential_action" befindet sich ein Buchstabe, der angibt, ob die Aktion "cascade" (C), "delete" (D), "set null" (N) oder "restrict" (R) ist.</p> <p>In der Spalte "event" befindet sich ein Buchstabe, der angibt, welches Ereignis die Ausführung der Aktion veranlasst: A=insert und delete, B=insert und update, C=update, D=delete, E=delete und update, I=insert, U=update, M=insert, delete und update.</p> <p>In der Spalte "trigger_time" wird angezeigt, ob die Aktion nach (after - A) oder vor (before - B) dem Trigger-Ereignis ausgeführt wird.</p> |



**Siehe auch**

- „SYSCONSTRAINT-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSCHECK-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSFKEY-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSIDX-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSTRIGGER-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]



---

# Transaktionen und Isolationsstufen

Um die Datenintegrität zu gewährleisten, ist es wichtig, dass Sie Zustände definieren können, in denen die Informationen in Ihrer Datenbank **konsistent** sind. Das Konzept der Konsistenz wird am besten durch ein Beispiel erläutert:

## Beispiel für Konsistenz

Angenommen, in Ihrer Datenbank werden Geldkonten verwaltet, und Sie möchten Geld von einem Konto auf ein anderes überweisen. Der Zustand der Datenbank ist konsistent, bevor und nachdem das Geld überwiesen wurde, jedoch nicht, nachdem Sie den Betrag einem Konto belastet haben und bevor Sie den Betrag dem anderen Konto gutgeschrieben haben. Während der Überweisung des Geldes ist die Datenbank in einem konsistenten Zustand, solange der Gesamtbetrag auf den Konten so hoch ist, wie er vor der Überweisung war. Wenn die Überweisung erst halb durchgeführt wurde, ist die Datenbank in einem nicht-konsistenten Zustand. Es müssen entweder sowohl die Last- als auch die Gutschrift verarbeitet werden, oder keine von beiden.

## Transaktionen sind logische Arbeitseinheiten

Eine **Transaktion** ist eine logische Arbeitseinheit. Jede Transaktion ist eine Sequenz von logisch verknüpften Anweisungen, die eine Aufgabe ausführen und die Datenbank von einem konsistenten Zustand in einen anderen versetzen. Die Eigenschaften eines konsistenten Zustandes hängen von der jeweiligen Datenbank ab.

Die Anweisungen in einer Transaktion werden als unteilbare Einheit behandelt: Entweder werden alle ausgeführt oder keine. Am Ende jeder Transaktion können Sie Ihre Änderungen **festschreiben**, um sie damit dauerhaft gültig zu machen. Wenn aus irgendeinem Grund einige Anweisungen in einer Transaktion nicht richtig verarbeitet wurden, werden alle vorübergehenden Änderungen gelöscht oder **zurückgesetzt**. Man könnte auch sagen, dass Transaktionen **atomar** sind.

Die Gruppierung von Anweisungen in Transaktionen ist der Schlüssel sowohl zum Schutz der Datenkonsistenz bei Datenträger- oder Systemausfall als auch zum Verwalten gleichzeitiger Datenbankvorgänge. Transaktionen können gefahrlos verflochten werden, und die Fertigstellung jeder Transaktion stellt einen Punkt dar, an dem die Daten der Datenbank konsistent sind. Jede Transaktion muss so entworfen werden, dass sie eine Aufgabe ausführt, die Ihre Datenbank von einem konsistenten Status in den nächsten versetzt.

Bei einem Systemausfall oder einem Datenbankabsturz im normalen Betrieb führt SQL Anywhere eine automatische Wiederherstellung der Daten durch, wenn die Datenbank das nächste Mal gestartet wird. Bei der automatischen Wiederherstellung werden alle abgeschlossenen Transaktionen wieder hergestellt, und alle Transaktionen, die beim Auftreten des Fehlers noch nicht festgeschrieben waren, werden zurückgesetzt. Durch die atomare Eigenschaft der Transaktionen wird sichergestellt, dass Datenbanken in einem konsistenten Zustand wieder hergestellt werden.

## Siehe auch

- „Sicherung und Datenwiederherstellung“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Parallelität“ auf Seite 884

## Transaktionen

SQL Anywhere erwartet, dass Sie Ihre Anweisungen in Transaktionen zusammenfassen. Wenn Sie eine solche Transaktion festschreiben, dann werden die Änderungen in der Datenbank permanent gespeichert. Wenn Sie Daten ändern, werden Ihre Änderungen im Transaktionslog aufgezeichnet und erst dann dauerhaft gemacht, wenn Sie die COMMIT-Anweisung eingeben.

Transaktionen beginnen mit einem der folgenden Ereignisse:

- Erste Anweisung nach erfolgter Verbindung mit einer Datenbank
- Erste Anweisung nach dem Ende einer Transaktion

Transaktionen werden mit einem der folgenden Ereignisse abgeschlossen:

- Eine COMMIT-Anweisung schreibt die Änderungen in der Datenbank fest.
- Eine ROLLBACK-Anweisung macht alle durch die Transaktionen durchgeführten Änderungen rückgängig.
- Eine Anweisung wird ausgeführt und schreibt gleichzeitig auch alle Änderungen automatisch fest: Dies gilt für die meisten Datendefinitionsanweisungen, wie z.B. ALTER, CREATE, COMMENT und DROP.
- Durch die Unterbrechung der Verbindung zur Datenbank wird ein indirektes Zurücksetzen ausgeführt.
- ODBC und JDBC haben eine Autocommit-Einstellung, die COMMIT nach jeder Anweisung erzwingt. Standardmäßig erfordern ODBC und JDBC, dass Autocommit aktiviert und jede Anweisung eine einzelne Transaktion ist. Wenn Sie die Möglichkeiten von Transaktionen nutzen, müssen Sie Autocommit ausschalten.
- Wenn die Datenbankoption "chained" auf "Off" gesetzt wird, hat dies ähnliche Auswirkungen wie das Erzwingen von "Autocommit" nach jeder Anweisung. Standardmäßig ist "chained" bei Verbindungen, die jConnect- oder Open Client-Anwendungen benutzen, auf "Off" gesetzt.

Sie können ermitteln, welche Verbindungen ausstehende Transaktionen haben, indem Sie eine Verbindung zu einer Datenbank mit dem SQL Anywhere-Konsolendienstprogramm (dbconsole) herstellen. Überprüfen Sie den Bereich **Verbindungen**, um zu sehen, welche Verbindung nicht festgeschriebene Vorgänge aufweist.

### Optionen für Interactive SQL

Interactive SQL stellt zwei Optionen zur Verfügung, mit denen Sie steuern können, wann und wie Transaktionen beendet werden:

- Wenn Sie die Option "auto\_commit" auf "On" setzen, schreibt Interactive SQL automatisch die Ergebnisse jeder erfolgreichen Anweisung fest und führt nach jeder fehlerhaften Anweisung automatisch ein ROLLBACK durch.
- Die Einstellung der Option "commit\_on\_exit" legt fest, was mit den nicht festgeschriebenen Änderungen passiert, wenn Sie Interactive SQL beenden. Wird diese Option auf "On" gestellt

(Standardwert), so führt Interactive SQL einen COMMIT-Befehl aus. Ansonsten werden Ihre nicht festgeschriebenen Änderungen mit einem ROLLBACK-Befehl ungeschehen gemacht.

#### Eine Datenquelle in Interactive SQL benutzen

In der Standardeinstellung arbeitet ODBC im Autocommit-Modus. Auch wenn Sie die auto\_commit-Option in Interactive SQL auf "Off" gesetzt haben, überschreibt die ODBC-Einstellung die Einstellungen von Interactive SQL. Sie können die ODBC-Einstellung mit dem SQL\_ATTR\_AUTOCOMMIT-Verbindungsattribut ändern. ODBC-Autocommit ist von der Option "chained" unabhängig.

SQL Anywhere unterstützt auch Transact-SQL-Anweisungen, wie z.B. BEGIN TRANSACTION, um die Kompatibilität mit Adaptive Server Enterprise zu gewährleisten.

#### Startzeit einer Transaktion ermitteln

Die Datenbankeigenschaft "TransactionStartTime" gibt die Uhrzeit zurück, zu der die Datenbank erstmals nach einem COMMIT oder ROLLBACK geändert wurde. Sie können diese Eigenschaft verwenden, um die Startzeit der frühesten Transaktion für alle aktiven Verbindungen zu ermitteln. Beispiel:

```
BEGIN
 DECLARE connid int;
 DECLARE earliest char(50);
 DECLARE connstart char(50);
 SET connid=next_connection(null);
 SET earliest = NULL;
 lp: LOOP
 IF connid IS NULL THEN LEAVE lp END IF;
 SET connstart = CONNECTION_PROPERTY('TransactionStartTime',connid);
 IF connstart <> '' THEN
 IF earliest IS NULL
 OR CAST(connstart AS TIMESTAMP) < CAST(earliest AS TIMESTAMP) THEN
 SET earliest = connstart;
 END IF;
 END IF;
 SET connid=next_connection(connid);
 END LOOP;
 SELECT earliest
END
```

#### Siehe auch

- „SQL-Anweisungen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Autocommit-Modus und manueller Commit-Modus“ [[SQL Anywhere Server - Programmierung](#)]
- „chained-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „auto\_commit-Option [Interactive SQL]“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „commit\_on\_exit-Option [Interactive SQL]“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Transact-SQL-Kompatibilität“ auf Seite 648
- „SQL Anywhere-Konsolendienstprogramm (dbconsole)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Verbindungen mit ausstehenden Transaktionen ermitteln (SQL)“ [[SQL Anywhere Server - Datenbankadministration](#)]

## Parallelität

Parallelität ist die Fähigkeit des Datenbankservers, mehrere Transaktionen gleichzeitig zu verarbeiten. Nur aufgrund der speziellen Verfahren innerhalb des Datenbankservers ist es möglich, dass gleichzeitige Transaktionen einander nicht stören und es zu keinem inkonsistenten und fehlerhaften Datenbestand kommt.

### Wer muss über Parallelität Bescheid wissen

Parallelität betrifft alle Datenbankadministratoren und Entwickler. Auch wenn Sie nur mit einer Einzelplatz-Datenbank arbeiten, müssen Sie sich mit Parallelität befassen, falls Sie Anforderungen von mehreren Anwendungen oder auch nur von mehreren Verbindungen in einer einzelnen Anwendung verarbeiten. Diese Anwendungen und Verbindungen können einander genauso überlagern, wie im Falle von mehreren Benutzern in einer Netzwerkumgebung.

### Transaktionsgröße wirkt sich auf die Parallelität aus

Die Art, wie Sie SQL-Anweisungen in Transaktionen zusammenfassen, hat wesentliche Auswirkungen auf die Datenintegrität und die Systemperformance. Wenn eine Transaktion zu kurz ist und keine vollständige logische Arbeitseinheit enthält, können in der Datenbank Inkonsistenzen auftreten. Wenn eine Transaktion zu lang ist und einige nicht zusammenhängende Aktionen enthält, ist es sehr leicht möglich, dass ein ROLLBACK-Befehl unnötigerweise Arbeit löscht, die ohne Schwierigkeiten in der Datenbank hätte festgeschrieben werden können.

Wenn Ihre Transaktionen lang sind, kann die Parallelität verringert werden, indem die gleichzeitige Bearbeitung von anderen Transaktionen verhindert wird.

Es gibt viele Faktoren, die sich auf die richtige Länge einer Transaktion auswirken, je nach Art der Anwendung und der Umgebung.

### Siehe auch

- „SQL Anywhere-Datenbankserver“ [[SQL Anywhere Server - Datenbankadministration](#)]

## Savepoints innerhalb von Transaktionen

Sie können wichtige Zustände innerhalb einer Transaktion erkennen und mithilfe von **Savepoints** wahlweise zu diesen zurückkehren, um Gruppen von zusammengehörigen Anweisungen zu trennen.

Eine SAVEPOINT-Anweisung definiert einen Zwischenpunkt in einer Transaktion. Sie können alle Änderungen bis zu diesem Punkt rückgängig machen, indem sie den Befehl ROLLBACK TO SAVEPOINT verwenden. Sobald eine RELEASE SAVEPOINT-Anweisung ausgeführt oder die Transaktion beendet wurde, können Sie den Savepoint nicht mehr benutzen. Savepoints haben keine Auswirkungen auf COMMITs. Wenn ein COMMIT ausgeführt wird, werden alle Änderungen in der Datenbank als permanent festgeschrieben.

Durch RELEASE SAVEPOINT oder ROLLBACK TO SAVEPOINT werden keine Sperren freigegeben: Sperren werden erst am Ende einer Transaktion freigegeben.

### Savepoints benennen und verschachteln

Wenn Sie benannte, verschachtelte Savepoints verwenden, können Sie viele aktive Savepoints innerhalb einer Transaktion setzen. Änderungen zwischen einem SAVEPOINT und einem RELEASE SAVEPOINT können aufgehoben werden, indem das Zurücksetzen bis zu einem früheren Savepoint oder auf die Transaktion selbst ausgedehnt wird. Änderungen innerhalb einer Transaktion werden erst zu einem festen Bestandteil der Datenbank, wenn sie festgeschrieben sind. Alle Savepoints werden bei der Beendigung einer Transaktion freigegeben.

Savepoints können im Massenbetriebsmodus nicht verwendet werden. Bei der Verwendung von Savepoints gibt es nur geringe zusätzliche Overheads.

## Isolationsstufen und Konsistenz

In SQL Anywhere können Sie kontrollieren, inwieweit Vorgänge in einer Transaktion für die Vorgänge in anderen gleichzeitigen Transaktionen sichtbar sind. Dazu setzen Sie eine Datenbankoption namens **Isolationsstufe**.

In SQL Anywhere können Sie außerdem die Isolationsstufen einzelner Tabellen in einer Abfrage mit entsprechenden Tabellen-Hints kontrollieren.

SQL Anywhere unterstützt die folgenden Isolationsstufen:

| Isolationsstufe                                            | Merkmale                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0—read uncommitted<br>(nicht festgeschriebene Daten lesen) | <ul style="list-style-type: none"> <li>• Lesen der Zeile mit oder ohne Schreibsperre zulässig</li> <li>• Es werden keine Lesesperren angewandt.</li> <li>• Keine Garantie, dass gleichzeitige Transaktionen nicht die Zeile ändern oder Änderungen an der Zeile zurücksetzen</li> <li>• Entspricht den Tabellen-Hints NOLOCK und READUNCOMMITTED.</li> <li>• Lässt Dirty Reads, nicht wiederholbare Lesevorgänge und Phantomzeilen zu</li> </ul> |

| Isolationsstufe                                       | Merkmale                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1 - read committed (festgeschriebene Daten lesen)     | <ul style="list-style-type: none"> <li>• Nur Lesen von Zeilen ohne Schreibsperre zulässig.</li> <li>• Lesesperre wird nur für die aktuelle Zeile gesetzt und gehalten, aber freigegeben, wenn der Cursor von der Zeile geht</li> <li>• Keine Garantie, dass Daten nicht während der Transaktion geändert werden</li> <li>• Entspricht dem Tabellen-Hint READCOMMITTED.</li> <li>• Verhindert Dirty Reads</li> <li>• Lässt nicht wiederholbare Lesevorgänge und Phantomzeilen zu</li> </ul> |
| 2 - repeatable read (wiederholbare Lesevorgänge)      | <ul style="list-style-type: none"> <li>• Nur Lesen von Zeilen ohne Schreibsperre zulässig.</li> <li>• Die Lesesperre wird gesetzt, während die einzelnen Zeilen in der Ergebnismenge gelesen werden, und bis zum Ende der Transaktion gehalten.</li> <li>• Entspricht dem Tabellen-Hint REPEATABLE READ.</li> <li>• Verhindert Dirty Reads und nicht wiederholbare Lesevorgänge</li> <li>• Lässt Phantomzeilen zu</li> </ul>                                                               |
| 3 - serializable (serialisierbar)                     | <ul style="list-style-type: none"> <li>• Nur Lesen von Zeile ohne Schreibsperre im Ergebnis zulässig</li> <li>• Lesesperren werden gesetzt, wenn der Cursor geöffnet wird, und gehalten, bis die Transaktion endet.</li> <li>• Entspricht den Tabellen-Hints HOLDLOCK und SERIALIZABLE.</li> <li>• Verhindert Dirty Reads, nicht wiederholbare Lesevorgänge und Phantomzeilen</li> </ul>                                                                                                   |
| snapshot <sup>1</sup>                                 | <ul style="list-style-type: none"> <li>• Es werden keine Lesesperren angewandt.</li> <li>• Lesen aller Zeilen erlaubt.</li> <li>• Ein Datenbank-Snapshot der festgeschriebenen Daten wird angefertigt, wenn die erste Zeile von der Transaktion gelesen oder aktualisiert wird.</li> </ul>                                                                                                                                                                                                 |
| statement-snapshot (Anweisungs-Snapshot) <sup>1</sup> | <ul style="list-style-type: none"> <li>• Es werden keine Lesesperren angewandt.</li> <li>• Lesen aller Zeilen erlaubt.</li> <li>• Ein Datenbank-Snapshot der festgeschriebenen Daten wird angefertigt, wenn die erste Zeile von der Anweisung gelesen wird.</li> </ul>                                                                                                                                                                                                                     |



| Isolationsstufe                                                                   | Merkmale                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| readonly-statement-snapshot (schreibgeschützter Anweisungs-Snapshot) <sup>1</sup> | <ul style="list-style-type: none"> <li>• Es werden keine Lesesperren angewandt.</li> <li>• Lesen aller Zeilen erlaubt.</li> <li>• Ein Datenbank-Snapshot der festgeschriebenen Daten wird angefertigt, wenn die erste Zeile von einer Nur-Lesen-Anweisung gelesen wird.</li> <li>• Verwendet für eine aktualisierbare Anweisung die Isolationsstufe (0, 1, 2 oder 3), die mit der Option "updatable_statement_isolation" festgelegt wurde.</li> </ul> |

<sup>1</sup> Snapshot-Isolation muss für die Datenbank aktiviert sein, indem Sie Option "allow\_snapshot\_isolation" auf "On" setzen.

Die Standardisolationsstufe ist 0, außer für Verbindungen mit Open Client, jConnect und TDS, bei denen die Standardisolationsstufe 1 ist.

Auf Sperren basierende Isolationsstufen verhindern manche oder alle Interferenzen. Stufe 3 bedeutet die höchste Isolationsstufe. Niedrigere Stufen erlauben mehr Inkonsistenzen, haben aber normalerweise eine bessere Performance. Die Standardeinstellung ist Stufe 0 (Read Uncommitted).

Die Snapshot-Isolationsstufen verhindern alle Interferenzen zwischen Lese- und Schreibvorgängen. Es kann jedoch zu Interferenzen zwischen Schreibzugriffen kommen. Einige Inkonsistenzen sind möglich und die Performance in Konfliktsituationen ist die gleiche wie auf Isolationsstufe 0. Die Performance, die nicht im Zusammenhang mit Konflikten steht, ist schlechter, da Zeilenversionen gespeichert und verwendet werden müssen.

Allgemein gilt, dass jede Isolationsstufe durch die benötigten Sperrentypen charakterisiert wird und wie Sperren behandelt werden, die von anderen Transaktionen kommen. Bei Isolationsstufe 0 benötigt der Datenbankserver nur Schreibsperren. Er verwendet diese Sperren, um zu gewährleisten, dass nicht zwei Transaktionen einander widersprechende Änderungen durchführen. So setzt zum Beispiel eine Stufe 0-Transaktion eine Schreibsperre für eine Zeile, bevor sie sie aktualisiert oder löscht, und jede neue Zeile wird mit einer bereits eingerichteten Schreibsperre eingefügt.

Stufe 0-Transaktionen führen keine Überprüfung der Zeilen durch, die sie lesen. Wenn zum Beispiel eine Stufe 0-Transaktion eine Zeile liest, überprüft sie nicht, welche Sperren möglicherweise von anderen Transaktionen für diese Zeile gesetzt wurden. Da keine Überprüfungen notwendig sind, sind Stufe 0-Transaktionen besonders schnell. Diese Geschwindigkeit geht auf Kosten der Konsistenz. Jedes Mal, wenn Transaktionen eine Zeile lesen, für die es eine Schreibsperre durch eine andere Transaktion gibt, laufen Sie Gefahr, nicht festgeschriebene Daten zu erhalten. Bei Stufe 1 überprüfen Transaktionen, ob Schreibsperren vorliegen, bevor sie eine Zeile lesen. Obwohl hierfür ein Arbeitsgang mehr notwendig ist, ist bei diesen Transaktionen sichergestellt, dass sie nur festgeschriebene Daten lesen.

### Hinweise

Alle Isolationsstufen garantieren, dass jede Transaktion vollständig oder überhaupt nicht ausgeführt wird, und dass keine Aktualisierungen verloren gehen.

Die Isolation besteht nur zwischen Transaktionen: Mehrere Cursor innerhalb derselben Transaktion können einander nicht überlagern.

### Siehe auch

- „FROM-Klausel“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „MobiLink-Isolationsstufen“ [*MobiLink - Serveradministration*]
- „So aktivieren Sie die Snapshot-Isolation“ auf Seite 891

## Snapshot-Isolation

Blockierungen und Deadlocks können auftreten, wenn Benutzer gleichzeitig dieselben Daten lesen bzw. schreiben. Die Snapshot-Isolation hat das Ziel, Parallelität und Konsistenz zu verbessern, indem unterschiedliche Datenversionen aufbewahrt werden. Wenn Sie die Snapshot-Isolation in einer Transaktion verwenden, liefert der Datenbankserver auf alle Leseanforderungen eine festgeschriebene Version der Daten. Er tut dies, ohne Lesesperren zu setzen, und vermeidet Interferenzen mit Benutzern, die Daten schreiben.

Ein **Snapshot** ist eine Datenmenge, die in der Datenbank festgeschrieben wurde. Wenn Sie die Snapshot-Isolation verwenden, benutzen alle Abfragen innerhalb einer Transaktion die gleiche Datenmenge. Für Datenbanktabellen werden keine Sperren gesetzt, wodurch es anderen Transaktionen möglich ist, auf die Daten zuzugreifen und diese zu ändern. SQL Anywhere unterstützt drei Snapshot-Isolationsstufen, mit denen Sie steuern können, wann ein Snapshot aufgezeichnet wird:

- **snapshot** Benutzt einen Snapshot festgeschriebener Daten aus der Zeit, als die erste Zeile von der Transaktion gelesen, eingefügt, aktualisiert oder gelöscht wurde.
- **statement-snapshot** Benutzt einen Snapshot der festgeschriebenen Daten von dem Zeitpunkt, zu dem die erste Zeile von der Anweisung gelesen wird. Jede Anweisung innerhalb der Transaktion sieht einen Snapshot von Daten von einem anderen Zeitpunkt.
- **readonly-statement-snapshot** Für Nur-Lesen-Anweisungen wird ein Snapshot festgeschriebener Daten aus der Zeit benutzt, als die erste Zeile gelesen wurde. Jede Nur-Lesen-Anweisung innerhalb der Transaktion sieht einen Snapshot von Daten aus einer anderen Zeit. Benutzen Sie für Insert-, Update- und Delete-Anweisungen die Isolationsstufe, die mit der Option "updateable\_statement\_isolation" festgelegt wurde: 0 (Standardeinstellung), 1, 2 oder 3.

Sie können auch angeben, wann der Snapshot bei einer Transaktion beginnt, indem Sie die Anweisung BEGIN SNAPSHOT verwenden.

Snapshot-Isolation ist häufig sinnvoll, wie z.B. unter folgenden Umständen:

- **Anwendungen mit vielen Lesezugriffen und wenigen Aktualisierungen** Snapshot-Transaktionen setzen Schreibsperren nur für Anweisungen, welche die Datenbank ändern. Wenn eine

Transaktion hauptsächlich Lesezugriffe durchführt, setzt die Snapshot-Transaktion keine Lesesperren, die zu Interferenzen mit den Transaktionen anderer Benutzer führen könnten.

- **Anwendungen mit lang dauernden Transaktionen, während andere Benutzer auf Daten zugreifen müssen** Snapshot-Transaktionen setzen keine Lesesperren, wodurch andere Benutzer die Daten lesen und aktualisieren können, während der Snapshot durchgeführt wird.
- **Anwendungen, die eine konsistente Datenmenge aus der Datenbank lesen müssen** Da ein Snapshot eine festgeschriebene Datenmenge eines bestimmten Zeitpunkts zeigt, können Sie die Snapshot-Isolation benutzen, um konsistente Daten zu erhalten, die sich während der Transaktion nicht verändern, sogar wenn andere Benutzer die Daten ändern, während Ihre Transaktion ausgeführt wird.

Die Snapshot-Isolation wirkt sich nur auf Basistabellen und globale temporäre Tabellen aus, die von allen Benutzern gemeinsam genutzt werden. Ein Lesezugriff auf einen anderen Tabellentyp liefert niemals eine alte Version der Daten und initiiert niemals einen Snapshot. Eine Aktualisierung eines anderen Tabellentyps initiiert nur dann einen Snapshot, wenn die Option "isolation\_level" auf "snapshot" gesetzt wurde und die Aktualisierung eine Transaktion initiiert.

Die folgenden Anweisungen können nicht ausgeführt werden, wenn mit der WITH HOLD-Klausel geöffnete Cursor vorhanden sind, die entweder Anweisungs- oder Transaktions-Snapshots verwenden:

- „ALTER INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ALTER TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „DROP INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „REFRESH MATERIALIZED VIEW-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „REORGANIZE TABLE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „REFRESH TEXT INDEX-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Wenn Cursor mit der WITH HOLD-Klausel geöffnet werden, ist ein Snapshot von allen Zeilen sichtbar, die zur Snapshot-Startzeit festgeschrieben wurden. Ebenfalls sichtbar sind alle Änderungen, die von der aktuellen Verbindung seit dem Start der Transaktion, innerhalb der der Cursor geöffnet war, abgeschlossen wurden.

TRUNCATE TABLE ist nur erlaubt, wenn eine schnelle Kürzung nicht durchgeführt wird, weil in diesem Fall einzelne DELETE-Anweisungen im Transaktionslog aufgezeichnet werden.

Falls eine dieser Anweisungen aus einer Nicht-Snapshot-Transaktion durchgeführt wird, werden bereits laufende Snapshot-Transaktionen, die danach die Tabelle benutzen, einen Fehler erhalten, der angibt, dass sich das Schema geändert hat.

Die Übereinstimmungsprüfung von materialisierten Ansichten vermeidet die Verwendung einer Ansicht, wenn sie nach dem Start des Snapshots für eine Transaktion aktualisiert wurde.

Snapshot-Isolationsstufen werden in allen Programmierschnittstellen unterstützt. Sie können die Isolationsstufe mit der SET OPTION-Anweisung festlegen. Informationen zur Snapshot-Isolation finden Sie unter „[Snapshot-Isolation](#)“ auf Seite 888.

- „[isolation\\_level-Option](#)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- ADO und OLE DB: „[ADO-Transaktionen](#)“ [[SQL Anywhere Server - Programmierung](#)]
- ADO.NET: [SATransaction.IsolationLevel-Eigenschaft](#) [[SQL Anywhere .NET](#)] [[SQL Anywhere Server - Programmierung](#)]

### Zeilenversionen

Wenn Snapshot-Isolation für eine Datenbank aktiviert wird, fügt der Datenbankserver jedes Mal, wenn eine Zeile aktualisiert wird, eine Kopie der Originalzeile zu der Version hinzu und speichert sie in der temporären Datei. Die Zeileneinträge der Originalversion werden gespeichert, bis alle aktiven Snapshot-Transaktionen, die möglicherweise auf die Originalzeilenwerte zugreifen müssen, abgeschlossen sind. Eine Transaktion mit Snapshot-Isolation berücksichtigt nur festgeschriebene Werte. Wenn die Aktualisierung einer Zeile nicht festgeschrieben oder zurückgesetzt wurde, bevor eine Snapshot-Transaktion begann, muss die Snapshot-Transaktion in der Lage sein, auf den Originalzeilenwert zuzugreifen. Dies ermöglicht es Transaktionen, mithilfe der Snapshot-Isolation Daten zu betrachten, ohne Sperren für die zugrunde liegenden Tabellen einzurichten.

Die Datenbankeigenschaft "VersionStorePages" liefert die Anzahl der Seiten in der temporären Datei, die derzeit für die Versionsspeicherung benutzt wird. Um diesen Wert abzurufen, führen Sie die folgende Abfrage aus:

```
SELECT DB_PROPERTY ('VersionStorePages');
```

Alte Zeilenversionseinträge werden entfernt, wenn sie nicht mehr benötigt werden. Alte Versionen von BLOBs werden in der Originaltabelle gespeichert, nicht aber in der temporären Datei, bis sie nicht mehr benötigt werden, und Indexeinträge für alte Zeilenversionen werden im Originalindex gespeichert, bis sie nicht mehr benötigt werden.

Informationen über den Umfang des freien Speicherplatzes in der temporären Datei erhalten Sie, indem Sie die Systemprozedur "sa\_disk\_free\_space" verwenden.

Wenn ein Trigger ausgelöst wird, der Zeilenwerte aktualisiert, werden auch die Originalwerte dieser Zeilen in der temporären Datei gespeichert.

Wenn Sie Ihre Anwendung so einrichten, dass sie kürzere Transaktionen und Snapshots verwendet, wird der Bedarf an temporärem Speicherplatz reduziert.

Wenn Sie Probleme mit dem Anwachsen der temporären Datei haben, können Sie ein GrowTemp-Systemereignis einrichten. Damit legen Sie fest, welche Aktionen beim Erreichen einer bestimmten Größe der temporären Datei durchgeführt werden sollen.

### Einführung in Snapshot-Transaktionen

Snapshot-Transaktionen setzen Schreibsperrungen für Aktualisierungen. Für eine Transaktion oder Anweisung, die einen Snapshot verwendet, werden jedoch niemals Lesesperrungen gesetzt. Daher blockieren Lesevorgänge keine Schreibvorgänge und umgekehrt, aber Schreibvorgänge können andere Schreibvorgänge blockieren, wenn sie versuchen, die gleichen Zeilen zu aktualisieren.

Bei der Snapshot-Isolation beginnt eine Transaktion nicht mit der BEGIN TRANSACTION-Anweisung. Stattdessen beginnt sie mit der ersten READ-, INSERT-, UPDATE oder DELETE-Anweisung innerhalb der Transaktion, abhängig von der Snapshot-Isolationsstufe, die für die Transaktion benutzt wird. Im folgenden Beispiel wird gezeigt, wann eine Transaktion für eine Snapshot-Isolation beginnt:

```
SET OPTION PUBLIC.allow_snapshot_isolation = 'On';
SET TEMPORARY OPTION isolation_level = 'snapshot';
SELECT * FROM Products; --transaction begins and the statement only
 --sees changes that are already committed
INSERT INTO Products
SELECT ID + 30, Name, Description,
'Extra large', Color, 50, UnitPrice, NULL
FROM Products
WHERE Name = 'Tee Shirt';
COMMIT; --transaction ends
```

### Siehe auch

- „BEGIN SNAPSHOT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „TRUNCATE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „sa\_disk\_free\_space-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Systemereignisse“ [*SQL Anywhere Server - Datenbankadministration*]

## So aktivieren Sie die Snapshot-Isolation

Die Snapshot-Isolation wird mit der Option "allow\_snapshot\_isolation" für eine Datenbank aktiviert oder deaktiviert. Wenn die Option auf "On" gesetzt ist, werden die Zeilenversionen in der temporären Datei aufbewahrt und die Verbindungen können jede der Snapshot-Isolationsstufen benutzen. Wenn die Option auf "Off" gesetzt ist, führt jeder Versuch, die Snapshot-Isolation zu verwenden, zu einem Fehler.

Wenn es einer Datenbank ermöglicht wird, die Snapshot-Isolation zu verwenden, kann sich dies auf die Performance auswirken, da Kopien aller geänderten Zeilen aufbewahrt werden müssen, unabhängig von der Anzahl der Transaktionen, welche die Snapshot-Isolation verwenden.

Die folgende Anweisung aktiviert die Snapshot-Isolation für eine Datenbank:

```
SET OPTION PUBLIC.allow_snapshot_isolation = 'On';
```

Die Einstellung der Option "allow\_snapshot\_isolation" kann geändert werden, auch wenn Benutzer mit der Datenbank verbunden sind. Wenn Sie die Einstellung dieser Option von "Off" auf "On" ändern, müssen alle laufenden Transaktionen abgeschlossen werden, bevor neue Transaktionen die Snapshot-Isolation benutzen können. Wenn Sie die Einstellung dieser Option von "On" auf "Off" ändern, müssen alle noch ausstehenden Transaktionen, welche die Snapshot-Isolation benutzen, abgeschlossen werden, bevor der Datenbankserver die Aufbewahrung von Zeilenversionsinformationen beendet.

Sie zeigen die aktuelle Einstellung der Snapshot-Isolation einer Datenbank an, indem Sie den Wert der Datenbankeigenschaft SnapshotIsolationState abfragen:

```
SELECT DB_PROPERTY ('SnapshotIsolationState');
```

Die SnapshotIsolationState-Eigenschaft hat einen der folgenden Werte:

- **On** Snapshot-Isolation ist für die Datenbank aktiviert.
- **Off** Snapshot-Isolation ist für die Datenbank deaktiviert.
- **in\_transition\_to\_on** Snapshot-Isolation ist aktiviert, sobald die aktuellen Transaktionen abgeschlossen sind.
- **in\_transition\_to\_off** Snapshot-Isolation ist deaktiviert, sobald die aktuellen Transaktionen abgeschlossen sind.

Wenn die Snapshot-Isolation für eine Datenbank aktiviert ist, müssen die Zeilenversionen für eine Transaktion aufbewahrt werden, bis die Transaktion festgeschrieben oder zurückgesetzt wird, auch dann, wenn keine Snapshots verwendet werden. Es ist daher am besten, die Option "allow\_snapshot\_isolation" auf "Off" zu setzen, wenn die Snapshot-Isolation nie benutzt wird.

### Siehe auch

- „Cursor-Sensitivität und Isolationsstufen“ [[SQL Anywhere Server - Programmierung](#)]

## Beispiel einer Snapshot-Isolation

Im folgenden Beispiel werden zwei Verbindungen zur SQL Anywhere-Beispieldatenbank benutzt, um zu zeigen, wie die Snapshot-Isolation eingesetzt werden kann, um Konsistenz ohne Blockierungen zu gewährleisten.

### Beispiel: Snapshot-Isolation verwenden

1. Führen Sie den folgenden Befehl aus, um eine Interactive SQL-Verbindung (Connection1) zur SQL Anywhere-Beispieldatenbank herzustellen:

```
dbisql -c "DSN=SQL Anywhere 16 Demo;ConnectionName=Connection1"
```

2. Führen Sie den folgenden Befehl aus, um eine Interactive SQL-Verbindung (Connection2) zur SQL Anywhere-Beispieldatenbank herzustellen:

```
dbisql -c "DSN=SQL Anywhere 16 Demo;ConnectionName=Connection2"
```

3. Führen Sie in "Connection1" die folgende Anweisung aus, um die Isolationsstufe auf 1 zu setzen ("read committed", d.h., festgeschriebene Daten lesen).

```
SET OPTION isolation_level = 1;
```

4. Führen Sie in "Connection1" die folgende Anweisung aus:

```
SELECT * FROM Products;
```

| ID  | Name      | Description | Size   | Color  | Quantity | ... |
|-----|-----------|-------------|--------|--------|----------|-----|
| 300 | Tee Shirt | Tank Top    | Small  | White  | 28       | ... |
| 301 | Tee Shirt | V-neck      | Medium | Orange | 54       | ... |

| ID  | Name         | Description | Size              | Color | Quantity | ... |
|-----|--------------|-------------|-------------------|-------|----------|-----|
| 302 | Tee Shirt    | Crew Neck   | One size fits all | Black | 75       | ... |
| 400 | Baseball Cap | Cotton Cap  | One size fits all | Black | 112      | ... |
| ... | ...          | ...         | ...               | ...   | ...      | ... |

5. Führen Sie in "Connection2" die folgende Anweisung aus:

```
UPDATE Products
SET Name = 'New Tee Shirt'
WHERE ID = 302;
```

6. Führen Sie in "Connection1" erneut die Anweisung SELECT aus:

```
SELECT * FROM Products;
```

Die SELECT-Anweisung wird gesperrt (nur die Schaltfläche **Stopp** kann ausgewählt werden) und kann nicht fortgesetzt werden, weil die UPDATE-Anweisung in "Connection2" weder festgeschrieben noch zurückgesetzt wurde. Die SELECT-Anweisung muss auf den Abschluss der Transaktion in "Connection2" warten, bevor sie fortfahren kann. Damit wird sichergestellt, dass die SELECT-Anweisung keine nicht festgeschriebenen Daten in das Ergebnis einliest.

7. Führen Sie in "Connection2" die folgende Anweisung aus:

```
ROLLBACK;
```

Die Transaktion in "Connection2" wird abgeschlossen, und die SELECT-Anweisung in "Connection1" wird fortgesetzt.

Mit der Snapshot-Isolationsstufe der Anweisung wird die gleiche Parallelität wie mit Isolationsstufe 1 erreicht, jedoch ohne Blockierungen.

8. Führen Sie in "Connection1" die folgende Anweisung aus, um die Snapshot-Isolation zu erlauben:

```
SET OPTION PUBLIC.allow_snapshot_isolation = 'On';
```

9. Führen Sie in "Connection1" die folgende Anweisung aus, um die Isolationsstufe auf "statement-snapshot" zu setzen:

```
SET TEMPORARY OPTION isolation_level = 'statement-snapshot';
```

10. Führen Sie in "Connection1" die folgende Anweisung aus:

```
SELECT * FROM Products;
```

11. Führen Sie in "Connection2" die folgende Anweisung aus:

```
UPDATE Products
SET Name = 'New Tee Shirt'
WHERE ID = 302;
```

12. Führen Sie in "Connection1" erneut die Anweisung SELECT aus:

```
SELECT * FROM Products;
```

Die SELECT-Anweisung wird ohne Sperren ausgeführt, umfasst jedoch nicht die Daten aus der UPDATE-Anweisung, die durch "Connection2" ausgeführt wurde.

13. Schließen Sie in "Connection2" die Transaktion ab, indem Sie die folgende Anweisung ausführen:

```
COMMIT;
```

14. Schließen Sie in "Connection1" die Transaktion ab (die Abfrage der Tabelle "Products") und führen Sie dann die SELECT-Anweisung erneut aus, um die aktualisierten Daten anzuzeigen:

```
COMMIT;
SELECT * FROM Products;
```

| ID  | Name          | Description | Size              | Color  | Quantity | ... |
|-----|---------------|-------------|-------------------|--------|----------|-----|
| 300 | Tee Shirt     | Tank Top    | Small             | White  | 28       | ... |
| 301 | Tee Shirt     | V-neck      | Medium            | Orange | 54       | ... |
| 302 | New Tee Shirt | Crew Neck   | One size fits all | Black  | 75       | ... |
| 400 | Baseball Cap  | Cotton Cap  | One size fits all | Black  | 112      | ... |
| ... | ...           | ...         | ...               | ...    | ...      | ... |

15. Machen Sie die Änderungen der SQL Anywhere-Beispieldatenbank mit folgender Anweisung rückgängig:

```
UPDATE Products
SET Name = 'Tee Shirt'
WHERE id = 302;
COMMIT;
```

### Siehe auch

- [Aufgabe auf Seite 931](#)
- [Aufgabe auf Seite 937](#)
- [Aufgabe auf Seite 943](#)

## Aktualisierungskonflikte und Snapshot-Isolation

Mit der Snapshot-Isolation kann es zu Aktualisierungskonflikten kommen, wenn eine Transaktion eine alte Version einer Zeile sieht und versucht, diese zu aktualisieren oder zu löschen. Wenn das passiert, meldet der Datenbankserver einen Fehler, wenn er den Konflikt erkennt. Bei einer festgeschriebenen Änderung tritt das ein, wenn ein Aktualisieren oder Löschen versucht wird. Bei einer nicht-



festgeschriebenen Änderung wird das Aktualisieren oder Löschen blockiert und der Datenbankserver gibt einen Fehler zurück, wenn die Änderung festgeschrieben wird.

Aktualisierungskonflikte können nicht auftreten, wenn "readonly-statement-snapshot" benutzt wird, da aktualisierbare Anweisungen mit einer Nicht-Snapshot-Isolation ausgeführt werden und jeweils die neueste Version der Datenbank berücksichtigen. Die Isolationsstufe "readonly-statement-snapshot" bietet daher viele Vorteile der Snapshot-Isolation, ohne große Änderungen einer Anwendung zu erfordern, die ursprünglich für die Ausführung auf einer anderen Isolationsstufe entwickelt wurde. Bei der Verwendung der Isolationsstufe readonly-statement-snapshot gilt:

- Für Nur-Lesen-Anweisungen werden niemals Sperren gesetzt.
- Nur-Lesen-Anweisungen berücksichtigen immer einen festgeschriebenen Status der Datenbank.

## Typische Arten von Inkonsistenz

Es gibt drei typische Arten von Inkonsistenz, die bei der Ausführung von gleichzeitigen Transaktionen auftreten können. Diese Liste ist nicht vollständig, da auch andere Inkonsistenzen vorkommen. Diese drei Arten werden im ISO SQL/2008-Standard erwähnt und anhand der Verhaltensweisen definiert, die auf den niedrigeren Isolationsstufen auftreten können.

- **Dirty Reads** Transaktion A ändert eine Zeile; diese Änderung wird aber nicht festgeschrieben oder zurückgesetzt. Transaktion B liest die geänderte Zeile. Transaktion A ändert entweder die Zeile nochmals, bevor sie festgeschrieben wird, oder setzt die Änderung zurück. In beiden Fällen hat Transaktion B die Zeile in einem Status gesehen, der niemals festgeschrieben wurde.
- **Nicht-wiederholbare Lesevorgänge** Transaktion A liest eine Zeile. Transaktion B ändert oder löscht die Zeile und gibt einen COMMIT-Befehl ein. Wenn Transaktion A danach versucht, dieselbe Zeile wieder zu lesen, wird die Zeile geändert oder gelöscht.
- **Phantomzeilen** Transaktion A liest eine Reihe von Zeilen, die einige Bedingungen erfüllen. Transaktion B führt dann einen INSERT- oder UPDATE-Befehl für eine Zeile aus, die vorher die Bedingungen von A nicht erfüllt hat. Transaktion B schreibt diese Änderungen fest. Diese neu festgeschriebenen Zeilen erfüllen nun die Bedingungen der Transaktion A. Falls Transaktion A den Lesevorgang wiederholt, erhält sie die aktualisierte Menge von Zeilen.

### Isolationsstufen und Dirty Reads, nicht-wiederholbare Lesevorgänge und Phantomzeilen

SQL Anywhere erlaubt Dirty Reads, nicht wiederholbare Lesevorgänge und Phantomzeilen, abhängig von der verwendeten Isolationsstufe. In der folgenden Tabelle wird durch ein X angezeigt, dass das Verhalten für die jeweilige Isolationsstufe erlaubt ist.

| Isolationsstufe                                         | Dirty Reads | Nicht-wiederholbare Lesevorgänge | Phantomzeilen |
|---------------------------------------------------------|-------------|----------------------------------|---------------|
| 0-read uncommitted (nicht festgeschriebene Daten lesen) | X           | X                                | X             |

| Isolationsstufe                                 | Dirty Reads    | Nicht-wiederholbare Lesevorgänge | Phantomzeilen  |
|-------------------------------------------------|----------------|----------------------------------|----------------|
| readonly-statement-snapshot                     | X <sup>1</sup> | X <sup>2</sup>                   | X <sup>3</sup> |
| 1-read committed (festgeschriebene Daten lesen) |                | X                                | X              |
| statement-snapshot                              |                | X <sup>2</sup>                   | X <sup>3</sup> |
| 2-repeatable read (wiederholbare Lesevorgänge)  |                |                                  | X              |
| 3-serializable (serialisierbar)                 |                |                                  |                |
| snapshot                                        |                |                                  |                |

<sup>1</sup> Dirty Reads können bei aktualisierbaren Anweisungen innerhalb einer Transaktion auftreten, wenn die Isolationsstufe, die durch die Option "updateable\_statement\_isolation" festgelegt wurde, die Dirty Reads nicht verhindert.

<sup>2</sup> Nicht wiederholbare Lesevorgänge können bei Anweisungen innerhalb einer Transaktion auftreten, wenn die Isolationsstufe, die durch die Option "updateable\_statement\_isolation" festgelegt wurde, die nicht wiederholbaren Lesevorgänge nicht verhindert. Nicht wiederholbare Lesevorgänge können vorkommen, weil jede Anweisung einen neuen Snapshot beginnt, sodass eine Anweisung möglicherweise Änderungen erkennt, die eine andere Anweisung nicht erkennt.

<sup>3</sup> Phantomzeilen können bei Anweisungen innerhalb einer Transaktion auftreten, wenn die Isolationsstufe, die durch die Option "updateable\_statement\_isolation" festgelegt wurde, die Phantomzeilen nicht verhindert. Phantomzeilen können vorkommen, weil jede Anweisung einen neuen Snapshot beginnt, sodass eine Anweisung möglicherweise Änderungen erkennt, die eine andere Anweisung nicht erkennt.

Diese Tabelle illustriert zwei Punkte:

- Jede Isolationsstufe eliminiert eine der drei typischen Inkonsistenzarten.
- Jede Stufe eliminiert die Inkonsistenzarten, die auf allen niedrigeren Stufen eliminiert werden.
- Bei Snapshot-Isolationsstufen für Anweisungen können nicht wiederholbare Lesevorgänge und Phantomzeilen innerhalb einer Transaktion vorkommen, nicht aber innerhalb einer einzelnen Anweisung in einer Transaktion.

Die Isolationsstufen haben verschiedene Namen unter ODBC. Diese Namen basieren auf den Namen der Inkonsistenzarten, die sie verhindern.

#### Siehe auch

- [Der Parameter ValuePtr auf Seite 899](#)
- [„Praktische Einführung: Einführung in Dirty Reads“ auf Seite 928](#)
- [„Praktische Einführung: Einführung in nicht-wiederholbare Lesevorgänge“ auf Seite 932](#)
- [„Praktische Einführung: Einführung in Phantomzeilen“ auf Seite 938](#)

## Cursor-Instabilität

Eine weitere signifikante Inkonsistenzart ist die **Cursor-Instabilität**. Wenn diese Inkonsistenz vorhanden ist, kann eine Transaktion eine Zeile ändern, die durch den Cursor einer anderen Transaktion referenziert wird. Die Cursorstabilität stellt sicher, dass Anwendungen, die Cursor verwenden, keine Inkonsistenzen in die Daten der Datenbank einbringen.

### Beispiel

Transaktion A liest eine Zeile unter Verwendung eines Cursors. Transaktion B ändert diese Zeile und schreibt sie fest. Ohne zu realisieren, dass die Zeile geändert wurde, ändert sie Transaktion A.

### Cursor-Instabilität eliminieren

SQL Anywhere bietet **Cursor-Stabilität** auf den Isolationsstufen 1, 2 und 3. Die Cursorstabilität gewährleistet, dass keine anderen Transaktionen die Informationen ändern können, die in der gegenwärtigen Zeile Ihres Cursors enthalten sind. Bei den Informationen in einer Zeile des Cursors kann es sich um die Kopie von Informationen in einer bestimmten Tabelle oder die Kombination von Daten aus verschiedenen Zeilen von mehreren Tabellen handeln. Wenn Sie innerhalb einer SELECT-Anweisung einen Join oder eine Unterabfrage-Bedingung verwenden, betrifft es meistens mehr als eine Tabelle.

Cursor werden nur dann verwendet, wenn Sie SQL Anywhere über eine andere Anwendung benutzen.

Ein weiterer Gesichtspunkt in Bezug auf Anwendungen, die Cursor verwenden, besteht darin, ob Änderungen der zugrunde liegenden Daten für die Anwendung sichtbar sind. Sie können die Änderungen, die für Anwendungen sichtbar sind, steuern, indem Sie die Sensitivität des Cursors festlegen.

### Siehe auch

- „Gespeicherte Prozeduren, Trigger, Batches und benutzerdefinierte Funktionen“ auf Seite 81
- „Anwendungsentwicklung mit SQL“ [*SQL Anywhere Server - Programmierung*]
- „SQL Anywhere-Cursor“ [*SQL Anywhere Server - Programmierung*]

## So legen Sie die Isolationsstufe fest

Jede Verbindung zur Datenbank hat ihre eigene Isolationsstufe. Zusätzlich kann die Datenbank für jeden Benutzer oder jede benutzererweiterte Rolle eine Standardisolationsstufe speichern. Mit der PUBLIC-Einstellung für die Datenbankoption "isolation\_level" können Sie eine Standardisolationsstufe festlegen.

Sie können auch die Isolationsstufe mit Tabellen-Hints setzen, aber dies ist eine erweiterte Funktion, die zur Einstellung der Isolationsstufe für eine bestimmte Anweisung verwendet wird.

Sie können die Isolationsstufe Ihrer Verbindung und die Ihrer Benutzer-ID zugeordnete Standardstufe ändern, indem Sie die SET OPTION-Anweisung verwenden. Sie können auch die Isolationsstufen für andere Benutzer oder Gruppen verändern.

### Standardisolationsstufe

Wenn Sie mit einer Datenbank verbunden sind, bestimmt der Datenbankserver Ihre anfängliche Isolationsstufe wie folgt:

1. Eine Standardisolationsstufe kann für jeden Benutzer und jede Rolle eingestellt werden. Wenn in der Datenbank eine Stufe für Ihre Benutzer-ID gespeichert ist, dann wird sie vom Datenbankserver verwendet.
2. Falls nicht, überprüft der Datenbankserver die Gruppen, zu denen Sie gehören, bis er eine Stufe findet. Alle Benutzer sind Berechtigungsempfänger der Rolle PUBLIC. Wenn vorher keine anderen Einstellungen gefunden werden, verwendet SQL Anywhere die PUBLIC zugeordnete Stufe.

### Hinweis

Wenn Sie die Snapshot-Isolation verwenden, müssen Sie zunächst die Snapshot-Isolation für die Datenbank aktivieren.

### Beispiel

- **Beispiel 1: Einstellen der Isolationsstufe für den aktuellen Benutzer** Führen Sie die Anweisung SET OPTION aus. Die folgende Anweisung setzt die Isolationsstufe für den aktuellen Benutzer auf die Stufe 3:

```
SET OPTION isolation_level = 3;
```

- **Beispiel 2: Setzen Sie die Isolationsstufe für einen Benutzer oder für die Rolle PUBLIC**

1. Stellen Sie eine Verbindung mit der Datenbank her.
2. Führen Sie die SET OPTION-Anweisung aus und fügen Sie vor "isolation\_level" den Namen des Berechtigungsempfängers und einen Punkt ein. Die folgende Anweisung setzt zum Beispiel die Standardisolation für die PUBLIC-Rolle auf 3.

```
SET OPTION PUBLIC.isolation_level = 3;
```

- **Beispiel 3: Einstellen der Isolationsstufe für die aktuelle Verbindung** Führen Sie die Anweisung SET OPTION mit dem Schlüsselwort TEMPORARY aus. Die folgende Anweisung setzt beispielsweise die Isolationsstufe für die Dauer der aktuellen Verbindung auf die Stufe 3:

```
SET TEMPORARY OPTION isolation_level = 3;
```

### Siehe auch

- „Benutzersicherheit (Rollen und Privilegien)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „SET OPTION-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- WITH Tabellen-Hint-Klausel, FROM-Klausel [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „So aktivieren Sie die Snapshot-Isolation“ auf Seite 891
- „Änderungen an Isolationsstufen während einer Transaktion“ auf Seite 900

## Isolationsstufen in ODBC-fähigen Anwendungen

ODBC-Anwendungen rufen SQLSetConnectAttr mit dem Attribut SQL\_ATTR\_TXN\_ISOLATION auf sowie ValuePtr, das entsprechend der Isolationsstufe festgelegt ist:

**Der Parameter ValuePtr**

| ValuePtr                               | Isolationsstufe             |
|----------------------------------------|-----------------------------|
| SQL_TXN_READ_UNCOMMITTED               | 0                           |
| SQL_TXN_READ_COMMITTED                 | 1                           |
| SQL_TXN_REPEATABLE_READ                | 2                           |
| SQL_TXN_SERIALIZABLE                   | 3                           |
| SA_SQL_TXN_SNAPSHOT                    | snapshot                    |
| SA_SQL_TXN_STATEMENT_SNAPSHOT          | statement-snapshot          |
| SA_SQL_TXN_READONLY_STATEMENT_SNAPSHOT | readonly-statement-snapshot |

**Isolationsstufe über ODBC ändern**

Sie können die Isolationsstufe Ihrer Verbindung via ODBC ändern, indem Sie die Funktion `SQLSetConnectAttr` in der Bibliothek `ODBC32.dll` verwenden.

Die Funktion `SQLSetConnectAttr` verwendet vier Parameter: Den Wert des ODBC-Verbindungs-Handles, die Tatsache, dass Sie die Isolationsstufe einstellen wollen, den Wert, der der Isolationsstufe entspricht, und null. Die der Isolationsstufe entsprechenden Werte werden in der nachstehenden Tabelle angezeigt.

| Zeichenfolge                           | Wert |
|----------------------------------------|------|
| SQL_TXN_ISOLATION                      | 108  |
| SQL_TXN_READ_UNCOMMITTED               | 1    |
| SQL_TXN_READ_COMMITTED                 | 2    |
| SQL_TXN_REPEATABLE_READ                | 4    |
| SQL_TXN_SERIALIZABLE                   | 8    |
| SA_SQL_TXN_SNAPSHOT                    | 32   |
| SA_SQL_TXN_STATEMENT_SNAPSHOT          | 64   |
| SA_SQL_TXN_READONLY_STATEMENT_SNAPSHOT | 128  |

Verwenden Sie die `SET OPTION`-Anweisung nicht, um eine Isolationsstufe von einer ODBC-Anwendung aus zu ändern. Da der ODBC-Treiber die Anweisungen nicht syntaktisch analysiert, wird die Ausführung einer Anweisung in ODBC nicht vom ODBC-Treiber erkannt. Dies könnte zu unerwartetem Sperrverhalten führen.

### Beispiel

Der folgende Funktionsaufruf setzt die Isolationsstufe auf "statement-snapshot":

```
SQLSetConnectAttr (dbc, SA_SQL_ATTR_TXN_ISOLATION, (SQLPOINTER*)
SA_SQL_TXN_STATEMENT_SNAPSHOT, 0);
```

ODBC verwendet die Isolationsfunktion, um die jeweiligen Sperrenoptionen der Datenbank zu unterstützen. So können Sie zum Beispiel im PowerBuilder das Sperrenattribut des Transaktionsobjektes verwenden, um bei der Verbindung mit der Datenbank die Isolationsstufe einzustellen. Beim Sperrenattribut handelt es sich um eine wie folgt eingestellte Zeichenfolge:

```
SQLCA.lock = "RU"
```

Die Sperre ist nur zum Zeitpunkt der CONNECT-Anweisung gültig. Änderungen am Sperrenattribut nach CONNECT haben keine Auswirkung auf die Verbindung.

## Änderungen an Isolationsstufen während einer Transaktion

Verschiedene Isolationsstufen können für verschiedene Teile einer einzelnen Transaktion geeignet sein. SQL Anywhere gibt Ihnen die Möglichkeit, die Isolationsstufe Ihrer Datenbank während einer Transaktion zu ändern.

Wenn Sie die Option "isolation\_level" während einer Transaktion ändern, wirkt sich die neue Einstellung nur auf die folgenden Elemente aus:

- Auf Cursor, die nach der Änderung geöffnet werden.
- Auf Anweisungen, die nach der Änderung ausgeführt werden.

Es kann sinnvoll sein, die Isolationsstufe während einer Transaktion zu ändern, um zu steuern, wie viele Sperren Ihre Transaktion setzt. Es kann auch sein, dass eine Transaktion eine umfangreiche Tabelle lesen muss, sich im Detail aber nur mit wenigen Zeilen befasst. Wenn sich eine Inkonsistenz nicht ernsthaft auf Ihre Transaktion auswirkt, sollten Sie eine niedrige Isolationsstufe einstellen, während Sie eine große Tabelle lesen, damit Sie nicht die Arbeit von anderen Benutzern verzögern.

Außerdem empfiehlt es sich in gewissen Fällen, die Isolationsstufe während der Transaktion zu ändern, wenn zum Beispiel nur eine Tabelle oder Gruppe von Tabellen einen serialisierten Zugriff erfordert.

Ein Beispiel, bei dem die Isolationsstufe während einer Transaktion geändert wird, finden Sie unter [„Praktische Einführung: Einführung in Phantomzeilen“ auf Seite 938](#).

#### Hinweis

Sie können auch die Isolationsstufe (nur Stufen 0 bis 3) mit Tabellen-Hints festlegen, aber dies ist eine erweiterte Funktion, die nur bei Bedarf verwendet werden sollte. Weitere Hinweise finden Sie im Abschnitt zum WITH *table-hint* unter [„FROM-Klausel“ \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#).

### Isolationsstufen ändern, wenn die Snapshot-Isolation benutzt wird

Wenn Sie die Snapshot-Isolation verwenden, können Sie die Isolationsstufe innerhalb einer Transaktion ändern. Sie können dies durchführen, indem Sie die Einstellung der Option "isolation\_level" ändern oder

indem Sie Tabellen-Hints benutzen, die sich auf die Isolationsstufe in einer Abfrage auswirken. Sie können jederzeit die Stufen statement-snapshot, readonly-statement-snapshot sowie die Isolationsstufen 0 bis 3 verwenden. Sie können die Snapshot-Isolationsstufe jedoch nicht in einer Transaktion benutzen, wenn sie auf einer anderen Isolationsstufe als "Snapshot" begann. Eine Transaktion wird durch eine Aktualisierung initialisiert und dauert bis zum nächsten COMMIT oder ROLLBACK an. Falls die erste Aktualisierung auf einer anderen Isolationsstufe als "Snapshot" stattfindet, liefert jede Anweisung, die versucht, die Snapshot-Isolationsstufe zu benutzen, bevor die Transaktion festgeschrieben oder zurückgesetzt wird, den Fehler "-1065 (SQLE\_NON\_SNAPSHOT\_TRANSACTION)". Beispiel:

```
SET OPTION PUBLIC.allow_snapshot_isolation = 'On';

BEGIN TRANSACTION
 SET OPTION isolation_level = 3;
 INSERT INTO Departments
 (DepartmentID, DepartmentName, DepartmentHeadID)
 VALUES(700, 'Foreign Sales', 129);
 SET TEMPORARY OPTION isolation_level = 'snapshot';
 SELECT * FROM Departments;
```

## Isolationsstufe anzeigen

Sie können die Funktion CONNECTION\_PROPERTY verwenden, um die Isolationsstufe für die aktuelle Verbindung anzuzeigen.

### Voraussetzungen

Sie müssen mit einer Datenbank verbunden sein.

### Aufgabe

- Führen Sie die folgende Anweisung aus:

```
SELECT CONNECTION_PROPERTY('isolation_level');
```

### Ergebnisse

Die Isolationsstufe für die aktuelle Verbindung wird zurückgegeben.

### Siehe auch

- „CONNECTION\_PROPERTY-Funktion [System]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- isolation\_level-Verbindungseigenschaft [[SQL Anywhere Server - Datenbankadministration](#)]
- „isolation\_level-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

## Transaktion blockieren und Deadlock

Wenn eine Transaktion ausgeführt wird, setzt der Datenbankserver Sperren auf Zeilen, um zu verhindern, dass diese Zeilen von anderen Transaktionen bearbeitet werden. **Sperren** steuern die Anzahl und Arten der zulässigen Überlagerungen.

SQL Anywhere verwendet die **Transaktionsblockierung**, damit Transaktionen gleichzeitig ohne Interferenzen oder mit nur geringen Interferenzen ausgeführt werden können. Jede Transaktion kann eine Sperre setzen, die verhindert, dass andere gleichzeitige Transaktionen eine bestimmte Zeile verändern oder auch auf sie zugreifen können. Dieses Schema für die Transaktionsblockierung verhindert einige Arten von Überschneidungen in jedem Fall. Beispiel: Eine Transaktion, die eine bestimmte Zeile einer Tabelle aktualisiert, setzt eine Sperre für diese Zeile, um zu gewährleisten, dass keine andere Transaktion die gleiche Zeile zur gleichen Zeit aktualisiert oder löscht.

### Siehe auch

- „blocking\_others\_timeout-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „blocking\_timeout-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „blocking-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

## Transaktion blockieren

Wenn eine Transaktion versucht, einen Vorgang auszuführen, aber durch eine Sperre einer anderen Transaktion daran gehindert wird, entsteht ein Konflikt, und die Transaktion, die den Vorgang ausführen möchte, kann nicht weiter abgearbeitet werden.

Transaktionen können einen Zustand erreichen, in dem die Weiterverarbeitung nicht möglich ist.

### Siehe auch

- „Deadlocks“ auf Seite 903
- „blocking\_others\_timeout-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „blocking\_timeout-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „blocking-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

## Die Blocking-Option

Wenn zwei Transaktionen eine Lesesperre für die gleiche Zeile gesetzt haben, hängt das Verhalten bei dem Versuch einer der Transaktionen, diese Zeile zu ändern, von der Einstellung der Datenbankoption "blocking" ab. Um diese Zeile zu ändern, muss die Transaktion die andere blockieren. Das kann sie aber nicht, solange die andere Transaktion die Zeile blockiert hat.

- Wenn die Option "blocking" auf "On" gesetzt ist (die Standardeinstellung), dann wartet die Transaktion, die einen Schreibvorgang vornehmen will, bis die andere Transaktion ihre Lesesperre freigegeben hat. Erst danach erfolgt der Schreibvorgang.
- Wenn die Option "blocking" auf "Off" gesetzt ist, erhält die Anweisung, die einen Schreibvorgang versucht, eine Fehlermeldung.

Wenn die Option "blocking" auf "Off" gesetzt ist, wird die Anweisung beendet, statt angehalten, und partiell durchgeführte Änderungen werden zurückgesetzt. In diesem Fall versuchen Sie, die Transaktion später noch einmal auszuführen.

Blockierungen sind häufiger bei höheren Isolationsstufen festzustellen, da dort eine größere Anzahl von Sperren und Überprüfungen durchgeführt wird. Höhere Isolationsstufen ergeben gewöhnlich weniger



Parallelität. Wie viel weniger dies ist, hängt von der jeweiligen Art der gleichzeitig ablaufenden Transaktionen ab.

### Siehe auch

- „blocking-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „blocking\_others\_timeout-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „blocking\_timeout-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]

## Deadlocks

Das Blockieren von Transaktionen kann zu einem **Deadlock** führen, wobei sich mehrere Transaktionen in einem Zustand befinden, in dem keine mehr weiter verarbeitet werden kann.

### Gründe für einen Deadlock

Ein Deadlock kann aus zwei Gründen entstehen:

- **Zyklischer Blockierkonflikt** Transaktion A wird durch Transaktion B blockiert, und Transaktion B wird durch Transaktion A blockiert. Mehr Zeit wird hier das Problem nicht lösen. Eine der Transaktionen muss abgebrochen werden, damit die andere fortfahren kann. Die gleiche Situation kann auch entstehen, wenn mehr als zwei Transaktionen in einem Zyklus blockiert sind.

Um einen transaktionalen Deadlock zu eliminieren, wählt SQL Anywhere eine am Deadlock beteiligte Verbindung aus, setzt die Änderungen für die auf dieser Verbindung aktiven Transaktion zurück und meldet einen Fehler. SQL Anywhere wählt die zurückzusetzende Verbindung unter Verwendung eines internen heuristischen Werts aus, der die Verbindung mit der kürzesten verbliebenen Wartezeit bevorzugt, wie durch die blocking\_timeout-Option festgelegt. Wenn alle Verbindungen auf unbegrenzten Wartezustand gesetzt sind, wird die Verbindung, die bewirkte, dass der Datenbankserver den Deadlock erkannte, als Opferverbindung ausgewählt.

- **Alle Worker-Threads sind blockiert** Wenn eine Transaktion blockiert wird, erfolgt keine Freigabe Ihres Worker-Threads. Beispiel: Wird ein Datenbankserver mit drei Worker-Threads konfiguriert und die Transaktionen A, B und C werden durch die Transaktion D blockiert, die derzeit keine Anforderungen ausführt, dann ist eine Deadlock-Situation entstanden, da es keine verfügbaren Worker-Threads mehr gibt. Diese Situation wird als Thread-Deadlock bezeichnet.

Angenommen der Datenbankserver hat  $n$  Worker-Threads. Ein Thread-Deadlock tritt auf, wenn  $n-1$  Worker-Threads blockiert sind und der letzte Worker-Thread im Begriff ist, zu blockieren. Der Kernel des Datenbankservers kann nicht zulassen, dass dieser Worker-Thread blockiert, da dies dazu führen würde, dass alle Worker-Threads blockiert wären und der Datenbankserver hängen würde. Stattdessen beendet der Datenbankserver die Aufgabe, die dabei ist, den letzten Worker-Thread zu blockieren, setzt die Änderungen für die auf dieser Verbindung aktiven Transaktion zurück und gibt einen Fehler zurück (SQLCODE -307, SQLSTATE 40W06).

Bei Datenbankservern mit Dutzenden oder Hunderten von Verbindungen können Thread-Deadlocks in Fällen vorkommen, in denen viele lange laufende Anforderungen auftreten. Dies liegt an der Größe der Datenbank oder weil es zu Blockierungen kommt. In diesem Fall kann die Erhöhung der Multiprogramming-Stufe des Datenbankservers eine geeignete Lösung sein. Das Anwendungsdesign kann aufgrund von übermäßigen oder unbeabsichtigten Konflikten auch Thread-Deadlocks bewirken.

In diesen Fällen kann die Skalierung der Anwendung auf größere Datenmengen das Problem verschlimmern und eine Erhöhung der Multiprogramming-Stufe des Datenbankservers löst das Problem möglicherweise nicht.

Die Anzahl der Datenbank-Threads, die der Server benutzt, hängt von den jeweiligen Datenbankeinstellungen ab.

### Siehe auch

- „log\_deadlocks-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „blocking\_timeout-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sa\_report\_deadlocks-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Threading-Verhalten“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Alle Threads sind blockiert“ [[Fehlermeldungen](#)]
- „Datenbankserverkonfiguration der Multiprogramming-Stufe“ [[SQL Anywhere Server - Datenbankadministration](#)]

### Wie Sie ermitteln, was in einem Deadlock blockiert ist

Sie können die Systemprozedur `sa_conn_info` verwenden, um zu ermitteln, welche Verbindungen in einem Deadlock blockiert sind. Diese Prozedur gibt eine Ergebnismenge zurück, die aus einer Zeile für jede Verbindung besteht. Eine Spalte der Ergebnismenge listet auf, warum die Verbindung blockiert und mit welcher Verbindung diese Blockierung eingetreten ist. Die Ergebnismenge gibt an, ob eine Verbindung blockiert ist, und welche Verbindung sie blockiert.

Sie können auch ein Deadlock-Ereignis verwenden, um Aktionen durchzuführen, wenn der Deadlock eintritt. Der Event-Handler kann mit der Prozedur `sa_report_deadlock` Informationen über die Bedingungen erhalten, die zum Deadlock geführt haben. Um weitere Details über den Deadlock vom Datenbankserver abzurufen, verwenden Sie die Option `log_deadlocks` und die Funktion `RememberLastStatement`.

Das folgende Beispiel zeigt, wie Sie ein Tabellen- und Systemereignis einrichten, das verwendet werden kann, um Informationen über Deadlocks bei ihrem Auftreten zu beziehen. Wenn bei Ihrer Anwendung häufig Deadlocks auftreten, können Sie die Anwendungsprofilerstellung verwenden, um die Ursache von Deadlocks zu diagnostizieren.

### Beispiel: Einleiten von Aktionen bei Deadlocks

1. Erstellen Sie eine Tabelle, um die von der Systemprozedur "sa\_report\_deadlocks" zurückgegebenen Daten zu speichern:

```
CREATE TABLE DeadlockDetails(
 deadlockId INT PRIMARY KEY DEFAULT AUTOINCREMENT,
 snapshotId BIGINT,
 snapshotAt TIMESTAMP,
 waiter INTEGER,
 who VARCHAR(128),
 what LONG VARCHAR,
 object_id UNSIGNED BIGINT,
 record_id BIGINT,
 owner INTEGER,
 is_victim BIT,
 rollback_operation_count UNSIGNED INTEGER);
```

- Erstellen Sie ein Ereignis, das ausgelöst wird, wenn ein Deadlock auftritt.

Dieses Ereignis kopiert die Ergebnisse der Systemprozedur "sa\_report\_deadlocks" in eine Tabelle und benachrichtigt den Administrator über den Deadlock.

```
CREATE EVENT DeadlockNotification
TYPE Deadlock
HANDLER
BEGIN
 INSERT INTO DeadlockDetails WITH AUTO NAME
 SELECT snapshotId, snapshotAt, waiter, who, what, object_id, record_id,
 owner, is_victim, rollback_operation_count
 FROM sa_report_deadlocks ();
COMMIT;
CALL xp_startmail (mail_user = 'George Smith',
 mail_password = 'mypwd');
CALL xp_sendmail(recipient='DBAdmin',
 subject='Deadlock details added to the
DeadlockDetails table.');
CALL xp_stopmail ();
END;
```

- Setzen Sie die Option "log\_deadlocks" auf "On":

```
SET OPTION PUBLIC.log_deadlocks = 'On';
```

- Aktivieren Sie die Protokollierung der zuletzt ausgeführten Anweisung:

```
CALL sa_server_option('RememberLastStatement', 'YES');
```

#### Siehe auch

- „log\_deadlocks-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „sa\_report\_deadlocks-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „sa\_server\_option-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „sa\_conn\_info-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE EVENT-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Praktische Einführung: Deadlocks diagnostizieren“ auf Seite 265

## Deadlocks aus Sybase Central anzeigen

Sie können ein Diagramm mit Deadlocks anzeigen lassen, die in der Datenbank aufgetreten sind, seitdem die Option "log\_deadlocks" auf "On" gesetzt wurde.

### Voraussetzungen

Sie müssen die Systemprivilegien MONITOR und SET ANY SYSTEM OPTION haben.

### Aufgabe

- Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
- Wählen Sie in Sybase Central im linken Fensterausschnitt die Datenbank aus und klicken Sie auf **Datei » Optionen**.

3. Aktivieren Sie die log\_deadlocks-Option:
  - a. Klicken Sie in der Liste **Optionen** auf **log\_deadlocks**.
  - b. Klicken Sie im Feld **Wert** auf **On**.
  - c. Klicken Sie auf **Jetzt permanent setzen**.
  - d. Klicken Sie auf **Schließen**.
4. Im rechten Fensterausschnitt klicken Sie auf die Registerkarte **Deadlocks**.

### Ergebnisse

Ein Deadlock-Diagramm mit den aufgetretenen Deadlocks wird angezeigt. Jeder Knoten im Deadlock-Diagramm steht für eine Verbindung und zeigt folgende Details an: die Verbindung, die am Deadlock beteiligt war, den Benutzernamen und die SQL-Anweisung, deren Ausführung die Verbindung versuchte, als der Deadlock auftrat. Es gibt zwei Arten von Deadlocks: Verbindungs-Deadlocks und Thread-Deadlocks. Verbindungs-Deadlocks weisen einen Zirkelbezug bei den Knoten auf. Ein Thread-Deadlock weist keinen Zirkelbezug bei den Knoten auf und die Anzahl der Knoten entspricht dem Thread-Limit der Datenbank plus eins.

### Nächste Schritte

Sie können die Anwendungsprofilerstellung verwenden, um die Ursache von Deadlocks zu diagnostizieren.

### Siehe auch

- „log\_deadlocks-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Transaktion blockieren und Deadlock“ auf Seite 901
- „Wie Sie ermitteln, was in einem Deadlock blockiert ist“ auf Seite 904

## Funktionsweise von Sperren

Eine Sperre ist ein Kontrollmechanismus im Mehrbenutzerbetrieb, der die Integrität der Daten während der parallelen Ausführung mehrerer Transaktionen schützt. SQL Anywhere wendet Sperren automatisch an, um zu verhindern, dass zwei Verbindungen Daten gleichzeitig ändern oder dass Verbindungen Daten lesen, die aktuell geändert werden. Sperren verbessern die Konsistenz von Abfrageergebnissen, indem sie Informationen, die gerade aktualisiert werden, schützen.

Der Datenbankserver setzt diese Sperren automatisch und bedarf keiner expliziten Instruktion. Er hält alle durch eine Transaktion gesetzten Sperren bis zum Abschluss der Transaktion, beispielsweise durch eine COMMIT-Anweisung oder durch eine ROLLBACK-Anweisung, aufrecht. Hierzu gibt es eine Ausnahme.

Die Transaktion, die auf die Zeile zugreift, hält auch die Sperre. Abhängig vom Typ der Sperre haben andere Transaktionen begrenzten Zugriff auf die gesperrte Zeile oder gar keinen.

### Siehe auch

- „Sperrendauer“ auf Seite 922

## Sperrentypen

Zur Gewährleistung der Datenbankkonsistenz und zur Unterstützung der entsprechenden Isolationsstufen zwischen Transaktionen verwendet SQL Anywhere folgende Arten von Sperren:

- **Schemasperren** Diese Sperren kontrollieren die Möglichkeit, Schemaänderungen durchzuführen. Beispielsweise kann eine Transaktion, welche die Struktur einer Tabelle durch Einfügen einer neuen Spalte ändert, diese Tabelle sperren, sodass andere Transaktionen durch die Schemaänderung nicht beeinflusst werden. In einem solchen Fall ist es besonders wichtig, dass andere Transaktionen am Zugriff gehindert werden, damit keine Fehler auftreten.
- **Zeilen Sperren** Diese Sperren werden benutzt, um die Konsistenz zwischen parallelen Transaktionen auf Zeilenebene zu gewährleisten. Eine Transaktion kann beispielsweise eine bestimmte Zeile sperren, um zu verhindern, dass die Zeile durch eine andere Transaktion geändert wird, und eine Transaktion muss eine Schreibsperre auf eine Zeile setzen, wenn sie beabsichtigt, die Zeile zu ändern. Um die Parallelität zu maximieren, können die Zeilenbereiche mit und ohne Schlüssel separat gesperrt werden. Das Aktualisieren von Nicht-Schlüsselspalten einer Zeile beeinträchtigt nicht das Einfügen und Löschen von Fremdzeilen, die auf diese Zeile verweisen.
- **Tabellensperren** Diese Sperren werden benutzt, um die Konsistenz zwischen parallelen Transaktionen auf Tabellenebene zu gewährleisten. Wenn beispielsweise `LOCK TABLE ... IN EXCLUSIVE MODE` oder `REFRESH MATERIALIZED VIEW ... WITH EXCLUSIVE MODE` ausgeführt wird, benötigen Sie eine exklusive Tabellensperre.
- **Positionssperren** Diese Sperren werden benutzt, um die Konsistenz innerhalb eines sequenziellen oder indexierten Table-Scans zu gewährleisten. Transaktionen suchen normalerweise Zeilen sequenziell oder in der Reihenfolge ab, die von einem Index vorgeschrieben wird. In beiden Fällen kann eine Sperre auf die Suchposition gesetzt werden. Beispiel: Wenn in einem Index eine Sperre gesetzt wird, kann damit verhindert werden, dass eine andere Transaktion eine Zeile mit einem bestimmten Wert oder Wertbereich einfügt.

Schemasperren bieten einen Mechanismus, der verhindert, dass sich Schemaänderungen unbeabsichtigt auf aktive Transaktionen auswirken. Zeilen Sperren, Tabellensperren und Positionssperren haben jeweils separate Zwecke, aber sie beeinflussen sich gegenseitig. Jeder Sperrentyp verhindert bestimmte Inkonsistenzen. Je nach der gewählten Isolationsstufe verwendet der Datenbankserver einige oder alle Sperrentypen, um das erforderliche Maß an Konsistenz aufrecht zu halten.

## Sperrendauer

Die verschiedenen Sperrenklassen können für verschiedene Zeiträume aufrechterhalten werden:

- **Position** Positionssperren sind kurzfristige Sperren, beispielsweise Lesesperren für bestimmte Zeilen, mit denen die Cursorstabilität auf Isolationsstufe 1 gewährleistet wird.
- **Transaktion** Transaktionssperren sind Zeilen-, Tabellen- oder Positionssperren, die bis zum Ende einer Transaktion bestehen bleiben.
- **Verbindung** Verbindungssperren sind Schemasperren, die über das Ende einer Transaktion hinaus bestehen bleiben, wie etwa Schemasperren, die erstellt werden, wenn Cursors des Typs `WITH HOLD` benutzt werden.

## So erhalten Sie Informationen über Sperren

Zur Diagnose eines Sperrenproblems in der Datenbank kann es hilfreich sein, den Inhalt der gesperrten Zeilen zu kennen. Sie können die aktuell in einer Datenbank aktiven Sperren anzeigen, indem Sie entweder die sa\_locks-Systemprozedur oder die Registerkarte **Sperren** in Sybase Central benutzen. Beide Methoden liefern die benötigten Informationen, darunter die Verbindung, die die Sperre hält, die Sperrendauer und den Sperrentyp.

### Hinweis

Aufgrund der flüchtigen Natur von Sperren in der Datenbank sind die Zeilen, die in Sybase Central sichtbar sind oder die von der sa\_locks-Systemprozedur zurückgegeben werden, beim Abschluss einer Abfrage möglicherweise nicht mehr vorhanden.

### Sperren mit Sybase Central anzeigen

Sie können Sperren in Sybase Central anzeigen. Markieren Sie eine Datenbank im linken Fensterausschnitt und klicken Sie auf die Registerkarte **Sperren** im rechten Fensterausschnitt. Diese Registerkarte zeigt für jede Sperre die Verbindungs-ID, die Benutzer-ID, den Tabellennamen, den Sperrentyp und den Sperrennamen an.

### Sperren mit der sa\_locks-Systemprozedur anzeigen

Die Ergebnismenge der sa\_locks-Systemprozedur enthält die row\_identifier-Spalte, in der Sie feststellen können, auf welche Zeile in einer Tabelle sich die Sperre bezieht. Um die Werte zu ermitteln, die in der gesperrten Zeile gespeichert sind, können Sie die Ergebnisse der sa\_locks-Systemprozedur mit einer bestimmten Tabelle verknüpfen, indem Sie die RowId der Tabelle im Join-Prädikat benutzen. Beispiel:

```
SELECT S.conn_id, S.user_id, S.lock_class, S.lock_type, E.*
FROM sa_locks() S JOIN Employees E WITH(NOLOCK)
ON ROWID(E) = S.row_identifier
WHERE S.table_name = 'Employees';
```

### Hinweis

Möglicherweise ist es nicht erforderlich, den Tabellen-Hint NOLOCK anzugeben. Falls die Abfrage jedoch auf einer anderen Isolationsstufe als 0 benutzt wird, könnte die Abfrage blockiert sein, bis die Sperren freigegeben werden, wodurch die Effizienz dieser Prüfmethode etwas eingeschränkt wird.

### Siehe auch

- „sa\_locks-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „FROM-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „ROWID-Funktion [Verschiedene]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Schemasperren

Schemasperren serialisieren Änderungen eines Datenbankschemas und stellen sicher, dass Transaktionen, die eine Tabelle verwenden, nicht durch Schemaänderungen beeinträchtigt werden, die durch andere Verbindungen initiiert werden. Eine gemeinsam genutzte Schemasperre könnte beispielsweise verhindern,

dass eine ALTER TABLE-Anweisung eine Spalte aus einer Tabelle löscht, wenn diese Tabelle von einem geöffneten Cursor über eine andere Verbindung gelesen wird.

Es gibt zwei Klassen von Schemasperren: gemeinsame oder exklusive.

### **Gemeinsame Sperren**

Eine gemeinsame Schemasperre wird gesetzt, wenn sich eine Transaktion direkt oder indirekt auf eine Tabelle in der Datenbank bezieht. Gemeinsame Schemasperren verursachen keine gegenseitigen Konflikte. Beliebige viele Transaktionen können gleichzeitig gemeinsame Schemasperren für die gleiche Tabelle setzen. Die gemeinsame Schemasperre bleibt bestehen, bis die Transaktion durch COMMIT oder ROLLBACK abgeschlossen wird.

Eine Verbindung, die eine gemeinsam genutzte Schemasperre hält, kann Tabellendaten ändern, sofern die Änderung nicht mit anderen Verbindungen in Konflikt steht. Das Tabellenschema wird im gemeinsam genutzten (Lese-)Modus gesperrt.

### **Exklusivsperrern**

Eine exklusive Schemasperre wird gesetzt, wenn das Schema einer Tabelle geändert wird, normalerweise durch die Verwendung einer DDL-Anweisung. Die ALTER TABLE-Anweisung ist ein Beispiel einer DDL-Anweisung, die eine exklusive Sperre für eine Tabelle setzt, bevor sie sie ändert. Eine exklusive Schemasperre für eine Tabelle kann immer nur von einer Verbindung gesetzt werden – alle anderen Versuche, das Schema einer Tabelle zu sperren (gemeinsam oder exklusiv) werden geblockt oder führen zu einer Fehlermeldung. Eine Verbindung auf Isolationsstufe 0, der Isolationsstufe mit den geringsten Einschränkungen, kann keine Zeilen aus einer Tabelle lesen, deren Schema im exklusiven Modus gesperrt wurde.

Nur die Verbindung, die die exklusive Tabellenschemasperre hält, kann die Tabellendaten ändern. Das Tabellenschema wird für die exklusive Verwendung durch eine einzelne Verbindung gesperrt.

## **Zeilensperren**

Zeilensperren verhindern Aktualisierungsverluste und andere Arten von Transaktionsinkonsistenzen. Sie gewährleisten, dass eine Zeile, die durch eine Transaktion geändert wird, nicht durch eine andere Transaktion geändert werden kann, bevor die erste Transaktion abgeschlossen ist, entweder durch eine implizite oder explizite COMMIT-Anweisung oder durch Abbrechen der Änderungen durch eine ROLLBACK-Anweisung.

Es gibt drei Klassen von Zeilensperren: Lesesperren (gemeinsam), Schreibsperren (exklusiv) und Absichtssperren. Der Datenbankserver setzt diese Sperren automatisch für jede Transaktion.

### **Lesesperren**

Wenn eine Transaktion eine Zeile liest, legt die Isolationsstufe der Transaktion fest, ob eine Lesesperre gesetzt wird. Sobald für eine Zeile eine Lesesperre vorliegt, kann keine Transaktion eine Schreibsperre dafür setzen. Das Setzen einer Lesesperre gewährleistet, dass eine Zeile nicht von einer anderen Transaktion geändert oder gelöscht wird, während sie gelesen wird. Beliebige viele Transaktionen können gleichzeitig Lesesperren für eine beliebige Zeile setzen, weshalb Lesesperren manchmal auch als gemeinsame Sperren oder Nicht-Exklusivsperrern bezeichnet werden.

Lesesperren können für verschiedene Zeitspannen aufrechterhalten werden. Auf den Isolationsstufen 2 und 3 werden alle von einer Transaktion gesetzten Lesesperren aufrechterhalten, bis die Transaktion durch COMMIT oder ROLLBACK festgeschrieben wird. Diese Lesesperren werden als langfristige Lesesperren bezeichnet.

Für Transaktionen, die auf Isolationsstufe 1 ausgeführt werden, setzt der Datenbankserver eine kurzfristige Lesesperre für die Zeile, in der ein Cursor positioniert ist. Wenn die Anwendung den Cursor durchläuft, wird die kurzfristige Lesesperre für die zuvor aktive Zeile freigegeben und eine neue kurzfristige Lesesperre wird für die nächste Zeile gesetzt. Diese Technik wird als **Cursorstabilität** bezeichnet. Da die Anwendung eine Lesesperre für die aktuelle Zeile aufrechterhält, können andere Transaktionen die Zeile nicht ändern, bis die Anwendung die Zeile verlässt. Es kann mehr als eine Sperre gesetzt werden, wenn sich der Cursor auf einer Abfrage befindet, die mehrere Tabellen umfasst. Kurzfristige Lesesperren werden nur gesetzt, wenn die Position innerhalb eines Cursors über mehrere Abfragen aufrechterhalten werden muss (normalerweise würde es sich dabei um FETCH-Anweisungen der Anwendung handeln). Kurzfristige Lesesperren werden beispielsweise nicht gesetzt, wenn eine SELECT COUNT(\*)-Abfrage verarbeitet wird, da ein Cursor, der für diese Anweisung geöffnet wird, niemals auf eine bestimmte Zeile der Basistabelle gesetzt wird. In einem solchen Fall braucht der Datenbankserver nur die Semantik der festgeschriebenen Lesevorgänge zu garantieren, das heißt, dass die von der Anweisung verarbeiteten Zeilen durch andere Transaktionen festgeschrieben wurden.

Transaktionen auf Isolationsstufe 0 ("read uncommitted") setzen keine langfristigen oder kurzfristigen Lesesperren, wodurch es auch nicht zu Konflikten mit anderen Transaktionen kommt (außer bei exklusiven Schemasperren). Transaktionen auf Isolationsstufe 0 können jedoch nicht festgeschriebene Änderungen verarbeiten, die durch andere parallele Transaktionen durchgeführt wurden. Sie können die Verarbeitung nicht festgeschriebener Änderungen durch die Snapshot-Isolation verhindern.

### Schreibsperren

Eine Transaktion setzt eine Schreibsperre, wenn eine Zeile eingefügt, aktualisiert oder gelöscht wird. Dieses Verhalten trifft für Transaktionen auf allen Isolationsstufen zu, einschließlich der Isolationsstufen 0 und der Snapshot-Isolationsstufe. Keine andere Transaktion kann eine Schreib-, Absichts- oder Lesesperre für die gleiche Zeile setzen, wenn eine Schreibsperre gesetzt ist. Schreibsperren werden auch als exklusive Sperren bezeichnet, da zur gleichen Zeit nur eine einzige Transaktion über eine Exklusivsperre für eine Zeile verfügen kann. Keine Transaktion kann eine Schreibsperre setzen, während eine andere Transaktion eine beliebige Sperre für die gleiche Zeile hat. Gleichermaßen gilt: Sobald eine Transaktion eine Schreibsperre setzt, werden Anforderungen anderer Transaktionen, die Zeile zu sperren, abgewiesen.

### Absichtssperren

Absichtssperren, die auch als Sperren für beabsichtigte Aktualisierungen bezeichnet werden, geben an, dass eine Absicht zum Ändern einer bestimmten Zeile besteht. Absichtssperren werden in folgenden Situationen gesetzt:

- Eine Transaktion führt eine FETCH FOR UPDATE-Anweisung aus.
- Eine Transaktion führt eine SELECT ... FOR UPDATE BY LOCK-Anweisung aus.
- Eine Transaktion benutzt SQL\_CONCUR\_LOCK als Parallelitätsbasis in einer ODBC-Anwendung (durch den Parameter SQL\_ATTR\_CONCURRENCY des ODBC API-Aufrufs SQLSetStmtAttr).



- Eine Transaktion führt eine SELECT...FROM T WITH (UPDLOCK)-Anweisung aus.

Absichtssperren verursachen keine Konflikte mit Lesesperren, sodass das Setzen einer Absichtssperre andere Transaktionen nicht davon abhält, die gleiche Zeile zu lesen. Absichtssperren verhindern jedoch, dass andere Transaktionen eine Absichts- oder Schreibsperre für dieselbe Zeile setzen, und garantieren damit, dass die Zeile vor einer Aktualisierung nicht von einer anderen Transaktion geändert wird.

Wenn eine Absichtssperre von einer Transaktion angefordert wird, die die Snapshot-Isolation benutzt, wird die Absichtssperre nur gesetzt, wenn es sich bei der Zeile um eine nicht geänderte Zeile in der Datenbank handelt, die allen parallelen Transaktionen zur Verfügung steht. Wenn es sich bei der Zeile jedoch um eine Snapshot-Kopie handelt, wird keine Absichtssperre gesetzt, da die Originalzeile bereits durch eine andere Transaktion geändert wurde. Jeder Versuch der Snapshot-Transaktion, die Zeile zu aktualisieren, schlägt fehl, und es wird ein Snapshot-Aktualisierungskonflikt gemeldet.

#### Siehe auch

- „Snapshot-Isolation“ auf Seite 888

## Tabellensperren

Zusätzlich zu Sperren für Zeilen unterstützt SQL Anywhere auch Sperren für Tabellen. Tabellensperren unterscheiden sich von Schemasperren: Eine Tabellensperre setzt eine Sperre für alle Zeilen in der Tabelle, im Gegensatz zu einer Sperre für das Tabellenschema. Es gibt drei Arten von Tabellensperren: gemeinsame Sperren, Schreibabsichtssperren und Exklusivsperren.

Tabellensperren werden zum Ende einer Transaktion freigegeben, wenn ein COMMIT oder ROLLBACK ausgeführt wird.

Die folgende Tabelle zeigt die Sperrenkombinationen, die Konflikte bewirken:

|           | Gemeinsam | Absicht  | Exklusiv |
|-----------|-----------|----------|----------|
| Gemeinsam |           | Konflikt | Konflikt |
| Absicht   | Konflikt  |          | Konflikt |
| Exklusiv  | Konflikt  | Konflikt | Konflikt |

#### Gemeinsame Tabellensperren

Eine gemeinsame Tabellensperre erlaubt mehreren Transaktionen, die Daten in einer Basistabelle zu lesen. Eine Transaktion, die eine gemeinsame Tabellensperre für eine Basistabelle hält, kann die Tabelle ändern, sofern keine andere Transaktion eine Sperre, welcher Art auch immer, für die Zeilen hält, die geändert werden.

Eine gemeinsame Tabellensperre wird beispielsweise gesetzt, indem eine LOCK TABLE ... IN SHARED MODE-Anweisung ausgeführt wird. Die Anweisungen REFRESH MATERIALIZED VIEW und REFRESH TEXT INDEX unterstützen ebenfalls eine WITH SHARE Mode-Klausel, die Sie verwenden

können, um gemeinsame Tabellensperren für die Basistabellen zu erstellen, während der Aktualisierungsvorgang stattfindet.

### Tabellensperren für Schreibabsicht

Eine Tabellensperre für Schreibabsicht, die auch als Absichtstabellensperre bezeichnet wird, wird implizit gesetzt, wenn durch eine Transaktion zum ersten Mal eine Schreibsperre für eine Zeile gesetzt wird. Eine Absichtstabellensperre wird gesetzt, wenn eine Zeile aktualisiert, eingefügt oder gelöscht wird. Ebenso wie gemeinsame Tabellensperren werden auch Absichtstabellensperren aufrechterhalten, bis die Transaktion mit COMMIT oder ROLLBACK abgeschlossen wird. Absichtstabellensperren stehen in Konflikt mit gemeinsamen und exklusiven Tabellensperren, nicht aber mit anderen Absichtstabellensperren.

### Exklusive Tabellensperren

Eine exklusive Tabellensperre hindert andere Transaktionen daran, das Schema oder Daten in einer Tabelle zu ändern oder neue Daten hinzuzufügen. Anders als bei exklusiven Schemasperren können jedoch Transaktionen, die auf Isolationsstufe 0 ausgeführt werden, die Zeilen einer Tabelle lesen, die eine exklusive Tabellensperre hat. Nur jeweils eine Transaktion kann zu einem bestimmten Zeitpunkt eine Exklusivsperre für eine Tabelle halten. Exklusive Tabellensperren stehen in Konflikt mit allen anderen Tabellen- und Zeilensperren.

Sie erwerben eine exklusive Tabellensperre implizit, wenn Sie die LOAD TABLE-Anweisung verwenden.

Mit der LOCK TABLE ... IN EXCLUSIVE MODE-Anweisung setzen Sie explizit eine exklusive Tabellensperre. Die Anweisungen REFRESH MATERIALIZED VIEW und REFRESH TEXT INDEX enthalten auch eine WITH EXCLUSIVE Mode-Klausel, die Sie verwenden können, um gemeinsame exklusive Tabellensperren für die Basistabellen zu setzen, während der Aktualisierungsvorgang stattfindet.

### Siehe auch

- „Schemasperren“ auf Seite 908
- „LOCK TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REFRESH MATERIALIZED VIEW-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REFRESH TEXT INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

## Positionssperren

Zusätzlich zu Zeilensperren implementiert SQL Anywhere auch Positionssperren, eine Art von Schlüsselbereichssperren, die Anomalien aufgrund von Phantomen oder Phantomzeilen verhindern. Positionssperren sind nur relevant, wenn der Datenbankserver Transaktionen verarbeitet, die auf Isolationsstufe 3 ablaufen.

Transaktionen, die auf Isolationsstufe 3 ablaufen, sind serialisierbar. Das Verhalten einer Transaktion auf Isolationsstufe 3 sollte nicht durch parallele Aktualisierungsaktivitäten anderer Transaktionen beeinflusst werden. Insbesondere können auf Isolationsstufe 3 keine Transaktionen durch INSERTs oder UPDATEs — Phantome — beeinflusst werden, die Zeilen einfügen, die sich auf die Ergebnisse einer Berechnung auswirken könnten. SQL Anywhere verwendet Positionssperren, um solche Aktualisierungen zu

verhindern. Diese zusätzlichen Sperren unterscheiden Isolationsstufe 2 ("repeatable read", d.h., wiederholbare Lesevorgänge) von Isolationsstufe 3.

Um das Erstellen von Phantomzeilen zu verhindern, setzt SQL Anywhere Sperren für Positionen innerhalb eines physischen Scans einer Tabelle. Bei einem sequenziellen Scan basiert die Scan-Position auf dem Zeilenidentifizierer der aktuellen Zeile. Bei einem Index-Scan basiert die Scan-Position auf dem Index-Schlüsselwert der aktuellen Zeile (der eindeutig oder nicht eindeutig sein kann). Durch die Sperre einer Suchposition verhindert eine Transaktion Einfügungen anderer Transaktionen in Bezug auf einen bestimmten Wertebereich in dieser Zeilenreihenfolge. Dieses Verhalten gilt für INSERT- und UPDATE-Anweisungen, die den Wert eines indizierten Attributs ändern. Wenn eine Scan-Position gesperrt ist, wird die UPDATE-Anweisung als Anforderung angesehen, mit DELETE den Indexeintrag zu löschen, unmittelbar gefolgt von einer INSERT-Anforderung.

SQL Anywhere unterstützt zwei Arten von Positionssperren: Phantomsperrern und Einfügesperren. Bei beiden Arten handelt es sich um gemeinsame Sperren, da beliebig viele Transaktionen die gleiche Art von Sperren für dieselben Zeilen setzen können. Bei Phantomsperrern und Anti-Phantomsperrern kommt es jedoch zu Konflikten.

## Phantomsperrern

Eine Phantomsperrre, die auch als Anti-Einfügesperre bezeichnet wird, wird auf eine Scan-Position gesetzt, um das anschließende Erstellen von Phantomzeilen durch andere Transaktionen zu verhindern. Wenn eine Phantomsperrre gesetzt wird, verhindert sie, dass andere Transaktionen eine Zeile in eine Tabelle unmittelbar vor der Zeile einfügen, für die eine Anti-Einfügesperre gesetzt wurde. Eine Phantomsperrre ist eine langfristige Sperre, die bis zum Abschluss einer Transaktion aufrechterhalten wird.

Phantomsperrern werden nur durch Transaktionen gesetzt, die auf Isolationsstufe 3 ausgeführt werden. Es handelt sich dabei um die einzige Isolationsstufe, die Konsistenz im Hinblick auf Phantome garantiert.

Für einen Index-Scan werden Phantomsperrern für jede durch den Index gelesene Zeile gesetzt, und am Ende des Index-Scans wird eine zusätzliche Phantomsperrre gesetzt, um Einfügungen in den Index am Ende des entsprechenden Indexbereichs zu verhindern. Phantomsperrern mit Index-Scans verhindern, dass Phantome durch das Einfügen neuer Zeilen in die Tabelle erstellt werden, oder dass ein indizierter Wert aktualisiert wird, der das Erstellen eines Indexeintrags an einer Stelle verursachen würde, die durch die Phantomsperrre abgedeckt ist.

Bei einem sequenziellen Scan werden Phantomsperrern für jede Zeile in einer Tabelle gesetzt, um zu verhindern, dass eine Einfügung die Ergebnismenge beeinflusst. Scans der Isolationsstufe 3 haben oft negative Auswirkungen auf die Datenbankparallelität. Während eine oder mehrere Phantomsperrern in Konflikt mit einer Einfügesperre stehen und eine oder mehrere Lesesperren in Konflikt mit einer Schreibsperre, gibt es zwischen Phantom-/Einfügesperren und Lese-/Schreibsperren keine Interaktion. Obwohl zum Beispiel eine Schreibsperre nicht für eine Zeile gesetzt werden kann, die eine Lesesperre enthält, kann sie für eine Zeile gesetzt werden, die lediglich über eine Phantomsperrre verfügt. Dem Datenbankserver stehen aufgrund dieser flexiblen Anordnung mehrere Optionen offen, aber das bedeutet, dass der Datenbankserver allgemein die zusätzliche Vorsichtsmaßnahme treffen muss, eine Lesesperre zu setzen, wenn er eine Phantomsperrre setzt. Andernfalls könnte eine andere Transaktion die Zeile löschen.

## Einfügesperren

Eine Einfügesperre, manchmal auch als Anti-Phantomsperrre bezeichnet, ist eine kurzfristige Sperre, die auf eine Scan-Position platziert wird, um das Recht zum Einfügen einer Zeile zu reservieren. Die Sperre wird nur für die Dauer der Einfügung aufrechterhalten. Sobald die Zeile innerhalb einer Datenbankseite ordnungsgemäß eingefügt wurde, erhält sie eine Schreibsperre, um die Konsistenz zu gewährleisten, und die Einfügesperre wird freigegeben. Eine Transaktion, die eine Einfügesperre für eine Zeile setzt, hält andere Transaktionen davon ab, eine Phantomsperrre für dieselbe Zeile zu setzen. Einfügesperren sind erforderlich, da der Datenbankserver auf einen Scan-Vorgang der Isolationsstufe 3 durch eine aktive Verbindung vorbereitet sein muss, der durch jede neue Anforderung auftreten könnte. Phantom- und Einfügesperren stehen nicht miteinander in Konflikt, wenn sie von derselben Transaktion gehalten werden.

## Sperrenkonflikte

SQL Anywhere verwendet Schema-, Zeilen-, Tabellen und Positionssperren nach Bedarf, um die verlangte Konsistenzstufe zu gewährleisten. Sie müssen die Verwendung einer bestimmten Sperre nicht explizit anfordern. Stattdessen legen Sie die Konsistenzstufe fest, die durch Auswählen der für diese Anforderung am besten geeigneten Isolationsstufe eingehalten wird. Die Kenntnis der einzelnen Sperrentypen hilft Ihnen bei der Wahl der Isolationsstufe und beim Verständnis für die Auswirkungen auf die Performance. Denken Sie daran, dass eine Transaktion sich nicht selbst durch das Setzen von Sperren blockieren kann. Ein Sperrenkonflikt kann lediglich zwischen zwei oder mehreren Transaktionen auftreten.

Im Allgemeinen tritt ein Sperrenkonflikt auf, wenn eine Transaktion versucht, eine Exklusivsperrre für eine Zeile zu setzen, an der eine andere Transaktion eine Sperre hält, oder versucht, eine gemeinsame Sperre für eine Zeile zu setzen, in der eine andere Transaktion eine Exklusivsperrre hält. Die Transaktion muss warten, bis eine andere Transaktion abgeschlossen ist. Die Transaktion, die warten muss, wird von einer anderen Transaktion **blockiert**.

Wenn der Datenbankserver einen Sperrenkonflikt erkennt, der verhindert, dass eine Transaktion sofort abläuft, kann er entweder die Ausführung der Transaktion aussetzen, oder die Transaktion beenden, Änderungen zurücksetzen und einen Fehler zurückgeben. Sie kontrollieren den Vorgang, indem Sie die Option "blocking" einstellen. Wenn die Option "blocking" auf **On** gesetzt ist, wartet die zweite Transaktion.

## Konflikte bei Sperren

Während jeder der vier Sperrentypen speziellen Zwecken dient, beeinflussen sich alle Typen gegenseitig und können daher einen Sperrenkonflikt zwischen Transaktionen verursachen. Um die Datenbankkonsistenz zu gewährleisten, sollte immer nur eine Transaktion eine Zeile zu einer bestimmten Zeit ändern. Andernfalls könnten zwei Transaktionen gleichzeitig versuchen, einen Wert in zwei verschiedene neue Werte zu ändern. Daher ist es wichtig, dass eine Zeilenschreibsperre exklusiv ist. Im Gegensatz dazu ergibt sich kein Problem, wenn mehrere Transaktionen gleichzeitig eine Zeile lesen wollen. Da keine Transaktion sie ändert, gibt es keinen Konflikt. Daher können Zeilenlesesperren von vielen Verbindungen gemeinsam genutzt werden.

Die folgende Tabelle zeigt die Sperrenkombination, die einen Konflikt bewirkt: Schemasperren werden nicht berücksichtigt, da sie sich nicht auf Zeilen beziehen.

| <b>Zeilensperren</b>   | <b>LesenPS</b> | <b>Lesen</b> | <b>Absicht</b> | <b>SchreibenKeinPS</b> | <b>Schreiben</b> |
|------------------------|----------------|--------------|----------------|------------------------|------------------|
| <b>LesenPS</b>         |                |              |                |                        | Konflikt         |
| <b>Lesen</b>           |                |              |                | Konflikt               | Konflikt         |
| <b>Absicht</b>         |                |              | Konflikt       | Konflikt               | Konflikt         |
| <b>SchreibenKeinPS</b> |                | Konflikt     | Konflikt       | Konflikt               | Konflikt         |
| <b>Schreiben</b>       | Konflikt       | Konflikt     | Konflikt       | Konflikt               | Konflikt         |

| <b>Tabellensperren</b> | <b>Gemeinsam</b> | <b>Absicht</b> | <b>Exklusiv</b> |
|------------------------|------------------|----------------|-----------------|
| <b>Gemeinsam</b>       |                  | Konflikt       | Konflikt        |
| <b>Absicht</b>         | Konflikt         |                | Konflikt        |
| <b>Exklusiv</b>        | Konflikt         | Konflikt       | Konflikt        |

| <b>Positionssperren</b> | <b>Phantom</b> | <b>Einfügen</b> |
|-------------------------|----------------|-----------------|
| <b>Phantom</b>          |                | Konflikt        |
| <b>Einfügen</b>         | Konflikt       |                 |

Siehe auch

- „sa\_locks-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Sperren bei Abfragen

Welche Sperren SQL Anywhere verwendet, wenn ein Benutzer eine SELECT-Anweisung eingibt, hängt von der Isolationsstufe der Transaktion ab. Unabhängig von der Isolationsstufe setzen alle SELECT-Anweisungen gemeinsame Schemasperren für die referenzierten Tabellen.

### SELECT-Anweisungen auf Isolationsstufe 0

Bei einer SELECT-Anweisung auf Isolationsstufe 0 sind keine Sperren erforderlich. Die Transaktionen sind nicht vor Änderungen durch andere Transaktionen geschützt. Sie bzw. der Datenbankbenutzer müssen die Ergebnisse dieser Abfragen richtig interpretieren und sich der Flüchtigkeit der abgerufenen Informationen bewusst sein.

### SELECT-Anweisungen auf Isolationsstufe 1

SQL Anywhere verwendet, wenn eine Transaktion auf Isolationsstufe 1 ausgeführt wird, nicht viel mehr Sperren als auf Isolationsstufe 0. Der Datenbankserver ändert sein Verhalten nur auf zwei Arten.

Der erste Unterschied hat nichts mit dem Setzen von Sperren zu tun, sondern vielmehr mit deren Respektierung. Auf Isolationsstufe 0 kann eine Transaktion jede Zeile lesen, sogar wenn eine andere Transaktion eine Schreibsperre gesetzt hat. Im Unterschied dazu muss jede Transaktion mit der Isolationsstufe 1 vor dem Lesen einer Zeile überprüfen, ob es dafür eine Schreibsperre gibt. Sie kann nicht über schreibgesperrte Zeilen hinaus lesen, da dies bedeuten könnte, nicht festgeschriebene Daten zu lesen. Die Verwendung des READPAST-Hints erlaubt es dem Server, mit Schreibsperren versehene Zeilen zu ignorieren, aber obwohl die Transaktion nicht mehr blockiert, stimmt ihre Semantik nicht mehr mit derjenigen der Isolationsstufe 1 überein.

Der zweite Unterschied betrifft die Cursorstabilität. Cursorstabilität wird durch das Setzen einer kurzfristigen Lesesperre für die aktuelle Zeile des Cursors erzielt. Diese Lesesperre wird freigegeben, wenn sich der Cursor verschiebt. Es kann mehr als eine Zeile davon betroffen sein, wenn der Inhalt des Cursors das Ergebnis einer Verbindung ist. In diesem Fall setzt der Datenbankserver kurzfristige Lesesperren für alle Zeilen, die Informationen zur aktuellen Zeile des Cursor beigetragen haben, und gibt diese Sperren frei, wenn eine andere Zeile des Cursor als aktuelle Zeile ausgewählt wurde.

### **SELECT-Anweisungen auf Isolationsstufe 2**

Auf Isolationsstufe 2 ändert der Datenbankserver seine Prozeduren, um die Semantik wiederholbarer Lesevorgänge zu gewährleisten. Wenn eine SELECT-Anweisung Werte aus jeder Zeile einer Tabelle liefert, setzt der Datenbankserver eine Lesesperre für jede Zeile der Tabelle, sobald er sie liest. Enthält aber die SELECT-Anweisung eine WHERE-Klausel oder eine andere Bedingung, die die Zeilen im Ergebnis einschränkt, dann liest der Datenbankserver stattdessen jede Zeile, vergleicht die Werte in der Zeile mit der Bedingung und setzt eine Lesesperre für die Zeile, wenn sie die Bedingung erfüllt. Die gesetzten Lesesperren sind langfristige Lesesperren und werden aufrecht erhalten, bis die Transaktion durch eine implizite oder explizite COMMIT- oder ROLLBACK-Anweisung abgeschlossen wird. Wie bei Isolationsstufe 1 wird die Cursorstabilität auch auf Isolationsstufe 2 gewährleistet und Dirty Reads sind nicht erlaubt.

### **SELECT-Anweisungen auf Isolationsstufe 3**

Bei Transaktionen auf Isolationsstufe 3 muss der Datenbankserver gewährleisten, dass alle Transaktionspläne serialisierbar sind. Besonders müssen, zusätzlich zu den Anforderungen der Isolationsstufe 2, Phantomzeilen verhindert werden, sodass die erneute Ausführung der gleichen Anweisung unter allen Umständen garantiert die gleichen Ergebnisse liefert.

Um diese Anforderung zu erfüllen, verwendet der Datenbankserver Lese- und Phantomsperrern. Wenn eine SELECT-Anweisung auf Isolationsstufe 3 ausgeführt wird, setzt der Datenbankserver eine Lesesperre für jede Zeile, die während der Berechnung der Ergebnismenge verarbeitet wird. Dadurch wird gewährleistet, dass keine anderen Transaktionen diese Zeilen ändern können, bevor die Transaktion abgeschlossen ist.

Diese Anforderung ähnelt den Vorgängen, die der Datenbankserver auf Isolationsstufe 2 durchführt. Der Unterschied besteht jedoch darin, dass eine Sperre für jede gelesene Zeile gesetzt werden muss, egal ob diese Zeilen die Prädikate in den Klauseln WHERE, ON oder HAVING von SELECT erfüllen. Wenn Sie zum Beispiel die Namen aller Angestellten in der Verkaufsabteilung auswählen, muss der Server alle Zeilen mit Informationen über Verkaufspersonal sperren, egal ob die Transaktion auf Isolationsstufe 2 oder 3 ausgeführt wird. Auf Isolationsstufe 3 muss der Server aber auch Lesesperren für jede Zeile mit Angestellten gesetzt werden, die nicht in der Verkaufsabteilung arbeiten. Andernfalls könnte eine andere

Transaktion möglicherweise einen anderen Angestellten in die Verkaufsabteilung transferieren, während die erste Transaktion noch ausgeführt wird.

Es gibt zwei Auswirkungen, wenn eine Lesesperre für jeden Zeilenlesevorgang gesetzt werden muss:

- Der Datenbankserver muss möglicherweise mehr Sperren setzen, als auf Isolationsstufe 2 notwendig wären. Die Anzahl der gesetzten Phantomsperrern ist größer als die Anzahl der Lesesperren, die für den Scan gesetzt werden. Diese Verdoppelung des Sperren-Overheads verlängert die Ausführungszeit der Anforderung.
- Das Setzen von Lesesperren für jede gelesene Zeile hat negative Auswirkungen auf die Parallelität der Aktualisierungsvorgänge für die Tabelle in der Datenbank.

Die Anzahl der vom Datenbankserver gesetzten Phantomsperrern kann stark schwanken und hängt von der Ausführungsstrategie ab, die vom Abfrageoptimierer gewählt wurde. Der SQL Anywhere-Abfrageoptimierer versucht, sequenzielle Scans auf Isolationsstufe 3 zu vermeiden, da sich diese nachteilig auf die Parallelität des Gesamtsystems auswirken können. Die Fähigkeit des Optimierers zur Vermeidung hängt jedoch von den Prädikaten in der Anweisung und von den relevanten Indizes ab, die in den referenzierten Tabellen verfügbar sind.

Nehmen wir zum Beispiel an, Sie wollen Informationen über den Mitarbeiter mit der Mitarbeiter-ID 123 auswählen. Da EmployeeID der Primärschlüssel der Employees-Tabelle ist, wird der Abfrageoptimierer mit großer Wahrscheinlichkeit eine index-basierte Strategie unter Verwendung des Primärschlüsselindexes wählen, um die Zeile auf effiziente Weise zu finden. Außerdem besteht keine Gefahr, dass eine andere Transaktion die Kennung eines anderen Angestellten auf 123 ändern könnte, da Primärschlüsselwerte eindeutig sein müssen. Der Server kann garantieren, dass kein zweiter Angestellter die gleiche Kennung hat, indem eine Lesesperre für die Zeile gesetzt wird, die die Information über den Angestellten mit der Kennung 123 enthält.

Im Gegensatz dazu müsste der Datenbankserver eine größere Anzahl von Sperren setzen, würde er alle Angestellten in der Verkaufsabteilung auswählen. Da es keinen relevanten Index gibt, muss der Datenbankserver jede Zeile in der Mitarbeitertabelle lesen und für jeden Mitarbeiter prüfen, ob er zur Verkaufsabteilung gehört. Wenn dies der Fall ist, müssen sowohl Lese- als auch Phantomsperrern für jede Zeile in der Tabelle gesetzt werden.

### **SELECT-Anweisungen und Snapshot-Isolation**

SELECT-Anweisungen, die mit den Isolationsstufen snapshot, statement-snapshot oder readonly-statement-snapshot ausgeführt werden, setzen keine Lesesperren. Der Grund dafür ist, dass jede Snapshot-Transaktion (oder -Anweisung) einen Snapshot eines festgeschriebenen Zustands zu einem Zeitpunkt in der Vergangenheit berücksichtigt. Der jeweilige Zeitpunkt hängt davon ab, welche der drei Snapshot-Isolationsstufen von der Anweisung benutzt werden. Lesetransaktionen blockieren niemals Aktualisierungstransaktionen, und Aktualisierungstransaktionen blockieren keine Lesevorgänge. Daher kann die Snapshot-Isolation beträchtliche Vorteile im Hinblick auf die Parallelität bieten, zusätzlich zu den offensichtlichen Konsistenzvorteilen. Es gibt jedoch einen Nachteil: Die Snapshot-Isolation kann sehr kostspielig sein. Dies liegt daran, dass wegen der Konsistenzgarantie der Snapshot-Isolation Kopien von geänderten Zeilen für andere Transaktionen gespeichert, verfolgt und schließlich gelöscht werden müssen.

**Siehe auch**

- „FROM-Klausel“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Sperren bei Einfügungen

INSERT-Vorgänge erstellen neue Zeilen. SQL Anywhere setzt während Einfügungen unterschiedliche Sperrentypen ein, um die Datenintegrität zu gewährleisten. Die folgende Reihenfolge der Vorgänge gilt für INSERT-Anweisungen, die auf einer beliebigen Isolationsstufe ausgeführt werden.

1. Es wird eine gemeinsame Schemasperre für die Tabelle gesetzt, falls eine solche nicht bereits vorhanden ist.
2. Es wird eine Tabellensperre für Schreibabsicht für die Tabelle gesetzt, falls eine solche nicht bereits vorhanden ist.
3. Es wird eine nicht gesperrte Position auf einer Seite gesucht, um die neue Zeile zu speichern. Um Sperrenkonflikte zu minimieren, belegt der Datenbankserver den Platz, der durch gelöschte (aber noch nicht festgeschriebene) Zeilen verfügbar wurde, nicht sofort wieder. Der Tabelle kann eine neue Seite zugeordnet werden (und die Datenbankdatei könnte anwachsen), um die neue Zeile unterzubringen.
4. In der neuen Zeile werden die Werte eingetragen.
5. In der Tabelle, in der die Zeile hinzugefügt wird, wird eine Einfügesperre gesetzt. Einfügesperren sind Exklusivsperrern, d.h., sobald die Einfügesperre gesetzt ist, kann keine andere Transaktion auf Isolationsstufe 3 die Einfügung durch das Setzen einer Phantomsperre blockieren.
6. Die neue Zeile wird mit einer Schreibsperre versehen. Die Einfügesperre wird freigegeben, sobald die Schreibsperre gesetzt wurde.
7. Die Zeile wird in die Tabelle eingefügt. Andere Transaktionen auf Isolationsstufe 0 können jetzt zum ersten Mal erkennen, dass die neue Zeile existiert. Diese anderen Transaktionen können jedoch die neue Zeile wegen der zuvor gesetzte Schreibsperre weder ändern noch löschen.
8. Alle betroffenen Indizes werden aktualisiert und die Eindeutigkeit wird geprüft, wo dies erforderlich ist. Primärschlüsselwerte müssen eindeutig sein. Andere Spalten können ebenfalls so definiert werden, dass sie nur eindeutige Werte enthalten. Wenn solche Spalten definiert wurden, muss auch die Eindeutigkeit überprüft werden.
9. Wenn es sich bei der Tabelle um eine Fremdtabelle handelt, wird eine gemeinsame Schemasperre für die Primärtabelle gesetzt (falls nicht bereits vorhanden). Für die entsprechende Primärzeile in der Primärtabelle wird eine Lesesperre gesetzt, wenn die eingefügten Werte für die Fremdschlüsselspalte nicht gleich NULL sind. Der Datenbankserver muss gewährleisten, dass die Primärzeile noch vorhanden ist, wenn die COMMIT-Anweisungen der Transaktion ausgeführt werden. Dies geschieht, indem er eine Lesesperre für die Primärzeile setzt. Sobald die Lesesperre gesetzt ist, kann eine andere Transaktion diese Zeile zwar lesen, aber keine Transaktion kann sie löschen oder aktualisieren.

Falls die entsprechende Primärzeile nicht vorhanden ist, kommt es zu einer Verletzung der Integritätsregel zur Erhaltung der referenziellen Integrität.



Nach dem letzten Schritt können alle für die Tabelle definierten AFTER INSERT-Trigger ausgelöst werden. Bei der Verarbeitung innerhalb von Triggern verhalten Sperren sich genauso wie bei Anwendungen. Sobald die Transaktion festgeschrieben (falls alle referenziellen Integritätsregeln erfüllt wurden) oder zurückgesetzt wurde, werden alle langfristigen Sperren freigegeben.

### Eindeutigkeit

Sie können sicherstellen, dass alle Werte in einer bestimmten Spalte oder Kombination von Spalten eindeutig sind. Der Datenbankserver erfüllt diese Aufgabe immer, indem er für die eindeutige Spalte einen Index erstellt, auch wenn Sie dies nicht ausdrücklich verlangen.

Insbesondere müssen alle Primärschlüsselwerte eindeutig sein. Der Datenbankserver erstellt automatisch einen Index für den Primärschlüssel jeder Tabelle. Verlangen Sie vom Datenbankserver nicht, einen Index für den Primärschlüssel zu erstellen, da ein solcher Index redundant wäre.

### Waisen und referenzielle Integrität

Ein Fremdschlüssel ist eine Referenz auf einen Primärschlüssel oder eine UNIQUE-Regel, normalerweise in einer anderen Tabelle. Wenn dieser Primärschlüssel nicht existiert, wird der bezugslose Fremdschlüssel **Waise** genannt. SQL Anywhere stellt automatisch sicher, dass Ihre Datenbank keine Zeilen enthält, die die referenzielle Integrität verletzen. Dieser Prozess wird **Überprüfen der referenziellen Integrität** genannt. Der Datenbankserver überprüft die referenzielle Integrität, indem er Waisen zählt.

### wait\_for\_commit

Sie können den Datenbankserver anweisen, mit der Überprüfung der referenziellen Integrität bis zum Ende Ihrer Transaktion zu warten. In diesem Modus können Sie eine Zeile einfügen, die einen Fremdschlüssel enthält. Dann fügen Sie eine Primärzeile ein, die den fehlenden Primärschlüssel enthält. Beide Vorgänge müssen in derselben Transaktion erfolgen.

Damit der Datenbankserver die Überprüfung der referenziellen Integrität bis zur Festschreibung verzögert, müssen Sie den Wert der Option "wait\_for\_commit" auf "On" stellen. Standardmäßig ist diese Option auf "Off" gesetzt. Um sie auf "On" zu setzen, müssen Sie die folgende Anweisung ausführen:

```
SET OPTION wait_for_commit = On;
```

Wenn der Server beim Einfügen eines neuen Fremdschlüsselwertes keine übereinstimmende Primärzeile findet und "wait\_for\_commit" auf "On" gesetzt ist, erlaubt der Server die Einfügung als Waise. Wenn Fremdzeilen als Waisen eingefügt werden, werden nach der Einfügung folgende Schritte durchgeführt:

- Der Server setzt eine gemeinsame Schemasperre für die Primärtabelle (falls nicht bereits vorhanden). Der Server setzt außerdem eine Sperre für Schreibabsicht für die Primärtabelle.
- Der Server fügt eine Hilfszeile in die Primärtabelle ein. Es wird nicht tatsächlich eine Zeile in die Primärtabelle eingefügt, sondern der Server erstellt für Sperrenzwecke einen eindeutigen Zeilenidentifizierer für diese Zeile. Für diese Hilfszeile wird dann eine Schreibsperre gesetzt. Anschließend fügt der Server die entsprechenden Werte in den Primärschlüsselindex der Primärtabelle ein.

Bevor eine Transaktion festgeschrieben wird, überprüft der Datenbankserver, ob die referenzielle Integrität aufrechterhalten bleibt, indem er die Anzahl der Waisen zählt, die durch Ihre Transaktion geschaffen wurden. Am Ende jeder Transaktion muss diese Anzahl 0 sein.

## Sperren während einer Aktualisierung

Der Datenbankserver ändert die in einem bestimmten Datensatz enthaltene Information mit folgenden Schritten. Wie bei Einfügungen gilt diese Reihenfolge der Vorgänge für alle Transaktionen unabhängig von ihrer Isolationsstufe.

1. Es wird eine gemeinsame Schemasperre für die Tabelle gesetzt, falls eine solche nicht bereits vorhanden ist.
2. Es wird eine Tabellensperre für Schreibabsicht für jede zu aktualisierende Tabelle gesetzt, falls eine solche nicht bereits vorhanden ist.
  - a. Bei jeder zu aktualisierende Tabelle, die Trigger aufweist, müssen anschließend die temporären Tabellen für die alten und neuen Werte nach Bedarf erstellt werden.
  - b. Die zu aktualisierenden Zeilen werden festgelegt. Während Zeilen durchsucht werden, sind sie gesperrt.

Bei den Isolationsstufen 2 und 3 gibt es folgende Abweichungen vom Standard-Sperrverhalten: Sperren für Schreibabsicht auf Zeilenebene werden anstelle von Lesesperren gesetzt und es können Sperren für Schreibabsicht bei Zeilen gesetzt werden, die schließlich als Kandidaten für eine Aktualisierung verworfen werden.

- c. Für jede in Schritt 2 festgelegte Kandidatenzeile wird die verbleibende Sequenz durchgeführt.
3. Die betroffene Zeile wird mit einer Schreibsperre versehen.
4. Jeder der betroffenen Spaltenwerte wird gemäß der UPDATE-Anweisung aktualisiert.
5. Wurden indizierte Werte geändert, dann werden neue Indexeinträge hinzugefügt. Die Original-Indexeinträge für die Zeile bleiben bestehen, werden aber als gelöscht gekennzeichnet. Neue Indexeinträge für die neuen Werte werden eingefügt, während eine kurzfristige Einfügesperre besteht. Falls erforderlich, verifiziert der Server die Indexeindeutigkeit.
6. Wenn eine Verletzung der Eindeutigkeit aufgetreten ist, wird eine temporäre Tabelle erstellt, um die alten und neuen Werte der Zeile zu speichern. Die alten und neuen Werte werden in die temporäre Tabelle kopiert und die Zeile in der Basistabelle wird gelöscht. Vorhandene DELETE-Trigger werden nicht ausgelöst. Verschieben Sie die Schritte 7 bis 9 ans Ende der zeilenweisen Verarbeitung.
7. Falls Fremdschlüsselwerte in der Zeile geändert wurden, wird eine gemeinsame Schemasperre für die Primärtabelle(n) gesetzt und neue Fremdschlüsselwerte werden gemäß der entsprechenden Prozedur eingefügt.

Falls zutreffend, wird ebenfalls die Prozedur für WAIT\_FOR\_COMMIT durchgeführt.

8. Wenn die Tabelle eine Primärtabelle in einer referenziellen Integritätsbeziehung ist und es sich bei dem UPDATE-Vorgang der Beziehung nicht um RESTRICT handelt, werden die betroffenen Zeilen in den Fremdtabellen ermittelt, indem zuerst eine gemeinsame Schemasperre für die Tabellen gesetzt wird, eine Tabellensperre für Schreibabsicht für jede Tabelle, und indem Schreibsperren für alle betroffenen Zeilen gesetzt werden, die wie erforderlich geändert werden. Dieser Prozess wird

möglicherweise kaskadierend durch eine verschachtelte Hierarchie von referenziellen Integritätsregeln durchgeführt.

#### 9. Lösen Sie die erforderlichen AFTER ROW-Trigger aus.

Nach dem letzten Schritt wird nun, sofern eine temporäre Tabelle erforderlich war, jede Zeile aus der temporären Tabelle in die Basistabelle eingefügt (ohne dass jedoch INSERT-Trigger ausgelöst werden). Wenn die Zeileneinfügung erfolgreich ist, werden die oben genannten Schritte 7 bis 9 ausgeführt und die alten und neuen Zeilenwerte in die alten und neuen temporären Tabellen kopiert, damit vorhandene AFTER STATEMENT UPDATE-Trigger alle geänderten Zeilen richtig verarbeiten können. Nachdem alle gespeicherten Zeilen verarbeitet wurden, werden die AFTER STATEMENT UPDATE-Trigger in der entsprechenden Reihenfolge ausgelöst. Bei COMMIT verifiziert der Server die referenzielle Integrität, indem er sicherstellt, dass die Anzahl der durch diese Transaktion erzeugten Waisen gleich 0 ist, und er gibt alle Sperren frei.

Die Änderung eines Spaltenwertes kann eine Vielzahl von Vorgängen nach sich ziehen. Der Datenbankserver hat wesentlich weniger Arbeit, wenn die Spalte, die Sie ändern, nicht Teil eines Primär- oder Fremdschlüssels ist. Die Arbeit ist noch geringer, wenn die Spalte in keinem Index enthalten ist, weder explizit noch implizit, falls die Spalte als eindeutig definiert wurde.

Die Überprüfung der referenziellen Integrität während eines UPDATE-Vorgangs ist nicht einfacher, als wenn die Überprüfung während eines INSERT-Vorgangs durchgeführt wird. Wenn Sie den Wert eines Primärschlüssels ändern, können Sie dadurch Waisen schaffen. Wenn Sie die neuen Werte einfügen, muss der Datenbankserver erneut eine Überprüfung auf Waisen durchführen.

#### Siehe auch

- „Sperren bei Einfügungen“ auf Seite 918
- „Isolationsstufen und Konsistenz“ auf Seite 885

## Sperren während des Löschens

Der DELETE-Vorgang verläuft fast in derselben Prozedur wie der INSERT-Vorgang, nur in umgekehrter Reihenfolge. Wie bei Einfügungen und Aktualisierungen gilt diese Reihenfolge der Vorgänge für alle Transaktionen unabhängig von ihrer Isolationsstufe.

1. Es wird eine gemeinsame Schemasperre für die Tabelle gesetzt, falls eine solche nicht bereits vorhanden ist.
2. Es wird eine Tabellensperre für Schreibabsicht für die Tabelle gesetzt, falls eine solche nicht bereits vorhanden ist.
  - a. Die zu aktualisierenden Zeilen werden festgelegt. Während Zeilen durchsucht werden, sind sie gesperrt.

Bei den Isolationsstufen 2 und 3 gibt es folgende Abweichungen vom Standard-Sperrverhalten: Sperren für Schreibabsicht auf Zeilenebene werden anstelle von Lesesperren gesetzt und es können Sperren für Schreibabsicht bei Zeilen gesetzt werden, die schließlich als Kandidaten für eine Aktualisierung verworfen werden.

- b. Für jede in Schritt 2 festgelegte Kandidatenzeile wird die verbleibende Sequenz durchgeführt.
3. Für die zu löschende Zeile wird eine Schreibsperrung gesetzt.
4. Die Zeile wird aus der Tabelle entfernt, sodass sie nicht mehr von anderen Transaktionen gesehen werden kann. Die Zeile kann erst gelöscht werden, wenn die Transaktion festgeschrieben wurde, da ansonsten die Transaktion nicht mehr zurückgesetzt werden kann. Die Indexeinträge für die gelöschte Zeile werden aufbewahrt, aber als gelöscht gekennzeichnet, bis die Transaktion abgeschlossen ist. Damit wird verhindert, dass andere Transaktionen die gleiche Zeile neu einfügen.
5. Wenn die Tabelle eine Primärtabelle in einer referenziellen Integritätsbeziehung ist und es sich bei dem DELETE-Vorgang der Beziehung nicht um RESTRICT handelt, werden die betroffenen Zeilen in den Fremdtabellen ermittelt, indem zuerst eine gemeinsame Schemasperrung für die Tabellen gesetzt wird, eine Tabellensperrung für Schreibabsicht für jede Tabelle, und indem Schreibsperrungen für alle betroffenen Zeilen gesetzt werden, die wie erforderlich geändert werden. Dieser Prozess wird möglicherweise kaskadierend durch eine verschachtelte Hierarchie von referenziellen Integritätsregeln durchgeführt.

Die Transaktion kann festgeschrieben werden, wenn die referenzielle Integrität dadurch nicht beeinträchtigt wird. Zur Überprüfung der referenziellen Integrität protokolliert der Datenbankserver alle Waisen, die als Nebenwirkung des Löschvorgangs geschaffen wurden. Bei COMMIT protokolliert der Server den Vorgang in der Transaktionslogdatei und gibt alle Sperren frei.

### Siehe auch

- „Isolationsstufen und Konsistenz“ auf Seite 885

## Sperrendauer

Sperren werden in der Regel von einer Transaktion gehalten, bis sie abgeschlossen ist. Dieses Verhalten verhindert, dass andere Transaktionen Änderungen vornehmen, die es unmöglich machen, die ursprüngliche Transaktion zurückzusetzen. Bei Isolationsstufe 3 müssen alle Sperren gehalten werden, bis eine Transaktion beendet ist, um Transaktionsserialisierung zu garantieren.

Die einzigen Sperren, die nicht bis zum Ende einer Transaktion gehalten werden, sind Cursorstabilitätssperren. Diese Zeilensperren werden so lange gehalten, bis die betreffende Zeile die aktuelle Zeile eines Cursors ist. In den meisten Fällen ist die Zeitspanne kürzer als die Dauer der Transaktion, aber für WITH HOLD-Cursor können Cursorstabilitätssperren für die Dauer der Verbindung gehalten werden.

### Siehe auch

- „LOCK TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „OPEN-Anweisung [ESQL] [SP]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

## Richtlinien zum Auswählen der Isolationsstufen

Die Wahl der Isolationsstufe hängt von der Aufgabe ab, die die Anwendung ausführt. Der vorliegende Abschnitt enthält einige Richtlinien für das Auswählen der Isolationsstufen.

Um eine Isolationsstufe auszuwählen, müssen Sie die notwendige Balance zwischen Konsistenz und Fehlerfreiheit einerseits und der Notwendigkeit gleichzeitiger Transaktionen andererseits finden. Wenn eine Transaktion nur einen oder zwei spezifische Werte in einer Tabelle berücksichtigt, ist es unwahrscheinlich, dass sie sich genau so stark mit anderen Prozessen überlagert wie eine Transaktion, die viele große Tabellen durchsucht und deshalb eventuell viele Zeilen oder ganze Tabellen sperrt und sehr lange brauchen kann, bis sie abgeschlossen ist.

Wenn bei Ihren Transaktionen beispielsweise Geld von einem Bankkonto auf ein anderes überwiesen wird, werden Sie sicherstellen wollen, dass die übertragenen Daten korrekt sind. Wenn Sie jedoch nur eine ungefähre Schätzung des Anteils inaktiver Konten benötigen, ist es Ihnen möglicherweise nicht wichtig, ob Ihre Transaktion auf andere wartet. Sie sind vielleicht sogar bereit, etwas an Genauigkeit zu opfern, um nicht mit anderen Benutzern der Datenbank zu kollidieren.

Außerdem wirkt sich eine Überweisung vielleicht nur auf die zwei Zeilen aus, die die beiden Kontostände enthalten, während alle Konten gelesen werden müssen, um die grobe Berechnung zu erhalten. Aus diesem Grund ist es unwahrscheinlich, dass die Überweisung andere Transaktionen verzögert.

SQL Anywhere bietet vier Isolationsstufen: 0, 1, 2 und 3. Stufe 3 bedeutet die vollständige Isolation und gewährleistet, dass sich alle Transaktionen so überlappen, dass die Abfolgeplanung serialisierbar ist.

Wenn Sie die Snapshot-Isolation für eine Datenbank aktiviert haben, stehen drei weitere Isolationsstufen zur Verfügung: Snapshot, statement-snapshot und readonly-statement-snapshot.

### Snapshot-Isolationsstufe wählen

Die Snapshot-Isolation bietet Vorteile sowohl für die Parallelität als auch für die Konsistenz. Die Verwendung der Snapshot-Isolation hat Kostennachteile, da alte Zeilenversionen so lange aufbewahrt werden, wie sie von laufenden Transaktionen möglicherweise benötigt werden. Snapshots mit langer Ausführungszeit können daher die Speicherung vieler alter Zeilenversionen erfordern. Normalerweise haben Snapshots für statement-snapshot eine kürzere Ausführungszeit als Snapshots für snapshot. Die Stufe statement-snapshot bietet daher möglicherweise Speicherplatzvorteile gegenüber Snapshot, jedoch auf Kosten der Konsistenz (jede Anweisung innerhalb der Transaktion sieht die Datenbank zu einem unterschiedlichen Zeitpunkt).

Für die meisten Situationen wird die Snapshot-Isolationsstufe empfohlen, da diese eine einzige Ansicht der Datenbank während der gesamten Transaktion bietet.

Die Anweisungs-Snapshot-Isolationsstufe bietet weniger Konsistenz, kann aber nützlich sein, wenn lang andauernde Transaktionen dazu führen, dass die Versionsspeicherung in der temporären Datei zu viel Speicherplatz belegt.

Die Isolationsstufe readonly-statement-snapshot bietet weniger Konsistenz als statement-snapshot, verhindert aber, dass es zu Aktualisierungskonflikten kommt. Daher ist diese Isolationsstufe am besten für die Portierung von Anwendungen geeignet, die ursprünglich dazu gedacht waren, unter anderen Isolationsstufen zu laufen.

### Siehe auch

- „Snapshot-Isolation“ auf Seite 888
- „Cursor-Sensitivität und Isolationsstufen“ [*SQL Anywhere Server - Programmierung*]

## Serialisierbare Abfolgeplanung

Um Transaktionen gleichzeitig zu bearbeiten, muss der Datenbankserver einige Teile der Anweisungen einer Transaktion ausführen, danach Teile von anderen Transaktionen, bevor wieder weitere Vorgänge von der ersten Transaktion durchgeführt werden. Die Reihenfolge, in der die Abläufe der verschiedenen Transaktionen miteinander verwoben sind, nennt sich **Abfolgeplanung**.

Wenn Transaktionen auf diese Art gleichzeitig durchgeführt werden, kann dies zu verschiedenen Ergebnissen führen, darunter auch zu den drei Inkonsistenzen, die im vorherigen Abschnitt beschrieben wurden. Manchmal wäre der endgültige Zustand der Datenbank auch erreicht worden, wenn die Transaktionen sequenziell durchgeführt worden wären, das heißt, dass eine Transaktion immer vollständig abgeschlossen wird, bevor die nächste beginnt. Eine Abfolgeplanung wird als **serialisierbar** bezeichnet, wenn die sequenzielle Ausführung der Transaktionen in einer beliebigen Reihenfolge zum gleichen Zustand der Datenbank geführt hätte.

Serialisierbarkeit ist das allgemein akzeptierte Kriterium für Fehlerfreiheit. Ein serialisierbares Schema wird als fehlerfrei akzeptiert, da die Datenbank nicht durch die gleichzeitige Ausführung der Transaktionen beeinträchtigt wird.

Die Isolationsstufe wirkt sich auf die Serialisierbarkeit einer Transaktion aus. Bei Isolationsstufe 3 sind alle Ablaufplanungen serialisierbar. Die Standardeinstellung ist 0.

### Serialisierbar bedeutet, dass sich die Parallelität nicht auswirkt

Selbst wenn Transaktionen sequenziell ausgeführt werden, kann der endgültige Zustand der Datenbank von der Reihenfolge abhängen, in der diese Transaktionen durchgeführt werden. Beispiel: Wenn eine Transaktion eine bestimmte Zelle auf den Wert 5 setzt, und eine andere auf 6, wird der endgültige Wert der Zelle dadurch bestimmt, welche Transaktion zuletzt durchgeführt wird.

Das Wissen um die Serialisierbarkeit eines Schemas löst die Frage nicht, in welcher Reihenfolge die Transaktionen am besten ausgeführt werden, sondern gibt nur an, dass Parallelität keine Auswirkung hat. Die durch die sequenzielle Ausführung einer Reihe von Transaktionen erzielten Ergebnisse gelten alle als richtig.

### Nicht serialisierbare Schemata führen zu Inkonsistenzen

Die Inkonsistenzen sind typisch für Probleme, die auftreten, wenn der Zeitplan nicht serialisierbar ist. In jedem Fall kam es zu einer Inkonsistenz, da sich die Anweisungen überlappten. Das erzeugte Ergebnis wäre nicht möglich, wenn alle Transaktionen sequenziell ausgeführt worden wären. So kann zum Beispiel ein Dirty Read nur auftreten, wenn eine Transaktion Zeilen selektieren kann, während eine andere Transaktion in derselben Zeile Daten einfügt oder aktualisiert.

#### Siehe auch

- „Typische Arten von Inkonsistenz“ auf Seite 895

## Typische Transaktionen für verschiedene Isolationsstufen

Verschiedene Isolationsstufen eignen sich für bestimmte Arten von Aufgaben. Entscheiden Sie mithilfe der nachstehenden Informationen, welche Isolationsstufe sich am besten für welche Aufgabe eignet.

### Typische Stufe 0-Transaktionen

Transaktionen, die Daten durchsuchen oder eintragen, können mehrere Minuten dauern und eine große Anzahl von Zeilen lesen. Wird dafür Isolationsstufe 2 oder 3 verwendet, kann die Parallelität darunter leiden. Für diese Art von Transaktionen verwendet man gewöhnlich die Isolationsstufen 0 oder 1.

Eine Decision Support-Anwendung, die große Mengen an Informationen von der Datenbank liest, um statistische Zusammenfassungen zu erstellen, wird beispielsweise nicht wesentlich beeinflusst, wenn sie einige Zeilen liest, die später geändert werden. Ist für so eine Anwendung eine hohe Isolationsstufe erforderlich, so können Lesesperren für große Datenmengen gesetzt werden, wodurch andere Anwendungen keinen Schreibzugang dazu haben.

### Typische Stufe 1-Transaktionen

Isolationsstufe 1 ist bei Cursors nützlich, da diese Kombination Cursorstabilität gewährleistet, ohne die Sperranforderungen wesentlich zu erhöhen. SQL Anywhere erzielt diesen Vorteil durch die vorzeitige Freigabe von Lesesperren, die für die aktuelle Zeile eines Cursors gesetzt wurden. Diese Sperren müssen bei Transaktionen mit Isolationsstufen 2 oder 3 bis zum Abschluss der Transaktion andauern, damit wiederholbare Lesevorgänge gewährleistet sind.

Zum Beispiel eignet sich eine Transaktion, die die Inventarlisten über einen Cursor aktualisiert, für diese Isolationsstufe, da keine Änderungen aufgrund von Kauf und Verkauf verloren gehen würden und diese häufigen Änderungen nur eine minimale Auswirkung auf andere Transaktionen haben.

### Typische Stufe 2-Transaktionen

Bei Isolationsstufe 2 können Zeilen, die Ihrem Kriterium entsprechen, nicht von anderen Transaktionen verändert werden. Sie können diese Isolationsstufe verwenden, wenn Sie Zeilen öfter als einmal lesen müssen und sich darauf verlassen wollen, dass sich die erste erhaltene Ergebnismenge nicht ändert.

Da aber diese Isolationsstufe relativ viele Lesesperren erfordert, sollte sie nur verwendet werden, wenn dies unbedingt erforderlich ist. Wie auch bei Transaktionen mit Isolationsstufe 3 kann eine sorgfältige Planung Ihrer Datenbank und der Indizes die Anzahl der erforderlichen Sperren verringern und die Performance Ihrer Datenbank wesentlich steigern.

### Typische Stufe 3-Transaktionen

Isolationsstufe 3 ist für Transaktionen notwendig, die große Sicherheit verlangen. Die Eliminierung von Phantomzeilen ermöglicht Ihnen die Ausführung von Operationen mit mehreren Schritten auf einer Anzahl von Zeilen, ohne dass Sie befürchten müssen, dass während Ihrer Transaktion neue Zeilen auftauchen und das Ergebnis verfälschen.

Auch wenn Isolationsstufe 3 ein hohes Maß an Integrität bietet, sollte sie in großen Systemen, die eine Vielzahl von gleichzeitigen Transaktionen unterstützen, sparsam verwendet werden. SQL Anywhere bringt auf dieser Isolationsstufe die meisten Sperren an und erhöht somit die Wahrscheinlichkeit, dass eine Transaktion den Ablauf von vielen anderen behindert.

## Parallelitätsverbesserung bei den Isolationsstufen 2 und 3

Die Isolationsstufen 2 und 3 verwenden viele Sperren, sodass eine gute Planung für solche Datenbanken, die diese Isolationsstufen häufig benutzen, besonders wichtig ist. Wenn Sie serialisierbare Transaktionen



verwenden müssen, sollten Sie bei der Planung der Datenbank, und besonders der Indizes, die Geschäftsregeln für Ihr Projekt berücksichtigen. Sie können die Performance auch steigern, indem Sie große Transaktionen in mehrere kleinere aufteilen und die Zeit verkürzen, während der die Zeilen gesperrt sind.

Obwohl serialisierbare Transaktionen das größte Potenzial haben, andere Transaktionen zu blockieren, sind sie deshalb nicht unbedingt weniger effizient. Bei der Bearbeitung dieser Transaktionen kann SQL Anywhere bestimmte Optimierungen ausführen, die die Performance trotz der größeren Anzahl von Sperren verbessert. Da beispielsweise alle gelesenen Zeilen gesperrt werden müssen, unabhängig davon ob sie den Suchkriterien entsprechen, kann der Datenbankserver das Lesen der Zeilen und das Anbringen von Sperren kombinieren.

## Tipps zur Verminderung der Auswirkungen der Sperren

Um das Anbringen einer großen Anzahl von Sperren zu vermeiden, was sich auf die Ausführung anderer gleichzeitiger Transaktionen auswirken kann, wird empfohlen, möglichst keine Transaktionen auf Isolationsstufe 3 auszuführen.

Ist es aber notwendig, Isolationsstufe 3 zu verwenden, dann können Sie die Auswirkungen auf die Parallelität von Transaktionen mindern, indem Sie die Abfrage so gestalten, dass nur die absolut notwendigen Zeilen und Indizeinträge gelesen werden. Dadurch kann die Transaktion mit Isolationsstufe 3 rascher ablaufen, und was vielleicht noch wichtiger ist, es werden weniger Sperren angebracht.

Wenn zumindest ein Vorgang auf Isolationsstufe 3 ausgeführt wird, führt möglicherweise das Hinzufügen eines Index zu einer Erhöhung der Transaktionsgeschwindigkeit. Ein Index kann zwei Vorteile haben:

- Ein Index ermöglicht es, Zeilen auf effiziente Art zu finden.
- Suchvorgänge, die Indizes verwenden, benötigen möglicherweise weniger Sperren.

### Siehe auch

- [„Funktionsweise von Sperren“ auf Seite 906](#)
- [„Tools für Performanceüberwachung und Diagnose“ auf Seite 153](#)

## Praktische Einführung in Isolationsstufen

Jede Isolationsstufen verhält sich anders. Welche Isolationsstufe für Sie geeignet ist, hängt von Ihrer Datenbank und von den beabsichtigten Vorgängen ab. Die folgenden praktischen Einführungen helfen Ihnen, die passenden Isolationsstufen für unterschiedliche Aufgaben zu ermitteln.

### Siehe auch

- [„Isolationsstufen und Konsistenz“ auf Seite 885](#)
- [Einführung in Snapshot-Transaktionen auf Seite 890](#)
- [„Typische Arten von Inkonsistenz“ auf Seite 895](#)
- [„Sperren bei Abfragen“ auf Seite 915](#)



## Praktische Einführung: Einrichten des Szenarios für die praktischen Einführungen zur Isolationsstufe

Alle praktischen Einführungen zur Isolationsstufe verwenden erfundene Szenarios, bei denen ein Sales Manager und ein Accountant dieselben Daten gleichzeitig ändern. Richten Sie Ihre Datenbank für die praktische Einführung zu Isolationsstufen ein, indem Sie zwei Interactive SQL-Fenster öffnen, um als Sales Manager und Accountant handeln zu können.

### Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

### Aufgabe

1. Starten Sie Interactive SQL. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Interactive SQL**.
2. Im Fenster **Verbinden** stellen Sie als "Sales Manager" eine Verbindung mit der SQL Anywhere-Beispieldatenbank her.
  - a. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Mit ODBC-Datenquelle verbinden**.
  - b. Klicken Sie auf **ODBC-Datenquellenname** und geben Sie im Feld darunter **SQL Anywhere 16 Demo** ein.
  - c. Klicken Sie auf die Registerkarte **Erweitert** und geben Sie **Sales Manager** in das Feld **ConnectionName** ein.
  - d. Klicken Sie auf **Verbinden**.
3. Starten Sie eine zweite Instanz von Interactive SQL.
4. Im Fenster **Verbinden** stellen Sie als Accountant eine Verbindung mit der SQL Anywhere-Beispieldatenbank her.
  - a. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Mit ODBC-Datenquelle verbinden**.
  - b. Klicken Sie auf **ODBC-Datenquellenname** und geben Sie im Feld darunter **SQL Anywhere 16 Demo** ein.
  - c. Klicken Sie auf die Registerkarte **Erweitert** und geben Sie **Accountant** in das Feld **ConnectionName** ein.
  - d. Klicken Sie auf **Verbinden**.

### Ergebnisse

Sie sind mit der Beispieldatenbank als Sales Manager und Accountant verbunden.

### Nächste Schritte

Arbeiten Sie eine der praktischen Einführungen zur Isolationsstufe durch.

### Siehe auch

- [„Praktische Einführung: Einführung in Dirty Reads“ auf Seite 928](#)
- [„Praktische Einführung: Einführung in nicht-wiederholbare Lesevorgänge“ auf Seite 932](#)
- [„Praktische Einführung: Einführung in Phantomzeilen“ auf Seite 938](#)
- [„Praktische Einführung: Einführung in Phantomsperren“ auf Seite 945](#)

## Praktische Einführung: Einführung in Dirty Reads

Die folgende praktische Einführung zeigt den Typ von Inkonsistenz, der auftreten kann, wenn mehrere Transaktionen gleichzeitig ausgeführt werden: Dirty Reads. Bei diesem Szenario greifen zwei Angestellte einer kleinen Handelsfirma gleichzeitig auf die Datenbank der Firma zu. Bei der ersten Person handelt es sich um den Sales Manager (Verkaufsleiter) der Firma, die zweite Person ist der Accountant (Buchhalter).

Der "Sales Manager" möchte den Preis für ein von der Firma verkauftes T-Shirtmodell um \$0.95 erhöhen, hat aber Schwierigkeiten mit der Syntax der SQL-Sprache. Was der Sales Manager nicht weiß, ist, dass der Accountant gleichzeitig versucht, den Einzelhandelswert des gegenwärtigen Inventars für einen Bericht zu berechnen, den er bei der nächsten Besprechung der Geschäftsleitung vorlegen möchte.

#### Hinweis

Damit diese praktische Einführung funktioniert, darf die Option **Datenbanksperren automatisch freigeben** in Interactive SQL nicht aktiviert sein. Sie können die Einstellung dieser Option überprüfen, indem Sie auf **Extras » Optionen** klicken und anschließend im linken Fensterausschnitt auf **SQL Anywhere**.

### Privilegien

Sie müssen die Systemprivilegien SELECT ANY TABLE, UPDATE ANY TABLE und SET ANY SYSTEM OPTION haben.

Für diese praktische Einführung wird davon ausgegangen, dass Sie mit der Beispieldatenbank als Sales Manager und Accountant verbunden sind. Siehe [„Praktische Einführung: Einrichten des Szenarios für die praktischen Einführungen zur Isolationsstufe“ auf Seite 927](#).

### Siehe auch

- [„Isolationsstufen und Konsistenz“ auf Seite 885](#)
- [Einführung in Snapshot-Transaktionen auf Seite 890](#)
- [„Typische Arten von Inkonsistenz“ auf Seite 895](#)
- [„Sperren bei Abfragen“ auf Seite 915](#)

## Lektion 1: Dirty Read erstellen

Der Accountant führt eine Berechnung aus, während der Sales Manager dabei ist, einen Preis heraufzusetzen. Die Berechnung des Accountants beruhte auf fehlerhaften Daten, die der Sales Manager eingegeben hatte und gerade korrigieren wollte.

## Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Einführung in Dirty Reads](#)“.

## Aufgabe

1. Führen Sie als Sales Manager die folgenden Anweisungen aus, um den Preis aller T-Shirts um 0,95 US-Dollar zu erhöhen:

```
UPDATE GROUPO.Products
 SET UnitPrice = UnitPrice + 95
 WHERE Name = 'Tee Shirt';
SELECT ID, Name, UnitPrice
 FROM GROUPO.Products;
```

Die folgende Ergebnismenge wird zurückgegeben:

| ID  | Name         | UnitPrice |
|-----|--------------|-----------|
| 300 | Tee Shirt    | 104.00    |
| 301 | Tee Shirt    | 109.00    |
| 302 | Tee Shirt    | 109.00    |
| 400 | Baseball Cap | 9.00      |
| ... | ...          | ...       |

Der Sales Manager stellt sofort fest, dass 0,95 statt 95 einzugeben waren, aber bevor er seinen Fehler korrigieren kann, greift der Accountant von einem anderen Büro aus auf die Datenbank zu.

2. Der Accountant der Firma ist besorgt, dass zuviel Geld im Lager steckt. Führen Sie als Accountant die folgende Anweisung aus, um den Gesamt-Einzelhandelswert des Lagerbestandes zu berechnen:

```
SELECT SUM(Quantity * UnitPrice)
 AS Inventory
 FROM GROUPO.Products;
```

Das folgende Ergebnis wird zurückgegeben:

| Inventory |
|-----------|
| 21453.00  |

Leider ist diese Berechnung falsch. Der "Sales Manager" hat irrtümlich den Preis für die T-Shirts um \$ 95 erhöht und das Ergebnis spiegelt diesen fehlerhaften Preis wider. Dieser Fehler zeigt eine typische Art von Inkonsistenz auf, die als **Dirty Read** bekannt ist. Sie haben als Accountant auf Daten zugegriffen, die der Sales Manager eingegeben, aber noch nicht festgeschrieben hat.

3. Korrigieren Sie als Sales Manager Ihren Fehler, indem Sie die ersten Änderungen zurücksetzen und die richtige UPDATE-Anweisung eingeben. Überprüfen Sie, ob die neuen Werte auch richtig sind.

```
ROLLBACK;
UPDATE GROUPO.Products
SET UnitPrice = UnitPrice + 0.95
WHERE NAME = 'Tee Shirt';
```

Die folgende Ergebnismenge wird zurückgegeben:

| ID  | Name         | UnitPrice |
|-----|--------------|-----------|
| 300 | Tee Shirt    | 9.95      |
| 301 | Tee Shirt    | 14.95     |
| 302 | Tee Shirt    | 14.95     |
| 400 | Baseball Cap | 9.00      |
| ... | ...          | ...       |

4. Der Accountant weiß nicht, dass der von ihm berechnete Betrag nicht stimmt. Sie können den richtigen Wert sehen, wenn Sie die SELECT-Anweisung im Fenster des Accountants nochmals ausführen.

```
SELECT SUM(Quantity * UnitPrice)
AS Inventory
FROM GROUPO.Products;
```

| Inventory |
|-----------|
| 6687.15   |

5. Beenden Sie die Transaktion im Sales Manager-Fenster. Der Sales Manager würde eine COMMIT-Anweisung eingeben, um die Änderungen festzuschreiben, aber Sie sollten stattdessen eine ROLLBACK-Anweisung ausführen, um zu vermeiden, dass die lokale Kopie der SQL Anywhere-Beispieldatenbank geändert wird.

```
ROLLBACK;
```

### Ergebnisse

Der Accountant erhält unwissentlich fehlerhafte Informationen von der Datenbank, da der Datenbankserver sowohl die Arbeit des Sales Managers, als auch die des Accountants gleichzeitig bearbeitet.

### Nächste Schritte

Gehen Sie weiter zu [„Lektion 2: Dirty Reads mithilfe der Snapshot-Isolation vermeiden“ auf Seite 931](#).

## Lektion 2: Dirty Reads mithilfe der Snapshot-Isolation vermeiden

Die Snapshot-Isolation verhindert Dirty Reads, indem andere Datenbankverbindungen bei Abfragen nur festgeschriebene Daten lesen können. Der Accountant kann die Snapshot-Isolation verwenden, um sicherzustellen, dass seine Abfrage nicht durch nicht festgeschriebene Daten beeinträchtigt werden.

### Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe „[Lektion 1: Dirty Read erstellen](#)“ auf Seite 928.

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Einführung in Dirty Reads](#)“.

### Aufgabe

1. Führen Sie als Sales Manager die folgende Anweisung aus, um die Snapshot-Isolation für die Datenbank zu aktivieren:

```
SET OPTION PUBLIC.allow_snapshot_isolation = 'ON';
```

2. Erhöhen Sie als Sales Manager die Preise aller T-Shirts um \$0.95:
  - a. Führen Sie die folgende Anweisung aus, um den Preis zu aktualisieren:

```
UPDATE GROUP0.Products
SET UnitPrice = UnitPrice + 0.95
WHERE Name = 'Tee Shirt';
```

- b. Berechnen Sie den gesamten Einzelhandelswert des Lagerbestandes, indem Sie den neuen T-Shirt-Preis für den Sales Manager verwenden:

```
SELECT SUM(Quantity * UnitPrice)
AS Inventory
FROM GROUP0.Products;
```

Das folgende Ergebnis wird zurückgegeben:

| Inventory |
|-----------|
| 6687.15   |

3. Führen Sie als Accountant die folgenden Anweisungen aus, um den gesamten Einzelhandelswert des Lagerbestandes zu berechnen: Da diese Transaktion die Snapshot-Isolationsstufe verwendet, wird das Ergebnis nur für die Daten berechnet, die in der Datenbank festgeschrieben wurden.

```
SET OPTION isolation_level = 'Snapshot';
SELECT SUM(Quantity * UnitPrice)
AS Inventory
FROM GROUP0.Products;
```

Das folgende Ergebnis wird zurückgegeben:

| Inventory |
|-----------|
| 6538.00   |

- Schreiben Sie als "Sales Manager" Ihre Änderungen der Datenbank fest, indem Sie die folgende Anweisung ausführen:

```
COMMIT;
```

- Führen Sie als Accountant die folgenden Anweisungen aus, um den aktualisierten Einzelhandelswert des aktuellen Lagerbestandes anzuzeigen:

```
COMMIT;
SELECT SUM(Quantity * UnitPrice)
 AS Inventory
 FROM GROUPO.Products;
```

Das folgende Ergebnis wird zurückgegeben:

| Inventory |
|-----------|
| 6687.15   |

Da der Snapshot für die Transaktion des Accountant mit dem ersten Lesevorgang begann, müssen Sie eine COMMIT-Anweisung ausführen, um die Transaktion zu beenden und es dem Accountant zu ermöglichen, die Änderungen zu sehen, die nach dem Beginn der Snapshot-Transaktion für die Daten durchgeführt wurden.

- Führen Sie als "Sales Manager" die folgende Anweisung aus, um die Änderungen des T-Shirt-Preises rückgängig zu machen und die SQL Anywhere-Beispieldatenbank wieder in ihren Originalzustand zu versetzen:

```
UPDATE GROUPO.Products
SET UnitPrice = UnitPrice - 0.95
WHERE Name = 'Tee Shirt';
COMMIT;
```

### Ergebnisse

Der Accountant hat Dirty Reads erfolgreich vermieden, indem er die Snapshot-Isolation aktiviert hat.

### Nächste Schritte

(Optional) Setzen Sie die Beispieldatenbank (*demo.db*) in ihren ursprünglichen Zustand zurück. Siehe „[Neuerstellung der Beispieldatenbank \(demo.db\)](#)“ [*SQL Anywhere 16 - Einführung*].

## Praktische Einführung: Einführung in nicht-wiederholbare Lesevorgänge

Die folgende praktische Einführung zeigt den Typ von Inkonsistenz, der auftreten kann, wenn mehrere Transaktionen gleichzeitig ausgeführt werden: nicht wiederholbarer Lesevorgang. Bei diesem Szenario

greifen zwei Angestellte einer kleinen Handelsfirma gleichzeitig auf die Datenbank der Firma zu. Bei der ersten Person handelt es sich um den Sales Manager (Verkaufsleiter) der Firma, die zweite Person ist der Accountant (Buchhalter).

Der "Sales Manager" möchte für Plastik-Schirmmützen einen neuen Preis anbieten. Der Accountant will die Preise für einige Waren auf einer kürzlich getätigten Bestellung überprüfen.

Dieses Beispiel beginnt mit beiden Verbindungen auf Isolationsstufe 1 und nicht auf Isolationsstufe 0, die die Standardisolation für die Beispieldatenbank von SQL Anywhere ist. Wenn Sie die Isolationsstufe auf 1 setzen, wird verhindert, dass Dirty Reads auftreten.

#### **Hinweis**

Damit diese praktische Einführung funktioniert, darf die Option **Datenbanksperren automatisch freigeben** in Interactive SQL nicht aktiviert sein. Sie können die Einstellung dieser Option überprüfen, indem Sie auf **Extras » Optionen** klicken und anschließend im linken Fensterausschnitt auf **SQL Anywhere**.

### **Privilegien**

Sie müssen die Systemprivilegien SELECT ANY TABLE, UPDATE ANY TABLE und SET ANY SYSTEM OPTION haben.

Für diese praktische Einführung wird davon ausgegangen, dass Sie mit der Beispieldatenbank als Sales Manager und Accountant verbunden sind. Siehe [„Praktische Einführung: Einrichten des Szenarios für die praktischen Einführungen zur Isolationsstufe“](#) auf Seite 927.

### **Siehe auch**

- [„Isolationsstufen und Konsistenz“](#) auf Seite 885
- [Einführung in Snapshot-Transaktionen](#) auf Seite 890
- [„Typische Arten von Inkonsistenz“](#) auf Seite 895
- [„Sperren bei Abfragen“](#) auf Seite 915

## **Lektion 1: Nicht wiederholbaren Lesevorgang erstellen**

Erstellen Sie einen nicht wiederholbaren Lesevorgang, bei dem der Accountant versucht, eine Zeile zu lesen, die vom Sales Manager geändert wird, und zwei verschiedene Ergebnisse während derselben Transaktion erhält.

### **Voraussetzungen**

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: [„Praktische Einführung: Einführung in nicht-wiederholbare Lesevorgänge“](#).

### **Aufgabe**

1. Setzen Sie die Isolationsstufe für die Verbindung des "Accountants" auf 1, indem Sie die folgende Anweisung ausführen:

```
SET TEMPORARY OPTION isolation_level = 1;
```

2. Setzen Sie die Isolationsstufe im Fenster des Sales Managers auf 1, indem Sie die folgende Anweisung ausführen:

```
SET TEMPORARY OPTION isolation_level = 1;
```

3. Führen Sie als Accountant die folgende Anweisung aus, um die Preise der Schirmmützen aufzulisten:

```
SELECT ID, Name, UnitPrice
FROM GROUPO.Products;
```

| ID  | Name         | UnitPrice |
|-----|--------------|-----------|
| 300 | Tee Shirt    | 9.00      |
| 301 | Tee Shirt    | 14.00     |
| 302 | Tee Shirt    | 14.00     |
| 400 | Baseball Cap | 9.00      |
| 401 | Baseball Cap | 10.00     |
| 500 | Visor        | 7.00      |
| 501 | Visor        | 7.00      |
| ... | ...          | ...       |

4. Als Sales Manager führen Sie die folgenden Anweisungen aus, um einen neuen Verkaufspreis für die Plastischirmmütze einzuführen:

```
SELECT ID, Name, UnitPrice FROM GROUPO.Products
WHERE Name = 'Visor';
UPDATE GROUPO.Products
SET UnitPrice = 5.95 WHERE ID = 501;
COMMIT;
SELECT ID, Name, UnitPrice FROM GROUPO.Products
WHERE Name = 'Visor';
```

| ID  | Name  | UnitPrice |
|-----|-------|-----------|
| 500 | Visor | 7.00      |
| 501 | Visor | 5.95      |

5. Vergleichen Sie den Preis für die Schirmmützen (Visor) im Fenster des Sales Managers mit dem Preis für die gleichen Schirmmützen im Fenster des Accountants. Als Accountant führen Sie die SELECT-Anweisung erneut aus und sehen den neuen Verkaufspreis des Sales Managers.

```
SELECT ID, Name, UnitPrice
FROM GROUPO.Products;
```



| ID  | Name         | UnitPrice |
|-----|--------------|-----------|
| 300 | Tee Shirt    | 9.00      |
| 301 | Tee Shirt    | 14.00     |
| 302 | Tee Shirt    | 14.00     |
| 400 | Baseball Cap | 9.00      |
| 401 | Baseball Cap | 10.00     |
| 500 | Visor        | 7.00      |
| 501 | Visor        | 5.95      |
| ... | ...          | ...       |

Diese Inkonsistenz wird **nicht-wiederholbarer Lesevorgang** genannt, da der Accountant nach dem zweiten Ausführen der gleichen SELECT-Anweisung in der *gleichen Transaktion* nicht mehr dasselbe Ergebnis erhält.

Hätte der Accountant seine Transaktion beispielsweise vor der erneuten Verwendung von SELECT mit einer COMMIT- oder ROLLBACK-Anweisung abgeschlossen, wäre die Situation anders. Die Datenbank steht für die gleichzeitige Verwendung mehrerer Benutzer zur Verfügung, und es ist absolut zulässig, dass jemand die Werte vor oder nach der Transaktion des Accountants verändert. Die geänderten Ergebnisse sind nur deshalb inkonsistent, da die Änderungen während der Transaktion geschahen. Ein solches Ereignis macht den Ablauf unserialisierbar.

- Der Accountant bemerkt dieses Verhalten und beschließt, dass er in Zukunft nicht mehr will, dass sich die Preise ändern, während er sie sich ansieht. Nicht wiederholbare Lesevorgänge werden auf Isolationsstufe 2 eliminiert. Führen Sie als Accountant folgende Anweisungen aus:

```
SET TEMPORARY OPTION isolation_level = 2;
SELECT ID, Name, UnitPrice
FROM GROUP0.Products;
```

- Der Sales Manager entscheidet, dass es besser ist, mit dem Abverkauf der Schirmmützen bis zur nächsten Woche zu warten, damit er den niedrigeren Preis nicht für eine große Bestellung, die er für den nächsten Tag erwartet, hergeben muss. Versuchen Sie als Sales Manager, die folgenden Anweisungen auszuführen. Die Ausführung der Anweisung beginnt und plötzlich scheint das Fenster einzufrieren.

```
UPDATE GROUP0.Products
SET UnitPrice = 7.00
WHERE ID = 501;
```

Der Datenbankserver muss auf Isolationsstufe 2 wiederholbare Lesevorgänge garantieren. Da der Accountant Isolationsstufe 2 verwendet, setzt der Datenbankserver eine Lesesperre auf jede Zeile der Tabelle "Products", die vom Accountant gelesen wird. Wenn der "Sales Manager" versucht, den Preis zurück zu ändern, muss seine Transaktion eine Schreibsperre für die Zeile mit den Schirmmützen in

der Tabelle "Products" setzen. Da Schreibsperren Exklusivsperren sind, muss seine Transaktion warten, bis die Transaktion des "Accountants" die Lesesperre freigibt.

- Der Accountant ist mit der Durchsicht der Preise fertig. Er möchte nicht riskieren, die Datenbank unbeabsichtigt zu ändern, also beendet er seine Transaktion mit einem ROLLBACK-Befehl.

**ROLLBACK ;**

Wenn der Datenbankserver diese Anweisung ausführt, wird die Transaktion des "Sales Manager" abgeschlossen.

| ID  | Name  | UnitPrice |
|-----|-------|-----------|
| 500 | Visor | 7.00      |
| 501 | Visor | 7.00      |

- Der Sales Manager kann nun seine Transaktion abschließen. Er will seine Änderung festschreiben und den ursprünglichen Preis wiederherstellen:

**COMMIT ;**

### Ergebnisse

Der Accountant erhält unterschiedliche Ergebnisse in derselben Transaktion. Er aktiviert daher die Snapshot-Isolation Stufe 2 zur Vermeidung von nicht wiederholbaren Lesevorgängen. Die Änderung des Accountants in der Datenbank hindert den Sales Manager, daran, Änderungen an der Datenbank durchzuführen.

Als Sie die Isolation des Accountants von Stufe 1 auf Stufe 2 erhöht haben, hat der Datenbankserver Lesesperren verwendet, wo es vorher keine gab. Ab da setzte er eine Lesesperre für seine Transaktion, und zwar für jede Zeile, die der Accountant auswählte.

In der vorigen praktischen Einführung für das Fenster des "Sales Managers" während der Ausführung der UPDATE-Anweisung ein. Der Datenbankserver begann die Anweisung auszuführen, als er feststellte, dass die Transaktion des "Accountants" eine Lesesperre für die Zeile gesetzt hatte, die der Sales Manager ändern musste. An diesem Punkt setzte der Datenbankserver einfach mit der Ausführung der Aktualisierung aus. Sobald der Accountant seine Transaktion mit dem ROLLBACK-Befehl abgeschlossen hatte, gab der Datenbankserver seine Sperren automatisch frei. Da es keine weiteren Hindernisse mehr gab, beendete der Datenbankserver die Ausführung des UPDATE-Befehls des Sales Managers.

### Nächste Schritte

Gehen Sie weiter zu „[Lektion 2: Nicht wiederholbare Lesevorgänge mithilfe der Snapshot-Isolation vermeiden](#)“ auf Seite 937.

**Siehe auch**

- „Lektion 2: Nicht wiederholbare Lesevorgänge mithilfe der Snapshot-Isolation vermeiden“ auf Seite 937
- „Die Blocking-Option“ auf Seite 902

## Lektion 2: Nicht wiederholbare Lesevorgänge mithilfe der Snapshot-Isolation vermeiden

Sie können die Snapshot-Isolation auch verwenden, um das Vermeiden von Blockierungen zu unterstützen. Da Transaktionen, welche die Snapshot-Isolation verwenden, nur festgeschriebene Daten erkennen, wird die Transaktion des "Sales Managers" durch die Transaktion des Accountant nicht blockiert.

**Voraussetzungen**

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe „Lektion 1: Nicht wiederholbaren Lesevorgang erstellen“ auf Seite 933.

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „Praktische Einführung: Einführung in nicht-wiederholbare Lesevorgänge“.

**Aufgabe**

1. Führen Sie als Accountant die folgenden Anweisungen aus, um die Snapshot-Isolation für die Datenbank zu aktivieren und festzulegen, welche Snapshot-Isolationsstufe benutzt wird:

```
SET OPTION PUBLIC.allow_snapshot_isolation = 'On';
SET TEMPORARY OPTION isolation_level = 'snapshot';
```

2. Führen Sie als Accountant die folgende Anweisung aus, um die Preise der Schirmmützen aufzulisten:

```
SELECT ID, Name, UnitPrice
FROM GROUP0.Products
ORDER BY ID;
```

| ID  | Name         | UnitPrice |
|-----|--------------|-----------|
| 300 | Tee Shirt    | 9.00      |
| 301 | Tee Shirt    | 14.00     |
| 302 | Tee Shirt    | 14.00     |
| 400 | Baseball Cap | 9.00      |
| 401 | Baseball Cap | 10.00     |
| 500 | Visor        | 7.00      |

| ID  | Name  | UnitPrice |
|-----|-------|-----------|
| 501 | Visor | 7.00      |
| ... | ...   | ...       |

3. Als Sales Manager führen Sie die folgenden Anweisungen aus, um einen neuen Verkaufspreis für die Plastischirmmütze einzuführen:

```
UPDATE GROUPO.Products
SET UnitPrice = 5.95 WHERE ID = 501;
COMMIT;
SELECT ID, Name, UnitPrice FROM GROUPO.Products
WHERE Name = 'Visor';
```

4. Der Accountant führt seine Abfrage erneut aus und sieht die Preisänderung nicht, da die Daten, die zum Zeitpunkt des ersten Lesevorgangs festgeschrieben waren, für die Transaktion benutzt werden.

```
SELECT ID, Name, UnitPrice
FROM GROUPO.Products;
```

5. Setzen Sie als Sales Manager den Preis für die Plastik-Schirmmützen wieder auf den Originalpreis zurück:

```
UPDATE GROUPO.Products
SET UnitPrice = 7.00
WHERE ID = 501;
COMMIT;
```

Der Datenbankserver setzt keine Lesesperren auf die Zeilen in der Tabelle "Products", die der Accountant liest, da der Accountant einen Snapshot der festgeschriebenen Daten sieht, der angefertigt wurde, bevor der Sales Manager Änderungen an der Tabelle "Products" durchgeführt hat.

6. Der Accountant ist mit der Durchsicht der Preise fertig. Er möchte nicht riskieren, die Datenbank unbeabsichtigt zu ändern, also beendet er seine Transaktion mit einem ROLLBACK-Befehl.

```
ROLLBACK;
```

### Ergebnisse

Sie haben erfolgreich versucht, nicht wiederholbare Lesevorgänge durch die Snapshot-Isolation zu vermeiden.

## Praktische Einführung: Einführung in Phantomzeilen

Die folgende praktische Einführung zeigt den Typ von Inkonsistenz, der auftreten kann, wenn mehrere Transaktionen gleichzeitig ausgeführt werden: Phantomzeile. Bei diesem Szenario greifen zwei Angestellte einer kleinen Handelsfirma gleichzeitig auf die Datenbank der Firma zu. Bei der ersten Person handelt es sich um den Sales Manager (Verkaufsleiter) der Firma, die zweite Person ist der Accountant (Buchhalter).

Der Sales Manager möchte neue Abteilungen für Auslandsumsätze und Großkunden erstellen. Der Accountant will alle Abteilungen überprüfen, die im Unternehmen existieren.

Dieses Beispiel beginnt mit beiden Verbindungen auf Isolationsstufe 2 und nicht auf Isolationsstufe 0, die die Standardisolation für die Beispieldatenbank von SQL Anywhere ist. Wenn Sie die Isolationsstufe auf 2 setzen, wird verhindert, dass Dirty Reads oder nicht wiederholbare Lesevorgänge auftreten.

#### Hinweis

Damit diese praktische Einführung funktioniert, darf die Option **Datenbanksperren automatisch freigeben** in Interactive SQL nicht aktiviert sein. Sie können die Einstellung dieser Option überprüfen, indem Sie auf **Extras » Optionen** klicken und anschließend im linken Fensterausschnitt auf **SQL Anywhere**.

### Privilegien

Sie müssen die Systemprivilegien SELECT ANY TABLE, INSERT ANY TABLE, DELETE ANY TABLE und SET ANY SYSTEM OPTION haben.

Für diese praktische Einführung wird davon ausgegangen, dass Sie mit der Beispieldatenbank als Sales Manager und Accountant verbunden sind. Siehe [„Praktische Einführung: Einrichten des Szenarios für die praktischen Einführungen zur Isolationsstufe“](#) auf Seite 927.

### Siehe auch

- [„Isolationsstufen und Konsistenz“](#) auf Seite 885
- [Einführung in Snapshot-Transaktionen](#) auf Seite 890
- [„Typische Arten von Inkonsistenz“](#) auf Seite 895
- [„Sperren bei Abfragen“](#) auf Seite 915

## Lektion 1: Phantomzeilen erstellen

Erstellen Sie eine Phantomzeile, indem der Sales Manager eine Zeile einfügt, während der Accountant benachbarte Zeilen liest. Diese Aktion führt dazu, dass die neue Zeile als Phantom erscheint.

### Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: [„Praktische Einführung: Einführung in Phantomzeilen“](#).

### Aufgabe

1. Setzen Sie die Isolationsstufe im Fenster des Sales Managers und des Accountants auf 2, indem Sie die folgende Anweisung in jedem Fenster ausführen:

```
SET TEMPORARY OPTION isolation_level = 2;
```

2. Führen Sie als Accountant die folgende Anweisung aus, um alle Abteilungen aufzulisten:

```
SELECT * FROM GROUP0.Departments
ORDER BY DepartmentID;
```

| DepartmentID | DepartmentName | DepartmentHeadID |
|--------------|----------------|------------------|
| 100          | R & D          | 501              |
| 200          | Sales          | 902              |
| 300          | Finance        | 1293             |
| 400          | Marketing      | 1576             |
| 500          | Shipping       | 703              |

3. Der Sales Manager beschließt, eine neue Abteilung aufzubauen, die sich auf den ausländischen Markt konzentriert. Philip Chin, der die Mitarbeiternummer (EmployeeID) 129 hat, leitet die neue Abteilung. Führen Sie als Sales Manager die folgende Anweisung aus, um einen neuen Eintrag für die neue Abteilung zu erstellen, der als neue Zeile am Ende der Tabelle im Sales Manager-Fenster erscheint:

```
INSERT INTO GROUPO.Departments
(DepartmentID, DepartmentName, DepartmentHeadID)
VALUES(600, 'Foreign Sales', 129);
COMMIT;
```

4. Führen Sie als Sales Manager die folgende Anweisung aus, um alle Abteilungen aufzulisten:

```
SELECT * FROM GROUPO.Departments
ORDER BY DepartmentID;
```

| DepartmentID | DepartmentName | DepartmentHeadID |
|--------------|----------------|------------------|
| 100          | R & D          | 501              |
| 200          | Sales          | 902              |
| 300          | Finance        | 1293             |
| 400          | Marketing      | 1576             |
| 500          | Shipping       | 703              |
| 600          | Foreign Sales  | 129              |

5. Der Accountant weiß aber nichts von der neuen Abteilung. Bei Isolationsstufe 2 bringt der Datenbankserver Sperren an, um zu gewährleisten, dass sich keine Zeile verändert, bringt aber keine Sperren an, die verhindern, dass andere Transaktionen neue Zeilen einfügen.

Der Accountant findet die neue Zeile nur, wenn er noch einmal die SELECT-Anweisung ausführt. Führen Sie als Accountant erneut die SELECT-Anweisung aus, um die neue Zeile, die der Tabelle hinzugefügt wurde, zu sehen.

```
SELECT * FROM GROUPO.Departments
ORDER BY DepartmentID;
```

| DepartmentID | DepartmentName | DepartmentHeadID |
|--------------|----------------|------------------|
| 100          | R & D          | 501              |
| 200          | Sales          | 902              |
| 300          | Finance        | 1293             |
| 400          | Marketing      | 1576             |
| 500          | Shipping       | 703              |
| 600          | Foreign Sales  | 129              |

Die neue Zeile wird als **Phantomzeile** bezeichnet, da sie vom Blickwinkel des "Accountants" aus wie eine Erscheinung aus dem Nichts aufgetaucht ist. Der Accountant ist auf Isolationsstufe 2 verbunden. Auf dieser Stufe setzt der Datenbankserver nur Sperren für Zeilen, die er verwendet. Die anderen Zeilen bleiben unberührt. Daher hält den "Sales Manager" nichts davon ab, eine neue Zeile einzufügen.

6. Der Accountant möchte in Zukunft solche Überraschungen vermeiden und erhöht daher die Isolationsstufe seiner gegenwärtigen Transaktion auf 3. Führen Sie als Accountant die folgenden Anweisungen aus:

```
SET TEMPORARY OPTION isolation_level = 3;
SELECT * FROM GROUPO.Departments
ORDER BY DepartmentID;
```

7. Der Sales Manager möchte gerne eine zweite Abteilung hinzufügen, die sich auf Verkaufsiniziativen für Großkunden konzentriert. Führen Sie als Sales Manager die folgende Anweisung aus:

```
INSERT INTO GROUPO.Departments
(DepartmentID, DepartmentName, DepartmentHeadID)
VALUES(700, 'Major Account Sales', 902);
```

Während der Ausführung wird das Fenster des Sales Managers angehalten, da die Sperren des Accountants die Anweisung blockieren. Klicken Sie in der Symbolleiste auf **Stopp**, um diesen Eintrag zu unterbrechen.

Als der Accountant die Isolationsstufe auf 3 erhöhte und noch einmal alle Zeilen in der Tabelle "Departments" auswählte, brachte der Datenbankserver Anti-Einfügesperren für jede Zeile der Tabelle an. Außerdem setzte er eine zusätzliche Phantomsperr, um Eingaben am Ende der Tabelle zu blockieren. Als der "Sales Manager" versuchte, am Ende der Tabelle eine neue Zeile einzufügen, war es diese letzte Sperre, die seine Anweisung blockierte.

Beachten Sie, dass die Sales Manager-Anweisung blockiert wurde, obwohl noch eine Verbindung auf Isolationsstufe 2 bestand. Der Datenbankserver platziert Anti-Einfügesperren, wie Lesesperren, entsprechend den Anforderungen der Isolationsstufe und den Anweisungen jeder Transaktion. Sobald diese Sperren angebracht sind, müssen sie von allen anderen gleichzeitig ablaufenden Transaktionen respektiert werden.

8. Um zu vermeiden, dass die SQL Anywhere-Beispieldatenbank geändert wird, sollten Sie die unvollständige Transaktion zurücksetzen, welche die Zeile "Major Account Sales" einfügt, und eine zweite Transaktion benutzen, um die Abteilung "Foreign Sales" zu löschen.
  - a. Führen Sie als Accountant die folgende Anweisung aus, um die Isolationsstufe zu senken, sodass der Sales Manager Änderungen in der Datenbank rückgängig machen kann:

```
SET TEMPORARY OPTION isolation_level=3;
```

- b. Führen Sie als Sales Manager die folgenden Anweisungen aus, um die aktuelle Transaktion zurückzusetzen, die gerade eingefügte Zeile zu löschen und diesen Vorgang festzuschreiben.

```
ROLLBACK;
DELETE FROM GROUPO.Departments
WHERE DepartmentID = 600;
COMMIT;
```

### Ergebnisse

Der Accountant erhält unterschiedliche Ergebnisse jedes Mal, wenn die SELECT-Anweisung ausgeführt wird, daher aktiviert er die Snapshot-Isolation Stufe 3 zur Vermeidung von Phantomzeilen. Die Änderung des Accountants in der Datenbank hindert den Sales Manager, daran, Änderungen an der Datenbank durchzuführen.

### Nächste Schritte

Gehen Sie weiter zu [„Lektion 2: Phantomzeilen mithilfe der Snapshot-Isolation vermeiden“](#) auf Seite 942.

### Siehe auch

- [„Lektion 2: Phantomzeilen mithilfe der Snapshot-Isolation vermeiden“](#) auf Seite 942
- [„Funktionsweise von Sperren“](#) auf Seite 906

## Lektion 2: Phantomzeilen mithilfe der Snapshot-Isolation vermeiden

Sie können die Snapshot-Isolationsstufe verwenden, um die Konsistenz auf der gleichen Stufe aufrechtzuerhalten wie auf Isolationsstufe 3, ohne dass es zu Blockierungen kommt. Die Anweisung des Sales Managers wird nicht blockiert und der Accountant sieht keine Phantomzeile.

### Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe [„Lektion 1: Phantomzeilen erstellen“](#) auf Seite 939.

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: [„Praktische Einführung: Einführung in Phantomzeilen“](#).



## Aufgabe

1. Führen Sie als Accountant die folgenden Anweisungen aus, um die Snapshot-Isolation zu aktivieren:

```
SET OPTION PUBLIC. allow_snapshot_isolation = 'On';
SET TEMPORARY OPTION isolation_level = 'snapshot';
```

2. Führen Sie die folgende Anweisung aus, um alle Abteilungen aufzulisten:

```
SELECT * FROM GROUPO.Departments
ORDER BY DepartmentID;
```

| DepartmentID | DepartmentName | DepartmentHeadID |
|--------------|----------------|------------------|
| 100          | R & D          | 501              |
| 200          | Sales          | 902              |
| 300          | Finance        | 1293             |
| 400          | Marketing      | 1576             |
| 500          | Shipping       | 703              |

3. Der Sales Manager beschließt, eine neue Abteilung aufzubauen, die sich auf den ausländischen Markt konzentriert. Philip Chin, der die Mitarbeiternummer (EmployeeID) 129 hat, leitet die neue Abteilung. Führen Sie als Sales Manager die folgende Anweisung aus, um einen neuen Eintrag für die neue Abteilung zu erstellen, der als neue Zeile am Ende der Tabelle im Sales Manager-Fenster erscheint:

```
INSERT INTO GROUPO.Departments
(DepartmentID, DepartmentName, DepartmentHeadID)
VALUES(600, 'Foreign Sales', 129);
COMMIT;
```

4. Führen Sie als Sales Manager die folgende Anweisung aus, um alle Abteilungen aufzulisten:

```
SELECT * FROM GROUPO.Departments
ORDER BY DepartmentID;
```

| DepartmentID | DepartmentName | DepartmentHeadID |
|--------------|----------------|------------------|
| 100          | R & D          | 501              |
| 200          | Sales          | 902              |
| 300          | Finance        | 1293             |
| 400          | Marketing      | 1576             |
| 500          | Shipping       | 703              |
| 600          | Foreign Sales  | 129              |

5. Der Accountant kann seine Abfrage erneut ausführen und sieht die neue Zeile nicht, da die Transaktion nicht festgeschrieben wurde.

```
SELECT * FROM GROUP0.Departments
ORDER BY DepartmentID;
```

| DepartmentID | DepartmentName | DepartmentHeadID |
|--------------|----------------|------------------|
| 100          | R & D          | 501              |
| 200          | Sales          | 902              |
| 300          | Finance        | 1293             |
| 400          | Marketing      | 1576             |
| 500          | Shipping       | 703              |

6. Der Sales Manager möchte gerne eine zweite Abteilung hinzufügen, die sich auf Verkaufshinitiativen für Großkunden konzentriert. Führen Sie als Sales Manager die folgende Anweisung aus:

```
INSERT INTO GROUP0.Departments
(DepartmentID, DepartmentName, DepartmentHeadID)
VALUES(700, 'Major Account Sales', 902);
```

Die Änderung des Sales Managers wird nicht blockiert, da der Accountant die Snapshot-Isolation benutzt.

7. Der Accountant muss seine Snapshot-Transaktion beenden, um die Änderungen zu sehen, die der Sales Manager in der Datenbank festgeschrieben hat.

```
COMMIT;
SELECT * FROM GROUP0.Departments
ORDER BY DepartmentID;
```

Nun sieht der Accountant die Abteilung "Foreign Sales", nicht aber die Abteilung "Major Account Sales".

| DepartmentID | DepartmentName | DepartmentHeadID |
|--------------|----------------|------------------|
| 100          | R & D          | 501              |
| 200          | Sales          | 902              |
| 300          | Finance        | 1293             |
| 400          | Marketing      | 1576             |
| 500          | Shipping       | 703              |
| 600          | Foreign Sales  | 129              |

8. Um zu vermeiden, dass die SQL Anywhere-Beispieldatenbank geändert wird, sollten Sie die unvollständige Transaktion zurücksetzen, welche die Zeile "Major Account Sales" einfügt, und eine zweite Transaktion benutzen, um die Abteilung "Foreign Sales" zu löschen.
  - a. Führen Sie als Sales Manager die folgende Anweisung aus, um die aktuelle Transaktion zurückzusetzen, die gerade eingefügte Zeile zu löschen und diesen Vorgang festzuschreiben.

```
ROLLBACK;
DELETE FROM GROUPO.Departments
WHERE DepartmentID = 600;
COMMIT;
```

## Ergebnisse

Sie haben erfolgreich versucht, Phantomzeilen durch die Snapshot-Isolation zu vermeiden.

## Praktische Einführung: Einführung in Phantomsperren

In dieser praktischen Einführung haben sowohl der Accountant als auch der Sales Manager Aufgaben, welche die Tabellen "SalesOrder" und "SalesOrderItems" betreffen. Der Accountant muss die Anzahl der Provisionsschecks überprüfen, die an die Vertriebsmitarbeiter ausgezahlt wurden, während der Sales Manager feststellt, dass einige Bestellungen fehlen, und sich anschickt, sie hinzuzufügen.

### Voraussetzungen

Sie müssen die Systemprivilegien SELECT ANY TABLE, INSERT ANY TABLE und DELETE ANY TABLE haben.

Für diese praktische Einführung wird davon ausgegangen, dass Sie mit der Beispieldatenbank als Sales Manager und Accountant verbunden sind. Siehe [„Praktische Einführung: Einrichten des Szenarios für die praktischen Einführungen zur Isolationsstufe“](#) auf Seite 927.

#### Hinweis

Damit diese praktische Einführung funktioniert, darf die Option **Datenbanksperren automatisch freigeben** in Interactive SQL nicht aktiviert sein. Sie können die Einstellung dieser Option überprüfen, indem Sie auf **Extras » Optionen** klicken und anschließend im linken Fensterausschnitt auf **SQL Anywhere**.

### Kontext und Bemerkungen

In dieser praktischen Einführung werden Phantomsperren gezeigt. Eine **Phantomsperre** ist eine gemeinsame Sperre, die auf eine indizierte Suchposition gesetzt wird, um Phantomzeilen zu verhindern. Wenn eine Transaktion mit Isolationsstufe 3 Zeilen auswählt, die ein angegebenes Kriterium erfüllen, bringt der Datenbankserver Anti-Einfügesperren an, damit andere Transaktionen keine Zeilen einfügen können, die ebenfalls diesem Kriterium entsprechen. Die Anzahl der für Ihre Transaktion gesetzten Sperren hängt sowohl vom Suchkriterium als auch vom Aufbau der Datenbank ab.

## Aufgabe

1. Setzen Sie die Isolationsstufe im Fenster des Sales Managers und des Accountants auf 2, indem Sie die folgende Anweisung in jedem Fenster ausführen:

```
SET TEMPORARY OPTION isolation_level = 2;
```

2. Die Handelsvertreter erhalten monatlich eine Provision, die als Prozentsatz der Umsätze für den jeweiligen Monat berechnet wird. Der Accountant bereitet die Provisionszahlungen für April 2001 vor. Seine erste Aufgabe besteht darin, den Gesamtumsatz jedes Vertreters für diesen Monat zu berechnen. Die Preise, Bestellinformationen und Daten der Angestellten sind in getrennten Tabellen gespeichert. Verknüpfen Sie diese Tabellen unter Verwendung von Fremdschlüsselbeziehungen, um die notwendigen Teilm Informationen zusammenzusetzen.

Führen Sie als Accountant die folgende Anweisung aus:

```
SELECT EmployeeID, GivenName, Surname,
 SUM(SalesOrderItems.Quantity * UnitPrice)
 AS "April sales"
FROM GROUPO.Employees
 KEY JOIN GROUPO.SalesOrders
 KEY JOIN GROUPO.SalesOrderItems
 KEY JOIN GROUPO.Products
WHERE '2001-04-01' <= OrderDate
 AND OrderDate < '2001-05-01'
GROUP BY EmployeeID, GivenName, Surname
ORDER BY EmployeeID;
```

| EmployeeID | GivenName | Surname   | April sales |
|------------|-----------|-----------|-------------|
| 129        | Philip    | Chin      | 2160.00     |
| 195        | Marc      | Dill      | 2568.00     |
| 299        | Rollin    | Overbey   | 5760.00     |
| 467        | James     | Klobucher | 3228.00     |
| ...        | ...       | ...       | ...         |

3. Der "Sales Manager" bemerkt, dass ein großer Auftrag von Philip Chin nicht in die Datenbank eingegeben wurde. Philip will seine Provision immer pünktlich erhalten, also gibt der "Sales Manager" den fehlenden Auftrag ein, der am 25. April gebucht wurde.

Führen Sie als "Sales Manager" die folgenden Anweisungen aus. Die Bestellung und die Elemente werden in getrennten Tabellen eingegeben, da eine Bestellung viele Elemente umfassen kann. Sie sollten zuerst den Eintrag für die Bestellung durchführen, bevor Sie Elemente hinzufügen. Um die referenzielle Integrität aufrechtzuerhalten, gestattet der Datenbankserver eine Transaktion, mit der Elemente einer Bestellung hinzugefügt werden, nur dann, wenn diese Bestellung bereits vorhanden ist.

```
INSERT into GROUPO.SalesOrders
VALUES (2653, 174, '2001-04-22', 'r1',
 'Central', 129);
INSERT into GROUPO.SalesOrderItems
```

```
VALUES (2653, 1, 601, 100, '2001-04-25');
COMMIT;
```

4. Der Accountant kann nicht wissen, dass der Sales Manager gerade einen neuen Auftrag hinzugefügt hat. Wäre der neue Auftrag früher eingegeben worden, so wäre er in der Berechnung von Philip Chins Aprilumsätzen berücksichtigt.

Berechnen Sie im Fenster des "Accountants" noch einmal den Gesamtumsatz für den Monat April. Verwenden Sie die gleiche Anweisung. Es wird Ihnen auffallen, dass Philip Chins Aprilumsatz plötzlich \$4560,00 beträgt.

| EmployeeID | GivenName | Surname   | April sales |
|------------|-----------|-----------|-------------|
| 129        | Philip    | Chin      | 4560.00     |
| 195        | Marc      | Dill      | 2568.00     |
| 299        | Rollin    | Overbey   | 5760.00     |
| 467        | James     | Klobucher | 3228.00     |
| ...        | ...       | ...       | ...         |

Nehmen wir an, dass der Accountant alle im April gebuchten Aufträge markiert, um anzuzeigen, dass die Provision bezahlt wurde. Der vom "Sales Manager" gerade eingegebene Auftrag wird in der zweiten Suchabfrage gefunden und als bezahlt markiert, obwohl er nicht in Phillips Gesamtumsatz für den Monat April berücksichtigt wurde.

5. Bei Isolationsstufe 3 bringt der Datenbankserver Anti-Einfügesperren an, um zu gewährleisten, dass keine anderen Transaktionen eine Zeile hinzufügen können, die den Such- oder Auswahlkriterien entspricht.

Führen Sie als Sales Managers die folgenden Anweisungen aus, um den neuen Auftrag zu entfernen:

```
DELETE
FROM GROUP0.SalesOrderItems
WHERE ID = 2653;
DELETE
FROM GROUP0.SalesOrders
WHERE ID = 2653;
COMMIT;
```

6. Führen Sie als Accountant die folgenden Anweisungen aus:

```
ROLLBACK;
SET TEMPORARY OPTION isolation_level = 3;
```

7. Führen Sie die folgende Abfrage aus:

```
SELECT EmployeeID, GivenName, Surname,
 SUM(SalesOrderItems.Quantity * UnitPrice)
 AS "April sales"
FROM GROUP0.Employees
KEY JOIN GROUP0.SalesOrders
```

```
KEY JOIN GROUPO.SalesOrderItems
KEY JOIN GROUPO.Products
WHERE '2001-04-01' <= OrderDate
AND OrderDate < '2001-05-01'
GROUP BY EmployeeID, GivenName, Surname;
```

Da Sie die Isolationsstufe auf 3 gesetzt haben, bringt der Datenbankserver automatisch Anti-Einfügesperren an, die gewährleisten, dass der Sales Manager keine April-Aufträge eingeben kann, solange der Accountant seine Transaktion nicht beendet hat.

8. Versuchen Sie als Sales Manager, Philip Chins fehlenden Auftrag einzugeben, indem Sie die folgende Anweisung ausführen.

```
INSERT INTO GROUPO.SalesOrders
VALUES (2653, 174, '2001-04-22',
 'r1','Central', 129);
```

Das Sales Manager-Fenster reagiert nicht mehr und der Vorgang wird nicht abgeschlossen. Klicken Sie in der Symbolleiste auf **Stopp**, um diesen Eintrag zu unterbrechen.

9. Der "Sales Manager" kann den Auftrag zwar nicht im April eingeben, aber der Eintrag im Mai sollte doch möglich sein.

Ändern Sie das Datum in der Anweisung auf den 05. Mai und versuchen Sie es noch einmal.

```
INSERT INTO GROUPO.SalesOrders
VALUES (2653, 174, '2001-05-05', 'r1',
 'Central', 129);
```

Das "Sales Manager"-Fenster reagiert nicht mehr. Klicken Sie in der Symbolleiste auf **Stopp**, um diesen Eintrag zu unterbrechen. Obwohl der Datenbankserver nicht mehr als die notwendigen Sperren zum Vermeiden von Einfügungen anbringt, haben diese Sperren das Potenzial, sich mit vielen Transaktionen zu überlagern.

Der Datenbankserver bringt Sperren in Tabellenindizes an. Er bringt zum Beispiel eine Phantomsperre in einem Index an, damit keine Zeile direkt davor eingefügt werden kann. Ist aber kein entsprechender Index vorhanden, so muss jede Zeile in der Tabelle gesperrt werden. Es gibt Situationen, in denen Anti-Einfügesperren einige Einfügungen in einer Tabelle verhindern, andere aber erlauben.

10. Um zu vermeiden, dass die SQL Anywhere-Beispieldatenbank geändert wird, sollten Sie die Änderungen in der Tabelle SalesOrders zurücksetzen. Im Fenster Sales Manager und Accountant führen Sie die folgende Anweisung aus:

```
ROLLBACK;
```

11. Fahren Sie beide Instanzen von Interactive SQL herunter.

### Ergebnisse

Sie haben die praktische Einführung zum Verständnis von Phantomsperren abgeschlossen.

**Siehe auch**

- „Isolationsstufen und Konsistenz“ auf Seite 885
- Einführung in Snapshot-Transaktionen auf Seite 890
- „Typische Arten von Inkonsistenz“ auf Seite 895
- „Sperrren bei Abfragen“ auf Seite 915

## Primärschlüssel-Generierung und Parallelität

Sie werden auf Situationen stoßen, wo die Datenbank automatisch eine eindeutige Zahl generieren sollte. Wenn Sie zum Beispiel eine Tabelle erstellen, in der alle Verkaufsrechnungen gespeichert werden, ziehen Sie es wahrscheinlich vor, dass die Datenbank automatisch eindeutige Rechnungsnummern vergibt, und nicht das Verkaufspersonal diese aussucht.

**Beispiel**

Man kann zum Beispiel Rechnungsnummern erhalten, indem zur vorherigen Nummer jeweils 1 addiert wird. Diese Methode funktioniert aber nicht, wenn mehr als eine Person Rechnungen in die Datenbank eingibt. Zwei Mitarbeiter könnten die gleiche Rechnungsnummer verwenden.

Das Problem kann auf mehr als eine Art gelöst werden:

- Ordnen Sie jeder Person, die neue Rechnungen eingibt, einen Bereich an Rechnungsnummern zu.

Sie können dieses Schema umsetzen, indem Sie eine Tabelle mit den Spalten Benutzername und Rechnungsnummer erstellen. Die Tabelle hätte für jeden Benutzer, der Rechnungen eingibt, eine Zeile. Jedesmal, wenn der Benutzer eine Rechnung eingibt, wird die Zahl in der Tabelle erhöht und für die neue Rechnung verwendet. Damit alle Tabellen in der Datenbank bearbeitet werden können, sollte eine Tabelle aus drei Spalten bestehen: Tabellename, Benutzername und letzter Schlüsselwert. Sie sollten regelmäßig überprüfen, dass jede Person genug Nummern hat.

- Erstellen Sie eine Tabelle mit den Spalten "table name" und "last key value".

Eine Zeile in dieser Tabelle enthält die letzte verwendete Rechnungsnummer. Die Rechnungsnummer wird jeweils automatisch erhöht, wenn ein Benutzer eine Rechnung hinzufügt, eine neue Verbindung herstellt, die Rechnungsnummer erhöht oder eine Änderung festschreibt. Andere Benutzer können auf neue Rechnungsnummern zugreifen, da die Zeile sofort durch eine separate Transaktion aktualisiert wird.

- Verwenden Sie eine Spalte mit einem Standardwert von NEWID mit dem binären UNIQUEIDENTIFIER-Datentyp, um einen universell eindeutigen Bezeichner (Universally Unique Identifier = UUID) zu generieren.

Sie können Tabellenzeilen mit UUID- und GUID-Werten eindeutig kennzeichnen. Da die auf einem Computer generierten Werte nicht mit den auf einem anderen Computer generierten Werten übereinstimmen, können die UUID- und GUID-Werte als Schlüssel in Replikations- und Synchronisationsumgebungen verwendet werden.

- Verwenden Sie eine Spalte mit einem AUTOINCREMENT-Standardwert. Beispiel:

```
CREATE TABLE Orders (
 OrderID INTEGER NOT NULL DEFAULT AUTOINCREMENT,
```

```
OrderDate DATE,
primary key(OrderID)
);
```

Wenn beim Einfügen in die Tabelle kein Wert für die AUTOINCREMENT-Spalte angegeben ist, wird ein eindeutiger Wert generiert. Ist ein Wert vorgegeben, wird er auch verwendet. Ist der Wert größer als der derzeitige Höchstwert für die Spalte, so wird dieser Wert als Startpunkt für nachfolgende Einfügungen verwendet. Der Wert der letzten eingefügten Zeile in einer AUTOINCREMENT-Spalte steht als globale Variable @@identity zur Verfügung.

**Siehe auch**

- „Der Standardwert NEWID“ auf Seite 858

# Verwendung einer Sequenz zum Generieren von eindeutigen Werten

Sie können mit einer **Sequenz** Werte generieren, die in mehreren Tabellen eindeutig sind oder sich von einer Reihe von natürlichen Zahlen unterscheiden. Eine Sequenz wird mit der CREATE SEQUENCE-Anweisung erstellt. Sequenzwerte werden als BIGINT-Werte zurückgegeben.

Für jede Verbindung wird der jeweils zuletzt angeforderte Werts als aktueller Wert gespeichert.

Beim Erstellen einer Sequenz wird in ihrer Definition die Anzahl der Sequenzwerte festgelegt, die der Datenbankserver im Sequenzcache vorhält. Wenn diese Werte verbraucht sind, wird der Sequenzcache neu gefüllt. Wenn der Datenbankserver ausfällt, können die im Cache gespeicherten Sequenzwerte übersprungen werden.

**Werte in einer Sequenz abrufen**

Mit folgender Anweisung können Sie den aktuellen Wert in der Sequenz abrufen:

```
SELECT sequence-name.CURRVAL;
```

Verwenden Sie die folgende Anweisung zum Abrufen des nächsten Werts in der Sequenz:

```
SELECT sequence-name.NEXTVAL;
```

**Wählen zwischen Sequenzen und AUTOINCREMENT-Werten**

| AUTOINCREMENT-Verhalten                                            | Sequenz-Verhalten                                                                                                                              |
|--------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Für eine einzelne Spalte in einer Tabelle festgelegt               | Als Datenbankobjekt gespeichert und kann überall verwendet werden, wo ein Ausdruck erlaubt ist                                                 |
| Spalte muss Ganzzahl-Datentyp oder numerisch exakter Datentyp sein | Werte können überall referenziert werden, wo ein Ausdruck verwendet werden kann, und müssen nicht dem Standardwert für eine Spalte entsprechen |



| AUTOINCREMENT-Verhalten                                                                                                                                                                                                                                         | Sequenz-Verhalten                                                                                                                                              |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Werte können nur für eine einzelne Spalte in einer Tabelle verwendet werden                                                                                                                                                                                     | Werte können in mehreren Tabellen verwendet werden                                                                                                             |
| Werte sind Teil der Menge von natürlichen Zahlen (1, 2, 3, ...)                                                                                                                                                                                                 | Kann andere Werte als die Menge von natürlichen Zahlen generieren                                                                                              |
| Werte müssen zunehmen                                                                                                                                                                                                                                           | Werte können zunehmen oder abnehmen                                                                                                                            |
| Ein eindeutiger Wert, der um eins größer als der vorherige Höchstwert in der Spalte ist, wird standardmäßig generiert<br><br>Die sa_reset_identity-Systemprozedur kann verwendet werden, um den AUTOINCREMENT-Wert für die nächste einzufügende Zeile zu ändern | Inkrementeinheit kann festgelegt werden                                                                                                                        |
| Wenn der nächste zu generierende Wert den Höchstwert überschreitet, der in der Spalte gespeichert werden kann, wird NULL zurückgegeben                                                                                                                          | Kann wählen, das Generieren von Werte zuzulassen, nachdem der Höchstwert und Mindestwert erreicht wurde, oder einen Fehler zurückzugeben (Angabe von NO CYCLE) |

### Sequenzbeispiel

Nehmen wir eine Sequenz, die zum Generieren von Vorfällennummern für eine Kunden-Hotline verwendet wird. Angenommen, Kunden können bei zwei verschiedenen Arten von Problemen anrufen: bei falschen Rechnungen und nicht erfolgten Lieferungen.

```
CREATE SEQUENCE incidentSequence
 MINVALUE 1000
 MAXVALUE 100000;

CREATE TABLE reportedBillingMistake(
 incidentID INT PRIMARY KEY DEFAULT (incidentSequence.nextval),
 billNumber INT,
 valueOnBill NUMERIC(10,2),
 expectedValue NUMERIC(10,2),
 comments LONG VARCHAR);

CREATE TABLE reportedMissingShipment(
 incidentID INT PRIMARY KEY DEFAULT(incidentSequence.nextval),
 orderNumber INT,
 comments LONG VARCHAR);
```

Die Verwendung von incidentSequence.nextval für die incidentID-Spalten garantiert, dass incidentIDs bei beiden Tabellen eindeutig sind. Es gibt keine Möglichkeit einer Unklarheit, ob es sich um einen Verrechnungs- oder Warenausgangsfehler handelt, wenn ein Kunde mit weiteren Anfragen zurückruft und eine Vorfällennummer angibt.

Um einen Rechnungsfehler einzufügen, sind die folgenden Anweisungen gleichwertig:

```
INSERT INTO reportedBillingMistake VALUES(DEFAULT, 12345, 100.00, 75.00,
'Bad bill');

INSERT INTO reportedBillingMistake
SELECT incidentSequence.nextval, 12345, 100.00, 75.00, 'Bad bill';
```

Um den soeben eingefügten incidentID-Wert zu finden, könnte die Verbindung, die die Einfügung (mit einer der zwei oben genannten Anweisungen) durchgeführt hat, folgende Anweisung ausführen:

```
SELECT incidentSequence.currval;
```

### Siehe auch

- „CREATE SEQUENCE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Der Standardwert "Autoincrement"“ auf Seite 856
- „Der Standardwert GLOBAL AUTOINCREMENT“ auf Seite 856
- Sequenzausdruck-Klausel, SELECT-Anweisung [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Sequenzen erstellen

Sie können Sybase Central verwenden, um eine Sequenz zu erstellen.

### Voraussetzungen

Sie müssen das CREATE ANY SEQUENCE-Systemprivileg oder das CREATE ANY OBJECT-Systemprivileg haben.

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf **Sequenzgeneratoren** und klicken Sie auf **Neu » Sequenzgenerator**.
3. Befolgen Sie die Anweisungen des **Assistenten zur Erstellung eines Sequenzgenerators**.

### Ergebnisse

Die Sequenz wurde erstellt.

### Siehe auch

- „CREATE SEQUENCE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „GRANT USAGE ON SEQUENCE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Verwendung einer Sequenz zum Generieren von eindeutigen Werten“ auf Seite 950
- Wählen zwischen Sequenzen und AUTOINCREMENT-Werten auf Seite 950

## Sequenzen ändern

Sie können Sybase Central verwenden, um eine Sequenz zu ändern.

## Voraussetzungen

Sie müssen Eigentümer der Sequenz sein oder eines der folgenden Privilegien haben:

- ALTER ANY SEQUENCE-Systemprivileg
- ALTER ANY OBJECT-Systemprivileg

## Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie auf einen Sequenzgenerator und klicken Sie dann auf **Eigenschaften**.

Auf der Registerkarte **Allgemein** können Sie die Einstellungen für die Sequenz ändern. Durch Klicken auf **Jetzt neu starten** wird eine ALTER SEQUENCE ... RESTART WITH *n*-Anweisung ausgeführt, wobei *n* dem Wert im Feld **Startwert** entspricht.

## Ergebnisse

Die Änderung tritt sofort in Kraft.

## Siehe auch

- „ALTER SEQUENCE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Sequenzen löschen

Sie können Sybase Central verwenden, um eine Sequenz zu löschen.

## Voraussetzungen

Sie müssen Eigentümer der Sequenz sein oder eines der folgenden Privilegien haben:

- DROP ANY SEQUENCE-Systemprivileg
- DROP ANY OBJECT-Systemprivileg

## Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie auf einen Sequenzgenerator und klicken Sie dann auf **Löschen**.

## Ergebnisse

Die Sequenz wird aus der Datenbank gelöscht. Wenn Sie eine Sequenz löschen, werden alle Synonyme für den Namen der Sequenz vom Datenbankserver automatisch gelöscht.

**Siehe auch**

- „DROP SEQUENCE-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## Datendefinitionsanweisung und Parallelität

Datendefinitionsanweisungen, die eine komplette Tabelle ändern, z.B. CREATE INDEX, ALTER TABLE und TRUNCATE TABLE, werden verhindert, wenn die Tabelle, auf die sich die Anweisung auswirkt, gerade von einer anderen Verbindung benutzt wird. Diese Datendefinitionsanweisungen können sehr zeitintensiv sein und die Datenbank bearbeitet keine Anforderungen, die sich auf die gleiche Tabelle beziehen, während die Anweisung bearbeitet wird.

Die CREATE TABLE-Anweisung verursacht keine Parallelitätskonflikte.

Die Anweisungen GRANT, REVOKE und SET OPTION verursachen ebenfalls keine Parallelitätskonflikte. Diese Anweisungen beeinflussen alle neuen SQL-Anweisungen, die zum Datenbankserver geschickt werden, aber keine bestehenden offenen Anweisungen.

GRANT- und REVOKE-Anweisungen sind nicht gestattet für Benutzer, die mit der Datenbank verbunden sind.

### **Datendefinitionsanweisungen und synchronisierte Datenbanken**

Die Verwendung von Datendefinitionsanweisungen in Datenbanken, die Synchronisation benutzen, erfordert spezielle Sorgfalt. Siehe [MobiLink - Serveradministration](#).

**Siehe auch**

- „Datendefinitionsanweisungen“ [[SQL Remote](#)]

## Zusammenfassung

Transaktionen und Sperren sind fast so wichtig wie die Beziehungen zwischen Tabellen. Die Integrität und Performance einer Datenbank kann durch die richtige Verwendung von Sperren und den sorgfältigen Aufbau von Transaktionen profitieren. Transaktionen und Sperren sind für jede Datenbank unerlässlich, die eine große Anzahl von Anweisungen gleichzeitig ausführt.

Transaktionen fassen SQL-Anweisungen in logische Arbeitseinheiten zusammen. Um Transaktionen abzuschließen, können Sie entweder alle durchgeführten Änderungen zurücksetzen oder die Änderungen festschreiben, um sie dauerhaft zu machen.

Bei einem Systemausfall sind Transaktionen für die Datenwiederherstellung unerlässlich. Sie spielen auch eine wichtige Rolle bei der Überlagerung der Anweisungen von gleichzeitig ablaufenden Transaktionen.

Für eine verbesserte Performance müssen mehrere Transaktionen gleichzeitig ausgeführt werden. Jede Transaktion besteht aus SQL-Anweisungskomponenten. Werden zwei oder mehrere Transaktionen gleichzeitig ausgeführt, muss der Datenbankserver die Ausführung der einzelnen Anweisungen planen. Im Gegensatz zu sequenziell ausgeführten Transaktionen können parallele Transaktionen zu Inkonsistenzen führen.

Vier Typen von Inkonsistenzen werden verwendet, um Isolationsstufen festzulegen:

- **Dirty Reads** Eine Transaktion liest Daten, die von einer anderen Transaktion geändert, aber noch nicht festgeschrieben wurden.
- **Nicht-wiederholbare Lesevorgänge** Eine Transaktion liest die gleiche Zeile zweimal und erhält unterschiedliche Ergebnisse.
- **Phantomzeilen** Eine Transaktion wählt Zeilen mit einem bestimmten Kriterium zweimal aus und findet in der zweiten Ergebnismenge neue Zeilen.
- **Verlorenes Update** Die Änderungen, die von einer Transaktion in einer Zeile durchgeführt wurden, gehen vollständig verloren, da eine andere Transaktion ein auf früheren Daten beruhendes Update speichern kann.

Ein Schema wird als serialisierbar bezeichnet, wenn die Wirkung durch die Ausführung der Anweisungen in Übereinstimmung mit dem Schema die gleiche ist, als wenn jede Transaktion sequenziell ausgeführt worden wäre. Schemata gelten als **korrekt**, wenn sie serialisierbar sind. Ein serialisierbares Schema verursacht keine der vorstehend angeführten Inkonsistenzen.

Sperren steuern die Anzahl und Arten der zulässigen Überlagerungen. SQL Anywhere bietet Ihnen vier Sperrstufen: Isolationsstufen 0, 1, 2 und 3. Bei der höchsten Isolationsstufe, Stufe 3, gewährleistet SQL Anywhere, dass das Schema serialisierbar ist, das heißt, dass das Ergebnis aller ausgeführten Transaktionen gleich ist wie bei einer sequenziellen Ausführung.

Leider können aber von einer Transaktion gesetzte Sperren den Fortschritt anderer Transaktionen behindern. Daher ist es wünschenswert, niedrigere Isolationsstufen zu verwenden, wenn die dadurch möglichen Inkonsistenzen akzeptiert werden können. Eine erhöhte Isolation für eine verbesserte Datenkonsistenz bedeutet häufig eine niedrigere Parallelität und Effizienz der Datenbank bei der Bearbeitung gleichzeitig ablaufender Transaktionen. Sie müssen häufig die Notwendigkeit für Konsistenz mit der Notwendigkeit für Performance abwägen, um die beste Isolationsstufe für jeden Vorgang zu ermitteln.

Widersprüchliche Sperrenanforderungen zwischen verschiedenen Transaktionen können zu Blockierungen oder Deadlocks führen. SQL Anywhere enthält Verfahren für beide Situationen und bietet Ihnen Optionen für deren Steuerung.

Transaktionen mit höheren Isolationsstufen müssen aber nicht *immer* eine negative Auswirkung auf die Parallelität haben. Andere Transaktionen werden nur behindert, wenn sie einen Zugriff auf gesperrte Zeilen verlangen. Sie können die Parallelität durch eine sorgfältige Planung Ihrer Datenbank und Transaktionen verbessern. So können Sie zum Beispiel die Zeit für Sperren verkürzen, indem Sie eine Transaktion in zwei kürzere aufteilen, oder Sie stellen fest, dass durch einen Index Ihre Transaktion auf einer höheren Isolationsstufe mit weniger Sperren auskommt.



---

# LDAP-Benutzerauthentifizierung

LDAP (Lightweight Directory Access Protocol) ist ein Branchenstandard für den Zugriff auf Verzeichnisdienste. Die LDAP-Benutzerauthentifizierung ermöglicht es Clientanwendungen, Benutzer-ID- und Kennwortinformationen an den Datenbankserver zu senden, wo diese von einem LDAP-Server statt über den Katalog authentifiziert werden. Die Authentifizierung durch einen LDAP-Server ermöglicht eine organisationsweite Kennwortverwaltung.

Die LDAP-Benutzerauthentifizierung ist ideal für Organisationen mit vorhandener Computerumgebung, die die Benutzeradministration vereinfachen und zentralisieren möchten, oder für Benutzer in einer neuen Computerumgebung, die Benutzer in unterschiedlichen Anwendungen und Datenbanken verwalten müssen und dies möglichst unkompliziert gestalten möchten.

Die Datenbankoption `login_mode` enthält einen LDAPUA-Modus, der die möglichen Authentifizierungsmechanismen um die LDAP-Benutzerauthentifizierung erweitert. Dank der Unterstützung der LDAP-Benutzerauthentifizierung kann SQL Anywhere in vorhandene, auf LDAP basierende unternehmensweite Verzeichniszugriffs-Frameworks integriert werden.

## Funktionsweise der LDAP-Benutzerauthentifizierung

Jedem Benutzer ist eine Login-Richtlinie zugeordnet. Die Login-Richtlinie kann für die Benutzerauthentifizierung optional ein primäres LDAP-Serverobjekt, ein sekundäres LDAP-Serverobjekt oder beide referenzieren. Sie können mehrere Login-Richtlinien mit verschiedenen Optionen definieren, um mehrere Domänen anzugeben.

Beim Authentifizieren eines Benutzers, dessen Login-Richtlinie ein LDAP-Serverobjekt angibt, werden der **Distinguished Name (DN)** für diesen Benutzer und das dem DN zugeordnete Kennwort verwendet, um eine Verbindung mit einem vertrauenswürdigen LDAP-Verzeichnisserver herzustellen.

Die Clientanwendung kennt in der Regel den LDAP-DN nicht und sendet ihn nicht an den Datenbankserver. Sie kennt aber die Datenbankbenutzer-ID und das Kennwort und sendet diese an den Server. Die Zuordnung von Datenbankbenutzer-ID und DN erfolgt anschließend, indem auf dem LDAP-Server mit einer vordefinierten Suchzeichenfolge nach dem DN einer gegebenen Benutzer-ID gesucht wird. Der DN, der aus einer Suche auf dem LDAP-Server an den Datenbankserver zurückgegeben wurde, wird in der ISYSUSER-Systemtabelle (`user_dn`-Spalte) mit der aktuellen UTC-Zeit (`user_dn_cached_at`-Spalte) gespeichert. Der gespeicherte DN-Wert wird weiter verwendet, bis die zugeordnete Login-Richtlinie mit einer `ALTER LOGIN POLICY...LDAP_REFRESH_DN=NOW`-Anweisung aktualisiert wird.

## LDAP-Serverkonfigurationsobjekte

Die Konfigurationsinformationen für Verbindungen mit einem LDAP-Server werden in einem **LDAP-Serverkonfigurationsobjekt** gespeichert. Jede Zeile in der ISYSLDAPSERVER-Systemtabelle enthält die Einstellungen für ein LDAP-Serverkonfigurationsobjekt. Jedes Konfigurationsobjekt enthält Attribute für Verbindungen, einschließlich einer LDAP-Server-Suchzeichenfolge, die verwendet wird, um den DN für eine bestimmte Benutzer-ID abzurufen. Diese Attribute und der Zustand eines LDAP-Servers können geprüft werden, indem die SYSLDAPSERVER-Ansicht abgefragt wird.

### Hochverfügbarkeit und LDAP-Benutzerauthentifizierung

Sie können Hochverfügbarkeit für einen LDAP-Server erreichen, indem Sie ihn für einen bestimmten Benutzer über die Login-Richtlinie des Benutzers als Primär- oder Sekundärserver kennzeichnen. Falls ein primärer LDAP-Server nicht erreichbar ist (z.B. wegen LDAP-Serverwartung, Netzwerkfehler oder LDAP-Serverausfall), wird der sekundäre LDAP-Server für die Benutzerauthentifizierung verwendet. Die Möglichkeit des Failovers von einem primären auf einen sekundären Server und des Failbacks von einem sekundären auf einen primären Server kann manuell mit SQL-Anweisungen verwaltet oder automatisch vom Datenbankserver ausgeführt werden, wenn dieser erkennt, dass ein Umschalten erforderlich ist.

### Vererbung für LDAP-Login-Richtlinienoptionen

Die Vererbung aus der Standard-Root-Login-Richtlinie erfolgt bei LDAP-Richtlinienoptionen als Gruppe. Wenn beispielsweise die Root-Login-Richtlinie einen primären LDAP-Server und eine Login-Richtlinie `policy_X` definiert, die keine LDAP-Richtlinienoptionen angibt, erben Benutzer, die `policy_X` zugeordnet sind, den primären LDAP-Server und die LDAP-Richtlinienoptionen aus der Root-Richtlinie.

Wenn eine andere Login-Richtlinie, `policy_Y`, nur einen sekundären LDAP-Server definiert, verwenden Benutzer, die `policy_Y` zugeordnet sind, ausschließlich den sekundären LDAP-Server und die dazugehörigen Einstellungen aus dieser Richtlinie. Die LDAPUA-Richtlinienoptionen aus der Root-Richtlinie werden nicht geerbt.

### Siehe auch

- „login\_mode-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Root-Login-Richtlinie“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Login-Richtlinien“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „ALTER LOGIN POLICY-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSLDAPSERVER-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSUSER-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## LDAP-Benutzerauthentifizierungsumgebung erstellen (SQL)

Richten Sie eine LDAP-Benutzerauthentifizierungsumgebung ein, indem Sie Folgendes erstellen: ein LDAP-Serverkonfigurationsobjekt, eine Login-Richtlinie, die den LDAP-Server verwendet, und Benutzer, die sich beim LDAP-Server authentifizieren, indem sie diese Login-Richtlinie verwenden.

### Voraussetzungen

Sie müssen einen LDAP-Server haben.

Sie müssen die folgenden Systemprivilegien haben:

- SET ANY SECURITY OPTION
- MANAGE ANY LDAP SERVER
- MANAGE ANY LOGIN POLICY
- MANAGE ANY USER



## Aufgabe

1. (Optional) Um die TLS-Verschlüsselung für die Kommunikation mit Ihrem LDAP-Server zu aktivieren, identifizieren Sie eine Zertifikatsdatei, die der Server verwenden muss. Die folgende Anweisung ist ein Beispiel dafür, wie die Zertifikatsdatei angegeben wird.

```
SET OPTION PUBLIC.trusted_certificates_file='c:\\certificates\\trusted.txt';
```

Um TLS-Verschlüsselung in den nachfolgenden Schritten zu aktivieren, muss die TLS OFF-Klausel in jedem Beispiel, in dem diese Klausel auftritt, auf TLS ON geändert werden.

2. Führen Sie eine VALIDATE LDAP SERVER-Anweisung aus, um die Verbindung mit einem LDAP-Server zu testen.

Die folgende Anweisung überprüft beispielsweise die Verbindungsattribute eines vorhandenen LDAP-Servers. Der Datenbankserver stellt mithilfe der angegebenen Anmeldeinformationen eine Verbindung mit dem LDAP-Server her.

```
VALIDATE LDAP SERVER
SEARCH DN
 URL 'ldap://iq10web:389/dc=sybase,dc=com?dn?sub?uid=*'
 ACCESS ACCOUNT 'cn=Manager,dc=sybase,dc=com'
 IDENTIFIED BY 'Not4YourEyes'
AUTHENTICATION URL 'ldap://iq10web:389/'
CONNECTION TIMEOUT 1000
CONNECTION RETRIES 3
TLS OFF;
```

Dieser LDAP-Server verwendet Port 389 für die Kommunikation. Im Beispiel wird die TLS-Verschlüsselung nicht aktiviert (TLS OFF). Ihre VALIDATE LDAP SERVER-Anweisung muss ohne Fehler ausgeführt werden, bevor Sie mit dem nächsten Schritt fortfahren.

3. Führen Sie eine CREATE LDAP SERVER-Anweisung aus, um ein LDAP-Serverkonfigurationsobjekt zu erstellen.

Die folgende Anweisung definiert beispielsweise ein LDAP-Serverkonfigurationsobjekt, das für die Benutzerauthentifizierung verwendet werden kann.

```
CREATE LDAP SERVER prim_ldap
SEARCH DN
 URL 'ldap://iq10web:389/dc=sybase,dc=com?dn?sub?uid=*'
 ACCESS ACCOUNT 'cn=Manager,dc=sybase,dc=com'
 IDENTIFIED BY 'Not4YourEyes'
AUTHENTICATION URL 'ldap://iq10web:389/'
CONNECTION TIMEOUT 1000
CONNECTION RETRIES 3
TLS OFF;
```

Anders als die VALIDATE LDAP SERVER-Anweisung versucht die CREATE LDAP SERVER-Anweisung nicht, eine Verbindung mit dem LDAP-Server herzustellen.

4. Führen Sie eine CREATE LDAP SERVER-Anweisung aus, um ein zweites LDAP-Serverkonfigurationsobjekt zu erstellen, das als Failover verwendet wird. Dieser Schritt ist optional, aber für die folgenden Schritte erforderlich.

Die folgende Anweisung definiert beispielsweise ein LDAP-Serverkonfigurationsobjekt, das als Failover für die Benutzerauthentifizierung verwendet werden kann.

```
CREATE LDAP SERVER sec_ldap
 SEARCH DN
 URL 'ldap://iq10web:390/dc=sybase,dc=com?dn?sub?uid=*'
 ACCESS ACCOUNT 'cn=Manager,dc=sybase,dc=com'
 IDENTIFIED BY 'Not4YourEyes'
 AUTHENTICATION URL 'ldap://iq10web:390/'
 CONNECTION TIMEOUT 1000
 CONNECTION RETRIES 3
 TLS OFF;
```

Dieser LDAP-Server verwendet Port 390 für die Kommunikation.

5. Führen Sie eine SET OPTION-Anweisung aus, um den Login-Modus so zu ändern, dass die LDAP-Benutzerauthentifizierung aktiviert wird.

Die folgende Anweisung aktiviert beispielsweise sowohl die Standardauthentifizierung als auch die LDAP-Benutzerauthentifizierung.

```
SET OPTION PUBLIC.login_mode='Standard,LDAPUA';
```

Die aktuelle Einstellung der login\_mode-Option kann folgendermaßen abgefragt werden:

```
SELECT connection_property('login_mode');
```

6. Führen Sie eine CREATE LOGIN POLICY-Anweisung aus, um eine neue Login-Richtlinie zu erstellen.

Die folgende Anweisung erstellt beispielsweise eine neue Login-Richtlinie, die für die Authentifizierung von Benutzern verwendet werden kann.

```
CREATE LOGIN POLICY ldap_policy_both
 LDAP_PRIMARY_SERVER=prim_ldap
 LDAP_SECONDARY_SERVER=sec_ldap
 LDAP_FAILOVER_TO_STD=ON;
```

Die Namen des primären und sekundären LDAP-Servers werden in dieser Login-Richtlinie angegeben. Wenn die Authentifizierung für den dieser Login-Richtlinie zugeordneten Benutzer fehlschlägt, wird eine Standardauthentifizierung versucht (sofern durch den Login-Modus zugelassen). Die aktuellen Einstellungen der Login-Richtlinie können folgendermaßen abgefragt werden:

```
SELECT lpo.* FROM sysloginpolicyoption AS lpo, sysloginpolicy AS lp
 WHERE lpo.login_policy_id = lp.login_policy_id
 AND lp.login_policy_name = 'ldap_policy_both';
```

7. Führen Sie eine CREATE USER-Anweisung aus, um eine neue Benutzer-ID mit der in einem vorherigen Schritt definierten Login-Richtlinie zu erstellen.

Die folgende Anweisung erstellt beispielsweise eine neue Benutzer-ID, die vom primären oder sekundären LDAP-Server authentifiziert wird.

```
CREATE USER ldap_user01 LOGIN POLICY ldap_policy_both;
```

Die IDENTIFIED BY-Klausel wird weggelassen. Die IDENTIFIED BY-Klausel wird verwendet, um das für die Standardauthentifizierung zu verwendende Kennwort anzugeben. Wenn mit dieser Klausel ein Kennwort angegeben wird, muss es nicht dasselbe sein, das vom LDAP-Server authentifiziert wird. Da dieses Kennwort bei der ersten erfolgreichen Authentifizierung des Benutzers beim LDAP-Server ersetzt wird, wird die IDENTIFIED BY-Klausel hier weggelassen.

8. Aktivieren Sie die LDAP-Server für die sofortige Verwendung. Die folgenden Anweisungen aktivieren den primären und sekundären LDAP-Server.

```
ALTER LDAP SERVER prim_ldap WITH ACTIVATE;
ALTER LDAP SERVER sec_ldap WITH ACTIVATE;
```

Der aktuelle Status aller LDAP-Server kann mit der folgenden Abfrage ermittelt werden.

```
call sa_get_ldapserver_status;
```

Die ldsrv\_state-Spalte der Ergebnismenge zeigt an, dass sich die beiden LDAP-Server im READY-Zustand befinden.

9. Führen Sie in Interactive SQL eine CONNECT-Anweisung aus, um die Verbindung mit der Datenbank über die LDAP-Benutzerauthentifizierung herzustellen.

Die folgende Anweisung verwendet beispielsweise die angegebene Benutzer-ID und das angegebene Kennwort, um eine Verbindung mit der Beispieldatenbank herzustellen.

```
CONNECT DATABASE demo USER ldap_user01 IDENTIFIED BY 'abcd1234';
```

Wenn die Benutzerauthentifizierung durch den LDAP-Server fehlschlägt, wird die Standardauthentifizierung versucht. Die Standardauthentifizierung kann fehlschlagen, wenn das Standardkennwort nicht mit dem angegebenen übereinstimmt (z.B. weil es nicht während der LDAP-Authentifizierung aktualisiert wurde).

## Ergebnisse

Der Benutzer wird über die LDAP-Benutzerauthentifizierung mit einem SQL Anywhere-Datenbankserver verbunden.

## Siehe auch

- „VALIDATE LDAP SERVER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE LDAP SERVER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „login\_mode-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „trusted\_certificates\_file-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „CREATE LOGIN POLICY-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „CREATE USER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „sa\_get\_ldapserver\_status-Systemprozedur“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

## LDAP-Benutzerauthentifizierungsumgebung erstellen (Sybase Central)

Führen Sie die folgenden Aufgaben aus, um eine LDAP-Benutzerauthentifizierungsumgebung einzurichten:

1. Erstellen Sie ein LDAP-Serverkonfigurationsobjekt.
2. Erstellen Sie eine Login-Richtlinie, die den LDAP-Server verwendet.
3. Erstellen Sie Benutzer, die sich mit der von Ihnen definierten Login-Richtlinie beim LDAP-Server authentifizieren.

Für diese Aufgaben benötigen Sie die folgenden Systemprivilegien:

- SET ANY SECURITY OPTION
- MANAGE ANY LDAP SERVER
- MANAGE ANY LOGIN POLICY
- MANAGE ANY USER

### Siehe auch

- „LDAP-Serverkonfigurationsobjekte erstellen“ auf Seite 962
- „Login-Richtlinien mit Verwendung eines LDAP-Servers erstellen“ auf Seite 964
- „Benutzer mit Authentifizierung bei einem LDAP-Server erstellen“ auf Seite 965

## LDAP-Serverkonfigurationsobjekte erstellen

Verwenden Sie Sybase Central, um zwei LDAP-Serverkonfigurationsobjekte zu erstellen.

### Voraussetzungen

Sie benötigen die Systemprivilegien MANAGE ANY LDAP SERVER und SET ANY SYSTEM OPTION.

### Kontext und Bemerkungen

Auf die einzelnen LDAP-Server wird über TCP/IP zugegriffen. Eine LDAP-Serverdefinition ist für jeden LDAP-Server erforderlich, den Sie für die Benutzerauthentifizierung verwenden möchten. Die unten gezeigten Antworten dienen nur Demonstrationszwecken. Geben Sie Ihre eigenen entsprechenden Antworten an.

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf **LDAP-Server** und klicken Sie dann auf **Neu » LDAP-Server**.

3. Geben Sie für den Namen des LDAP-Servers den Servernamen (z.B. **prim\_ldap**) ein und klicken Sie dann auf **Weiter**.
4. Wählen Sie den Typ der Netzwerkverschlüsselung aus (z.B. **Keine Verschlüsselung (ldap:)**).
5. Als Hostnamen und Portnummer des LDAP-Servers geben Sie den Servernamen und die Portnummer ein (z.B. **iq10web** und **389**).
6. Unter **Authentifizierungs-URL** sollten Sie eine URL sehen (z.B. **ldap://iq10web:389/**). Klicken Sie auf **Weiter**.
7. Die Such-URL können Sie direkt in das Feld **Such-URL** eingeben (z.B. **ldap://iq10web:389/dc=sybase,dc=com?dn?sub?uid=\***).
8. Als **Distinguished Name** geben Sie den Distinguished Name ein (z.B. **cn=Manager,dc=sybase,dc=com**).
9. Wenn für die Authentifizierung auf dem LDAP-Server ein Kennwort erforderlich ist, geben Sie es in das Feld **Kennwort** ein (z.B. **Not4YourEyes**).
10. Klicken Sie auf **Weiter**.
11. Als Verbindungs-Timeout wählen Sie das Verbindungs-Timeout-Intervall aus (z.B. **1 Sekunde**).
12. Wählen Sie für die Anzahl der Verbindungsversuche die Anzahl der Wiederholungen aus (z.B. **4**) und klicken Sie dann auf **Weiter**.
13. Um den LDAP-Server zu aktivieren, wählen Sie **Diesen LDAP-Server jetzt aktivieren** aus.
14. Wählen Sie die **Login-Modi** aus. (Mindestens **Standard** und **LDAPUA** sollten ausgewählt werden.)
15. Klicken Sie auf **Verbindung testen**, um Ihre LDAP-Verbindungsparameter zu überprüfen. Klicken Sie beispielsweise auf **Verbindung testen**, wenn sich das Dialogfeld öffnet. Wenn Sie die Verbindungsparameter richtig eingegeben haben, ist die Verbindung erfolgreich. Klicken Sie auf **Schließen** und anschließend auf **Weiter**.
16. (Optional) Fügen Sie einen Kommentar hinzu. Klicken Sie auf **Weiter**.
17. Die SQL-Anweisung, die ausgeführt wird, um das LDAP-Serverkonfigurationsobjekt zu erstellen, wird angezeigt. Klicken Sie auf **Fertig stellen**.

## Ergebnisse

Ein LDAP-Serverkonfigurationsobjekt wird erstellt und Verweise darauf werden zur ISYSLDAPSERVER-Systemtabelle hinzugefügt.

## Nächste Schritte

(Optional) Erstellen Sie ein zweites LDAP-Serverkonfigurationsobjekt, das für Failover verwendet wird. Führen Sie dieselben Schritte aus wie oben, jedoch mit **sec\_ldap** als Servername und Port 390.

Erstellen Sie nachdem Abschluss dieser Aufgabe eine Login-Richtlinie, die den LDAP-Server verwendet.

### Siehe auch

- „Login-Richtlinien mit Verwendung eines LDAP-Servers erstellen“ auf Seite 964
- „login\_mode-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „trusted\_certificates\_file-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „SYSLDAPSERVER-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

## Login-Richtlinien mit Verwendung eines LDAP-Servers erstellen

Verwenden Sie Sybase Central, um eine Login-Richtlinie zu erstellen, die den LDAP-Server verwendet.

### Voraussetzungen

Sie müssen das **MANAGE ANY LOGIN POLICY**-Systemprivileg haben.

### Kontext und Bemerkungen

Eine Login-Richtlinie ist erforderlich, wenn Sie einen LDAP-Server für die Benutzerauthentifizierung verwenden. Die unten gezeigten Antworten dienen nur Demonstrationszwecken. Geben Sie Ihre eigenen entsprechenden Antworten an.

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf **Login-Richtlinien** und klicken Sie auf **Neu » Login-Richtlinie**.
3. Geben Sie den Namen der Login-Richtlinie (z.B. **ldap\_policy\_both**) ein und klicken Sie dann auf **Weiter**.
4. Wählen Sie den **LDAP-Primärserver** aus (z.B. **prim\_ldap**).
5. (Optional) Wählen Sie den **LDAP-Sekundärserver** aus (z.B. **sec\_ldap**).
6. (Optional, wird aber empfohlen) Wählen Sie den **LDAP-Failover auf Standardauthentifizierung** aus (z.B. **On**).
7. Klicken Sie auf **Weiter**.
8. (Optional) Fügen Sie einen Kommentar hinzu. Klicken Sie auf **Weiter**.
9. Die SQL-Anweisung, die ausgeführt wird, um die Login-Richtlinie zu erstellen, wird angezeigt. Klicken Sie auf **Fertig stellen**.

## Ergebnisse

Eine Login-Richtlinie wird erstellt, die einen LDAP-Server für die Benutzerauthentifizierung verwendet, und Definitionen für diese Richtlinie werden zu den Systemtabellen SYSLOGINPOLICY und SYSLOGINPOLICYOPTION hinzugefügt.

## Nächste Schritte

Erstellen Sie einen Benutzer, der sich bei einem LDAP-Server authentifiziert.

## Siehe auch

- „LDAP-Serverkonfigurationsobjekte erstellen“ auf Seite 962
- „Benutzer mit Authentifizierung bei einem LDAP-Server erstellen“ auf Seite 965
- „CREATE LOGIN POLICY-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSLOGINPOLICY-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSLOGINPOLICYOPTION-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

# Benutzer mit Authentifizierung bei einem LDAP-Server erstellen

Erstellen Sie mit Sybase Central einen Benutzer, der sich bei einem LDAP-Server authentifiziert.

## Voraussetzungen

Sie müssen das MANAGE ANY USER-Systemprivileg haben.

## Kontext und Bemerkungen

Benutzer, die sich bei einem LDAP-Server authentifizieren, müssen in der Datenbank identifiziert werden. Die unten gezeigten Antworten dienen nur Demonstrationszwecken. Geben Sie Ihre eigenen entsprechenden Antworten an.

## Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf **Benutzer** und klicken Sie auf **Neu » Benutzer**.
3. Geben Sie den Namen des neuen Benutzers (z.B. **ldap\_user01**) ein und klicken Sie dann auf **Weiter**.
4. Achten Sie darauf, dass **Diesem Benutzer ein Kennwort zuweisen** nicht aktiviert ist.
5. Als **Login-Richtlinie** wählen Sie die Login-Richtlinie (z.B. **ldap\_policy\_both**) aus.
6. Klicken Sie auf **Weiter**.
7. (Optional) Fügen Sie einen Kommentar hinzu. Klicken Sie auf **Weiter**.

- Die SQL-Anweisung, die ausgeführt wird, um den Benutzer zu erstellen, wird angezeigt. Klicken Sie auf **Fertig stellen**.

### Ergebnisse

Ein Datenbankbenutzer wird erstellt, der einen LDAP-Server für die Authentifizierung verwendet. Wenn der LDAP-Server nicht verfügbar ist, wird die Standardauthentifizierung durchgeführt.

### Nächste Schritte

Verwenden Sie die Benutzer-ID und das LDAP-Kennwort, um eine Verbindung mit der Datenbank herzustellen.

### Siehe auch

- „LDAP-Serverkonfigurationsobjekte erstellen“ auf Seite 962
- „Login-Richtlinien mit Verwendung eines LDAP-Servers erstellen“ auf Seite 964
- „CREATE USER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSUSER-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

## LDAP-Serverkonfigurationsobjekt aktivieren oder unterbrechen

Aktivieren oder unterbrechen Sie ein LDAP-Serverkonfigurationsobjekt für die Verwendung mit einem LDAP-Server.

### Voraussetzungen

Sie müssen das MANAGE ANY LDAP SERVER-Systemprivileg haben.

### Kontext und Bemerkungen

Sie müssen ein LDAP-Serverkonfigurationsobjekt erneut aktivieren, nachdem Sie es geändert haben, oder ein LDAP-Serverkonfigurationsobjekt aussetzen, um es zu ändern oder zu löschen.

### Aufgabe

- Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
- Klicken Sie im linken Fensterausschnitt auf **LDAP-Server**.
- Rechtsklicken Sie im rechten Fensterausschnitt auf die LDAP-Serverkonfiguration und klicken Sie dann auf **Aktivieren** bzw. **Aussetzen**.
- Klicken Sie auf **OK**.

### Ergebnisse

Der Status des LDAP-Serverkonfigurationsobjekts wird aktualisiert.



### Siehe auch

- „ALTER LDAP SERVER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSLDAPSERVER-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

## LDAP-Serverkonfigurationsobjekte ändern

Ändern Sie ein LDAP-Serverkonfigurationsobjekt so, dass eine Verbindung mit einem LDAP-Server für die LDAP-Benutzerauthentifizierung hergestellt wird.

### Voraussetzungen

Sie müssen das **MANAGE ANY LDAP SERVER**-Systemprivileg haben.

### Kontext und Bemerkungen

Sybase Central setzt ein LDAP-Serverkonfigurationsobjekt aus, bevor es geändert wird, und reaktiviert es, sobald der Vorgang abgeschlossen ist.

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie im linken Fensterausschnitt auf **LDAP-Server**.
3. Rechtsklicken Sie im rechten Fensterausschnitt auf die LDAP-Serverkonfiguration, die Sie ändern möchten, und klicken Sie dann auf **Eigenschaften**.
4. Bearbeiten Sie die Eigenschaften auf den Registerkarten **Allgemein** und **Einstellungen** und klicken Sie dann auf **OK**.

### Ergebnisse

Die Einstellungen für das LDAP-Serverkonfigurationsobjekt werden in der **ISYSLDAPSERVER**-Systemtabelle aktualisiert.

### Siehe auch

- „LDAP-Serverkonfigurationsobjekt aktivieren oder unterbrechen“ auf Seite 966
- „ALTER LDAP SERVER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSLDAPSERVER-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

## LDAP-Serverkonfigurationsobjekte validieren

Stellen Sie eine Verbindung mit einem LDAP-Server her, um sicherzustellen, dass das LDAP-Konfigurationsobjekt richtig konfiguriert ist.

### Voraussetzungen

Sie müssen das **MANAGE ANY LDAP SERVER**-Systemprivileg haben.

Der LDAP-Server, der dem LDAP-Serverkonfigurationsobjekt zugeordnet ist, muss zugänglich sein.

### Kontext und Bemerkungen

Der Datenbankserver verwendet für die Verbindung mit dem LDAP-Server die Einstellungen im LDAP-Konfigurationsobjekt, um sicherzustellen, dass sie gültig sind.

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie im linken Fensterausschnitt auf **LDAP-Server**.
3. Rechtsklicken Sie im rechten Fensterausschnitt auf die LDAP-Serverkonfiguration und klicken Sie dann auf **Verbindung testen**.

### Ergebnisse

Der Datenbankserver validiert das LDAP-Serverkonfigurationsobjekt. Wenn während der Validierung Probleme auftreten, gibt der Datenbankserver Warnungen zurück.

### Siehe auch

- „[VALIDATE LDAP SERVER-Anweisung](#)“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „[SYSLDAPSERVER-Systemansicht](#)“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

## LDAP-Serverkonfigurationsobjekte löschen

Löschen Sie ein LDAP-Serverkonfigurationsobjekt, das für Verbindungen mit einem LDAP-Server verwendet wird.

### Voraussetzungen

Sie müssen das **MANAGE ANY LDAP SERVER**-Systemprivileg haben.

### Kontext und Bemerkungen

Sybase Central entfernt alle Verweise auf das LDAP-Serverkonfigurationsobjekt aus allen Login-Richtlinien, bevor es gelöscht wird.

Sybase Central stoppt automatisch das LDAP-Serverkonfigurationsobjekt, bevor es gelöscht wird.

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.

2. Klicken Sie im linken Fensterausschnitt auf **LDAP-Server**.
3. Rechtsklicken Sie im rechten Fensterausschnitt auf die LDAP-Serverkonfiguration und klicken Sie dann auf **Löschen**.
4. Bestätigen Sie Ihre Auswahl, indem Sie auf **Ja** klicken.

### Ergebnisse

Der Datenbankserver löscht das LDAP-Serverkonfigurationsobjekt und Verweise darauf werden aus der ISYSLDAPSERVER-Systemtabelle entfernt.

### Siehe auch

- „DROP LDAP SERVER-Anweisung“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSLDAPSERVER-Systemansicht“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]



---

# SQL Anywhere-Debugger

Sie können den SQL Anywhere-Debugger während der Entwicklung von gespeicherten Prozeduren, Triggern, Event-Handlern und benutzerdefinierten Funktionen für SQL verwenden.

Sie können den Debugger auch für Folgendes benutzen:

- **In Event-Handlern nach Fehlern suchen** Event-Handler sind eine Erweiterung von in SQL geschriebenen gespeicherten Prozeduren. Die Angaben zur Fehlersuche in gespeicherten Prozeduren in diesem Abschnitt gelten auch für die Fehlersuche in Event-Handlern.
- **Gespeicherte Prozeduren und Klassen durchsuchen** Sie können den Quellcode von SQL-Prozeduren durchsuchen.
- **Ausführung schrittweise verfolgen** Sie können den Code einer gespeicherten Prozedur Zeile für Zeile verfolgen. Sie können auch den Stack der aufgerufenen Funktionen in beiden Richtungen durchsuchen.
- **Breakpoints setzen** Sie führen den Code aus, bis Sie auf einen Breakpoint treffen, und halten an der betreffenden Stelle an.
- **Unterbrechungsbedingungen festlegen** Breakpoints enthalten Codezeilen, aber Sie können auch Bedingungen angeben, bei deren Eintritt der Code unterbrochen werden soll. Sie können z.B. bei einer Zeile stoppen, wenn diese zum zehnten Mal ausgeführt wird, oder wenn eine Variable einen bestimmten Wert erreicht.
- **Lokale Variable prüfen und ändern** Wenn die Ausführung an einem Breakpoint gestoppt wurde, können Sie die Werte von lokalen Variablen prüfen und ändern.
- **Ausdrücke prüfen und bei einem bestimmten Ausdruck unterbrechen** Wenn die Ausführung an einem Breakpoint unterbrochen wird, können Sie den Wert einer großen Auswahl von Ausdrücken prüfen.
- **Zeilenvariable prüfen und ändern** Zeilenvariable sind die ALTEN und NEUEN Werte von Triggern auf Zeilenebene. Sie können diese Werte prüfen und ändern.
- **Abfragen ausführen** Sie können Abfragen ausführen, wenn die Ausführung an einem Breakpoint in einer SQL-Prozedur gestoppt wurde. Auf diese Weise können Sie sich die Zwischenergebnisse in temporären Tabellen ansehen sowie Werte in Basistabellen überprüfen und den Abfrageausführungsplan ansehen.

## Voraussetzungen für die Benutzung des Debuggers

Es kann jeweils nur ein Benutzer den Debugger benutzen.

Bei der Verwendung des Debuggers über HTTP/SOAP-Verbindungen sollten Sie die Timeoutoptionen für den Port auf dem Server ändern. `-xs http{TO=600;KTO=0;PORT=8081}` setzt beispielsweise den Timeout auf 10 Minuten und deaktiviert den Keepalive-Timeout für Port 8081. Der Timeout (TO) ist

der Zeitraum zwischen empfangenen Paketen. Der Keep-Alive-Timeout (KTO) ist die Gesamtzeit, für die die Verbindung aufrechterhalten werden darf. Wenn Sie KTO auf 0 setzen, entspricht dies der Einstellung, dass nie eine Zeitüberschreitung eintritt.

Wenn Sie eine SQL Anywhere HTTP/SOAP-Clientprozedur verwenden, um den SQL Anywhere HTTP/SOAP-Dienst aufzurufen, in dem Sie ein Debugging durchführen, sollten Sie die `remote_idle_timeout`-Datenbankoption des Clients auf einen hohen Wert setzen, z.B. auf 150 (der Standardwert ist 15 Sekunden), um einen Timeout während der Debugging-Sitzung zu vermeiden.

### Siehe auch

- „Datenbankserveroption -xs“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „remote\_idle\_timeout-Option“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „KeepaliveTimeout-Protokolloption (KTO)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Timeout-Protokolloption (TO)“ [[SQL Anywhere Server - Datenbankadministration](#)]

## Praktische Einführung: Erste Schritte mit dem Debugger

In dieser praktischen Einführung wird veranschaulicht, wie Fehler in gespeicherten Prozeduren mithilfe des Debuggers gefunden werden können.

Die SQL Anywhere-Beispieldatenbank, *demo.db*, enthält eine gespeicherte Prozedur namens `debugger_tutorial`, die wiederum einen absichtlichen Fehler enthält. Die `debugger_tutorial`-Systemprozedur gibt eine Ergebnismenge zurück, die den Namen der Firma mit dem höchsten Auftragswert sowie den Auftragswert selbst enthält. Die Prozedur berechnet diese Werte durch Schleifendurchläufe durch die Ergebnismenge einer Abfrage, die Firmen und Aufträge auflistet. (Dieses Ergebnis könnte durch eine `SELECT FIRST`-Abfrage erreicht werden, ohne der Prozedur logische Schritte hinzufügen zu müssen. Diese Prozedur wird benutzt, um ein bequemes Beispiel zu erstellen.) Jedoch führt der Fehler in der `debugger_tutorial`-Systemprozedur dazu, dass sie nicht die angegebene Ergebnismenge zurückgibt.

### Privilegien

Sie müssen die `SA_DEBUG`-Systemrolle haben.

Zusätzlich benötigen Sie entweder das `EXECUTE ANY PROCEDURE`-Systemprivileg oder das `EXECUTE`-Privileg für die `debugger_tutorial`-Systemprozedur. Außerdem müssen Sie entweder das `ALTER ANY PROCEDURE`-Systemprivileg oder das `ALTER ANY OBJECT`-Systemprivileg haben.

### Siehe auch

- „SQL Anywhere-Debugger“ auf Seite 971
- „Voraussetzungen für die Benutzung des Debuggers“ auf Seite 971

## Lektion 1: Debugger-Start und Fehlersuche

Starten Sie den Debugger, um die gespeicherte Prozedur "debugger\_tutorial" auszuführen und den Fehler zu finden.

### Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Erste Schritte mit dem Debugger](#)“.

### Aufgabe

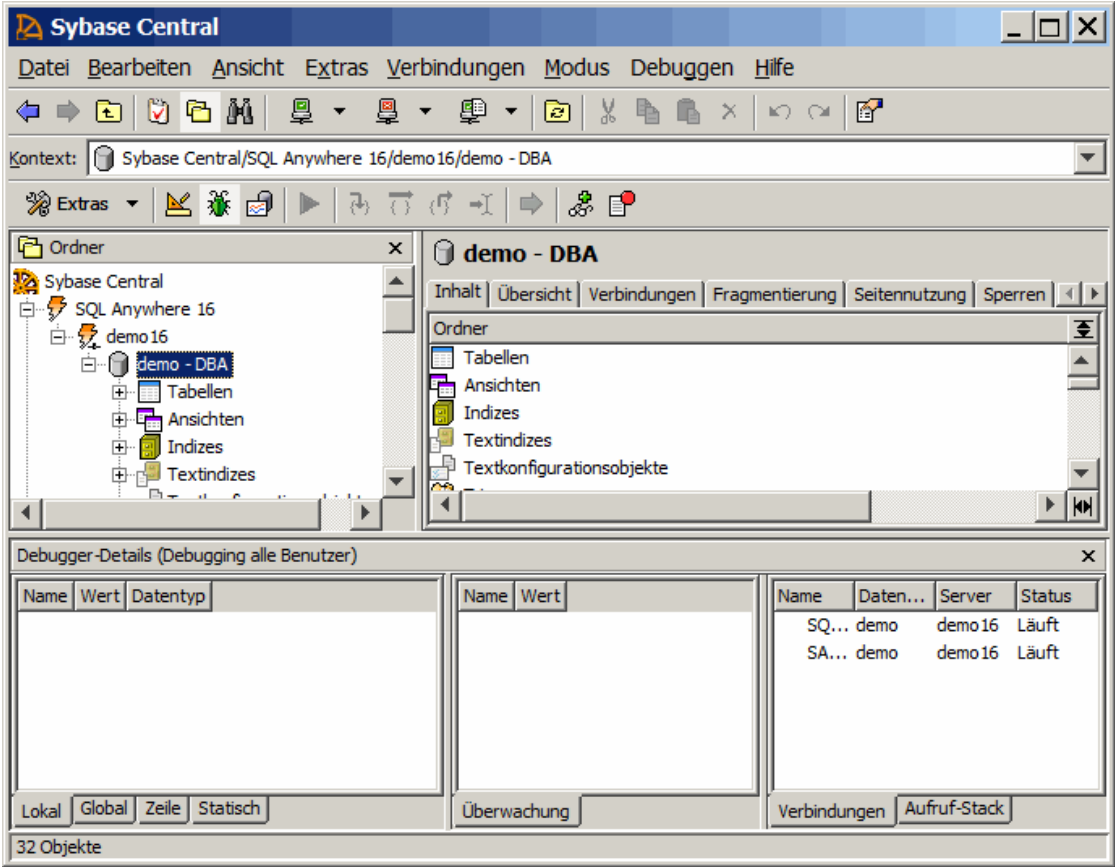
1. Erstellen Sie eine Kopie der Beispieldatenbank, die Sie in dieser praktischen Einführung verwenden.
  - a. Erstellen Sie ein Verzeichnis, z. B. `c:\demodb`, um die Datenbank zu speichern.
  - b. Geben Sie den folgenden Befehl ein, um die -Datenbank zu erstellen:

```
newdemo c:\demodb\demo.db
```

2. Starten Sie Sybase Central. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Sybase Central**.
3. Stellen Sie in Sybase Central folgendermaßen eine Verbindung mit `demo.db` her:
  - a. Klicken Sie auf **Verbindungen » Verbinden mit SQL Anywhere 16**.
  - b. Füllen Sie im Fenster **Verbinden** die folgenden Felder aus, um eine Verbindung mit der Datenbank herzustellen.
    - i. Im Feld **Benutzer-ID** geben Sie **DBA** ein.
    - ii. Im Feld **Kennwort** geben Sie **sql** ein.
    - iii. Wählen Sie in der Dropdown-Liste **Aktion** die Option **Eine Datenbank auf diesem Computer starten und eine Verbindung herstellen** aus.
    - iv. Geben Sie im Feld **Datenbankdatei** Folgendes ein: `c:\demodb\demo.db`.
    - v. Im Feld **Servername** geben Sie **demo\_server** ein.
  - c. Klicken Sie auf **Verbinden**.
4. Klicken Sie auf **Modus » Debug**.
5. In das Feld **Welchen Benutzer wollen Sie debuggen?** geben Sie **\*** ein und klicken auf **OK**.

Der Fensterausschnitt **Debugger-Details** erscheint unten in Sybase Central, und die Symbolleiste von Sybase Central enthält eine Reihe von Debugger-Werkzeugen.

Wenn Sie **\*** angeben, können Sie eine Fehlersuche für alle Benutzer durchführen. Um den Benutzer zu ändern, für den Sie die Fehlersuche durchführen, müssen Sie beenden und erneut den Debug-Modus aufrufen. Wenn Sie eine Benutzer-ID angeben, werden Informationen zu Verbindungen mit dieser Benutzer-ID erfasst und auf der Registerkarte **Verbindungen** angezeigt.



6. Doppelklicken Sie im linken Fensterausschnitt von Sybase Central auf **Prozeduren und Funktionen**.
7. Rechtsklicken Sie auf **Debugger\_Tutorial (GROUPO)** und klicken Sie auf **Von Interactive SQL aus ausführen**.

Interactive SQL wird geöffnet und die folgende Ergebnismenge erscheint:

| top_company | top_value |
|-------------|-----------|
| (NULL)      | (NULL)    |

Diese Ergebnismenge ist falsch. Im Rest der Einführung wird der Fehler diagnostiziert, der dieses Ergebnis hervorgerufen hat.

8. Schließen Sie alle offenen Fenster von Interactive SQL.

### Ergebnisse

Der Debugger wird gestartet und ein Fehler wurde in der gespeicherten Prozedur "debugger\_tutorial" gefunden.



## Nächste Schritte

Gehen Sie weiter zu „[Lektion 2: Fehlerdiagnose](#)“ auf Seite 975.

### Siehe auch

- „SQL Anywhere-Debugger“ auf Seite 971
- „Neuerstellung der Beispieldatenbank (demo.db)“ [[SQL Anywhere 16 - Einführung](#)]
- „Praktische Einführung: Verbindung mit der Beispieldatenbank herstellen“ [[SQL Anywhere Server - Datenbankadministration](#)]

## Lektion 2: Fehlerdiagnose

Diagnostizieren Sie die gespeicherte Prozedur `debugger_tutorial`, indem Sie Breakpoints setzen, den Code durchgehen und den Wert der Variablen überwachen, während die Prozedur ausgeführt wird.

### Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: „[Praktische Einführung: Erste Schritte mit dem Debugger](#)“.

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe „[Lektion 1: Debugger-Start und Fehlersuche](#)“ auf Seite 973.

### Aufgabe

1. Suchen Sie im linken Fensterausschnitt auf der Registerkarte **SQL** nach **debugger\_tutorial (GROUPO)** und der folgenden Anweisung:

```
OPEN cursor_this_customer;
```

2. Fügen Sie einen Breakpoint hinzu, indem Sie auf den senkrechten grauen Bereich links von der Anweisung klicken. Der Breakpoint wird als roter Kreis dargestellt.
3. Rechtsklicken Sie im linken Fensterausschnitt auf **Debugger\_Tutorial (GROUPO)** und klicken Sie auf **Von Interactive SQL aus ausführen**.

Im rechten Fensterausschnitt von Sybase Central erscheint ein gelber Pfeil über dem Breakpoint.

4. Klicken Sie im Fensterausschnitt **Debugger-Details** auf die Registerkarte **Lokal**, um eine Liste von lokalen Variablen in der Prozedur mit ihren aktuellen Werten und Datentypen anzuzeigen. Die Variablen **Top\_Company**, **Top\_Value**, **This\_Value** und **This\_Company** sind alle uninitialisiert und daher gleich NULL.
5. Drücken Sie F11, um durch die Prozedur zu scrollen. Die Werte der Variablen ändern sich, wenn Sie die folgende Zeile erreichen:

```
IF SQLSTATE = error_not_found THEN
```

6. Drücken Sie erneut zweimal F11, um festzustellen, welche Verzweigung die Ausführung nimmt. Der gelbe Pfeil bewegt sich zurück zum folgenden Text:

```
customer_loop: loop
```

Die IF-Prüfung wurde nicht als TRUE zurückgegeben. Der Grund dafür ist, dass ein Vergleich eines beliebigen Wertes mit NULL immer NULL zurückgibt. Durch NULL ist die Bedingung nicht erfüllt und der Code innerhalb der Anweisung IF...END IF wird nicht ausgeführt.

An dieser Stelle werden Sie möglicherweise bemerken, dass der Grund für das Problem darin liegt, dass **Top\_Value** nicht initialisiert wurde.

7. Prüfen Sie die Hypothese, dass das Problem durch die fehlende Initialisierung von **Top\_Value** hervorgerufen wird, ohne den Code der Prozedur zu ändern.
- Im Fenster **Debugger-Details** klicken Sie auf die Registerkarte **Lokal**.
  - Klicken Sie auf die Variable **Top\_Value**, geben Sie **3000** in das Feld **Wert** ein und drücken Sie die Eingabetaste.
  - Drücken Sie F11 solange, bis das Feld **Wert** der Variablen **This\_Value** größer als 3000 ist.
  - Klicken Sie auf den Breakpoint, sodass er grau wird.
  - Drücken Sie F5, um die Prozedur auszuführen.

Das Interactive SQL-Fenster wird angezeigt und zeigt die richtigen Ergebnisse:

| top_company | top_value |
|-------------|-----------|
| Chadwicks   | 8076      |

- Schließen Sie alle offenen Fenster von Interactive SQL.

## Ergebnisse

Die Hypothese wurde bestätigt. Das Problem liegt darin, dass die Variable **Top\_Value** nicht initialisiert ist.

## Nächste Schritte

Gehen Sie weiter zu [„Lektion 3: Fehlerbehebung“ auf Seite 976](#).

## Siehe auch

- [„Breakpoints“ auf Seite 977](#)
- [„Breakpoints setzen“ auf Seite 978](#)

# Lektion 3: Fehlerbehebung

Beheben Sie den Fehler, den Sie in der vorherigen Lektion ermittelt haben, indem Sie die Variable **Top\_Value** initialisieren.

## Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie die Rollen und Privilegien haben, die im Abschnitt "Privilegien" am Anfang dieser praktischen Einführung aufgeführt sind: [„Praktische Einführung: Erste Schritte mit dem Debugger“](#).

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe [„Lektion 1: Debugger-Start und Fehlersuche“ auf Seite 973](#).

## Aufgabe

1. Klicken Sie auf **Modus » Design**.
2. Im rechten Fensterausschnitt suchen Sie nach folgender Anweisung:  

```
OPEN cursor_this_customer;
```
3. Geben Sie darunter die folgende Zeile ein, die die Variable **Top\_Value** initialisiert:

```
SET top_value = 0;
```

4. Klicken Sie auf **Datei » Speichern**.
5. Führen Sie die Prozedur erneut aus und prüfen Sie, ob Interactive SQL die richtigen Ergebnisse anzeigt.
6. Schließen Sie alle offenen Fenster von Interactive SQL.

## Ergebnisse

Dieser Fehler ist behoben und die Prozedur wird wie erwartet ausgeführt. Sie haben die praktische Einführung zum Debugging abgeschlossen.

## Nächste Schritte

Löschen Sie das Verzeichnis, das die Kopie der Datenbank enthält, die in dieser praktischen Einführung verwendet wird, beispielsweise `c:\demodb`.

## Siehe auch

- [„Variablen“ auf Seite 981](#)
- [„Variable Werte anzeigen“ auf Seite 981](#)

# Breakpoints

Breakpoints steuern, wann der Debugger die Ausführung Ihres Quellcodes unterbricht.

Wenn Sie im Debug-Modus arbeiten und eine Verbindung auf einen Breakpoint stößt, ändert sich das Verhalten, abhängig von der ausgewählten Verbindung:

- Wenn noch keine Verbindung ausgewählt ist, wird die Verbindung automatisch ausgewählt und der Quellcode für die Prozedur wird angezeigt.
- Wenn bereits eine Verbindung ausgewählt ist und es diese Verbindung ist, die auf den Breakpoint stößt, wird der Quellcode für die Prozedur angezeigt.
- Wenn bereits eine Verbindung ausgewählt ist und es nicht die Verbindung ist, die auf den Breakpoint stößt, wird ein Fenster eingeblendet, in dem Sie aufgefordert werden, die Verbindung auszuwählen, die auf den Breakpoint gestoßen ist.

## Breakpoints setzen

Administratoren können Breakpoints in Sybase Central setzen. Ein Breakpoint gibt dem Debugger vor, an einer bestimmten Zeile die Ausführung zu unterbrechen. Standardmäßig gilt ein Breakpoint für alle Verbindungen.

### Voraussetzungen

Sie müssen die SA\_DEBUG-Systemrolle haben.

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie auf **Modus » Debug**.
3. Im Feld **Welchen Benutzer wollen Sie debuggen?** geben Sie \* ein, um alle Benutzer zu debuggen, oder geben Sie den Namen des Datenbankbenutzers ein, den Sie debuggen wollen.

| Option                                       | Aktion                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Rechter Fensterausschnitt von Sybase Central | <div>a. Doppelklicken Sie im linken Fensterausschnitt auf <b>Prozeduren und Funktionen</b> und wählen Sie eine Prozedur aus.</div> <div>b. Im rechten Fensterausschnitt klicken Sie auf die Zeile, in die Sie den Breakpoint einfügen wollen.<br/><br/>In dieser Zeile erscheint nun ein Cursor.</div> <div>c. Drücken Sie F9.<br/><br/>Links von der Codezeile erscheint ein roter Kreis.</div> |

| Option               | Aktion                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Menü<br><b>Debug</b> | <p>a. Wählen Sie <b>Debug » Breakpoints</b>.</p> <p>b. Klicken Sie auf <b>Neu</b>.</p> <p>c. In der Liste <b>Prozedur</b> wählen Sie eine Prozedur aus.</p> <p>d. Wenn nötig, füllen Sie die Felder <b>Bedingung</b> und <b>Anzahl</b> aus.</p> <p>Bei der Bedingung handelt es sich um einen SQL-Ausdruck. Wenn dieser als TRUE ausgewertet wird, unterbricht der Breakpoint die Ausführung.</p> <p>Die Anzahl gibt an, wie oft der Breakpoint durchlaufen werden muss, bevor die Ausführung angehalten wird. Wenn Sie den Wert 0 angeben, wird die Ausführung bei jedem Durchlauf des Breakpoints angehalten.</p> <p>e. Klicken Sie auf <b>OK</b>. Der Breakpoint wird bei der ersten ausführbaren Anweisung in der Prozedur gesetzt.</p> |

## Ergebnisse

Der Breakpoint wird gesetzt.

## Beispiel

Sie können einen Breakpoint festlegen, der sich auf eine Verbindung eines bestimmten Benutzers bezieht, indem Sie die folgende Bedingung eingeben:

```
CURRENT USER = 'user-name'
```

## Status von Breakpoints ändern

Ändern Sie den Status eines Breakpoints in Sybase Central.

### Voraussetzungen

Sie müssen die SA\_DEBUG-Systemrolle haben.

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie auf **Modus » Debug**.
3. Doppelklicken Sie im linken Fensterausschnitt auf **Prozeduren und Funktionen** und wählen Sie eine Prozedur aus.

| Option                                        | Aktion                                                                                                                                                                                                                                                |
|-----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Rechter Fenster-ausschnitt von Sybase Central | Klicken Sie im rechten Fensterausschnitt auf den Breakpoint-Indikator links von der Zeile, die Sie bearbeiten möchten. Der Breakpoint wechselt von aktiv zu inaktiv, ein weiterer Klick auf den Breakpoint entfernt diesen.                           |
| <b>Breakpoints-Fenster</b>                    | <ol style="list-style-type: none"> <li>Wählen Sie <b>Debug » Breakpoints</b>.</li> <li>Wählen Sie den Breakpoint aus und klicken Sie auf <b>Bearbeiten, Deaktivieren</b> oder <b>Entfernen</b>.</li> <li>Klicken Sie auf <b>Schließen</b>.</li> </ol> |

## Ergebnisse

Der Status des Breakpoints wird geändert.

## Breakpoint-Bedingungen ändern

Sie fügen Breakpoints Bedingungen hinzu, um den Debugger anzuweisen, die Ausführung bei diesem Breakpoint nur dann zu unterbrechen, wenn eine bestimmte Bedingung oder Anzahl erfüllt ist. Bei Prozeduren und Triggern muss es sich bei der Bedingung um eine SQL-Suchbedingung handeln.

### Voraussetzungen

Sie müssen die SA\_DEBUG-Systemrolle haben.

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie auf **Modus » Debug**.
3. Doppelklicken Sie im linken Fensterausschnitt auf **Prozeduren und Funktionen** und wählen Sie eine Prozedur aus.
4. Wählen Sie **Debug » Breakpoints**.
5. Wählen Sie den zu bearbeitenden Breakpoint und klicken Sie auf **Bearbeiten**.
6. In der Liste **Bedingung** klicken Sie auf eine Bedingung. Für einen Breakpoint, der nur für Verbindungen für eine bestimmte Benutzer-ID gilt, geben Sie beispielsweise folgende Bedingung ein:

```
CURRENT USER= 'user-name '
```

Bei dieser Bedingung gilt: *user-name* ist die Benutzer-ID, für die ein Breakpoint aktiviert werden soll.

7. Klicken Sie auf **OK** und dann auf **Schließen**.

## Ergebnisse

Die Bedingung wird auf den Breakpoint gesetzt.

# Variablen

Mit dem Debugger können Sie das Verhalten Ihrer Variablen ansehen und bearbeiten, während Sie durch Ihren Code durchgehen. Der Debugger verfügt über einen Fensterausschnitt **Debugger-Details**, in dem die unterschiedlichen Arten von Variablen dargestellt werden, die in gespeicherten Prozeduren verwendet werden. Der Fensterausschnitt **Debugger-Details** erscheint unten im Sybase Central-Fenster, wenn Sybase Central im Debuggen-Modus ausgeführt wird.

## Variable Werte anzeigen

Sie können Variablenwerte in Sybase Central anzeigen.

### Voraussetzungen

Sie müssen die SA\_DEBUG-Systemrolle haben.

Zusätzlich benötigen Sie das EXECUTE ANY PROCEDURE-Systemprivileg oder das EXECUTE-Privileg für die Prozedur.

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie auf **Modus » Debug**.
3. Im Feld **Welchen Benutzer wollen Sie debuggen?** geben Sie \* ein, um alle Benutzer zu debuggen, oder geben Sie den Namen des Datenbankbenutzers ein, den Sie debuggen wollen.
4. Doppelklicken Sie im linken Fensterausschnitt auf **Prozeduren und Funktionen** und wählen Sie eine Prozedur aus.
5. Im rechten Fensterausschnitt klicken Sie auf die Zeile, in die Sie den Breakpoint einfügen wollen.

In dieser Zeile erscheint nun ein Cursor.

6. Drücken Sie F9.

Links von der Codezeile erscheint ein roter Kreis.

7. Im Fensterausschnitt **Debugger-Details** klicken Sie auf die Registerkarte **Lokal**.
8. Im linken Fensterausschnitt rechtsklicken Sie auf die Prozedur und klicken auf **Aus Interactive SQL ausführen**.

9. Klicken Sie auf die Registerkarte **Lokal**.

### Ergebnisse

Die Variablen werden zusammen mit ihren Werten angezeigt.

## Globale Variablen

Globale Variablen werden von SQL Anywhere definiert und enthalten Informationen über die aktuelle Verbindung, Datenbank und andere Einstellungen. Sie werden im Fensterausschnitt **Debugger-Details** auf der Registerkarte **Global** eingeblendet.

Zeilenvariablen werden in Triggern benutzt, um die Werte von Zeilen aufzunehmen, auf die sich die Trigger-Anweisung auswirkt. Sie werden im Fensterausschnitt **Debugger-Details** auf der Registerkarte **Zeile** eingeblendet.

Statische Variablen werden in Java-Klassen verwendet. Sie werden auf der Registerkarte **Statisch** angezeigt.

### Siehe auch

- „Globale Variablen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Trigger“ auf Seite 98

## Aufruf-Stack anzeigen

Beim Debugging für verschachtelte Prozeduren können Sie die Sequenz der ausgeführten Aufrufe überprüfen. Auf der Registerkarte **Aufruf-Stack** wird eine Liste der Prozeduren angezeigt.

### Voraussetzungen

Sie müssen die SA\_DEBUG-Systemrolle haben.

Zusätzlich benötigen Sie das EXECUTE ANY PROCEDURE-Systemprivileg oder das EXECUTE-Privileg für die Prozedur.

### Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie auf **Modus » Debug**.
3. Im Feld **Welchen Benutzer wollen Sie debuggen?** geben Sie \* ein, um alle Benutzer zu debuggen, oder geben Sie den Namen des Datenbankbenutzers ein, den Sie debuggen wollen.
4. Doppelklicken Sie im linken Fensterausschnitt auf **Prozeduren und Funktionen** und wählen Sie eine Prozedur aus.



5. Im rechten Fensterausschnitt klicken Sie auf die Zeile, in die Sie den Breakpoint einfügen wollen.  
In dieser Zeile erscheint nun ein Cursor.
6. Drücken Sie F9.  
Links von der Codezeile erscheint ein roter Kreis.
7. Im Fensterausschnitt **Debugger-Details** klicken Sie auf die Registerkarte **Lokal**.
8. Im linken Fensterausschnitt rechtsklicken Sie auf die Prozedur und klicken auf **Aus Interactive SQL ausführen**.
9. Im Fensterausschnitt **Debugger-Details** klicken Sie auf die Registerkarte **Aufruf-Stack**.

### Ergebnisse

Die Namen der Prozeduren erscheinen auf der Registerkarte **Aufruf-Stack**. Die aktuelle Prozedur ist in der Liste ganz oben. Die Prozedur, von der sie aufgerufen wurde, steht direkt darunter.

## Fehlersuche in Verbindungen

Auf der Registerkarte **Verbindungen** werden die Verbindungen zur Datenbank angezeigt. Zu jeder Zeit können mehrere Verbindungen bestehen. Einige sind möglicherweise an einem Breakpoint angehalten worden, andere nicht.

Sie können eine andere Verbindung anzeigen, indem Sie auf dem Register **Verbindungen** doppelt auf eine Verbindung klicken.

Eine nützliche Technik besteht darin, einen Breakpoint so zu setzen, dass er die Ausführung für eine bestimmte Benutzer-ID unterbricht. Sie können dies erreichen, indem Sie eine Breakpoint-Bedingung im folgenden Format einrichten:

```
CURRENT USER = 'user-name'
```

Der spezielle SQL-Wert CURRENT USER enthält die Benutzer-ID der Verbindung.

### Siehe auch

- „Breakpoint-Bedingungen ändern“ auf Seite 980
- „CURRENT USER-Spezialwert“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]



---

# Index

## Symbole

- \* (Sternchen)
  - SELECT-Anweisung,296
- \*=
  - Transact-SQL-Outer-Joins,509
- im, Option
  - Tipps zur Performance-Verbesserung,247
- <
  - Vergleichsoperator in WHERE-Klauseln,310
- =\*
  - Transact-SQL-Outer-Joins,509
- >
  - Vergleichsoperator in WHERE-Klauseln,310
- @@error, globale Variable
  - Rückgabewerte,673
- @@identity, globale Variable
  - IDENTITY-Spalte,663

## A

- a\_init\_pre\_filter
  - Eintrittspunktfunktion für Vorfilter,468
- a\_init\_pre\_filter-Struktur
  - Info,458
- a\_init\_term\_breaker
  - Eintrittspunktfunktion für Begriffsegmentierer,469
- a\_init\_term\_breaker-Struktur
  - Info,461
- a\_server\_context-Struktur
  - Info,456
- a\_term-Struktur
  - Info,465
- a\_term\_breaker\_for-Enumeration
  - Info,462
- a\_text\_source-Schnittstelle
  - Info,459
- a\_word\_source-Schnittstelle
  - Info,463
- Abfolgeplanung
  - für Transaktionen,924
  - serialisierbar,924
- Abfrage Strategien für Speichermangel, Statistik
  - Beschreibung,218
- Abfrage, Phase der semantischen Transformationen
  - Abfrageverarbeitung,329
- Abfragealgorithmen
  - Abkürzungen in Ausführungsplänen,356
- Abfrageausführung
  - Ansichtenübereinstimmung,234
  - Info,376
  - Parallelität,376
- Abfrageergebnisse
  - exportieren,744,753,754
  - in Gruppen organisieren,474
- Abfrageergebnisse exportieren
  - Info,744
- Abfrageinterne Parallelität
  - abfrageinterne vs. abfrageübergreifende Parallelität,376
  - Exchange-Algorithmus,376
  - Info,376
- Abfragen
  - allgemeine Tabellenausdrücke,536
  - Anzahl vermindern,263
  - Ausführungspläne,341
  - Bypass-Abfragen, Definition,331
  - Daten aus einer Tabelle abrufen,291
  - Diagnostizieren der zu lang laufenden Abfragen, praktische Einführung,272
  - exportieren,744,753,754
  - Info,291
  - Liste der möglichen Optimierungen durch den Optimierer,340
  - Mengenoperationen,486
  - ohne Ausführung optimieren,342
  - optimieren,332
  - Optimierer, Bypass,331
  - Parallelität in,378
  - SELECT-Anweisung,291
  - semantische Transformationen,340
  - Transact-SQL-kompatible Abfragen schreiben,665
  - Unnötige Inner- und Outer-Joins eliminieren,241
  - unnötige Konvertierung der Groß- und Kleinschreibung eliminieren,241
  - Verarbeitungsphasen,329
  - Verzeichniszugriff-Proxy-Tabellen,800
  - zu lang laufende, Abfrageperformance überwachen,245
  - zu lang laufende, Performance-Probleme beheben,189
- Abfragen optimieren
  - Phasen,329
- Abfragen, die den Optimierer umgehen

- Eignung für ein Überspringen der Abfrageverarbeitungsphase,331
- Info,331
- Abfragen, die für ein Überspringen der Abfrageverarbeitungsphase in Frage kommen
- Info,331
- Abfragen, Probleme
- Ferndatenzugriff,823
- Abfragen, selbstblockierend
- Ferndatenzugriff,823
- Abfragen, syntaktische Analyse
- Ferndatenzugriff,819
- Abfragen, Vorverarbeitung
- Ferndatenzugriff,819
- Abfragenormalisierung
- Ferndatenzugriff,819
- Abfrageoptimierer
- Info,332
- Abfrageoptimierung
- IN-Listen-Prädikate,241
- Optimierer, Bypass,331
- Abfrageperformance
- Ausführungspläne lesen,341
- Cache-Lesevorgänge und -treffer,350
- Fehlen effektiver Indizes,350
- Prädikat-Selektivität,349
- Probleme mit der Datenfragmentierung identifizieren,350
- Quellen der Schätzung,349
- RowsReturned, Statistik,349
- Selektivitätsstatistiken,348
- Abfrageplan-Cacheseiten, Statistik
- Beschreibung,218
- Abfragespeicher
- Info über,221
- Abfragetransformation
- benutzerdefinierte Funktionen,241
- Abfragetransformationen
- Inlining von einfachen gespeicherten Prozeduren,241
- Abfrageübergreifende Parallelität
- abfrageinterne vs. abfrageübergreifende Parallelität,376
- Info,376
- Abfrageverarbeitung
- Phasen,329
- Abfragezeilen materialisiert/Sek., Statistik
- Beschreibung,218
- Abgeleitete Tabellen
- aus DML-Anweisungen auswählen,307
- Info,306
- Joins,517
- natürliche Joins,523
- Outer-Joins,508
- Schlüssel-Joins,533
- Abhängigkeiten
- Ansichtenabhängigkeiten,43
- Abkürzungen in Ausführungsplänen
- Info,356
- Absichtssperren
- Konflikte,915
- Snapshot-Isolation,911
- Absichtstabellensperren
- Info,912
- Absteigende Reihenfolge
- ORDER BY-Klausel,481
- access\_date\_time
- Verzeichniszugriffsserver,795
- Adaptive Server Enterprise
- Architektur,651
- Datentypkonvertierungen,830
- emulieren,657,658
- Kompatibilität,648,784
- Kompatibilität beim Datenimport/-export,784
- kompatible Objektnamen sicherstellen,661
- nach SQL Anywhere migrieren,774
- Serverklasse,829
- spezielle IDENTITY-Spalte,662
- Administratorrollen
- Adaptive Server Enterprise,653
- ADSODBC-Serverklasse
- Info,828
- Advantage, Datenbankserver
- ODBC-Serverklasse,828
- Aggregate
- Element in Ausführungsplänen,375
- Aggregatfunktionen
- äquivalente Formeln für OLAP,598
- auf gruppierte Daten anwenden,326
- Datentypen,472
- DISTINCT, Schlüsselwort,473
- Einführung,325
- Fenster (OLAP),571
- GROUP BY-Klausel,477
- Info,470
- mehrere Aggregatstufen,540

---

- mehrere Stufen,540
- NULL,473
- OLAP,572
- ORDER BY und GROUP BY,485
- Skalaraggregate,471
- Aktionen
  - CASCADE,874
  - RESTRICT,873
  - SET DEFAULT,874
  - SET NULL,873
- Aktivieren
  - materialisierte Ansichten,68
  - Prozedurprofilerstellung in Sybase Central,158
  - reguläre Ansichten mit SQL,54
  - reguläre Ansichten mit Sybase Central,52
- Aktiviert
  - Status von materialisierten Ansichten,76
- Aktualisieren
  - Ansichten,46
  - manuelle Ansichten ,67
  - mithilfe von Joins,634
  - Textindizes,385,424
- Aktualisierung
  - Typ zum Aktualisieren von Textindizes auswählen,424
- Aktualisierungen, Registerkarte
  - Indexberater, Ergebnisse,168
- Aktualisierungstyp
  - manuelle und Sofortansichten,59
- Aktualisierungstypen
  - für eine materialisierte Ansicht ändern,74
  - Textindizes,424
  - Textindizes ändern,386
- Aliase
  - für berechnete Spalten,301
  - Info,299
  - Korrelationsnamen,305
  - SQL Anywhere-Implementierung,647
- ALL
  - Schlüsselwort und UNION-Klausel,486
  - Unterabfrage-Tests,614
- ALL, Operator
  - Hinweise,614
  - Info,613
  - Unterabfrage-Tests,613
- All-rows, Optimierungsziel
  - Ziel des Optimierers auswählen,228
- Allgemeine Tabellenausdrücke
  - Beispiele,536
  - Datentypen für rekursive,545
  - Einschränkungen für rekursive,543
  - hierarchische Datenstrukturen untersuchen,543
  - Info,536
  - Konstantenmengen speichern,542
  - mehrere Aggregatstufen,540
  - Problem der kürzesten Entfernung,550
  - rekursive,543
  - Stücklistenproblem,547
  - typische Anwendungen,540
  - wo zulässig,539
- allow\_nulls\_by\_default, Option
  - für Transact-SQL-Kompatibilität einstellen,659
- allow\_snapshot\_isolation, Option
  - verwenden,891
- Alphabetische Reihenfolge
  - ORDER BY-Klausel,322
- ALTER INDEX-Anweisung
  - mit Snapshot-Isolation nicht verfügbar,889
- ALTER SERVER-Anweisung
  - Fremdserver ändern,793
- ALTER TABLE-Anweisung
  - CHECK-Integritätsregeln,860
  - Fremdschlüssel,24
  - mit Snapshot-Isolation nicht verfügbar,889
  - Parallelität,954
  - Primärschlüssel,19
- ALTER TRIGGER-Anweisung
  - Syntax,104
- ALTER-Anweisung
  - automatisches Festschreiben,882
- AND
  - logische Operatoren verwenden,319
- Ändern
  - CHECK-Integritätsregeln,864
  - Fremdserver,793
  - Prozeduren mit Sybase Central,90
  - Sequenzen,952
  - Spalten-Standardwerte,853
  - Textindizes,382
  - Trigger,104
- Ändern von Daten
  - Privilegien,624
  - UPDATE-Anweisungen mit Verletzungen von Integritätsregeln verarbeiten,636
- Änderungen festschreiben
  - Info,624

- Änderungen in Transaktionen zusammenfassen
  - Info, 882
- Änderungen rückgängig machen
  - Info, 625
- Anfängliche Cachegröße
  - Info, 223
- Anforderungen abbrechen
  - Ferndatenzugriff, 823
- Anforderungen Aktiv, Statistik
  - Beschreibung, 216
- Anforderungen GET DATA/Sek., Statistik
  - Beschreibung, 218
- Anforderungen Ungeplant, Statistik
  - Beschreibung, 216
- Anforderungen, Statistik
  - Beschreibung, 216
- Anforderungen, Registerkarte
  - Indexberater, Ergebnisse, 168
- Anforderungs-Trace-Analyse
  - ausführen, 192
  - Info, 192
- Anforderungsaustausch, Statistik
  - Beschreibung, 216
- Anforderungslog
  - Info, 195
  - Sicherheit, 195
- Anforderungsprotokollierung
  - Info, 195
  - mit clientseitigem Caching von Anweisungen, 196
- Anführungszeichen
  - Adaptive Server Enterprise, 316
  - Zeichenfolgen, 316
- Angemeldeter Benutzer
  - im Gegensatz zum effektiven Benutzer, 82
- ANSI
  - nicht-ANSI-Joins, 499
  - SQL, Standards und Inkonsistenzen, 895
- ANSI-Integritätsregeln für Aktualisierungen
  - Ausführungspläne, 372
- Ansicht erstellen, Assistent
  - verwenden, 49
- Ansichten
  - aktualisieren, 46
  - aktualisieren mit INSTEAD OF-Trigger, 111
  - allgemeine Tabellenausdrücke, 536
  - ändern und Ansichtenabhängigkeiten, 50
  - Beschränkungen für SELECT-Anweisungen bei regulären Ansichten, 46
  - Daten in regulären Ansichten durchsuchen, 55
  - DEAKTIVIERT-Status für reguläre Ansichten, 48
  - exportieren, 745
  - FROM-Klausel, 305
  - GÜLTIG-Status für reguläre Ansichten, 48
  - löschen mit Sybase Central, 51
  - mit Ansichten arbeiten, 41
  - mit Ansichtsabhängigkeiten arbeiten, 43
  - mit Textindexabfragen, 402
  - natürliche Joins, 523
  - Outer-Joins, 508
  - Programmvariable referenzieren, 541
  - Prüfoption und reguläre Ansichten, 47
  - reguläre Ansichten ändern, Hinweise, 50
  - reguläre Ansichten mit SQL aktivieren, 54
  - reguläre Ansichten mit SQL deaktivieren, 54
  - reguläre Ansichten mit Sybase Central aktivieren, 52
  - reguläre Ansichten mit Sybase Central ändern, 50
  - reguläre Ansichten mit Sybase Central deaktivieren, 52
  - reguläre Ansichten mit Sybase Central erstellen, 49
  - reguläre Ansichten verwenden, 46
  - Schlüssel-Joins, 533
  - Status regulärer Ansichten, 48
  - UNGÜLTIG-Status für reguläre Ansichten, 48
- Ansichtenabhängigkeiten
  - Info, 43
  - Informationen im Katalog, 45
  - Schemaänderungen, 43
  - Status regulärer Ansichten, 48
- Ansichtenübereinstimmung
  - Abfrageausführung, 234
  - Abfrageberechnung, 235
  - Algorithmus für die Ansichtenübereinstimmung, Ergebnisse von Ausführungsplänen, 372
  - Algorithmus für die Ansichtenübereinstimmung, Info, 235
  - Algorithmus, Anforderungen, 235
  - Ergebnisse von Ausführungsplänen, 372
  - Info, 235
  - mit Snapshot-Isolation verwenden, 889
- Ansichtsabhängigkeiten
  - Abhängigkeitsinformationen suchen, 45
- Ansichtsstatus
  - deaktiviert, 48
  - feststellen, 48
  - gültig, 48

---

- reguläre Ansichten,48
- Status von materialisierten Ansichten,76
- ungültig,48
- verstehen,48
- Anweisungen
  - Diagnostizieren langsamer Anweisungen, praktische Einführung,272
  - nicht unterstützte Transact SQL-Anweisungen,651
  - optimieren,333
  - Trennzeichen,146
  - zusammengesetzte,119
- Anweisungen, komplettes Passthrough
  - Ferndatenzugriff,820
- Anweisungen, partielles Passthrough
  - Ferndatenzugriff,820
- Anweisungen, Statistik
  - Beschreibung,216
- Anweisungscache-Fehlschläge, Statistik
  - Beschreibung,216
- Anweisungscache-Treffer, Statistik
  - Beschreibung,216
- Anweisungsvorbereitungen, Statistik
  - Beschreibung,216
- Anwendungsprofil, Assistent
  - automatischen Start aktivieren und deaktivieren,157
  - Info,157
  - starten,157
- Anwendungsprofilerstellung
  - Anforderungs-Trace-Analyse,192
  - erkennen, ob der Arbeitsspeicher ein einschränkender Faktor ist,190
  - erkennen, ob die CPU ein einschränkender Faktor ist,190
  - erkennen, ob die I/O-Bandbreite ein einschränkender Faktor ist,190
  - Indexberater,164
  - Info,156
  - praktische Einführungen,264
  - Produktionsdatenbank,171
  - Protokollierungsdatenbank,171
  - Protokollierungssitzung erstellen,186
  - Prozedurprofilerstellung,158
- Anwendungsprofilerstellung, Modus
  - Verwendung,156
- ANY, Operator
  - Info,612
  - Probleme,612
  - Unterabfragentests,612
- Anzahl Abfrage-Warten, Statistik
  - Beschreibung,213
- Anzahl Abfrageanforderungen, Statistik
  - Beschreibung,213
- Anzahl Zugriffsfehlschläge, Statistik
  - Beschreibung,213
- Anzeigen
  - Daten in regulären Ansichten,55
  - Ergebnisse der Prozedurprofilerstellung,162
- Apostrophe
  - Zeichenfolgen,316
- APPLY, Ausdrücke
  - Beispiel,518
  - Info,517
- Arbeitstabellen
  - Abfrageverarbeitung,239
  - Info,239
  - Performance-Tipps,256
- Architekturen
  - Adaptive Server Enterprise,651
- Arithmetische Operatoren
  - Abfrageergebnisse zusammenfassen,470
  - Überlauffehler,303
  - Vorrang von Ausdrücken und Operatoren,302
- Arten der Volltextsuche
  - Info,390
- AS-Schlüsselwort
  - Aliase,299
  - SQL Anywhere-Implementierung,647
- ASEODBC-Serverklasse
  - Info,829
- Assistent für die Anwendungsprofilerstellung
  - praktische Einführungen,264
- Assistent für die Datenbankprotokollierung
  - praktische Einführungen,264
- Assistent zum Definieren eines Primärschlüssels
  - Zugriff,18
- Assistent zum Erstellen einer Eindeutigkeits-Integritätsregel
  - Zugriff,863
- Assistent zum Erstellen einer globalen temporären Tabelle
  - Zugriff,12
- Assistent zum Erstellen einer Tabelle
  - Zugriff,5
- Assistent zum Erstellen eines Textkonfigurationsobjekts

- festgelegte Einstellungen,406
- Syntax,381
- Assistent zum Erstellen von Funktionen
  - Zugriff,94
- Assistent zum Erstellen von materialisierten Ansichten
  - Zugriff,65
- Assistent zum Erstellen von Spalten-Integritätsregeln
  - Zugriff,862
- Assistent zum Erstellen von Tabellen-Integritätsregeln
  - Zugriff,862
- Assistent zum Erstellen von Textindizes
  - verwenden,383
- Assistent zum Migrieren einer Datenbank
  - Vorgehensweise,775
- Assistent zur Erstellung eines Sequenzgenerators
  - Syntax,952
- Atomare Transaktionen
  - Info,881
- Atomare zusammengesetzte Anweisungen
  - Info,120
- Attribute
  - Abfrageergebnisse als XML erhalten,688
  - SQLCA.lock,898
- Auf Daten auf Clientcomputern zugreifen
  - Info,760
- Auf entfernte Daten zugreifen
  - Info,785
- Auffächerung
  - Indizes,40
- Aufruf-Stacks
  - im Debugger anzeigen,982
- Aufrufe
  - Statistiken in Zugriffsplänen,370
- Aufrufer
  - effektiver und angemeldeter Benutzer,82
  - Festlegen des von einer Datenbank benutzten Sicherheitsmodells,88
  - Prozeduren und Funktionen mit Aufruferprivilegien ausführen,82
  - Systemprozeduren vor Version 16.0 als Aufrufer oder Definierer ausführen,84
- Aufsteigende Reihenfolge
  - ORDER BY-Klausel,481
- Ausblenden
  - materialisierte Ansichten,69
- Ausdrücke
  - APPLY-Ausdrücke,517
  - NULL,319
- Äußere Referenzen
  - Definition,603
  - HAVING-Klausel,608
  - Info,603
- Ausführen
  - Abfragen mehr als einmal,323
  - SQL-Skripten,780
  - SQL-Skripten in Interactive SQL,780
  - Trigger,104
- Ausführen von SQL-Skriptdateien
  - Info,780
- Ausführliche Textpläne
  - Info über,343
  - mit SQL-Funktionen anzeigen,345
- Ausführungsphase
  - Abfrageverarbeitung,331
- Ausführungspläne
  - Abkürzungen,356
  - Ansichtenübereinstimmungsergebnisse,372
  - ausführliche Textpläne,343
  - Caching,337
  - grafische Pläne,347
  - kontextsensitive Hilfe,351
  - kurze Textpläne,342
  - lesen,341
  - ohne Ausführung einer Abfrage anzeigen,342
- Ausgabeumleitung
  - Info,744,753,754
- Auslagerungsplatz
  - Datenbankcache,225
- Ausnahmen
  - deklarieren,133
- Ausnahmeroutinen
  - Beispiel,141
  - Fehlerprotokollierung,141
  - Prozeduren und Trigger,136
  - verschachtelte zusammengesetzte Anweisungen,138
- Auswahl aus DML
  - Verwendung,307
- Authentifizierung
  - LDAP-Benutzerauthentifizierung, Info,957
- AUTO REFRESH-Klausel
  - Info,424
- AUTO, Modus
  - verwenden,693
- auto\_commit, Option



- Änderungen in Interactive SQL
  - zusammenfassen,882
- AutoCommit
  - Performance,246
- Autocommit, Modus
  - Transaktionen,882
- AUTOINCREMENT
  - IDENTITY-Spalte,662
  - negative Zahlen,856
  - signierte Datentypen,856
  - Standardwert,856
  - UltraLite-Anwendungen,856
  - Unterschiede zu Sequenzen,950
  - wann anzuwenden,949
- automatic\_timestamp, Option
  - für Transact-SQL-Kompatibilität einstellen,659
- Automation
  - Eindeutige Schlüssel generieren,949
- Automatische Joins
  - Fremdschlüssel,645
- Automatisches Festschreiben
  - ALTER-Anweisung,882
  - COMMENT-Anweisung,882
  - Datendefinitionsanweisungen,882
  - DROP-Anweisung,882
- AVG-Funktion
  - äquivalente mathematische Formel,598
  - Verwendung,572
- AvgDiskReads
  - Schätzungen in Zugriffsplänen,371
- AvgDiskReadTime
  - Schätzungen in Zugriffsplänen,371
- AvgDiskWrites
  - Schätzungen in Zugriffsplänen,371
- AvgRowCount
  - Schätzungen in Zugriffsplänen,371
- AvgRunTime
  - Schätzungen in Zugriffsplänen,371

## B

- Baselining
  - Prozedurprofilerstellung, praktische Einführung,284
- Basis-Aggregatfunktionen
  - OLAP,572
- Basistabellen
  - erstellen,5

- Info,5
- Batch-Modus
  - Interactive SQL,782
- Batches
  - Info,112
  - OUTPUT-Anweisung,114
  - schreiben,112
  - SELECT-Anweisungen verwenden,148
  - Steuerungsanweisungen,112
  - Tipps zum Schreiben,146
  - Transact-SQL,671
  - verglichen mit gespeicherten Prozeduren,112
  - zulässige Anweisungen,148
  - zulässige SQL-Anweisungen,148
  - zusammengesetzte Anweisungen,113
- BCP, Format
  - Import/Export mit ASE,784
- Bedingungen
  - für Breakpoints bearbeiten,980
  - GROUP BY-Klausel,328
  - mit logischen Operatoren verknüpfen,319
  - Musterübereinstimmung,313
- Bedingungen für die Diagnoseprotokollierung
  - Info,181
- Befehlstrennzeichen
  - einstellen,146
- BEGIN TRANSACTION-Anweisung
  - Einschränkungen bei der Transaktionsverwaltung,818
  - Ferndatenzugriff,818
- BEGIN-Anweisung
  - Batches,112
- Begriffe
  - Datenbank mit Volltextsuche durchsuchen,379
- Begriffe, Volltextsuche
  - Info,390
- Begriffs- und Phrasensuche
  - Volltextsuche,390
- Begriffsegmentierer
  - Ablauflogik für externe Begriffsegmentierer-Bibliothek,455
  - Beispiel für externen Begriffsegmentierer,451
  - externe Begriffsegmentierer-Bibliotheken,407
  - generischer Begriffsegmentieralgorithmus,407
  - Volltextsuche,406
  - Volltextsuche, externe Begriffsegmentierer-Bibliothek definieren,454
- Begriffsegmentierer-Bibliotheken

- Callbacks aus externen Bibliotheken,456
- Begriffslänge
  - Begriffslängen für Textindizes einstellen,406
- Begriffslängen
  - Volltextsuche,406
- Beispiel-Textkonfigurationsobjekte
  - Volltextsuche,415
- Beispieldatenbank
  - Schema von demo.db,493
- Beispielzeichenfolgen-Interpretationen
  - Volltextsuche,417
- Belegung physischen Speichers/Sek., Statistik
  - Beschreibung,213
- Benutzer-IDs
  - Adaptive Server Enterprise,654
  - Groß- und Kleinschreibung,660
  - Standardwert,855
- Benutzerauthentifizierung
  - mit einem LDAP-Server,957
- Benutzerdefinierte Datentypen
  - auf Spalten anwenden,866
  - CHECK-Integritätsregeln,861
  - erstellen,865
  - löschen,867
  - mit SQL erstellen,865
- Benutzerdefinierte Funktionen
  - aufrufen,95
  - Ausführungsprivilegien,97
  - Caching,339
  - Ergebnisse der Profilerstellung generieren und prüfen,158
  - erstellen,94
  - Info,93
  - Inlining als Teil der Abfragetransformation,241
  - löschen,97
  - Parameter,117
  - Thread-Sicherheit,93
  - Tipps zum Schreiben,146
- Benutzerdefinierte Statistiken
  - Liste,217
- Berechnete Spalten
  - Abfragen zum Verwenden von Sargable-Funktionen veranlassen,293
  - Begrenzungen,17
  - berechnete Spaltenausdrücke ändern,14
  - einfügen und aktualisieren,16
  - Indizes,33
  - konvertieren in nicht berechnete Spalte,14
  - mit berechneten Spalten arbeiten,13
  - neu berechnen,17
  - Trigger,16
- Berechtigungen
  - Adaptive Server Enterprise,654
  - Verzeichniszugriffsserver,795
- Bereiche
  - Diagnoseprotokollierung,174
- Bereiche der Diagnoseprotokollierung
  - Beschreibungen,174
  - Info,174
- Bereichsabfragen
  - Info,311
- Bereichsgrenzen
  - Element in Ausführungsplänen,374
- Bereitgestellte Seiten, Statistik
  - Beschreibung,213
- Berücksichtigung von Groß-/Kleinschreibung
  - Sortierreihenfolge,483
- Beste Pläne mit Kosten
  - Optimiererstatistiken-Feldbeschreibungen,362
- BETWEEN, Schlüsselwort
  - Bereichsabfragen,311
- BETWEEN-Suchbedingungen
  - Bereichsabfragen,311
- Bewahrte Tabellen
  - in Outer-Joins,505
- Bezeichner
  - Eindeutigkeit,661
  - Groß- und Kleinschreibung,660
  - qualifizieren,295
- Binäre große Objekte
  - einfügen,630
  - hinzufügen,630
- BINARY-Datentyp
  - SQL Anywhere-Implementierung,645
- Bitmaps
  - durchsuchen,260
- Bits
  - Element in Ausführungsplänen,375
- Blattseiten
  - Info,40
- Blockieren
  - Deadlock,903
  - Info,902
  - Transaktionen,901
- Blockierung
  - Anzeigen von Deadlocks in Sybase Central,905

---

- Fehlerbehandlung,904
- praktische Einführung,932
- Blocking, Option
  - verwenden,902
- Boolesche Suche
  - Volltextsuche,399
- Breakpoints
  - aktivieren,979
  - Bedingungen,980
  - deaktivieren,979
  - einzelne Verbindungen,980
  - individuelle Benutzer,980
  - Info,977
  - setzen,978
  - Status,979
  - Zähler,980
- Breakpoints aktivieren
  - Info,979
- Breakpoints deaktivieren
  - aktivieren,979
  - Info,979
- Breakpoints setzen
  - Debugger,978
- Breite Spalten
  - Info,259
- Buckets
  - Histogramme,334
- Bypass-Abfragen
  - Definition,331
  - erscheinen nicht in grafischen Plänen,351
  - Optimierer umgehen,331

## C

- Cache
  - anfängliche, minimale und maximale Größe,223
  - Anweisungen in gespeicherten Prozeduren,337
  - Anweisungen, die die Abfrageoptimierung umgehen,337
  - Ausführungspläne,337
  - Cache verwenden, um die Performance zu verbessern,220
  - Caching auf Anweisungsebene,337
  - dynamische Belegung,224
  - Größe überwachen,226
  - Lesevorgänge-Treffer-Verhältnis,350
  - Unix,225

- verschlüsselte Datenbanken benötigen größeren Cache,220
- Vorwärmung,227
- Cache Aufräumen bei Besuch, Statistik
  - Beschreibung,213
- Cache, im Cache abgelegte Pläne
  - Optimierer, Bypass,331
- Cache-Aufräumvorgänge, Statistik
  - Beschreibung,213
- Cache-Instabilität, Statistik
  - Beschreibung,213
- Cachedimensionierung
  - Performance,224
- Cacheersetzungen: Gesamtseiten/Sek., Statistik
  - Beschreibung,213
- Cachegröße
  - anfängliche, minimale und maximale Größe,223
  - Hinweise zu Windows Mobile,260
  - Hinweise zur Performance,260
  - Seitengrößen,260
  - überwachen,226
  - Unix,225
  - Windows,225
  - Windows Mobile,225
- Cachegröße Aktuell, Statistik
  - Beschreibung,207
- Cachegröße Maximum, Statistik
  - Beschreibung,207
- Cachegröße Minimum, Statistik
  - Beschreibung,207
- Cachegröße Spitze, Statistik
  - Beschreibung,207
- CacheHits, Eigenschaft
  - Knotenstatistiken-Feldbeschreibungen,368
  - Statistiken in Zugriffsplänen,370
- Cachelesevorgänge Gesamtseiten/Sek., Statistik
  - Beschreibung,207
- Cachelesevorgänge Index intern/Sek., Statistik
  - Beschreibung,207
- Cachelesevorgänge Index-Blatt/Sek., Statistik
  - Beschreibung,207
- Cachelesevorgänge Tabelle/Sek., Statistik
  - Beschreibung,207
- Cachelesevorgänge, Arbeitstabelle
  - Beschreibung,207
- CacheRead, Eigenschaft
  - Knotenstatistiken-Feldbeschreibungen,368
  - Statistiken in Zugriffsplänen,370

- CacheReadIndLeaf, Eigenschaft
  - Statistiken in Zugriffsplänen,370
- CacheReadTable, Eigenschaft
  - Statistiken in Zugriffsplänen,370
- Cacheseiten Datei verändert, Statistik
  - Beschreibung,213
- Cacheseiten Datei, Statistik
  - Beschreibung,213
- Cacheseiten Frei, Statistik
  - Beschreibung,213
- Cacheseiten verbraucht, Statistik
  - Beschreibung,213
- Cacheseiten Zugewiesene Strukturen, Statistik
  - Beschreibung,213
- Cachestatistiken
  - Liste,207
- Cachetreffer/Sek., Statistik
  - Beschreibung,207
- Cachevorwärmung
  - Info,227
- Caching
  - Anweisungen, die die Abfrageoptimierung umgehen,337
  - Ausführungspläne,337
  - benutzerdefinierte Funktionen,339
  - Unterabfragen,339
- CALL-Anweisung
  - Beispiele,91
  - in Prozeduren verwenden,81
  - Parameter,116
  - Steueranweisungen,118
- CASCADE, Aktion
  - Info,874
- CASE-Anweisung
  - Steueranweisungen,118
- CBSIZE-Verbindungsparameter
  - Performance,264
- cdata, Direktive
  - anwenden,704
- CHAR-Datentyp
  - SQL Anywhere-Implementierung,645
- CHECK, Bedingungen
  - Transact-SQL,652
- CHECK, Integritätsregeln
  - ändern,864
  - Domänen,861
  - löschen,864
  - Spalten,860
  - Tabellen,861
- CHECK, Integritätsregeln der Domänen vererben,861
- CHECK-Integritätsregeln
  - Tools für die Aufrechterhaltung der Datenintegrität,851
- Checkpoint, Statistik
  - Liste,208
- Checkpoint-Auslesevorgänge/Sek., Statistik
  - Beschreibung,208
- Checkpoint-Dringlichkeit, Statistik
  - Beschreibung,208
- Checkpoint-Inaktivitätszeit, Statistik
  - Beschreibung,208
- Checkpoint-Log, Statistik
  - Beschreibung,208
- Checkpoint-Logs
  - Info,256
- Checkpoints/Sek., Statistik
  - Beschreibung,208
- ChkptLog Bitmap-Größe, Statistik
  - Beschreibung,208
- ChkptLog Festschreiben auf Platte/Sek., Statistik
  - Beschreibung,208
- ChkptLog Log-Größe, Statistik
  - Beschreibung,208
- ChkptLog Schr. Seiten/Sek., Statistik
  - Beschreibung,208
- ChkptLog Schreiben in Bitmap/Sek., Statistik
  - Beschreibung,208
- ChkptLog Schreibvorgänge/Sek., Statistik
  - Beschreibung,208
- ChkptLog Seiten in Gebrauch, Statistik
  - Beschreibung,208
- ChkptLog Seitenabbilder gesp./Sek., Statistik
  - Beschreibung,208
- ChkptLog Verschieben Seiten/Sek., Statistik
  - Beschreibung,208
- ChkptLog Vorabbild/Sek. speichern, Statistik
  - Beschreibung,208
- Client-Dateien
  - importieren von und exportieren auf Clientcomputer,760
- Clientseitige Daten
  - Verlust von Daten verhindern, die von einem Client geladen werden,762
- Clientseitiges Caching von Anweisungen mit Anforderungsprotokollierung,196

---

|                                                   |                                          |
|---------------------------------------------------|------------------------------------------|
| Clientseitiges Laden                              | Einschränkungen bei der                  |
| Info,721                                          | Transaktionsverwaltung,818               |
| CLOSE-Anweisung                                   | COMMIT-Anweisung                         |
| Prozeduren für die Cursor-Verwaltung,128          | automatisches Festschreiben,882          |
| Clustered-Indizes                                 | Ferndatenzugriff,818                     |
| deklarieren,32                                    | Prozeduren und Trigger,146               |
| Indexberater, Ergebnisse,167                      | referenzielle Integrität,919             |
| Indexberater-Ergebnisse implementieren,168        | Transaktionen,882                        |
| verwenden,31                                      | UltraLite, Syntax,624                    |
| ClusteredHashGroupBy, Planelement                 | zusammengesetzte Anweisungen,120         |
| Abkürzungen im Plan,356                           | COMPUTE-Klausel                          |
| Comm Empfangene Anforderungen, Statistik          | CREATE TABLE,13                          |
| Beschreibung,209                                  | Transact-SQL SELECT-Anweisung, nicht     |
| Comm Empfangene Byte unkomprimiert/Sek.,          | unterstützte Syntax,665                  |
| Statistik                                         | CONNECTION_PROPERTY-Funktion             |
| Beschreibung,209                                  | Info,202                                 |
| Comm Empfangene Byte/Sek., Statistik              | CONTAINS-Suchbedingung                   |
| Beschreibung,209                                  | Begriffe löschen,447                     |
| Comm Empfangene Mehrfachpakete/Sek., Statistik    | conversion_error, Option                 |
| Beschreibung,209                                  | Auswirkung auf Textindizes,413           |
| Comm Empfangene Pakete unkomprimiert/Sek.,        | COUNT-Funktion                           |
| Statistik                                         | Aggregatfunktionen auf gruppierte Daten  |
| Beschreibung,209                                  | anwenden,326                             |
| Comm Empfangene Pakete/Sek., Statistik            | Info,472                                 |
| Beschreibung,209                                  | NULL,473                                 |
| Comm Freie Puffer, Statistik                      | Counter1, Statistik                      |
| Beschreibung,209                                  | Beschreibung,217                         |
| Comm Ges. Byte/Sek., Statistik                    | Counter2, Statistik                      |
| Beschreibung,209                                  | Beschreibung,217                         |
| Comm Gesamtpuffer, Statistik                      | Counter3, Statistik                      |
| Beschreibung,209                                  | Beschreibung,217                         |
| Comm Gesendete Byte unkomprimiert/Sek., Statistik | Counter4, Statistik                      |
| Beschreibung,209                                  | Beschreibung,217                         |
| Comm Gesendete Mehrfachpakete/Sek., Statistik     | Counter5, Statistik                      |
| Beschreibung,209                                  | Beschreibung,217                         |
| Comm Gesendete Pakete unkomprimiert/Sek.,         | COVAR_POP-Funktion                       |
| Statistik                                         | äquivalente mathematische Formel,598     |
| Beschreibung,209                                  | COVAR_SAMP-Funktion                      |
| Comm Gesendete Pakete/Sek., Statistik             | äquivalente mathematische Formel,598     |
| Beschreibung,209                                  | CPUTime                                  |
| Comm Sendefehler/Sek., Statistik                  | Knotenstatistiken-Feldbeschreibungen,368 |
| Beschreibung,209                                  | CREATE DATABASE-Anweisung                |
| Comm, eindeutige Clientadressen, Statistik        | Transact-SQL-kompatible Datenbanken      |
| Beschreibung,209                                  | erstellen,658                            |
| CommBufferSize-Verbindungsparameter               | CREATE DEFAULT-Anweisung                 |
| Performance,264                                   | nicht unterstützt,652                    |
| COMMIT TRANSACTION-Anweisung                      | CREATE DOMAIN-Anweisung                  |
|                                                   | Domänen verwenden,865                    |

- Transact-SQL-Kompatibilität,652
  - CREATE EXISTING TABLE-Anweisung
    - Proxy-Tabellen für Verzeichniszugriffsserver erstellen,796
    - Standort von Proxy-Tabellen definieren,806
    - verwenden,807
  - CREATE EXTERNLOGIN-Anweisung
    - externe Logins für Verzeichniszugriffsserver erstellen,796
    - verwenden,804
  - CREATE FUNCTION-Anweisung
    - Syntax,93
  - CREATE INDEX-Anweisung
    - mit Snapshot-Isolation nicht verfügbar,889
    - Parallelität,954
  - CREATE LDAP SERVER-Anweisung
    - LDAP-Server zur Benutzerauthentifizierung verwenden,958
  - CREATE PROCEDURE-Anweisung
    - Beispiele,89
    - Parameter,116
    - Syntax,101
    - verwenden,814
  - CREATE RULE-Anweisung
    - nicht unterstützt,652
  - CREATE SERVER-Anweisung
    - Fremdserver erstellen,788
    - IBM DB2-Datentypkonvertierungen,832
    - Microsoft SQL Server-Datentypkonvertierungen,838
    - ODBC- und ASE-Datentypkonvertierungen,830
    - Oracle-Datentypkonvertierungen,845
    - SAP HANA-Datentypkonvertierungen,834
    - SQL Anywhere-Datenbankzuordnung,825
    - Verzeichniszugriffsserver erstellen,796
  - CREATE TABLE-Anweisung
    - Fremdschlüssel,24
    - Parallelität,954
    - Primärschlüssel,19
    - Proxy-Tabellen,807
    - Proxy-Tabellen für Verzeichniszugriffsserver erstellen,796
    - Standort von Proxy-Tabellen definieren,806
    - Transact-SQL-kompatible Tabellen erstellen,664
  - CREATE TEXT CONFIGURATION-Anweisung
    - verwenden,406
  - CREATE VIEW-Anweisung
    - WITH CHECK OPTION-Klausel,47
  - create\_date\_time
    - Verzeichniszugriffsserver,795
  - CROSS APPLY-Klausel
    - Info,517
  - Cross-Joins
    - Info,503
  - CSV, Dateien
    - importieren,728
  - CUBE-Klausel
    - als Abkürzung für GROUPING SETS,558
    - Info,560
  - CUME\_DIST-Funktion
    - äquivalente mathematische Formel,598
    - Verwendung,595
  - CurrentCacheSize, Eigenschaft
    - Optimiererstatistiken-Feldbeschreibungen,362
  - Cursor
    - in Joins aktualisieren,499
    - in Prozeduren, Triggern und benutzerdefinierten Funktionen,128
    - Instabilität,897
    - LOOP-Anweisung,129
    - Prozeduren,129
    - Prozeduren und Trigger,131,135
    - SELECT-Anweisungen,129
    - SQL Anywhere-Implementierung,646
    - Stabilität,897
    - Verbindungslimit,262
  - Cursor offen, Statistik
    - Beschreibung,216
  - Cursor, Statistik
    - Beschreibung,216
  - Cursorstabilitätssperren
    - Info,922
    - WITH HOLD-Cursor,922
- ## D
- dataMenü
    - Abfrageergebnisse exportieren,744
  - DataSet
    - zum Export von Daten als XML verwenden,679
    - zum Importieren von XML,685
  - DATE-Format
    - Textindizes,413
  - date\_format, Option
    - Auswirkung auf Textindizes,413
  - date\_format-Option

- 
- ändern für Textkonfigurationsobjekte,382
  - Dateien
    - Fragmentierung,248
    - grafische Pläne,347
  - Dateifragmentierung
    - Info,249
  - Daten
    - als XML exportieren,678
    - Daten mit SQL in Tabellen anzeigen,10
    - Datenintegrität gewährleisten,849
    - Export-Tools,743
    - exportieren,743
    - Groß- und Kleinschreibung,660
    - Hinweise zum Datenexport,743
    - hinzufügen,626
    - hinzufügen, ändern und löschen,623
    - importieren,723
    - importieren und exportieren,721
    - Integrität und Korrektheit,924
    - Konsistenz,895
    - manuelle Ansichten aktualisieren,67
    - materialisierte Ansichten füllen,66
    - mit dem Exportassistenten,744
    - suchen,27
    - Tabellendaten mit Sybase Central anzeigen,9
    - ungültige,849
    - zum Ändern von Daten erforderliche Privilegien,624
  - Daten ändern
    - Daten mit mehr als einer Tabelle aktualisieren,634
    - INSERT-Anweisung,638
    - UPDATE-Anweisung,633
  - Daten auswählen
    - mit Unterabfragen,600
  - Daten einfügen
    - BLOBs,630
    - in bestimmte Spalten,628
    - INPUT-Anweisung,726
    - INSERT verwenden,626
    - INSERT-Anweisung,731
    - MERGE-Anweisung,732
    - Verhalten bei nicht festgelegten Spalten,627
  - Daten entladen, Fenster
    - verwenden,752
  - Daten exportieren
    - Abfrageergebnisse,744,753,754
    - Datenbank sichern,722
    - dbunload, Dienstprogramm,759
    - Dienstprogramm Entladen,750
    - Ergebnismengen,783
    - Info,743
    - Interactive SQL Export-Assistent,744
    - OUTPUT-Anweisung,745
    - Schemata,770
    - Tools,743
    - UNLOAD TABLE-Anweisung,748
    - UNLOAD-Anweisung,749
    - XML,678
  - Daten hinzufügen
    - INSERT verwenden,626
  - Daten importieren
    - Datenbank sichern,722
    - DEFAULTS-Option,741
    - Import-Assistent,724,739
    - in Datenbanken,723
    - INSERT-Anweisung verwenden,626
    - Konvertierungsfehler,739
    - LOAD TABLE-Anweisung,729
    - nicht übereinstimmende Tabellenstrukturen,741
    - NULL-Werte,741
    - Performance,721
    - Performance-Tipps,723
    - Proxy-Tabellen,738
    - Situationen für Import/Export,721
    - Tabellen,739
    - temporäre Tabellen,741
    - Tools,723
    - von anderen Datenbanken,738
    - XML mit dem DataSet-Objekt ,685
    - XML mit OPENXML-Operator,679
    - XML mit xp\_read\_file-Systemprozedur,683
    - XML-Dokumente,679
    - xp\_read\_file-Systemprozedur,683
  - Daten laden
    - Konvertierungsfehler,739
  - Daten löschen
    - DELETE-Anweisung,638
    - TRUNCATE TABLE-Anweisung,639
  - Daten verschieben
    - exportieren,743
    - importieren,723
  - Daten, Registerkarte
    - SQL Anywhere 16-Plug-In,9
  - Datenänderungsanweisungen
    - Info,623
  - Datenbank

- aktualisieren,623
- Datenbank entladen, Assistent
  - Vorgehensweise,750
- Datenbank entladen, Dienstprogramm (dbunload)
  - Tabellendaten exportieren,770,771
- Datenbank erstellen, Assistent
  - Transact-SQL-kompatible Datenbanken erstellen,657
- Datenbank extrahieren, Assistent
  - SQL Remote,774
- Datenbank migrieren, Assistent
  - Info,774
- Datenbank neu aufsetzen
  - Tipps zur Performancesteigerung,257
- Datenbank-Threads
  - blockierte,903
- Datenbankdateien
  - Dateifragmentierung,249
  - Fragmentierung,248
  - Performance,256
- Datenbanken
  - an Synchronisationssystemen beteiligte Datenbanken entladen und neu laden,767,769
  - entladen,756,763
  - entladen und neu laden,773
  - exportieren,756
  - für SQL Remote extrahieren,774
  - Groß- und Kleinschreibung berücksichtigen,660
  - Groß-/Kleinschreibung in ASE-kompatiblen Datenbanken,656
  - Listen von Systemobjekten in Interactive SQL anzeigen,3
  - migrieren, Assistent,775
  - mit Objekten arbeiten,1
  - nach SQL Anywhere migrieren,774
  - neu laden,763,772
  - Neuaufbau im Vergleich zu Export,764
  - Neuaufbau, Gründe,764
  - nicht an Synchronisationssystemen beteiligte Datenbanken entladen und neu laden,766
  - nicht an Synchronisationssystemen beteiligte Datenbanken neu aufbauen,766
  - Systemobjekte anzeigen,3
  - Tabellen aus mehreren Datenbanken verknüpfen,812
  - Transact-SQL-Kompatibilität,656
  - Upgrade des Datenbankdateiformats,763
  - Warnung vor Komprimierung von Datenbankdateien,263
  - XML importieren,679
  - XML speichern,677
  - XML-Unterstützung,677
- Datenbanken durchsuchen
  - Isolationsstufen,924
- Datenbanken entladen
  - im Format mit Kommatrennzeichen,765
- Datenbanken erstellen
  - externe Protokollierung,193
  - Transact-SQL-kompatible Datenbank,657,658
- Datenbanken migrieren
  - Info,774
  - Systemprozeduren sa\_migrate verwenden,776
- Datenbanken neu aufbauen
  - "dbunload" für an Synchronisationssystemen beteiligte Datenbanken verwenden,768,769
  - Befehlszeile,772
  - Info,763
  - MobiLink,767,769
  - nicht-replizierende Datenbanken,766
  - replizierende Datenbanken,767,769
  - Tabellenfragmentierung reduzieren,249
  - Tools,763
  - UNLOAD TABLE-Anweisung,770
- Datenbanken neu laden
  - Befehlszeile,772
- Datenbanken protokollieren
  - Info,171
- Datenbankobjekte
  - direkte Referenzen,45
  - indirekte Referenzen,45
  - mit Datenbankobjekten arbeiten,1
- Datenbankoptionen
  - Auswirkungen auf materialisierte Ansichten,61
  - Auswirkungen auf Textindizes,413
  - für Transact-SQL-Kompatibilität einstellen,659
  - Indexberater,168
  - Textkonfigurationsobjekt, Einstellungen,413
- Datenbankprotokollierung, Assistent
  - verwenden,186
- Datenbankseiten
  - Indexberater, Ergebnisse,167
- Dateneingabe
  - Isolationsstufen,924
- Datenimport
  - INPUT-Anweisung,726



---

INSERT-Anweisung,731  
 MERGE-Anweisung,732  
 Datenintegrität  
   Auswirkungen von nicht-serialisierbaren  
   Zeitplänen,924  
   erzwingen,868  
   gewährleisten,849  
   Info,849  
   Informationen in den Systemtabellen,877  
   Integritätsregeln,851  
   prüfen,874  
   Spalten-Standardwerte,852  
   Tools für die Aufrechterhaltung der  
   Datenintegrität,850  
   verlieren,872  
 Datenkonsistenz  
   Dirty Reads,895  
   Dirty Reads und Sperren,915  
   Dirty Reads, praktische Einführung,928  
   ISO SQL-Standard,895  
   Isolationsstufe 0,915  
   Korrektheit,924  
   mit Sperren garantieren,906  
   Phantomzeilen,895  
   Phantomzeilen und Sperren,916  
   praktische Einführung in nicht wiederholbare  
   Lesevorgänge,932  
   praktische Einführung in Phantomzeilen,938  
   praktische Einführung zu Phantomsperrern,945  
   Snapshot-Isolation,917  
   wiederholbare Lesevorgänge,895  
   wiederholbare Lesevorgänge und Sperren,915  
 Datenquellen  
   externe Server,824  
 Datentypen  
   Aggregatfunktionen,472  
   benutzerdefiniert,865  
   entfernte Prozeduren,814  
   EXCEPT-Klausel,486  
   in Sybase Central erstellen,865  
   INTERSECT-Klausel,486  
   löschen,867  
   mit SQL erstellen,865  
   rekursive Unterabfragen,545  
   Spalten zuordnen,865  
   Transact-SQL-Zeitstempel,661  
   UNION-Klausel,486  
 Datentypkonvertierungen  
   IBM DB2,832  
   Microsoft SQL Server,838  
   ODBC und ASE,830  
   Oracle Database,845  
   SAP HANA,834  
 Datenwiederherstellung  
   importieren und exportieren,722  
   Transaktionen,625  
 Datenwürfel  
   Info,560  
 Datum und Uhrzeit, Standardwerte  
   Info,854  
 Datumsangaben  
   Einführung in Suchbedingungen,320  
   Eingaberegeln,316  
   Prozeduren und Trigger,147  
   SQL Anywhere-Implementierung,644  
   suchen,316  
 Dauer  
   Sperren,907  
 DB2ODBC-Serverklasse  
   Info,832  
 DB\_PROPERTY-Funktion  
   Info,202  
 dbinit, Dienstprogramm  
   Transact-SQL-kompatible Datenbanken  
   erstellen,658  
 dbisql, Dienstprogramm  
   Datenbanken neu aufbauen,763  
 dbo, Benutzer  
   Adaptive Server Enterprise,653  
 dbspaces  
   verwalten,652  
 dbunload, Dienstprogramm  
   Tabellendaten exportieren,770  
   Tabellenschema exportieren,771  
   verwenden,757  
 dbxtract, Dienstprogramm  
   Daten extrahieren,774  
 DDL  
   automatisches Festschreiben,882  
   bei Transaktionen mit Snapshot-Isolation nicht  
   erlaubte Anweisungen,889  
   Info,1  
   Parallelität,954  
 Deadlock, Systemereignis  
   verwenden,904  
 Deadlocks

- Diagnose,904
- Gründe für,903
- in Sybase Central anzeigen,905
- Info,901
- melden,904
- praktische Einführung,265
- Transaktionsblockierung,903
- Deaktivieren
  - materialisierte Ansichten,68
  - reguläre Ansichten mit SQL,54
  - reguläre Ansichten mit Sybase Central,52
  - Trigger-Vorgänge,107
- Deaktiviert
  - Status von materialisierten Ansichten,76
- Debugger
  - Anforderungen,971
  - Aufruf-Stack anzeigen,982
  - Breakpoint-Bedingungen bearbeiten,980
  - Breakpoint-Status ändern,979
  - Breakpoints,977
  - Breakpoints setzen,978
  - Funktionen,971
  - HTTP-Funktionen,971
  - Info,971
  - Praktische Einführung,972
  - SOAP-Funktionen,971
  - Variable prüfen,981
  - Variablen anzeigen,981
  - Verbindungen,983
- Debugging
  - Praktische Einführung,972
  - Privilegien,971
- DECLARE-Anweisung
  - Prozeduren,133
  - Prozeduren für die Cursor-Verwaltung,128
  - zusammengesetzte Anweisungen,119
- default\_char
  - Standard-CHAR-Textkonfigurationsobjekt,414
  - Textkonfigurationsobjekte,406
- default\_nchar
  - Standard-NCHAR-Textkonfigurationsobjekt,414
  - Textkonfigurationsobjekte,406
- Definierer
  - effektiver Benutzer und angemeldeter Benutzer,82
  - Festlegen des von einer Datenbank benutzten Sicherheitsmodells,88
  - Prozeduren und Funktionen mit Aufruferprivilegien ausführen,82
  - Systemprozeduren vor Version 16.0 als Aufrufer oder Definierer ausführen,84
- Defragmentierung
  - alle Tabellen in einer Datenbank,249
  - einzelne Tabellen in einer Datenbank,249
  - Festplatte,249
  - Info,248
- delayed\_commits, Option
  - Tipps zur Performance-Verbesserung,248
- DELETE-Anweisung
  - Fehler,875
  - Prüfung der referenziellen Integrität bei DELETE,875
  - sperren während,921
  - Verwendung,638
- demo.db
  - Schema,493
- DENSE\_RANK-Funktion
  - äquivalente mathematische Formel,598
  - Verwendung,592
- DerivedTable, Planelement
  - Abkürzungen im Plan,356
- Derzeit aktiv, Statistik
  - Beschreibung,213
- Deterministische Funktionen
  - definiert,339
  - Nebenwirkungen,339
- Devices
  - verwalten,652
- Diagnoseprotokollierung
  - Daten interpretieren,188
  - Datenbank protokollieren,171
  - Datenbankeigenschaften in Verbindung mit der Protokollierung,172
  - externe Protokollierungsdatenbank erstellen,193
  - Info,170
  - konfigurieren,172
  - Produktionsdatenbank,171
  - Protokollierungsbedingungen,181
  - Protokollierungsbereiche,174
  - Protokollierungseinstellungen ermitteln,182
  - Protokollierungseinstellungen konfigurieren,184
  - Protokollierungseinstellungen während einer Protokollierungssitzung ändern,185
  - Protokollierungssitzung erstellen,186
  - Protokollierungsstufen,173
  - Protokollierungstypen,176
- Diagnoseprotokollierung konfigurieren

- 
- Info,172
  - Diagnoseprotokollierung, Typen
    - OPTIMIZATION\_LOGGING,176
    - OPTIMIZATION\_LOGGING\_WITH\_PLANS,176
  - Diagnoseprotokollierungssitzung erstellen,186
  - Diagnoseprotokollierungsstufen anpassen,175
    - Einstellung,184
    - Info,173
  - Diagnoseprotokollierungstypen
    - Info,176
  - Diagramme
    - mit Systemmonitor,203
  - Dienstprogramm Initialisierung (dbinit)
    - Transact-SQL-kompatible Datenbanken erstellen,658
  - Dienstprogramm, Entladen
    - Daten exportieren,750
    - Tools für den Neuaufbau,763
  - Direkte Abhängigkeit
    - Info,45
  - Direkte Referenzen
    - Datenbankobjekte,45
  - Direktive Diagramme
    - Info,550
  - Dirty Reads
    - Inkonsistenzen Dirty Reads,895
    - praktische Einführung,928
    - Sperren bei Abfragen,915
    - versus Isolationsstufen,895
  - DISK-Anweisungen
    - nicht unterstützt,652
  - DiskRead, Eigenschaft
    - Knotenstatistiken-Feldbeschreibungen,368
    - Statistiken in Zugriffsplänen,370
  - DiskReadIndInt, Eigenschaft
    - Statistiken in Zugriffsplänen,370
  - DiskReadIndLeaf, Eigenschaft
    - Statistiken in Zugriffsplänen,370
  - DiskReadTable, Eigenschaft
    - Statistik in Zugriffsplänen,370
  - DiskReadTime
    - Knotenstatistiken-Feldbeschreibungen,368
  - DiskWrite, Eigenschaft
    - Knotenstatistiken-Feldbeschreibungen,368
    - Statistiken in Zugriffsplänen,370
  - DiskWriteTime
    - Knotenstatistiken-Feldbeschreibungen,368
  - DistH, Planelement
    - Abkürzungen im Plan,356
  - DISTINCT, Schlüsselwort
    - Aggregatfunktionen,473
  - Distinct-Eliminierung
    - Info,241
  - DISTINCT-Klausel
    - doppelte Ergebnisse eliminieren,304
    - unnötige DISTINCT-Eliminierung,241
  - Distinct-Liste
    - Element in Ausführungsplänen,376
  - Distinguished Name (DN)
    - LDAP-Benutzerauthentifizierung,957
  - DistO, Planelement
    - Abkürzungen im Plan,356
  - DML
    - Info,623
    - Privilegien,624
    - Verwendung in Abfragen,307
  - DN
    - Distinguished Name, LDAP-Benutzerauthentifizierung,957
  - Dokumente
    - einfügen,630
    - Volltextsuche,448
    - Volltextsuche, Flusststeuerung,449
  - Domäne erstellen, Assistent
    - verwenden,865
  - Domänen
    - Anwendungsbeispiele,865
    - auf Spalten anwenden,866
    - CHECK-Integritätsregeln,861
    - CREATE DOMAIN,865
    - Groß- und Kleinschreibung,660
    - in Sybase Central erstellen,865
    - löschen,867
    - mit SQL erstellen,865
    - Spalten zuordnen,865
  - Doppelpunkte zum Trennen von Join-Strategien
    - Info,342
  - Doppelte Ergebnisse
    - eliminieren,304
  - DROP CONNECTION-Anweisung
    - Syntax,106
    - verwenden,816,817
  - DROP DATABASE-Anweisung

- Adaptive Server Enterprise,651
- DROP EXTERNLOGIN-Anweisung
  - verwenden,804
- DROP INDEX-Anweisung
  - mit Snapshot-Isolation nicht verfügbar,889
- DROP SERVER-Anweisung
  - Fremdserver löschen,792
  - Verzeichniszugriffsserver löschen,802
- DROP TABLE-Anweisung
  - Proxy-Tabellen von Verzeichniszugriffsservern löschen,802
- DROP-Anweisung
  - automatisches Festschreiben,882
  - Parallelität,954
- DT, Planelement
  - Abkürzungen im Plan,356
- DUMP DATABASE-Anweisung
  - nicht unterstützt,651
- DUMP TRANSACTION-Anweisung
  - nicht unterstützt,651
- Durchsuchen
  - reguläre Ansichten,55
- Dynamische Cachedimensionierung
  - Info über,224
  - Tipps zur Performance-Verbesserung,224
  - Unix,225
  - Windows,225
  - Windows Mobile,225

## E

- EAH, Planelement
  - Abkürzungen im Plan,356
- EAM, Planelement
  - Abkürzungen im Plan,356
- Effektiver Benutzer
  - im Gegensatz zum angemeldeten Benutzer,82
- Effizienz
  - Daten importieren,723
  - verbessern und Sperren,926
- EH, Planelement
  - Abkürzungen im Plan,356
- Eigentümer
  - Tabelle ändern,6
  - Verzeichniszugriffsserver,795
- Eindeutige Bezeichner
  - Tabellen,17
- Eindeutige Ergebnisse

- begrenzen,304
- Eindeutige Schlüssel
  - Generierung und Parallelität,949
- Eindeutigkeit
  - mit einem Index erzwingen,40
- Eindeutigkeits-Integritätsregeln
  - generierte Indizes,28
- Eindeutigkeits-Integritätsregeln konfigurieren
  - Info,863
- Einfache Abfragen
  - Info,331
- Einfügen
  - Daten in alle Spalten,627
  - Daten in Integritätsregeln,627
  - Daten in Standardwerte,627
  - NULL-Verhalten bei nicht festgelegten Spalten,627
  - Spaltendaten, INSERT-Anweisung,627,628
- Einfügen, Daten
  - mit SELECT,629
- Einfügesperren
  - Info,914
- Einführungen
  - Debugger,972
- Einschränkungen
  - Ferndatenzugriff,822
  - Ferndatenzugriff-Zeichensatzkonvertierung,787
  - Info,297
  - manuelle Ansichten in Sofortansichten ändern,62
- Einstellung
  - Diagnoseprotokollierungsstufen,184
- Einstellungen für die Diagnoseprotokollierung während einer Protokollierungssitzung ändern
  - Info,185
- Eintrittspunkte
  - externe Begriffsegmentierer,469
  - externe Vorfilter,468
- Einzeilige Unterabfragen
  - Info,600
- Element, Direktive
  - anwenden,701
- Elemente
  - Abfrageergebnisse als XML erhalten,688
  - XML aus relationalen Daten generieren,678
  - XML in Datenbanken speichern,677
- EM, Planelement
  - Abkürzungen im Plan,356
- Empfohlene Indizes, Registerkarte

- 
- Indexberater, Ergebnisse,167
  - Entfernen
    - Domänen,867
    - Fremdserver,792
    - Trigger,106
    - Verzeichniszugriffsserver,802
  - Entfernte Daten
    - für den Ferndatenzugriff nicht unterstützte Funktionen,822
    - nicht unterstützte Funktionen,822
    - Standort von Proxy-Tabellen angeben,806
    - zugreifen auf,785
    - Zuordnungen entfernter Tabellen,786
  - Entfernte Funktion
    - Datentypen,814
  - Entfernte Prozedur erstellen, Assistent
    - verwenden,815
  - Entfernte Prozeduraufrufe
    - Info,814
  - Entfernte Prozeduren
    - Aufrufe,814
    - Datentypen,814
    - erstellen,815
    - löschen,816,817
  - Entfernte Tabellen
    - auf einem Server auflisten,794
    - Info,786
    - Joins,811
    - Spalten auflisten,810
    - zugreifen auf,785
  - Entfernte Transaktionsverwaltung
    - Überblick,818
  - Entität-Integrität
    - Primärschlüssel,644
  - Entitäten
    - Integrität erzwingen,868
  - Entitätsintegrität
    - UPDATE-Anweisung,636
    - von Clientanwendung verletzt,868
  - Entlade-Tools
    - Assistent Datenbank entladen,750
    - Daten entladen-Fenster,752
    - Info,743
  - Entladen
    - Info,763
  - Entladen einer Datenbank
    - in Sybase Central,750
  - Entladen und neu laden
    - Datenbanken,773
  - Entladen und Neuladen
    - an Synchronisationssystemen beteiligte Datenbanken,767,769
    - nicht an Synchronisationssystemen beteiligte Datenbanken,766
  - Entladen von Datenbanken
    - exportieren,756
  - Entscheidungsunterstützung
    - Isolationsstufen,924
  - Entschlüsseln
    - materialisierte Ansichten mit Sybase Central,71
  - Enumerationsphase
    - Abfrageverarbeitung,330
  - Equi-Joins
    - Info,501
  - Ereignisse
    - Ergebnisse der Profilerstellung generieren und prüfen,158
    - zulässige Anweisungen,148
  - Ergebnismengen
    - Abfrage mehr als einmal ausführen,323
    - Anzahl der Zeilen beschränken,483
    - aus gespeicherten Prozeduren zurückgeben,124
    - entfernte Prozeduren,814
    - Fehlerbehebung,323
    - in eine Datei speichern,783
    - mehrere aus Prozeduren zurückgeben,124
    - mehrfache,126
    - Transact-SQL,672
    - Variable,127
  - Ergebnisse
    - des Indexberaters verstehen,167
  - Ergebnisse aus Prozeduren zurückgeben
    - Info,121
  - Ergebnisse der Prozedurprofilerstellung analysieren
    - Info,162
  - Ersatzzeilen
    - Info,919
  - Erstellen
    - benutzerdefinierte Funktionen,94
    - Datentypen in Sybase Central,865
    - Datentypen mit SQL,865
    - Diagnoseprotokollierungssitzung,186
    - Domänen in Sybase Central,865
    - Domänen mit SQL,865
    - entfernte Prozeduren,815
    - Externe Logins,804

- externe Protokollierungsdatenbank,193
- Fremdserver ,788
- Proxy-Tabellen in Sybase Central,807
- Prozeduren,89
- Sequenzen,952
- Spalten-Standardwerte,853
- temporäre Prozeduren,89
- Transact-SQL-kompatible Tabellen,664
- Trigger (SQL),101
- Trigger (Sybase Central),100
- Erzeugungswerte
  - Element in Ausführungsplänen,375
- EstCpuTime
  - Schätzungen in Zugriffsplänen,371
- EstDiskReads
  - Schätzungen in Zugriffsplänen,371
- EstDiskReadTime
  - Schätzungen in Zugriffsplänen,371
- EstDiskWrites
  - Schätzungen in Zugriffsplänen,371
- EstRowCount
  - Schätzungen in Zugriffsplänen,371
- EstRunTime
  - Schätzungen in Zugriffsplänen,371
- Excel
  - Architekturkonflikt,746
  - Daten in eine SQL Anywhere-Datenbank exportieren,746,747
  - Daten in eine SQL Anywhere-Datenbank importieren,728
  - Ferndatenzugriff,843
- Excel-Dateien
  - Unterstützung für externen Vorfilter und und Begriffsegmentierer-Bibliothek ,449
- EXCEPT-Klausel
  - Abfragen kombinieren,486
  - NULL,489
  - Regeln,487
  - Transact-SQL-Kompatibilität,665
  - verwenden,487
- Exchange, Planelement
  - Abkürzungen im Plan,356
- EXECUTE IMMEDIATE-Anweisung
  - Prozeduren,143
  - WITH RESULT SET-Klausel,143
- Existenztest
  - Info,614
  - Verneinung von,615
- EXISTS, Operator
  - Info,614
- Exklusive Tabellensperren
  - Info,912
- EXPLICIT, Modus
  - Abfragen schreiben,698
  - cdata-Direktive verwenden,704
  - Element-Direktive verwenden,701
  - Hide-Direktive verwenden,702
  - Syntax,696
  - verwenden,696
  - xml-Direktive verwenden,703
- Export-Assistent
  - verwenden,744
- Export-Tools
  - Dienstprogramm Entladen,750
  - Info,743
  - Interactive SQL Export-Assistent,744
  - OUTPUT-Anweisung,745
  - UNLOAD TABLE-Anweisung,748
  - UNLOAD-Anweisung,749
- Exportieren
  - Abfrageergebnisse,744,753,754
  - Ansichten,758
  - ASE-Kompatibilität,784
  - Info,743
  - NULL,755
  - relationale Daten als XML,679
  - Schemata,771
  - Tabellen,758
- Exportieren von Daten
  - mit UNLOAD-Anweisung in Datei,749
- Exportieren von Datenbank
  - Sybase Central,756
- Exportieren von Datenbanken
  - Befehlszeile,757
- Externe Begriffsegmentierer-Bibliotheken
  - Einstellungen für Textkonfigurationsobjekte,407
- Externe Logins
  - entfernen,804
  - erstellen,804
  - Fremdserver,804
  - Info,804
  - löschen,804
- Externe Server
  - ODBC,824
- Externer Vorfilter-Bibliotheken
  - Einstellungen für Textkonfigurationsobjekte,412

- 
- Externes Laden
    - Info,721
  - Externes Login erstellen, Assistent verwenden,804
  - Extrahieren
    - Datenbanken für SQL Remote,774
  - F**
  - FALSE, Bedingungen
    - NULL,318
  - FASTFIRSTROW, Tabellen-Hint
    - Ziel des Optimierers auswählen,228
  - Fehler
    - bei DELETE und UPDATE,875
    - Konvertierung,739
    - Transact-SQL,673
    - Transact-SQL-Verhalten mit Watcom SQL nachahmen,675
  - Fehlerbehandlung
    - Anwendungsprofilerstellung,189
    - ANY-Operator,612
    - Deadlocks,904
    - Ferndatenzugriff,822
    - GROUP BY-Klausel,326
    - natürliche Joins,521
    - ON EXCEPTION RESUME,133
    - Performance,219
    - Prozeduren und Trigger,132
  - Fehlerbehebung
    - Ergebnismenge scheint sich zu ändern,323
  - Fehlersuche
    - Anforderungen,971
    - Anwendungslogik,191
    - HTTP-Prozeduren,971
    - Info,971
    - SQL Anywhere-Debugger verwenden,971
  - Fehlersuchmodus
    - verwenden,971
  - Fenster (OLAP)
    - Auswirkung von ORDER BY-Klausel auf Standardwerte,567
    - Größe,566
    - Größe mit der RANGE -Klausel festlegen,566
    - Größe mit der ROWS-Klausel festlegen,566
    - Inline-Fenster definieren,564
    - Inlining und die WINDOW-Klausel,568
    - Klausel WINDOW in der SELECT-Anweisung,564
    - Reihenfolge der Klauselberechnung ,564
    - Standardwerte, wenn das Fenster nur teilweise definiert ist,567
  - Fenster-Aggregatfunktionen
    - Info,571
    - Liste der unterstützten Funktionen,571
    - OLAP,571
  - Fensterfunktionen
    - Aggregat, Liste,571
    - Info,564
    - Rangfunktionen, Liste,590
    - Zeilennummerierung,597
  - Ferndatenzugriff
    - Abfragenormalisierung,819
    - allgemeine Probleme bei Abfragen,823
    - Durchreichmodus,813
    - Einführung,785
    - Fehlerbehandlung,822
    - Fremdserver,787
    - Groß-/Kleinschreibung,822
    - Grundkonzepte,785
    - interne Vorgänge,818
    - komplettes Passthrough der Anweisung,820
    - Lotus Notes SQL,844
    - Microsoft Access,836
    - Microsoft Excel,843
    - Microsoft FoxPro,844
    - partielltes Passthrough der Anweisung,820
    - Performance-Einschränkungen,785
    - selbstblockierende Abfragen,823
    - Serverfunktionalität,819
    - Sybase IQ,836
    - syntaktische Analyse der Abfragen,819
    - Systemanbindungsprobleme,823
    - Verbindungsnamen,823
    - Vorverarbeitung der Abfragen,819
    - Zeichensatzkonvertierung, Einschränkung ,787
  - Festgeschriebene Daten lesen
    - Arten von Inkonsistenz,895
    - Einführung,885
    - Einstellung für ODBC,898
    - SELECT-Anweisungen,915
  - Festplatten-I/O-Vorgänge, Statistik
    - Liste,211
  - Festplatten-Lesezugriffe, Statistik
    - Liste,211

- Festplatten-Schreibzugriffe, Statistik
  - Liste, 212
- Festschreibung
  - wait\_for\_commit, Option, 919
- Festschreibungen verzögern
  - Tipps zur Performance-Verbesserung, 248
- FETCH-Anweisung
  - Prozeduren für die Cursor-Verwaltung, 128
- fetchst
  - Info, 245
- file\_name
  - Verzeichniszugriffsserver, 795
- Filter, Planelement
  - Abkürzungen im Plan, 356
- FIRST-Klausel
  - verwenden, 483
- First-row, Optimierungsziel
  - Ziel des Optimierers auswählen, 228
- FIRST\_VALUE-Funktion
  - Beispiele, 582
  - Verwendung, 572
- FirstRowRunTime
  - Knotenstatistiken-Feldbeschreibungen, 368
- FOR BROWSE-Klausel
  - Transact-SQL SELECT-Anweisung, nicht unterstützte Syntax, 665
- FOR JSON AUTO
  - Syntax, 716
  - verwenden, 716
- FOR JSON EXPLICIT
  - Syntax, 717
  - verwenden, 717
- FOR JSON RAW
  - Syntax, 716
  - verwenden, 716
- FOR JSON-Klausel
  - Info, 715
- FOR READ ONLY-Klausel
  - ignoriert, 666
- FOR XML AUTO
  - verwenden, 693
- FOR XML EXPLICIT
  - cdata-Direktive verwenden, 704
  - Syntax, 696
  - verwenden, 696
- FOR XML EXPLICIT, Modus
  - Element-Direktive verwenden, 701
  - Hide-Direktive verwenden, 702
  - xml-Direktive verwenden, 703
- FOR XML RAW
  - verwenden, 691
- FOR XML-Klausel
  - Abfrageergebnisse als XML erhalten, 688
  - AUTO-Modus verwenden, 693
  - BINARY-Datentyp, 688
  - Einschränkungen, 688
  - EXPLICIT-Modus, Syntax, 696
  - EXPLICIT-Modus verwenden, 696
  - IMAGE-Datentyp, 688
  - in Interactive SQL anzeigen, 705
  - LONG BINARY-Datentyp, 688
  - RAW-Modus verwenden, 691
  - VARBINARY-Datentyp, 688
  - Verwendung, 688
- FOR-Anweisung
  - Steueranweisungen, 118
- FOR-Klausel
  - Abfrageergebnisse als XML erhalten, 688
  - FOR XML AUTO verwenden, 693
  - FOR XML EXPLICIT verwenden, 696
  - FOR XML RAW verwenden, 691
- FORCE NO OPTIMIZATION-Klausel
  - Eignung zum Überspringen von Abfrageverarbeitungsphasen, 332
- FORCE OPTIMIZATION-Klausel
  - Eignung zum Überspringen von Abfrageverarbeitungsphasen, 332
- Formeln
  - OLAP, Aggregatfunktionen, 598
- FORWARD TO-Anweisung
  - native Anweisungen, 813
  - native Anweisungen an Fremdserversenden, 813
- FoxPro
  - Ferndatenzugriff, 844
- Fragmentierung
  - Dateien, 249
  - Fragmentierung, Registerkarte, 251
  - für Tabellen verringern, 249
  - Indizes, 254
  - Indizes, praktische Einführung, 277
  - Info, 248
  - Tabellen, 249
  - Tabellen, praktische Einführung, 281
  - Tabellenfragmentierung, 249
- Freigabe von Sperren
  - Ausnahmen, 922



- 
- Fremdschlüssel
    - einfügen,875
    - generierte Indizes,28
    - in Sybase Central anzeigen,23
    - in Sybase Central erstellen,23
    - Indizes,644
    - Integrität,644
    - MATCH-Klausel,644
    - mit SQL ändern,24
    - mit SQL erstellen,24
    - notwendig/optional,870
    - Performance,258
    - referenzielle Integrität,872
    - referenzielle Zyklen,871
    - Rollenname,525
    - Schlüssel-Joins,523
    - Sortierfolge,644
    - Terminologie,21
    - verwalten,21
    - Waisen,21
    - zusammengesetzte Fremdschlüssel,21
  - Fremdschlüssel erstellen, Assistent
    - verwenden,23
  - Fremdserver
    - Advantage Database Server,828
    - ändern,793
    - ASE ODBC,829
    - Eigenschaften auflisten,794
    - erstellen,788
    - erstellen, Assistent zum Erstellen von Fremdservern,789
    - externe Logins,804
    - Funktionalität des Fremdservers anzeigen,794
    - IBM DB2,832
    - in Sybase Central erstellen,789
    - Klassen,824
    - löschen,792
    - Lotus Notes SQL,844
    - Microsoft Access,836
    - Microsoft Excel,843
    - Microsoft FoxPro,844
    - MIRROR,826
    - mit Fremdservern arbeiten,787
    - MySQL,840
    - native Anweisungen senden,813
    - nicht unterstützt für UltraLite unter Mac OS X,827
    - ODBC,842
    - Oracle Database,845
    - SAP HANA,834
    - SQL Anywhere, SAODBC,826
    - SQL Server,838
    - Sybase IQ, IQODBC,836
    - Tabellen auf einem Fremdserver auflisten,794
    - Transaktionsverwaltung,817
    - UltraLite,827
  - Fremdserver erstellen, Assistent
    - verwenden,789
  - FROM-Klausel
    - abgeleitete Tabellen in,306
    - gespeicherte Prozeduren in,307
    - Info,305
    - Isolationsstufen,885
  - Full-Outer-Joins
    - SQL Anywhere-Implementierung,645
  - FullCompare, Eigenschaft
    - Statistiken in Zugriffsplänen,370
  - FullOuterHashJoin, Planelement
    - Abkürzungen im Plan,356
  - Funktionen
    - Assistent zum Erstellen von Funktionen,94
    - Caching,339
    - Ergebnisse der Profilerstellung generieren und prüfen,158
    - Fenster,571
    - Fenster-Rangfunktionen (OLAP),590
    - idempotent oder deterministisch,339
    - Inlining,241
    - mit benutzerdefinierten Funktionen arbeiten,93
    - SOUNDEX-Funktion,321
    - TRACEBACK,133
    - TSEQUAL,662
  - Funktionen für lineare Regression
    - OLAP,588
  - Fuzzy
    - Info,401
    - Volltextsuche, praktische Einführung,433
    - wie der Datenbankserver eine Fuzzy-Suche interpretiert,414
- ## G
- Gemeinsame Sperren
    - Info,909
  - Gemeinsame Tabellensperren
    - Info,911
  - GENERIC, Textindizes

- Präfixsuchen,396
- GENERIC-Textindizes
  - Volltextsuche, praktische Einführung,426
- Generieren
  - eindeutige Schlüssel,949
- Generischer Begriffsegmentieralgorithmus
  - Einstellungen für Textkonfigurationsobjekte,407
- Gesamtvorteile
  - Indexberater, Ergebnisse,168
- Geschäftsregeln
  - Info,849
- Geschätzt aktiv, Statistik
  - Beschreibung,213
- Geschätzte Blattseiten
  - Element in Ausführungsplänen,374
- Geschätzte Cacheseiten
  - Optimiererstatistiken-Feldbeschreibungen,362
- Geschätzte Seiten
  - Element in Ausführungsplänen,373
- Geschätzte Zeilen
  - Element in Ausführungsplänen,373
- Geschätzte Zeilengröße
  - Element in Ausführungsplänen,373
- Gespeicherte Prozedur, Sprache
  - Überblick,669
- Gespeicherte Prozeduren
  - Caching von Anweisungen,337
  - Debugging, praktische Einführung,972
  - Ergebnisse der Profilerstellung generieren und prüfen,158
  - in allgemeinen Tabellenausdrücken,541
  - Sybase Central zur Konvertierung von gespeicherten Prozeduren verwenden,671
  - Transact-SQL, gespeicherte Prozeduren, Übersicht,669
  - verglichen mit Batches,112
  - verwenden in der FROM-Klausel,307
- Gespeicherte Prozeduren in Transact-SQL
  - Überblick,669
- Gesperrte Tabellen
  - Elemente in Zugriffsplänen,373
- Gestreute Lesevorgänge
  - Performance,261
- Gleich-Operator
  - Vergleichsoperator in WHERE-Klauseln,310
- Gleichzeitige Transaktionen
  - blockieren,902
- GLOBAL AUTOINCREMENT
  - Standardwert,856
  - Unterschiede zu Sequenzen,950
- Globale temporäre Tabellen
  - gemeinsam genutzt,10
  - Info,10
  - nicht gemeinsam genutzt,10
  - Tabellenstrukturen zusammenführen,741
- Globale Variable
  - Debugger,982
- Globales Autoincrement
  - verglichen mit GUID und UUID,858
- go
  - Trennzeichen in Batchanweisungen,112
- Grafiken
  - einfügen,630
- Grafische Pläne
  - Abkürzungen,356
  - anpassen,347
  - anzeigen in Interactive SQL,355
  - Ausführungspläne lesen,347
  - Bypass-Abfragen,351
  - Detaillierte Knoteninformationen anzeigen,351
  - Info,347
  - Knotenstatistiken-Feldbeschreibungen,368
  - kontextsensitive Hilfe,351
  - Lokale bzw. globale Feldbeschreibungen,362
  - ohne Ausführung einer Abfrage anzeigen,342
  - Optimiererstatistiken-Feldbeschreibungen,362
  - Optimierung umgehen,351
  - Prädikat,353
  - Statistiken,348
  - Zugriff mit SQL-Funktionen,355
- GRANT-Anweisung
  - Parallelität,954
  - Transact-SQL,655
- GrByH, Planelement
  - Abkürzungen im Plan,356
- GrByHClust, Planelement
  - Abkürzungen im Plan,356
- GrByHSets, Planelement
  - Abkürzungen im Plan,356
- GrByO, Planelement
  - Abkürzungen im Plan,356
- GrByOSets, Planelement
  - Abkürzungen im Plan,356
- GrByS, Planelement
  - Abkürzungen im Plan,356
- GrBySSets, Planelement

- 
- Abkürzungen im Plan,356
  - Groß-/Kleinschreibung
    - Benutzer-IDs,660
    - Bezeichner,660
    - Daten,660
    - Datenbanken,660
    - Domänen,660
    - Ferndatenzugriff,822
    - Kennwörter,661
    - SQL,295
    - Transact-SQL-Kompatibilität,660
  - Groß-/Kleinschreibung berücksichtigen
    - ASE-kompatible Datenbanken erstellen,656
    - Tabellenamen,295
  - Größe
    - Verzeichniszugriffsserver,795
  - Größer als
    - Bereichsspezifikation,311
    - Vergleichsoperator in WHERE-Klauseln,310
  - Größer gleich
    - Vergleichsoperator in WHERE-Klauseln,310
  - GROUP BY ALL-Klausel
    - Transact-SQL SELECT-Anweisung, nicht unterstützte Syntax,665
  - GROUP BY-Klausel
    - Aggregatfunktionen,477
    - Aggregatfunktionen auf gruppierte Daten anwenden,326
    - Ausführung,474
    - Erläuterungen,474
    - Erweiterungen,554
    - Fehler,326
    - mit der WHERE- und der HAVING-Klausel verwenden,474
    - Performance,233
    - SQL-Standardwert-Entsprechung,478
    - SQL/2008-Standard,478
    - und ORDER BY,485
    - WHERE-Klausel,476
  - Group By-Liste
    - Element in Ausführungsplänen,375
  - GROUPING-Funktion
    - mit CUBE-Klausel (OLAP) verwendet,560
    - mit ROLLUP-Klausel (OLAP) verwendet,558
    - NULL-Platzhalter entdecken,562
  - Grundkonzepte für den Ferndatenzugriff
    - Überblick,785
  - Gruppen-Zugehörigkeitstest
    - =ANY,611
    - Info,621
    - Verneinung von,611
  - Gruppenlesevorgänge
    - Tabellen,260
  - Gruppieren
    - mehrere Spalten verwenden,476
    - Volltextsuche,400
  - Gruppierte Daten
    - Info,325
  - GUID
    - generieren,949
    - Standard-Spaltenwert,858
    - verglichen mit globalem Autoincrement,858
- ## H
- HANA
    - Ferndatenzugriff auf SAP HANA,834
  - HANA, Ferndatenzugriff
    - Info,834
  - HANA-Ferndatenzugriff
    - Datentypkonvertierungen,834
  - HANAODBC-Serverklasse
    - Info,834
  - HashAntisemijoin, Planelement
    - Abkürzungen im Plan,356
  - HashDistinct, Planelement
    - Abkürzungen im Plan,356
  - HashExcept, Planelement
    - Abkürzungen im Plan,356
  - HashExceptAll, Planelement
    - Abkürzungen im Plan,356
  - HashFilter, Planelement
    - Abkürzungen im Plan,356
  - HashGroupBy, Planelement
    - Abkürzungen im Plan,356
  - HashGroupBySets, Planelement
    - Abkürzungen im Plan,356
  - HashIntersect, Planelement
    - Abkürzungen im Plan,356
  - HashIntersectAll, Planelement
    - Abkürzungen im Plan,356
  - HashJoin, Planelement
    - Abkürzungen im Plan,356
  - HashSemijoin, Planelement
    - Abkürzungen im Plan,356
  - HashTableScan, Planelement

- Abkürzungen im Plan,356
- Haupt-Heap-Byte, Statistik
  - Beschreibung,218
- HAVING-Klausel
  - Gruppen von Daten sammeln,480
  - logische Operatoren,481
  - mit der GROUP BY-Klausel verwenden,474,480
  - mit oder ohne Aggregate,480
  - Performance,292
  - und WHERE-Klausel,328
  - Unterabfragen,607
- Heaps Abfrageverarbeitung, Statistik
  - Beschreibung,213
- Heaps Carver, Statistik
  - Beschreibung,213
- Heaps Neu positionierbar gesperrt, Statistik
  - Beschreibung,213
- Heaps Neu positionierbar, Statistik
  - Beschreibung,213
- Heuristik
  - Abfrageoptimierung,335
- HF, Planelement
  - Abkürzungen im Plan,356
- HFP, Planelement
  - Abkürzungen im Plan,356
- Hide, Direktive
  - anwenden,702
- Hierarchische Datenstrukturen
  - Stücklistenproblem,547
  - untersuchen,543
- Hints
  - Clustered-Indizes verwenden,31
  - Index-Hints ,27
- Hinzufügen
  - Daten in Datenbank,723
- Histogramme
  - aktualisieren,229
  - Info,334
- HOLDLOCK, Schlüsselwort
  - Transact-SQL,667
- Hostvariable
  - in Batches,112
- HTS, Planelement
  - Abkürzungen im Plan,356
- HTTP-Dienste
  - Fehlersuche,971
- HTTP-Funktionen
  - Fehlersuche,971

## I

- I/O
  - Bitmaps durchsuchen,260
- IAH, Planelement
  - Abkürzungen im Plan,356
- IAM, Planelement
  - Abkürzungen im Plan,356
- IBM DB2
  - Datentypkonvertierungen,832
  - Ferndatenzugriff auf IBM DB2,832
  - nach SQL Anywhere migrieren,774
- IBM DB2, Ferndatenzugriff
  - Info,832
- ID
  - Meta-Eigenschaftsnamen,682
- Idempotente Funktionen
  - definiert,339
- IDENTITY, Spalte
  - spezielle IDENTITY,662
  - Werte abrufen,663
- IF-Anweisung
  - Steueranweisungen,118
- IGNORE NULLS-Klausel
  - Verwendung in der Funktion LAST\_VALUE,583
- IH, Planelement
  - Abkürzungen im Plan,356
- IM, Planelement
  - Abkürzungen im Plan,356
- IMMEDIATE REFRESH-Klausel
  - Info,424
- Import-Assistent
  - Info,724
  - verwenden,739
- Import-Tools
  - Info,723
  - INPUT-Anweisung,726
  - INSERT-Anweisung,731
  - Interactive SQL Import-Assistent,724
  - LOAD TABLE-Anweisung,729
  - MERGE-Anweisung,732
  - Proxy-Tabellen,738
- Importieren
  - ASE-Kompatibilität,784
  - Info,723
  - temporäre Tabellen,10
  - Tools,723
- Importieren von Daten

- 
- Hinweise,723
  - Info,721
  - LOAD TABLE-Anweisung, Beispiel,739
  - IN, Bedingungen
    - Unterabfragen,610
  - IN, Liste
    - Optimierung,241
  - IN, Parameter
    - definiert,116
  - IN, Planelement
    - Abkürzungen im Plan,356
  - IN-Liste
    - Element in Ausführungsplänen,376
  - In-Memory-Modus
    - Tipps zur Performance-Verbesserung,247
  - IndAdd, Eigenschaft
    - Statistiken in Zugriffsplänen,370
  - Index erstellen, Assistent
    - verwenden,33
  - Index Hinz./Sek., Statistik
    - Beschreibung,213
  - Index Suchen/Sek., Statistik
    - Beschreibung,213
  - Index Volle Vergleiche/Sek., Statistik
    - Beschreibung,213
  - Index-Hints
    - Clustered-Indizes verwenden,31
  - Indexauffächerung
    - Info,40
  - Indexberater
    - bei einer Abfrage verwenden,165
    - bei einer Datenbank verwenden,165
    - Einführung,28
    - Empfehlungen für eine Abfrage erhalten,165
    - Empfehlungen für eine Datenbank abrufen,165
    - Empfehlungen verstehen,166
    - Ergebnisse bewerten,168
    - Ergebnisse implementieren,168
    - Ergebnisse verstehen,167
    - Info,164
    - MANAGE PROFILING-Systemprivileg erforderlich,164
    - Server-Status,168
    - Verbindungsstatus,168
  - Indexfragmentierung
    - Info,254
    - praktische Einführung,277
  - Indexfunktionen
    - Zeilennummerierung,597
  - Indexname
    - Element in Ausführungsplänen,374
  - IndexOnlyScan, Planelement
    - Abkürzungen im Plan,356
  - IndexScan, Planelement
    - Abkürzungen im Plan,356
  - Indexselektivität
    - Info,40
  - Indirekte Referenzen
    - Datenbankobjekte,45
  - Indizes
    - Ansicht mit Textindex abfragen,402
    - Auffächerung und Seitengrößen,261
    - bei häufig durchsuchten Spalten verwenden,27
    - berechnete Spalten,33
    - Blattseiten,40
    - Clusterbildung,31
    - die zu erstellenden Indizes ermitteln,28
    - Einschränkungen und Hinweise,27
    - Empfehlungen des Indexberaters verstehen,166
    - erstellen,33
    - Fragmentierung,254,277
    - gemeinsam genutzte physische Indizes ermitteln,38
    - Indexberater verwenden,164
    - Info,27
    - Kataloge,37
    - Korrelationen zwischen,169
    - Kosten und Vorteile,164
    - logische,37
    - löschen,36
    - mit Indizes arbeiten,27
    - mit Sybase Central neu erstellen,35
    - mit Sybase Central validieren,34
    - mögliche,167
    - nicht benutze,168
    - Nutzen und Sperren,926
    - Optimierung,27
    - Performance der HAVING-Klausel,292
    - Performance der WHERE-Klausel,292
    - Performance verbessern,233
    - physische,37
    - Prädikat-Analyse,292
    - Prädikate, sargable (als Suchargument nutzbar),292
    - Schiefe,254
    - Seitengrößen,261
    - Spaltenreihenfolge,29
    - SQL Anywhere-Implementierung,645

- Statistiklisten,213
- Struktur,40
- temporäre Tabellen,29
- Textindizes,423
- Transact-SQL-Kompatibilität und Benennung,661
- verwenden,27
  - zum Erfüllen eines Prädikats verwenden,231
  - zusammengesetzte,29
- Indizes gemeinsam nutzen
  - Info,37
- IndLookup, Eigenschaft
  - Statistiken in Zugriffsplänen,370
- Inhalt
  - Verzeichniszugriffsserver,795
- Initialisieren
  - materialisierte Ansichten,66
- Inkonsistenzen
  - Auswirkungen von nicht-serialisierbaren Zeitplänen,924
  - bei nicht wiederholbaren Lesevorgängen,915
  - Dirty Reads,895
  - Dirty Reads und Sperren,915
  - Dirty Reads, praktische Einführung,928
  - ISO SQL-Standard,895
  - nicht wiederholbare Lesevorgänge,895
  - Phantomzeilen,895
  - Phantomzeilen und Sperren,916
  - praktische Einführung in nicht wiederholbare Lesevorgänge,932
  - praktische Einführung in Phantomzeilen,938
  - praktische Einführung zu Phantomsperren,945
  - Sperren vermeiden,906
- Inlining
  - benutzerdefinierte Funktionen,241
  - einfache gespeicherte Prozeduren,241
- InList, Planelement
  - Abkürzungen im Plan,356
- Inner-Joins
  - Info,504
  - Join-Eliminierung durch Neuschreibungsoptimierung,241
  - SQL Anywhere-Implementierung,645
- INOUT, Parameter
  - definiert,116
- INPUT-Anweisung
  - Excel,728
  - materialisierte Ansichten,727
  - Textindizes,727
  - verwenden,728
- INSERT, Trigger
  - als Folge von INPUT-Anweisungen ausgelöst,721
- INSERT-Anweisung
  - Daten in alle Spalten einfügen,627
  - Daten in bestimmte Spalten einfügen,628
  - Hinweise zu materialisierten Ansichten,731
  - Hinweise zu Textindizes,731
  - Prüfung der referenziellen Integrität bei INSERT,874
  - redundante Daten,874
  - SELECT,626
  - sperren während,918
  - verwenden,731
  - zum Ändern von Daten verwenden,638
  - zum Hinzufügen von Daten,626
- INSTEAD OF-Trigger
  - Info,110
  - Rekursion,110
  - zum Aktualisieren von Ansichten,111
- instest
  - Info,245
- Integrität
  - erzwingen,868
  - Info,849
  - Informationen in den Systemtabellen,877
  - Integritätsregeln implementieren,851
  - prüfen,874
  - Spalten-Standardwerte,852
  - Tools für die Aufrechterhaltung der Datenintegrität,850
  - verlieren,872
- Integritätsprüfungen
  - CHECK-Integritätsregel,851
  - NOT NULL-Integritätsregel,851
  - RI-Integritätsregeln,851
  - Tabellen- und Spalten-Integritätsregeln,851
  - Trigger,851
- Integritätsregeln
  - CHECK-Integritätsregeln,861
  - Datenintegrität,859
  - eindeutige Integritätsregeln,863
  - in Sybase Central,862
  - Info,859
  - Integrität,859
  - referenzielle Integrität,859
- Integritätsregeln zur Erhaltung der referenziellen Integrität

---

Tools für die Aufrechterhaltung der  
Datenintegrität,851

Interactive SQL

- Abfrageergebnisse exportieren,744,753,754
- Änderungen in Transaktionen  
zusammenfassen,882
- Batch-Modus,782
- Batch-Vorgänge,782
- beenden,882
- Daten in Tabellen anzeigen,10
- Datenbanken neu aufbauen,763
- grafische Pläne anzeigen,355
- Indexberater,165
- Liste von Tabellen anzeigen,492
- mehrere Ergebnismengen anzeigen,126
- relationale Daten als XML exportieren,678
- Skriptdateien,780
- Skripten ausführen,780
- SQL-Skripten laden,783
- Tabellen importieren,739
- Trennzeichen für Anweisungen,146

Interne Vorgänge

- Ferndatenzugriff,818

Internes Laden

- Info,721

INTERSECT-Klausel

- Abfragen kombinieren,486
- NULL,489
- Regeln,487
- Transact-SQL-Kompatibilität,665
- verwenden,487

INTO CLIENT FILE-Klausel

- importieren von und exportieren auf  
Clientcomputer,761

INTO VARIABLE-Klausel

- importieren von und exportieren auf  
Clientcomputer,761

INTO-Klausel

- verwenden,122

Invocations

- Knotenstatistiken-Feldbeschreibungen,368

IO, Planelement

- Abkürzungen im Plan,356

IPerformance

- Indizes,27

IQODBC-Serverklasse

- Info,836

IS NULL, Schlüsselwort

Info,318

ISNULL-Funktion

- Info,318

ISO SQL, Standards

- Parallelität,895
- typische Inkonsistenzen,895

ISOLATION-Klausel

- Transact-SQL SELECT-Anweisung, nicht  
unterstützte Syntax,665

isolation\_level, Option

- Optimiererstatistiken-Feldbeschreibungen,362

Isolationsstufe 0

- Dirty Reads, praktische Einführung,928
- Info,885
- SELECT-Anweisung, Sperren,915

Isolationsstufe 1

- Info,885
- praktische Einführung in nicht wiederholbare  
Lesevorgänge,932
- SELECT-Anweisung, Sperren,915

Isolationsstufe 2

- Info,885
- Phantomzeilen, praktische Einführung,938
- praktische Einführung zu Phantomsperren,945
- SELECT-Anweisung, Sperren,916

Isolationsstufe 3

- Info,885
- Phantomzeilen, praktische Einführung,938
- SELECT-Anweisung, Sperren,916

Isolationsstufe: Anweisungs-Snapshot

- Info,885

Isolationsstufe: Festgeschriebene Anweisungen lesen

- Info,885

Isolationsstufe: Nicht festgeschriebene Anweisungen  
lesen

- Info,885

Isolationsstufe: Schreibgeschützter Anweisungs-  
Snapshot

- Info,885

Isolationsstufe: Serialisierbar

- Info,885

Isolationsstufe: Snapshot

- Info,885

Isolationsstufe: Wiederholbare Lesevorgänge

- Info,885

Isolationsstufen

- ändern,897
- anzeigen,901

- Arten von Inkonsistenz,895
- auswählen,922
- einstellen,897
- Implementierung auf Stufe 0,915
- Implementierung auf Stufe 1,915
- Implementierung auf Stufe 2,916
- Implementierung auf Stufe 3,916
- Info,885
- jeweils typische Transaktionen,925
- ODBC,898
- Parallelität bei den Stufen 2 und 3 verbessern,925
- praktische Einführung,926
- Snapshot-Isolationsstufe wählen,923
- versus typische Transaktionen,924
- wählen,922
- während einer Transaktion ändern,900
- Isolationsstufen und Konsistenz
- Info,885
- ISYSFKEY
  - Verwendung der Systemtabelle,37
- ISYSIDX
  - Index gemeinsam nutzen,38
  - Systemtabelle verwenden,37
- ISYSIDXCOL
  - Verwendung der Systemtabelle,37
- ISYSPHYSIDX
  - Index gemeinsam nutzen,38
  - Systemtabelle verwenden,37
- J**
- JDBC
  - materialisierte Ansichten als Kandidaten,237
- JH, Planelement
  - Abkürzungen im Plan,356
- JHA, Planelement
  - Abkürzungen im Plan,356
- JHAP, Planelement
  - Abkürzungen im Plan,356
- JHFO, Planelement
  - Abkürzungen im Plan,356
- JHO, Planelement
  - Abkürzungen im Plan,356
- JHPO, Planelement
  - Abkürzungen im Plan,356
- JHR, Planelement
  - Abkürzungen im Plan,356
- JHRO, Planelement
  - Abkürzungen im Plan,356
- JHS, Planelement
  - Abkürzungen im Plan,356
- JHSP, Planelement
  - Abkürzungen im Plan,356
- JM, Planelement
  - Abkürzungen im Plan,356
- JMFO, Planelement
  - Abkürzungen im Plan,356
- JMO, Planelement
  - Abkürzungen im Plan,356
- JNL, Planelement
  - Abkürzungen im Plan,356
- JNLA, Planelement
  - Abkürzungen im Plan,356
- JNLFO, Planelement
  - Abkürzungen im Plan,356
- JNLO, Planelement
  - Abkürzungen im Plan,356
- JNLS, Planelement
  - Abkürzungen im Plan,356
- Join-Bedingungen
  - Info,495
  - Typen,501
- Join-Operatoren
  - Transact-SQL,668
- Joins
  - abgeleitete Tabellen,517
  - automatische,645
  - Berechnung eines Inner-Joins,497
  - bewahrte Tabellen,505
  - CROSS APPLY- und OUTER APPLY-Joins,517
  - Cross-Joins,503
  - Cursor aktualisieren,499
  - Daten aus mehreren Tabellen abrufen,492
  - Datentypkonvertierung,498
  - DELETE-, UPDATE- und INSERT-Anweisungen,499
  - doppelte Korrelationsnamen,513
  - entfernte Tabellen verknüpfen,811
  - Equi-Joins,501
  - explizite Join-Bedingungen,495
  - Full-Outer-Join,645
  - generierte Join-Bedingungen,495
  - Info,492
  - Inner- und Outer-Joins,504
  - Inner-Joins,504
  - Join-Bedingungen,495



---

- Join-Eliminierung durch
- Neuschreibungsoptimierung,241
- kartesisches Produkt,503
- Kommas,503
- Kompatibilität mit Transact-SQL,668
- Left-Outer-Join,645
- mehr als zwei Tabellen,498
- natürliche,645
- natürliche Joins,519
- nicht-ANSI-Joins,499
- Nullwert-liefernde Tabellen,505
- oder Unterabfragen,605
- ON-Klausel,499
- Outer-Joins,505
- Right-Outer-Join,645
- Schlüssel,645
- Schlüssel-Joins,523
- Selbst-Joins,511
- sich aus APPLY-Ausdrücken ergebende,517
- spezielle,511
- Standard ist KEY JOIN,496
- Stern-Joins,513
- Suchbedingungen,501
- Tabellen aus mehreren lokalen Datenbanken verknüpfen,812
- Tabellenausdrücke,498
- Transact-SQL, Outer-Joins und Ansichten,510
- Transact-SQL, Outer-Joins und NULL,510
- Transact-SQL-Outer-Joins, Einschränkungen,509
- Unterabfragen konvertieren,616
- Unterabfragen zu Joins konvertieren,616
- Verhalten,495
- verknüpfte Tabellen,496
- verschachtelt,498
- WHERE-Klausel,502
- zwei Tabellen,497
- JSON-Format
  - Abfrageergebnisse abrufen,715
  - Info,715

## K

- Kapazitätsplanung
  - Info,190
- Kardinalität
  - Element in Ausführungsplänen,374
- Kartesische Produkte
  - Info,503

- Katalog
  - Abhängigkeitsinformationen finden,45
  - Adaptive Server Enterprise-Kompatibilität,653
  - Indexinformationen,37
- Kennwörter
  - Groß- und Kleinschreibung,661
- Klammern
  - in arithmetischen Anweisungen,302
  - UNION-Operatoren,486
- Klassen
  - Fremdserver,824
- Klauseln
  - COMPUTE,665
  - FOR BROWSE,665
  - FOR READ ONLY,666
  - GROUP BY ALL,665
  - Info,291
  - INTO,122
  - ISOLATION,665
  - PLAN,665
- Kleiner als
  - Bereichsspezifikation,311
  - Vergleichsoperator in WHERE-Klauseln,310
- Kleiner gleich
  - Vergleichsoperator in WHERE-Klauseln,310
- Kodierung
  - XML,677
- Kommas
  - Stern-Joins,513
  - Tabellenausdrücke verknüpfen,528
  - Tabellenausdrucksliste,503
- Kommentare
  - Prozeduren mit Sybase Central ändern,90
- Kommentierungsphase
  - Abfrageverarbeitung,329
- Kommunikation, Statistik
  - Liste,209
- Kompatibilität
  - Adaptive Server Enterprise mit Transact-SQL,648
  - automatische Konvertierung von gespeicherten Prozeduren,671
  - Datenbanken für Transact-SQL-Kompatibilität konfigurieren,656
  - Groß-/Kleinschreibung berücksichtigen,660
  - GROUP BY-Klausel,478
  - Import/Export mit ASE,784
  - Joins in Transact-SQL,668
  - kompatible SQL-Anweisungen schreiben,663

- nicht-ANSI-Joins,499
- NULL ausgeben,755
- Optionen für die Transact-SQL-Kompatibilität einstellen,659
- Server und Datenbanken,651
- SQL Anywhere-Kompatibilität mit Transact-SQL,648
- Transact-SQL,648
- Komplexe Outer-Joins
  - Info,507
- Komprimierung
  - Empfehlungen für Verbindungspakete,263
  - Warnung vor Komprimierung von Datenbank oder Logdateien,263
- Konfigurieren
  - Diagnoseprotokollierung,172
  - Einstellungen für die Diagnoseprotokollierung,184
- Konflikte
  - Snapshot-Isolation,894
  - sperren,914
  - Tabellensperren,911
  - Transaktionsblockierung,902
  - zyklische Blockierung,903
- Konkurrierende Trigger
  - Auslöserang,109
- Konsistenz
  - ISO SQL-Standard,895
  - praktische Einführung in nicht wiederholbare Lesevorgänge,932
- Konsistenzen
  - Auswirkungen von nicht-serialisierbaren Zeitplänen,924
  - Dirty Reads,895
  - Dirty Reads und Sperren,915
  - Dirty Reads, praktische Einführung,928
  - Info,881
  - Isolationsstufe,885
  - Isolationsstufe 0,915
  - Korrektheit und Zeitplan,924
  - mit Sperren garantieren,906
  - Phantomzeilen,895
  - Phantomzeilen und Sperren,916
  - praktische Einführung in Phantomzeilen,938
  - praktische Einführung zu Phantomsperren,945
  - Snapshot-Isolation,917
  - versus Isolationsstufen,895
  - versus typische Transaktionen,924
  - während Transaktionen,895
  - wiederholbare Lesevorgänge,895
  - wiederholbare Lesevorgänge und Sperren,915
- Kontext
  - Externe Bibliotheken für Volltextsuche,456
- Konvertierungsfehler beim Import
  - Info,739
- Kopieren
  - Daten mit INSERT,630
  - Prozeduren,92
- Korrelationsfunktion
  - OLAP,588
- Korrelationsnamen
  - Einschränkungen,305
  - in Selbst-Joins,511
  - Info,525
  - mit allgemeinen Tabellenausdrücken benutzen,538
  - Stern-Joins,513
  - Tabellennamen,305
- Korrelierte Unterabfragen
  - äußere Referenzen,603
  - Definition,603
  - Info,603
- Kosten
  - Indexberater, Ergebnisse,168
- Kostenbasierte Optimierung
  - Info,333
  - umgehen,331
- Kostenmodelle
  - für Festplattenzugriffe,333
  - Info,333
- Kostenträchtige Trigger ersetzen
  - Tipps zur Performance-Verbesserung,259
- Kostenvorteile insgesamt
  - Indexberater, Ergebnisse,168
- Kreuzprodukte
  - Info,503
- Kurze Textpläne
  - Info über,342
  - mit SQL-Funktionen anzeigen,344
- L**
- Laden
  - Hinweise zur Datenbank-Wiederherstellung,730
  - Hinweise zur Synchronisation,730
  - SQL-Skripten in Interactive SQL,783
- Laden von Tabellen
  - Datenbankspiegelung, Hinweise,730

- 
- Langfristige Lesesperren
    - Info,910
  - LAST\_VALUE-Funktion
    - Beispiele,582
    - Verwendung,572
  - LDAP
    - Benutzerauthentifizierung,957
    - LDAP-Serverkonfigurationsobjekte aktivieren,966
    - LDAP-Serverkonfigurationsobjekte unterbrechen,966
  - LDAP-Benutzerauthentifizierung
    - Benutzer mit Sybase Central erstellen,965
    - Info,957
    - LDAP-Serverkonfiguration mit Sybase Central ändern,966,967
    - LDAP-Serverkonfiguration mit Sybase Central erstellen,962
    - LDAP-Serverkonfiguration mit Sybase Central löschen,968
    - LDAP-Serverkonfiguration mit Sybase Central validieren,967
    - Login-Richtlinie mit Sybase Central erstellen,964
    - mit SQL einrichten,958
    - mit Sybase Central einrichten,962
    - Serverkonfigurationsobjekte,957
  - LDAP-Serverkonfigurationsobjekte
    - ändern,967
    - erstellen,962
    - löschen,968
    - validieren,967
  - LEAVE-Anweisung
    - Steueranweisungen,118
  - Leerl.-Checkpoints/Sek., Statistik
    - Beschreibung,208
  - Leerlauf aktiv/Sek., Statistik
    - Beschreibung,208
  - Leerlauf-Schreibvorgänge/Sek., Statistik
    - Beschreibung,208
  - Left-Outer-Joins
    - SQL Anywhere-Implementierung,645
  - LeftOuterHashJoin, Planelement
    - Abkürzungen im Plan,356
  - LesenPS-Sperren
    - Konflikte,915
  - Lesesperren
    - Info,909
    - Konflikte,915
    - langfristige,910
  - LIKE, Suchbedingung
    - Einführung,313
  - LIKE-Suchbedingung
    - Platzhalter,314
  - LIMIT-Klausel
    - verwenden,483
  - Links-Outer-Joins
    - Info,505
  - LOAD DATABASE-Anweisung
    - nicht unterstützt,651
  - LOAD TABLE-Anweisung
    - Hinweise,729
    - Hinweise zu materialisierten Ansichten,730
    - Hinweise zu Textindizes,730
    - Importieren von BCP-Formatdaten,784
    - Syntax,639,748
    - verwenden,739
  - LOAD TRANSACTION-Anweisung
    - nicht unterstützt,651
  - localname
    - Meta-Eigenschaftsname,682
  - Log, Registerkarte
    - Indexberater, Ergebnisse,168
  - Logdateien
    - Warnung vor Komprimierung von Logdateien,263
  - login\_mode-Option
    - LDAP-Benutzerauthentifizierung aktivieren,958
  - Logische Indizes
    - gemeinsam genutzte physische Indizes ermitteln,38
    - Info,37
  - Logische Operatoren
    - Bedingungen verknüpfen,319
    - HAVING-Klausel,481
  - Logs
    - Rollback-Log,885
  - Lokale temporäre Tabellen
    - benennen,10
    - Info,10
  - Lokale Variable
    - Debugger,981
  - LONG VARCHAR, Datentyp
    - XML speichern,677
  - LOOP-Anweisung
    - Prozeduren,129
    - Steueranweisungen,118
  - Loopback-Verbindungen
    - Info,812
  - Löschen
-

- Begriffe aus CONTAINS-Abfragen,447
- Begriffe aus dem Textindex,447
- benutzerdefinierte Datentypen,867
- CHECK-Integritätsregeln,864
- Datentypen,867
- entfernte Prozeduren ,816,817
- Fremdserver,792
- Prozeduren,93
- Sequenzen,953
- Spalten-Standardwerte,853
- Trigger,106
- Verzeichniszugriffsserver,802
- Lotus Notes
  - Ferndatenzugriff,844
- M**
- Mac OS X
  - nicht unterstützte Fremdserver für UltraLite unter Mac OS X,827
- MANUAL REFRESH-Klausel
  - Info,425
- Manuelle Ansichten
  - aktualisieren,67
  - Einschränkungen bei der Konvertierung in manuelle Ansichten,62
  - in eine Sofortansicht ändern,74
  - Info,59
  - materialisierte Ansichten mit manuellem Aktualisierungstyp,59
  - mit Sybase Central erstellen,65
  - Veraltung,59
- Massenimport von Daten
  - Performance,721
- Massenvorgänge
  - Auswirkungen auf die Performance,721
  - Datenwiederherstellung,722
  - Info,721
  - Tipps zur Performance-Verbesserung,262
- Master-Datenbank
  - nicht unterstützt,651
- Materialisieren, Ergebnismenge
  - Abfrageverarbeitung,239
- Materialisierte Ansichten
  - Abhängigkeiten, die Tabellenalternierungen blockieren,43
  - aktivieren und deaktivieren,68
  - aktualisieren mit Sybase Central,67
  - Ansichtenübereinstimmung mit Snapshot-Isolation verwenden,889
  - Ansichtsabhängigkeiten,43
  - auf Verwendung prüfen,234
  - COSTED,
    - Ansichtenübereinstimmungsergebnis,372
  - Datenaktualität und -konsistenz,56
  - den Aktualisierungstyp ändern,74
  - die Verwendung in der Optimierung mit Sybase Central aktivieren,72
  - die Verwendung in der Optimierung mit Sybase Central deaktivieren,72
  - Eigenschaften, Überblick,77
  - eine Sofortansicht erstellen,74
  - Einsatzmöglichkeiten für materialisierte Ansichten,57
  - Einschränkung bei der Erstellung von Sofortansichten,62
  - Einschränkung bei der Verwaltung von materialisierten Ansichten,61
  - entschlüsseln mit Sybase Central,71
  - Erstellungsoptionen für materialisierte Ansichten abfragen,73
  - Hinweise zu Datenbankoptionen,61
  - Hinweise zum Optimierer,58
  - Hinweise zum Speicherplatz,56
  - Info,56
  - Informationen über materialisierte Ansichten abrufen,73
  - Initialisieren mit Sybase Central,66
  - Kandidatenliste für die Verbindung ermitteln,237
  - kurze Gegenüberstellung zu regulären Ansichten und Tabellen,42
  - löschen mit Sybase Central,70
  - manuelle Ansicht in eine Sofortansicht ändern,74
  - manuelle und sofortige, Vergleich,59
  - mit Algorithmus für die Ansichtenübereinstimmung schätzen,235
  - mit Daten füllen mit Sybase Central,66
  - mit Sybase Central erstellen,65
  - nicht übereinstimmende Verbindungen und Optionen,237
  - Performance durch materialisierte Ansichten verbessern,234
  - Performance-Tipp,234
  - Plan-Caching,337
  - Sofortansicht in eine manuelle Ansicht ändern,74
  - Spaltenstatistiken,56

---

SQL Anywhere-Implementierung,646  
Status,76  
Status- und Eigenschaften, Diagramm,78  
Veraltung,60  
Veraltungsschwellenwert des Optimierers für materialisierte Ansichten einstellen,79  
verbergen,69  
verschlüsseln mit Sybase Central,71  
verwenden, um die Performance zu verbessern,57  
Verwendung durch Optimierer prüfen,238  
Wartungskosten,56  
materialized\_view\_optimization, Option  
Verwendung,79  
MAX-Funktion  
äquivalente mathematische Formel,598  
Verwendung,572  
max\_query\_tasks, Option  
abfrageinterne Parallelität kontrollieren,376  
Optimiererstatisiken-Feldbeschreibungen,362  
Maximum  
Cachegröße,223  
MAXIMUM TERM LENGTH, Einstellung  
Definition,410  
empfohlene Größe für N-Gramme,410  
Mehrere Ergebnismengen  
Anzeige in Interactive SQL,126  
Mehrfach vorhandene Zeilen  
mit UNION entfernen,486  
Mehrfach-Transaktionen  
Parallelität,884  
Mehrfachdatenbanken  
Joins,812  
Mehrzeilige Unterabfragen  
Info,600  
Mem Pages Abfrageverarbeitung, Statistik  
Beschreibung,213  
Mem Pages Ansichtsdefinitionen, Statistik  
Beschreibung,215  
Mem Pages Carver, Statistik  
Beschreibung,213  
Mem Pages Gesperrter Heap, Statistik  
Beschreibung,215  
Mem Pages Haupt-Heap, Statistik  
Beschreibung,215  
Mem Pages Prozedurdefinitionen, Statistik  
Beschreibung,215  
Mem Pages Rollback-Log, Statistik  
Beschreibung,215  
Mem Pages Seitenzuordnungen, Statistik  
Beschreibung,215  
Mem Pages Sperrentabelle, Statistik  
Beschreibung,215  
Mem Pages Triggerdefinitionen, Statistik  
Beschreibung,215  
Mem Pages Verbrauchter Cursor, Statistik  
Beschreibung,213  
Mem Pages Verlegbarkeit, Statistik  
Beschreibung,215  
Mem Pages Verlegungen/Sek., Statistik  
Beschreibung,215  
Mengenoperationen  
Info,486  
NULL,489  
Regeln,487  
MERGE-Anweisung  
Hinweise zu materialisierten Ansichten,734  
Hinweise zu Textindizes,734  
RAISERROR-Aktion verwenden,737  
Syntax,749  
verwenden,732  
MergeExcept, Planelement  
Abkürzungen im Plan,356  
MergeExceptAll, Planelement  
Abkürzungen im Plan,356  
MergeIntersect, Planelement  
Abkürzungen im Plan,356  
MergeIntersectAll, Planelement  
Abkürzungen im Plan,356  
MergeJoin, Planelement  
Abkürzungen im Plan,356  
MESSAGE-Anweisung  
Prozeduren,133  
Meta-Eigenschaftsnamen  
ID,682  
localname,682  
Microsoft Access  
Ferndatenzugriff,836  
nach SQL Anywhere migrieren,774  
Microsoft Excel  
Architekturkonflikt,746  
Daten in eine SQL Anywhere-Datenbank exportieren,746,747  
Daten in eine SQL Anywhere-Datenbank importieren,728  
Ferndatenzugriff,843  
Microsoft FoxPro

- Ferndatenzugriff,844
- Microsoft SQL Server
  - nach SQL Anywhere migrieren,774
- MIN-Funktion
  - äquivalente mathematische Formel,598
- Minimale Cachegröße
  - Info,223
- MINIMUM TERM LENGTH-Einstellung
  - Definition,409
- MIRROR-Serverklasse
  - Info,826
- MobiLink
  - Datenbanken neu aufbauen,767,769
- modified\_date\_time
  - Verzeichniszugriffsserver,795
- Mögliche Indizes
  - Indexberater,167
  - Info,167
- MSACCESSODBC-Serverklasse
  - Info,836
- MSSODBC-Serverklasse
  - Info,838
- MultIdx, Planelement
  - Abkürzungen im Plan,356
- MultipleIndexScan, Planelement
  - Abkürzungen im Plan,356
- Musterübereinstimmung
  - Einführung,313
- MySQL
  - ODBC-Serverklasse,840
- MYSQLODBC-Serverklasse
  - Info,840
- N**
- N-Gramme
  - Definition,407
  - empfohlene Größe,410
  - wie Begriffe aufgeteilt werden, Erklärung,406
  - zweistufiger Prozess der Generierung,414
- N-Gramme-
  - wie N-Gramme generiert werden,414
- Nach mehreren Spalten gruppieren
  - Info,476
- Nachbarschaftssuche
  - Volltextsuche,397
- nachgestellte Leerzeichen
  - Datenbanken erstellen,656
- Transact-SQL,656
- Vergleiche,310
- Namensbereiche
  - Indizes,661
  - Trigger,661
- Namespaces
  - in XML definieren,686
- Native Anweisungen
  - an Fremdserver senden,813
- Natürliche Joins
  - Fehler,521
  - Info,519
  - mit einer ON-Klausel,521
  - SQL Anywhere-Implementierung,645
  - SQL-Sprache,645
  - von Ansichten und abgeleiteten Tabellen,523
  - von Tabellenausdrücken,522
- NCHAR-Datentyp
  - SQL Anywhere-Implementierung,645
- NestedLoopsAntisemijoin, Planelement
  - Abkürzungen im Plan,356
- NestedLoopsJoin, Planelement
  - Abkürzungen im Plan,356
- NestedLoopsSemijoin, Planelement
  - Abkürzungen im Plan,356
- Neu berechnen
  - berechnete Spalten,17
- Neu erstellen
  - Indizes,35
- Neu starten
  - Sequenzen,952
- Neuaufbau
  - Ausfallzeiten minimieren,773
  - Datenbanken,763
  - Tools,763
  - Zweck,763
- Neuaufbau von Datenbanken
  - Hinweise,763
  - SQL Remote,767,769
- Neuaufbau-Tools
  - dbisql, Dienstprogramm,765
  - Dienstprogramm, entladen,765
  - Info,763
  - UNLOAD TABLE-Anweisung,770
- Neuschreibungsoptimierungen
  - Liste,340
- NEWID-Funktion
  - Standard-Spaltenwert,858

---

- wann anzuwenden,949
- NGRAM, Textindizes
  - Präfixsuchen,396
- NGRAM-Textindizes
  - Fuzzy, Volltextsuche, praktische Einführung,433
  - Nicht-Fuzzy, Volltextsuche, praktische Einführung,437
- Nicht benutzte Indizes, Registerkarte
  - Indexberater, Ergebnisse,168
- Nicht festgeschriebene Daten lesen
  - Arten von Inkonsistenz,895
  - Einführung,885
  - Einstellung für ODBC,898
  - SELECT-Anweisungen,915
- Nicht gleich
  - Vergleichsoperator in WHERE-Klauseln,310
- Nicht größer als
  - Vergleichsoperator in WHERE-Klauseln,310
- Nicht kleiner als
  - Vergleichsoperator in WHERE-Klauseln,310
- Nicht-ANSI-Joins
  - Info,499
- Nicht-deterministischen Funktionen
  - Nebenwirkungen,339
- Nicht-korrelierte Unterabfragen
  - Info,603
- Nicht-serialisierbare Transaktionszeitpläne
  - Auswirkungen,924
- Nicht-wiederholbare Lesevorgänge
  - Info,895
  - Isolationsstufen,895
  - praktische Einführung,932
- NOHOLDLOCK, Schlüsselwort
  - ignoriert,667
- Normalisierung
  - Performance-Nutzen,255
- NOT
  - logische Operatoren verwenden,319
- NOT BETWEEN, Schlüsselwort
  - Bereichsabfragen,311
- NOT NULL-Integritätsregeln
  - Tools für die Aufrechterhaltung der Datenintegrität,851
- NOT, Schlüsselwort
  - Beispiel,311
- Notes und Fernzugriff
  - Info,844
- Notwendig

- Fremdschlüssel,870
- NULL
  - Aggregatfunktion,473
  - Ausgabe,755
  - Daten importieren,741
  - durch einen Wert ersetzen,318
  - Eigenschaften,318
  - einfügen,627
  - Eliminierung von doppelten Nullwerten mithilfe der DISTINCT-Klausel,305
  - Ergebnisse in UNBEKANNT, wenn bei Vergleichen verwendet,318
  - EXCEPT-Klausel,489
  - INTERSECT-Klausel,489
  - Konvertierungsfehler ignorieren,739
  - Mengenoperatoren und NULL,489
  - Platzhalter in OLAP,562
  - Sortierreihenfolge,483
  - Spalten-Standardwert,659
  - Spaltendefinition,319
  - Standardparameter,318
  - Standardwert,859
  - Transact-SQL Outer-Joins,510
  - Transact-SQL-Kompatibilität,664
  - unbekannte Werte und WHERE-Klausel,317
  - UNION-Klausel,489
  - Unterschied zu Nullen und Leerstellen,317
  - vergleichen,318
- NULL ausgeben
  - Info,755
- Nullwert-liefernde Tabellen
  - in Outer-Joins,505

## O

- Objekte
  - Definitionen verbergen,149
  - qualifizierte Namen,1
  - sperrbare,907
- ODBC
  - Anwendungen und Sperren,898
  - externe Server,824
  - Isolationsstufen einstellen,898
  - materialisierte Ansichten als Kandidaten,237
- ODBC-Serverklassen
  - Adaptive Server Enterprise,829
  - Advantage Database Server,828
  - IBM DB2,832

- Info,824
- IQODBC,836
- Lotus Notes SQL,844
- Microsoft Access,836
- Microsoft Excel,843
- Microsoft FoxPro,844
- MySQL,840
- Oracle Database,845
- SAP HANA,834
- Serverklasse ODBC,842
- SQL Anywhere,826,838
- UltraLite,827
- odbcfet
  - Info,245
- Ohne Laden
  - SQL-Skriptdateien ausführen,781
- OLAP
  - Basis-Aggregatfunktionen,572
  - CUBE-Klausel,560
  - Einführung,552
  - Fenster-Aggregatfunktionen,571
  - Fenster-Rangfunktionen,590
  - Fensterfunktionen,571
  - Funktionen für lineare Regression,588
  - GROUP BY-Klauselerweiterungen,554
  - Info,552
  - Korrelationsfunktionen,588
  - OLAP-Performance verbessern,553
  - ROLLUP-Klausel,558
  - Standardabweichungsfunktionen,584
  - Varianzfunktionen,584
  - WITH CUBE-Klausel,562
  - WITH ROLLUP-Klausel,560
  - Zeilennummerierungsfunktionen,597
- OLAP-Funktionen
  - Formeln,598
- ON EXCEPTION RESUME-Klausel
  - Fehlerbehandlung,133
  - gespeicherte Prozeduren,132
  - Transact-SQL,675
  - Verarbeitungsroutine bei Ausnahmefehlern,137
- ON-Klausel
  - Einführung,499
  - Joins,499
  - Tabellen referenzieren,500
- OPEN-Anweisung
  - Prozeduren für die Cursor-Verwaltung,128
- OpenString, Planelement
- Abkürzungen im Plan,356
- OPENXML-Operator
  - mit xp\_read\_file verwenden,683
  - verwenden,679
- Operatoren
  - arithmetische,302
  - Bedingungen verknüpfen,319
  - Schlüsselwort NOT,311
  - Vorrang,302
- Optimieren von Abfragen
  - Ausführungspläne lesen,341
- Optimierer
  - Bypass,331
  - Info,332
  - Liste der Optimierungen für eine verbesserte Performance,340
  - materialisierte Ansichten verwenden,72
  - Phasen der Abfrageverarbeitung,329
  - Prädikat-Analyse,292
  - Quellen der Selektivitätsschätzung,336
  - semantische Transformationen,340
- Optimiererschätzungen
  - Info,334
- Optimierung
  - Ausführungspläne lesen,341
  - Info,332
  - kostenbasiert,333
  - Schritte,329
- Optimierung umgehen
  - Bypass-Abfragen,331
- Optimierungen
  - Liste der von Optimierer angewendeten Transformationen,340
- Optimierungsmethode
  - Optimiererstatistiken-Feldbeschreibungen,362
- Optimierungsphase
  - Abfrageverarbeitung,330
- Optimierungszeit
  - Optimiererstatistiken-Feldbeschreibungen,362
- Optimierungsziel
  - Ausführungspläne,372
- Optimierungsziel "Alle Zeilen"
  - Performance,239
- optimization\_goal, Option
  - Optimiererstatistiken-Feldbeschreibungen,362
- optimization\_level, Option
  - Optimiererstatistiken-Feldbeschreibungen,362
- optimization\_workload, Option



- 
- Optimiererstatistiken-Feldbeschreibungen,362
    - verwenden,233
  - Optionale Fremdschlüssel
    - Info,870
  - Optionen
    - blockieren,902
    - DEFAULTS,741
    - isolation\_level,897
  - OR
    - logische Operatoren verwenden,319
  - Oracle
    - nach SQL Anywhere migrieren,774
  - Oracle Database
    - Datentypkonvertierungen,845
  - Oracle Database und Ferndatenzugriff
    - Info,845
  - ORAODBC-Serverklasse
    - Info,845
  - ORDER BY und GROUP BY
    - Info,485
  - ORDER BY-Klausel
    - Abfrageergebnisse sortieren,481
    - Auswirkung auf teilweise definierte Fenster (OLAP),567
    - Beispiele,322
    - Einschränkungen bei der Definition von regulären Ansichten,46
    - erforderlich, damit Zeilen immer in derselben Reihenfolge erscheinen,323
    - Ergebnisse beschränken,483
    - GROUP BY,485
    - in Definition materialisierter Ansichten aufnehmen,62
    - mit Indizes die Performance steigern,324
    - Performance,233
    - zusammengesetzte Indizes,30
  - Order-by
    - Element in Ausführungsplänen,376
  - OrderedDistinct, Planelement
    - Abkürzungen im Plan,356
  - OrderedGroupBy, Planelement
    - Abkürzungen im Plan,356
  - OrderedGroupBySets, Planelement
    - Abkürzungen im Plan,356
  - OUT, Parameter
    - definiert,116
  - OUTER APPLY-Klausel
    - Beispiel,518
  - Info,517
  - Outer-Joins
    - Ansichten und abgeleitete Tabellen,508
    - Einschränkungen,509
    - Info,505
    - Join-Eliminierung durch Neuschreibungsoptimierung,241
    - Kompatibilität mit Transact-SQL,668
    - komplexe,507
    - Stern-Join, Beispiel,515
    - Transact-SQL,509
    - Transact-SQL und Ansichten,510
    - Transact-SQL, Einschränkungen,509
    - und Join-Bedingungen,506
  - OUTPUT-Anweisung
    - Abfrageergebnisse exportieren,744,753,754
    - beim Export von Daten als XML verwendet,678
    - Excel,746,747
    - Syntax,745
  - OVER-Klausel
    - Syntax in Funktionen als Fensterfunktionen,568
- ## P
- Paketgröße
    - Performance,264
  - ParallelHashAntisemijoin, Planelement
    - Abkürzungen im Plan,356
  - ParallelHashFilter, Planelement
    - Abkürzungen im Plan,356
  - ParallelHashSemijoin, Planelement
    - Abkürzungen im Plan,356
  - ParallelIndexScan, Planelement
    - Abkürzungen im Plan,356
  - Parallelität
    - DDL-Anweisungen,954
    - Funktionsweise von Sperren,906
    - in Abfragen,378
    - Info,376,884
    - Inkonsistenz,895
    - ISO SQL-Standard,895
    - Konsistenz,895
    - mit Indizes verbessern,926
    - Performance,884
    - Primärschlüssel,949
    - verbessern,925
    - Vorteile,884
  - ParallelLeftOuterHashJoin, Planelement

- Abkürzungen im Plan,356
- ParallelTableScan, Planelement
  - Abkürzungen im Plan,356
- Parameter
  - für Funktionen,325
- Parameter übergeben
  - an Funktionen,117
  - an Prozeduren,116
- Parse-Bäume
  - Abfrageverarbeitung,329
- Partieller Index-Scan
  - Info,40
- PARTITION, Schlüsselwort
  - Transact-SQL SELECT-Anweisung, nicht unterstützte Syntax,665
- PC, Planelement
  - Abkürzungen im Plan,356
- PCTFREE, Einstellung
  - Tabellenfragmentierung reduzieren,249
- PDF-Dateien
  - Unterstützung für externen Vorfilter und und Begriffsegmentierer-Bibliothek,449
- PERCENT\_RANK-Funktion
  - äquivalente mathematische Formel,598
  - Verwendung,596
- PercentTotalCost
  - Knotenstatistiken-Feldbeschreibungen,368
- Performance
  - Abfrage-Geschwindigkeit messen,245
  - Anwendungsprofilerstellung,156
  - Arbeitslast des Optimierers,228
  - Arbeitstabellen,239
  - Ausführungspläne lesen,341
  - Dateifragmentierung,249
  - Datenbank neu aufsetzen,257
  - empfohlene Seitengrößen,260
  - erweiterte Anwendungsprofilerstellung,170
  - gestreute Lesevorgänge,261
  - Index, Hinweise,231
  - Indizes verwenden,27
  - Info,219
  - kaskadierende referenzielle Aktionen vermeiden,256
  - Laufzeit, tatsächlich und geschätzt,350
  - Lesevorgänge-Treffer-Verhältnis des Cache,350
  - Liste der Verbesserungs-Tipps,219
  - Massenimport,721
  - mit dem Systemmonitor überwachen,203
  - Mit dem Windows-Systemmonitor überwachen,205
  - Optimiererschätzungen und tatsächliche Statistiken vergleichen,348
  - Optimierungsziel "Alle Zeilen",239
  - Paketgröße,264
  - Prädikat-Analyse,292
  - Quelle der Schätzung,349
  - Schlüssel,258
  - Seitengrößen,260
  - Selektivität,349
  - Statistiken im Windows-Systemmonitor,205
  - Systemmonitor-Statistiken,207
  - Tabellen- und Seitengrößen,260
  - Tools zum Überwachen und Verbessern der Performance,153
  - überwachen,201
  - überwachen und steigern,153
  - verbessern,31,219
  - verbessern durch materialisierte Ansichten,234
  - verbessern versus Sperren,926
  - WITH EXPRESS CHECK,238
- Performance überwachen
  - Abkürzungen in Ausführungsplänen,356
  - Ausführungspläne lesen,341
  - Statistiken im Systemmonitor,207
- Performance verbessern
  - auf Parallelitätsprobleme prüfen,228
  - Indizes,233
  - Integritätsregeln deklarieren,256
  - Massenvorgänge,721
  - Primärschlüssel-Breite vermindern,258
  - Statistiken von kleinen Tabellen erfassen, Empfehlung,248
  - Transaktionslog,247
  - verschiedene Dateien auf verschiedene Devices platzieren,256
  - Ziel des Optimierers auswählen,228
- Performance-Monitoring
  - Tools zum Messen von Abfragen,245
- Performance-Statistik
  - Überwachung,201
- Performance-Tools
  - grafische Pläne,347
  - Prozedurprofilerstellung mit Systemprozeduren,197
  - Zeitfolgen-Dienstprogramme,200
- PerformanceFetch

---

Info,245  
 PerformanceInsert  
   Info,245  
 PerformanceTraceTime  
   Info,245  
 PerformanceTransaction  
   Info,245  
 Phantomsperren  
   Info,913  
   praktische Einführung,945  
 Phantomzeilen  
   Dateninkonsistenzen,895  
   Praktische Einführung,938  
   Verhindern auf Isolationsstufe 2,916  
   versus Isolationsstufen,895  
 Phasen  
   Phasen der Abfrageverarbeitung,329  
 Phasen der Abfrageverarbeitung  
   Info,329  
 Phrasen  
   Sonderzeichen in Volltextsuche,393  
   Volltextsuche,393  
 Phrasen, Volltextsuche  
   Info,390  
 Physische Indizes  
   gemeinsam genutzte physische Indizes ermitteln,38  
   Info,37  
 Pl.-Lesezugr. Akt., Statistik  
   Beschreibung,211  
 Pl.-Lesezugr. Index-Blatt/Sek., Statistik  
   Beschreibung,211  
 Pl.-Lesezugr. Index-intern/Sek., Statistik  
   Beschreibung,211  
 Pl.-Lesezugr. Max. Akt., Statistik  
   Beschreibung,211  
 Pl.-Schreibzugr. Seiten/Sek., Statistik  
   Beschreibung,212  
 Pl.-Schreibzugr. Transaktionslog/Sek., Statistik  
   Beschreibung,212  
 Plan-Caching  
   Info,337  
 Plananzeige  
   Knotenstatistiken-Feldbeschreibungen,368  
   Lokale bzw. globale Feldbeschreibungen,362  
   Optimiererstatistiken-Feldbeschreibungen,362  
   Zugriff,355  
 Pläne  
   Abkürzungen,356  
   ausführliche Textpläne,343  
   Caching,337  
   grafische Pläne,347  
   kontextsensitive Hilfe,351  
   kurze Textpläne,342  
   lesen,341  
   ohne Ausführung einer Abfrage anzeigen,342  
   Pläne mit Kosten  
     Optimiererstatistiken-Feldbeschreibungen,362  
   Planerstellungphase  
     Abfrageverarbeitung,330  
   Platte Akt. I/O, Statistik  
     Beschreibung,211  
   Platte Max. I/O, Statistik  
     Beschreibung,211  
   Platten-Lesezugr. Gesamts./Sek., Statistik  
     Beschreibung,211  
   Platten-Lesezugr. Tabelle/Sek., Statistik  
     Beschreibung,211  
   Platten-Lesezugriffe, Arbeitstabellen-Statistik  
     Beschreibung,211  
   Platten-Schreibzugr. Aktive, Statistik  
     Beschreibung,212  
   Platten-Schreibzugr. Dat.festschr./Sek., Statistik  
     Beschreibung,212  
   Platten-Schreibzugr. DB-Erweiterung/Sek., Statistik  
     Beschreibung,212  
   Platten-Schreibzugr. Max. Akt., Statistik  
     Beschreibung,212  
   Platten-Schreibzugr. Temp. Erw./Sek., Statistik  
     Beschreibung,212  
   Platzhalter  
     Musterübereinstimmung,313  
     Zeichenfolgenvergleiche,313  
   Plus-Operator  
     NULL,319  
   Portierbare SQL  
     schreiben,663  
   Positionierte Aktualisierungen  
     Beispiel,131  
   Positionssperren  
     Dauer,907  
     Einfügesperren,914  
     Info,912  
     Phantomsperren,913  
   Prädikat  
     Element in Ausführungsplänen,375  
   Prädikat-Analyse

- Info,292
- Prädikate
  - in Ausführungsplänen lesen,353
  - IN-Listen optimieren,241
  - Optimierer,292
  - Performance,292
  - Verwendungszweck,309
- Prädikate, sargable (als Suchargument nutzbar)
  - Info,292
- Präfixbegriff
  - Info,394
- Präfixe, Volltextsuche
  - Info,390
- Präfixsuche
  - unerwartete Ergebnisse bei N-Gramm-Textindizes,394
  - Volltextsuche,394
- Präfixsuchen
  - in GENERIC-Textindizes,396
  - in NGRAM-Textindizes,396
- Praktische Einführungen
  - Anwendungsprofilerstellung,264
  - Baselining mit Prozedurprofilerstellung,284
  - Deadlocks,265
  - Dirty Reads,928
  - Fuzzy-Volltextsuche,433
  - GENERIC, Volltextsuche,426
  - Indexfragmentierung diagnostizieren,277
  - Isolationsstufen,926
  - langsame Anweisungen diagnostizieren,272
  - Nicht-Fuzzy-Volltextsuche,437
  - nicht-wiederholbare Lesevorgänge,932
  - Phantomsperrern,945
  - Phantomzeilen,938
  - Tabellenfragmentierung diagnostizieren,281
- PreFilter, Planelement
  - Abkürzungen im Plan,356
- PREPARE TRANSACTION-Anweisung
  - Ferndatenzugriff,818
- PREPARE-Anweisung
  - Ferndatenzugriff,818
- Primärschlüssel
  - AUTOINCREMENT,856
  - Beispiel,17
  - Entitätsintegrität,868
  - generieren,949
  - generierte Indizes,28
  - GLOBAL AUTOINCREMENT,856
  - in Sybase Central ändern,18
  - in Sybase Central erstellen,18
  - in Sybase Central löschen,18
  - Info,17
  - Integrität,644
  - mit NEWID UUID erstellen,858
  - mit Sequenzen generieren,950
  - mit SQL ändern,19
  - mit SQL erstellen,19
  - mit SQL löschen,19
  - Parallelität,949
  - Performance,258
  - Sortierfolge,644
  - Tools für die Aufrechterhaltung der Datenintegrität,850
  - verwalten,17
- Primärschlüsselspalte
  - Element in Ausführungsplänen,374
- Primärschlüsseltabelle
  - Element in Ausführungsplänen,374
- Primärschlüsseltabelle, geschätzte Zeilen
  - Element in Ausführungsplänen,374
- Privilegien
  - benutzerdefinierte Funktionen,97
  - Datenmanipulation,624
  - Debugging,971
  - Trigger,109
- Problem der kürzesten Entfernung
  - Info,550
- ProcCall, Planelement
  - Abkürzungen im Plan,356
- Produktionsdatenbank
  - Info,171
- Profilerstellung für Anwendungen
  - Info,156
- Profilerstellungsdatenbank
  - intern ggü. extern erstellen,171
- Programmvariable
  - allgemeine Tabellenausdrücke,541
- Projektionen
  - Info,297
- PROPERTY-Funktion
  - Info,202
- Protokollierung
  - Anwendungsprofilerstellung mithilfe der Datenbankprotokollierung,170
  - Datenbank protokollieren,171
  - Info,170

- 
- Protokollierungsdaten
    - Info,171
    - nicht als Teil eines UNLOAD-Vorgangs entladen,171
  - Protokollierungssitzung
    - Info,171
  - Proxy-Tabelle erstellen, Assistent verwenden,807
  - Proxy-Tabellen
    - Daten importieren,738
    - erstellen in Sybase Central,807
    - Info,805
    - löschen vom Verzeichniszugriffsserver,802
    - mit der CREATE EXISTING TABLE-Anweisung erstellen,807
    - mit der CREATE TABLE-Anweisung erstellen,807
    - mit SQL erstellen,807
    - Standort von Proxy-Tabellen angeben,806
    - Trennzeichen und Verzeichniszugriff,801
    - Verzeichniszugriff-Proxy-Tabellen abfragen,800
  - Prozedur erstellen, Assistent verwenden,89
  - Prozedur, Sprache
    - Überblick,669
  - Prozeduren
    - Assistent zum Erstellen von Prozeduren,89
    - Aufbau,115
    - aufrufen,91
    - Ausnahmeroutinen,136
    - Caching von Anweisungen,337
    - Cursor,128
    - Cursor verwenden,129
    - Datumsangaben,147
    - Eingabe überprüfen,147
    - entfernte Prozeduren erstellen,815
    - entfernte Prozeduren löschen,816,817
    - Ergebnismengen,124
    - Ergebnisse der Profilerstellung generieren und prüfen,158
    - Ergebnisse in Ergebnismengen zurückgeben,124
    - Ergebnisse zurückgeben,121
    - erstellen,89
    - EXECUTE IMMEDIATE-Anweisung,143
    - Fehlerbehandlung,131
    - Fehlerbehandlung in Transact-SQL,673
    - Hinweise zum Referenzieren von temporären Tabellen,12
    - Info,81
    - Inlining als Teil der Abfragetransformation,241
    - Konvertierung,671
    - kopieren,92
    - löschen,93
    - mehrere Ergebnismengen von Prozeduren,126
    - mit Sybase Central ändern,90
    - Parameter,116
    - Parameter deklarieren,116
    - Prozeduren und Funktionen mit Eigentümer- oder Aufruferprivilegien ausführen,82
    - Rückgabewerte,674
    - Savepoints,146
    - Standard-Fehlerbehandlung,132
    - Statistik,334
    - Tabellennamen,147
    - Tipps zum Schreiben,146
    - Tipps zum Schreiben von Prozeduren, Triggern, benutzerdefinierten Funktionen und Batches,146
    - Transact-SQL,671
    - Transact-SQL Fehlerbehandlung mit Watcom SQL nachahmen,675
    - Transact-SQL, Übersicht,669
    - Trennzeichen für Anweisungen,146
    - Überblick,81
    - übersetzen,671
    - variable Ergebnismengen von,127
    - verwenden,82
    - verwenden in der FROM-Klausel,307
    - Vorteile,82
    - Warnungen,135
    - WITH RESULT SET-Klausel,143
    - Zeitangaben,147
    - zulässige Anweisungen,148
  - Prozeduren ändern
    - Info,90
  - Prozeduren aufrufen
    - Info,91
  - Prozedurprofilerstellung
    - aktivieren,158,197
    - Baselining, praktische Einführung,284
    - deaktivieren,161
    - Ergebnisse der Profilerstellung analysieren,163
    - Ergebnisse der Profilerstellung verstehen,162
    - in Sybase Central,158
    - mit Systemprozeduren durchführen,197
    - mithilfe von "sa\_server\_option" deaktivieren,199
    - Objekte, für die ein Profil erstellt werden kann,162

- Profilerstellung mithilfe von "sa\_server\_option"  
zurücksetzen,198
- Profilerstellungsdaten mithilfe von  
Systemprozeduren abrufen,200
- Profilerstellungsfiler mithilfe von  
"sa\_server\_option" einstellen,198  
zurücksetzen,159
- Prozedurprofilerstellung deaktivieren  
Info,161
- Prozedurprofilerstellung zurücksetzen  
Info,159
- Prüfung der referenziellen Integrität  
beim Festschreiben,919  
verzögern,919
- Prüfwerte  
Element in Ausführungsplänen,375
- Publikationen  
Tabellen löschen,8
- Punktewerte  
Volltextsuche,403

## Q

- Qualifikationen  
Info,309
- Qualifizierende Bezeichner  
Datenbankobjekte,1,295
- Quantifizierter Vergleichstest  
Info,619  
Unterabfragen,609
- Quellcode  
Breakpoints setzen,978
- Quellen der Schätzung  
Quellen der Selektivitätsschätzung des  
Optimierers,336
- QueryMemActiveEst, Eigenschaft  
Optimiererstatistiken-Feldbeschreibungen,362
- QueryMemActiveMax, Eigenschaft  
Optimiererstatistiken-Feldbeschreibungen,362
- QueryMemLikelyGrant  
Optimiererstatistiken-Feldbeschreibungen,362
- QueryMemMaxUseful  
Knotenstatistiken-Feldbeschreibungen,368
- QueryMemMaxUseful, Eigenschaft  
Optimiererstatistiken-Feldbeschreibungen,362
- QueryMemNeedsGrant  
Optimiererstatistiken-Feldbeschreibungen,362
- QueryMemPages, Eigenschaft  
Optimiererstatistiken-Feldbeschreibungen,362
- quoted\_identifizier, Option  
für Transact-SQL-Kompatibilität einstellen,659  
Info,316

## R

- RAISERROR, Aktion  
bei Zusammenführungsvorgang verwenden,737
- RAISERROR-Anweisung  
ON EXCEPTION RESUME verwenden,675  
Transact-SQL,675
- Rang  
mit Aggregation verwenden,593
- RANGE-Klausel  
Standardwerte, wenn das Fenster nur teilweise  
definiert ist,567  
verwenden,566
- Rangfunktionen  
Beispiele,590  
mit Fenstern,590  
obere und untere Prozentsätze finden,597
- RANK-Funktion  
äquivalente mathematische Formel,598  
Verwendung,590
- RAW, Modus  
verwenden,691
- READ-Anweisung  
SQL-Skriptdateien ausführen,781
- READ\_CLIENT\_FILE-Funktion  
importieren von und exportieren auf  
Clientcomputer,760
- READCLIENTFILE-Systemprivileg  
von Clientcomputern importieren,760
- ReceivingTracingFrom  
Protokollierungskonfiguration,172
- Rechts-Outer-Joins  
Info,505
- RecursiveHashJoin, Planelement  
Abkürzungen im Plan,356
- RecursiveLeftOuterHashJoin, Planelement  
Abkürzungen im Plan,356
- RecursiveTable, Planelement  
Abkürzungen im Plan,356
- RecursiveUnion, Planelement  
Abkürzungen im Plan,356
- Referenzen  
Referenzen von anderen Tabellen anzeigen,23

---

- Referenzielle Integrität
  - Aktionen,873
  - beim Festschreiben überprüfen,919
  - CHECK-Integritätsregeln,860
  - erzwingen,869
  - Fremdschlüssel,872
  - Info,849
  - Informationen in den Systemtabellen,877
  - Integritätsregeln,851
  - Primärschlüssel,644
  - prüfen,874
  - Prüfung während DELETE durchgeführt,875
  - Prüfung während INSERT durchgeführt,874
  - Spalten-Standardwerte,852
  - Systemtrigger,873
  - Tools für die Aufrechterhaltung der
    - Datenintegrität,850
    - UPDATE-Anweisung,636
    - verlieren,872
    - von Clientanwendung verletzt,873
    - Waisen,919
- Referenzielle Integrität erzwingen
  - Info,869
- Referenzielle Integritätsaktionen
  - von Systemtrigger implementiert,874
- Referenzierendes Objekt
  - Info,43
- Referenziertes Objekt
  - Info,43
- REFRESH MATERIALIZED VIEW-Anweisung
  - mit Snapshot-Isolation nicht verfügbar,889
- Regeln
  - Transact-SQL,652
- REGR\_AVGX-Funktion
  - äquivalente mathematische Formel,598
- REGR\_AVGY-Funktion
  - äquivalente mathematische Formel,598
- REGR\_COUNT-Funktion
  - äquivalente mathematische Formel,598
- REGR\_INTERCEPT-Funktion
  - äquivalente mathematische Formel,598
- REGR\_R2-Funktion
  - äquivalente mathematische Formel,598
- REGR\_SLOPE-Funktion
  - äquivalente mathematische Formel,598
- REGR\_SXX-Funktion
  - äquivalente mathematische Formel,598
- REGR\_SXY-Funktion
  - äquivalente mathematische Formel,598
- REGR\_SYY-Funktion
  - äquivalente mathematische Formel,598
- Reguläre Ansichten
  - ändern mit Sybase Central,50
  - Daten in Ansichten durchsuchen,55
  - Info,46
  - kurze Gegenüberstellung von materialisierten
    - Ansichten und Tabellen,42
  - löschen mit Sybase Central,51
  - reguläre Ansichten mit SQL aktivieren,54
  - reguläre Ansichten mit SQL deaktivieren,54
  - reguläre Ansichten mit Sybase Central
    - aktivieren,52
    - reguläre Ansichten mit Sybase Central
      - deaktivieren,52
      - reguläre Ansichten mit Sybase Central erstellen,49
- Rekursive Abfragen
  - Einschränkungen,543
- Rekursive Unterabfragen
  - Datentyp-Deklarationen in,545
  - Info,543
  - Problem der kürzesten Entfernung,550
  - Stücklistenproblem,547
- Relationale Daten
  - als XML exportieren,678
  - XML,678
- Relativer Vorteil
  - Indexberater, Ergebnisse,167
- reload.sql
  - Datenbanken neu aufbauen,764
  - Datenbanken neu laden,772
  - entfernte Datenbanken neu aufbauen,763
  - Info,763
  - Tabellen exportieren,758
  - Tabellendaten exportieren,770,771
- REORGANIZE TABLE-Anweisung
  - mit Snapshot-Isolation nicht verfügbar,889
- Replikation
  - an Synchronisationssystemen beteiligte
    - Datenbanken neu aufbauen,767,769
    - Datenbanken neu aufbauen,767,769
- Reservierte Wörter
  - Fremdserver,822
- RESIGNAL-Anweisung
  - Syntax,137
- Ressourcenwächter
  - Definition,262

- RESTRICT, Aktion
    - Info, 873
  - RETURN-Anweisung
    - Syntax, 121
  - REVOKE-Anweisung
    - Parallelität, 954
    - Transact-SQL, 655
  - RI-Integritätsregeln
    - Info, 851
    - Tools für die Aufrechterhaltung der Datenintegrität, 851
  - Richtung
    - Element in Ausführungsplänen, 374
  - Right-Outer-Joins
    - SQL Anywhere-Implementierung, 645
  - RL, Planelement
    - Abkürzungen im Plan, 356
  - ROLLBACK-Anweisung
    - Prozeduren und Trigger, 146
    - Transaktionen, 882
    - Trigger, 670
    - UltraLite, Syntax, 625
    - zusammengesetzte Anweisungen, 120
  - Rollback-Logs
    - Datenwiederherstellung, 722
    - Savepoints, 885
  - Rollen
    - Adaptive Server Enterprise, 653
  - Rollennamen
    - Info, 525
  - ROLLUP, Vorgang
    - GROUP BY verstehen, 475
  - ROLLUP-Klausel
    - als Abkürzung für GROUPING SETS, 558
    - Info, 558
  - ROW\_NUMBER-Funktion
    - Verwendung, 598
  - RowConstructor, Planelement
    - Abkürzungen im Plan, 356
  - ROWID
    - Beispiel, 908
  - ROWID, Planelement
    - Abkürzungen im Plan, 356
  - RowIdScan, Planelement
    - Abkürzungen im Plan, 356
  - RowLimit, Planelement
    - Abkürzungen im Plan, 356
  - RowReplicate, Planelement
    - Abkürzungen im Plan, 356
  - ROWS, Planelement
    - Abkürzungen im Plan, 356
  - ROWS-Klausel
    - Standardwerte, wenn das Fenster nur teilweise definiert ist, 567
    - verwenden, 566
  - RowsReturned
    - Knotenstatistiken-Feldbeschreibungen, 368
    - Statistiken in Zugriffsplänen, 370
  - RR, Planelement
    - Abkürzungen im Plan, 356
  - RT, Planelement
    - Abkürzungen im Plan, 356
  - RU, Planelement
    - Abkürzungen im Plan, 356
  - Rückgabewerte
    - Prozeduren, 674
  - RunTime
    - Knotenstatistiken-Feldbeschreibungen, 368
    - Statistiken in Zugriffsplänen, 370
- ## S
- sa\_ansi\_standard\_packages-Systemprozedur
    - SQL Flagger, Verwendung, 642
  - sa\_audit\_string-Systemprozedur
    - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst, 85
  - sa\_clean\_database-Systemprozedur
    - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst, 85
  - sa\_column\_stats-Systemprozedur
    - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst, 85
  - sa\_conn\_activity-Systemprozedur
    - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst, 85
  - sa\_conn\_compression\_info-Systemprozedur
    - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst, 85
  - sa\_conn\_info-Systemprozedur
    - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst, 85
  - sa\_conn\_list-Systemprozedur
    - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst, 85
  - sa\_conn\_options-Systemprozedur



---

|                                                                    |                                                                    |
|--------------------------------------------------------------------|--------------------------------------------------------------------|
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_index_density-Systemprozedur                                    |
| sa_conn_properties-Systemprozedur                                  | durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_index_levels-Systemprozedur                                     |
| sa_db_info-Systemprozedur                                          | durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_install_feature-Systemprozedur                                  |
| sa_db_list-Systemprozedur                                          | durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_java_loaded_classes-Systemprozedur                              |
| sa_db_properties-Systemprozedur                                    | durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_load_cost_model-Systemprozedur                                  |
| SA_DEBUG-Rolle                                                     | durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 |
| Debugger,971                                                       | sa_locks-Systemprozedur                                            |
| sa_dependent_views-Systemprozedur                                  | verwenden,908                                                      |
| Verwendung,45                                                      | sa_make_object-Systemprozedur                                      |
| sa_disable_auditing_type-Systemprozedur                            | durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_materialized_view_can_be_immediate-Systemprozedur               |
| sa_disk_free_space-Systemprozedur                                  | durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_migrate-Systemprozedur                                          |
| sa_enable_auditing_type-Systemprozedur                             | Verwendung,776                                                     |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_migrate_create_fks-Systemprozedur                               |
| sa_external_library_unload-Systemprozedur                          | verwenden,778                                                      |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_migrate_create_remote_fks_list-Systemprozedur                   |
| sa_flush_cache-Systemprozedur                                      | verwenden,778                                                      |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_migrate_create_remote_table_list-Systemprozedur                 |
| sa_flush_statistics-Systemprozedur                                 | verwenden,778                                                      |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_migrate_create_tables-Systemprozedur                            |
| sa_get_histogram-Systemprozedur                                    | verwenden,778                                                      |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_migrate_data-Systemprozedur                                     |
| sa_get_request_profile-Systemprozedur                              | verwenden,778                                                      |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_migrate_drop_proxy_tables-Systemprozedur                        |
| sa_get_request_times-Systemprozedur                                | verwenden,778                                                      |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | sa_procedure_profile-Systemprozedur                                |
| sa_get_table_definition-Systemprozedur                             | detaillierte Profilerstellungsdaten erhalten,200                   |
| durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 | durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 |
|                                                                    | sa_procedure_profile_summary-Systemprozedur                        |
|                                                                    | durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85 |
|                                                                    | zusammenfassende Profilerstellungsdaten erhalten,200               |
|                                                                    | sa_recompile_views-Systemprozedur                                  |

- durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_refresh\_materialized\_views-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_refresh\_text\_indexes-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_remove\_tracing\_data-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_report\_deadlocks-Systemprozedur verwenden,904
- sa\_reset\_identity-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_save\_trace\_data-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_send\_udp-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_server\_option-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
  - Filter für Prozedurprofilerstellung einstellen,198
  - Prozedurprofilerstellung aktivieren,197
  - Prozedurprofilerstellung deaktivieren,199
  - Prozedurprofilerstellung zurücksetzen,198
- sa\_set\_tracing\_level-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- SA\_SQL\_TXN\_READONLY\_STATEMENT\_SNAPSHOT
  - ODBC-Isolationsstufe,898
- SA\_SQL\_TXN\_SNAPSHOT
  - ODBC-Isolationsstufe,898
- SA\_SQL\_TXN\_STATEMENT\_SNAPSHOT
  - ODBC-Isolationsstufe,898
- sa\_table\_fragmentation-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_table\_page\_usage-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_table\_stats-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_text\_index\_vocab\_nchar-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_unload\_cost\_model-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_user\_defined\_counter\_add-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_user\_defined\_counter\_set-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_validate-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sa\_verify\_password-Systemprozedur durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- SAODBC-Serverklasse
  - Info,826
- Saplan-Dateien
  - Info,347
- Savepoints
  - benennen,885
  - Benennung,885
  - innerhalb von Transaktionen,884
  - Prozeduren und Trigger,146
  - verschachteln,885
  - Verschachtelung,885
- Schema
  - Sperren,908
- Schemasperren
  - exklusive,909
  - gemeinsam genutzt,909
  - Info,908
- Schemata
  - exportieren,771
- Schlüssel
  - mit Sequenzen generieren,950
  - Performance,258
  - Primärschlüssel,17
- Schlüssel-Joins
  - Ansichten und abgeleitete Tabellen,533
  - bei mehrfachen Fremdschlüsseln,525
  - Info,523
  - Listen und Tabellenausdrücke ohne Kommas,531
  - mit ON-Klausel,524
  - ON-Klausel,501

- 
- Regeln,535
  - SQL Anywhere-Implementierung,645
  - Tabellenausdrücke,528
  - Tabellenausdrücke ohne Kommas,528
  - Tabellenausdruckslisten,529
  - Schlüsseltyp
    - Element in Ausführungsplänen,374
  - Schlüsselwerte
    - Element in Ausführungsplänen,374
  - Schlüsselwort IN
    - übereinstimmende Listen,312
  - Schlüsselwörter
    - Fremdserver,822
    - HOLDLOCK,667
    - NOHOLDLOCK,667
  - SchreibenKeinPS-Sperren
    - Konflikte,915
  - Schreibsperren
    - Info,910
    - Konflikte,915
  - Seiten
    - Element in Ausführungsplänen,375
    - Plattenspeicherzuordnung für eingefügte Zeilen,631
  - Seitengrößen
    - Hinweise zu Windows Mobile,260
    - Hinweise zur Performance,260
    - Info,260
    - Performance,260
    - Plattenspeicherzuordnung für eingefügte Zeilen,631
    - und Indizes,261
  - Seitenzuordnungen
    - durchsuchen,260
    - Element in Ausführungsplänen,373
  - Selbst-Joins
    - Info,511
  - SELECT, Anweisung
    - Schlüssel und Abfragezugriff,258
  - SELECT-Anweisung
    - Aliase,299
    - Aliase und SQL-Kompatibilität,647
    - angezeigte Zeichenfolgen,301
    - aus DML-Anweisungen auswählen,307
    - Cursor,129
    - Einschränkungen bei regulären Ansichten,46
    - INSERT,626
    - INTO-Klausel,122
    - Spaltenreihenfolge,298
    - Spaltentitel,299
    - Syntax,291
    - Transact-SQL-Kompatibilität,665
    - Unterabfragen,600
    - Variable,667
    - Zeichendaten,316
    - Zeilen angeben,309
  - SELECT-Liste
    - Aliase,299
    - Aliase und SQL-Kompatibilität,647
    - Ausführungspläne,372
    - berechnete Spalten,301
    - EXCEPT-Anweisungen,486
    - Info,296
    - INTERSECT-Anweisungen,486
    - Spaltenreihenfolge wirkt sich auf Reihenfolge in Ergebnissen aus,298
    - UNION-Anweisungen,486
    - UNION-Klausel,487
  - Selektivität
    - Ausführungsplan lesen,348
    - Element in Ausführungsplänen,374
    - in Ausführungsplänen lesen,353
    - Quellen der Optimiererschätzung,336
  - Selektivität im Plan
    - Info,353
  - Selektivitätsschätzungen
    - durch partiellen Index-Scan,40
    - in Ausführungsplänen lesen,353
  - Selektivitätsstatistiken
    - Info,348
  - self\_recursion, Option
    - Adaptive Server Enterprise,670
  - Semantische Transformationen
    - Liste,340
  - Semikola
    - Trennzeichen für Anweisungen,146
  - SendingTracingTo
    - Protokollierungskonfiguration,172
  - seq, Planelement
    - Abkürzungen im Plan,356
  - Sequenzen
    - aktuelle oder nächste Werte abfragen,950
    - ändern,952
    - Beispiel für die Erstellung und Verwendung einer Sequenz,951
    - eindeutige Schlüssel generieren,950

- erstellen,952
- Info,950
- löschen,953
- neu starten,952
- Unterschiede zu Autoinkrement,950
- Sequenzgenerator
  - ändern,952
  - erstellen,952
  - Info,950
  - löschen,953
  - Sequenz zum Generieren eindeutiger Schlüssel verwenden,950
  - Unterschiede zu Autoinkrement,950
- Sequenzielle Table-Scans
  - Plattenspeicherzuordnung und Performance,631
- Sequenzielle Transitionen
  - Element in Ausführungsplänen,374
- Sequenzielles Durchsuchen
  - Plattenspeicherzuordnung und Performance,631
- Serialisierbar
  - Abfolgeplanung,924
  - Arten von Inkonsistenz,895
  - Einführung,885
  - Einstellung für ODBC,898
  - Parallelität verbessern,925
  - SELECT-Anweisungen,915
- Serialisierbare Zeitpläne
  - Auswirkungen,924
  - Info,924
  - Sperren freigeben,922
- Server
  - mit Fremdservern arbeiten,787
  - Systemmonitor-Diagramme,203
- Server und Datenbanken
  - Kompatibilität,651
- Server-Status
  - Indexberater,168
- Serverfunktionalität
  - Ferndatenzugriff,819
- Serverklassen
  - Advantage Database Server,828
  - ASEODBC,829
  - DB2ODBC,832
  - definieren,786
  - DIRECTORY,795
  - HANAODBC,834
  - Info,787
  - IQODBC,836
  - MIRROR,826
  - MSACCESSODBC,836
  - MSSODBC,838
  - MySQL,840
  - ODBC,842
  - ODBC-basiert,824
  - ORAODBC,845
  - SAODBC,826
  - ULODBC,827
- Serverkonfigurationsobjekte
  - LDAP,957
  - LDAP, ändern,967
  - LDAP, erstellen,962
  - LDAP, löschen,968
  - LDAP, validieren,967
- Serverseitiges Laden
  - Info,721
- SET DEFAULT, Aktion
  - Info,874
- SET HIDDEN-Klausel
  - verwenden,149
- SET NULL, Aktion
  - Info,873
- SET OPTION-Anweisung
  - Transact-SQL,659
  - vom SQL Flagger ignoriert,643
- SET-Klausel
  - UPDATE-Anweisung,633
- SHARED, Schlüsselwort
  - Transact-SQL SELECT-Anweisung, nicht unterstützte Syntax,665
- Sicherheit
  - Anforderungslog,195
  - importieren von und exportieren auf Clientcomputer,761
  - Objekte verbergen,149
- Sicherheitsadministrator
  - Adaptive Server Enterprise,653
- Sicherheitsmodell
  - Aufrufer,84
  - Definierer,84
  - Festlegen des von einer Datenbank benutzten Sicherheitsmodells,88
  - Systemprozeduren vor Version 16.0 als Aufrufer oder Definierer ausführen,84
- SIGNAL-Anweisung
  - Prozeduren,133
  - Transact-SQL,675

---

SingleRowGroupBy, Planelement  
     Abkürzungen im Plan,356  
 Skalaraggregate  
     Info,471  
 Skalaraggregatfunktionen  
     Definition,477  
 Skriptdateien  
     ausführen, SQL,780  
 Skripten  
     in Interactive SQL ausführen,780  
     SQL-Dateien,780  
     SQL-Skriptdateien erstellen,780  
     SQL-Skriptdateien laden,783  
 Snapshot-Anzahl, Statistik  
     Beschreibung,216  
 Snapshot-Isolation  
     Absichtssperren,911  
     aktivieren,891  
     Aktualisierungskonflikte vermeiden,894  
     Info,888  
     Performance-Auswirkungen,923  
     SELECT-Anweisung, sperren,917  
     SQL Anywhere-Implementierung,647  
     Stufe wählen,923  
     Stufen während einer Transaktion ändern,900  
     Transaktionen,890  
     Übereinstimmung materialisierter Ansichten,889  
     Zeilenversionen,890  
 Snapshot-Isolationsstufe  
     schreibgeschützter Snapshot, verwenden,923  
 Snapshot-Isolationsstufen  
     Info,923  
 SnapshotIsolationState, Eigenschaft  
     verwenden,891  
 SOAP-Dienste  
     Fehlersuche,971  
 SOAP-Funktionen  
     Fehlersuche,971  
 Sofortansichten  
     auf manuelle Ansicht ändern,74  
     Einschränkungen beim Erstellen,62  
     erstellen,74  
     Info,59  
     materialisierte Ansichten mit manuellem  
         Aktualisierungstyp,59  
     nur geänderte Zeilen während eines Neuaufbaus  
         aktualisiert,59  
 Sort, Planelement  
     Abkürzungen im Plan,356  
 SortedGroupBySets, Planelement  
     Abkürzungen im Plan,356  
 Sortieren  
     Abfrageergebnisse ,322  
     mit einem Index,233  
 Sortierreihenfolge  
     ORDER BY-Klausel,481  
     Vergleiche,310  
 SortTopN, Planelement  
     Abkürzungen im Plan,356  
 SOUNDEX-Funktion  
     Info,321  
 sp\_addgroup-Systemprozedur  
     Transact-SQL,654  
 sp\_addlogin-Systemprozedur  
     Transact-SQL,654  
     Unterstützung,651  
 sp\_adduser-Systemprozedur  
     Transact-SQL,654  
 sp\_bindefault, Prozedur  
     Transact-SQL,652  
 sp\_bindrule, Prozedur  
     Transact-SQL,652  
 sp\_changegroup-Systemprozedur  
     Transact-SQL,654  
 sp\_copy\_directory-Systemprozedur  
     durch Aufrufer/Definierer-Einstellung für  
         Datenbank beeinflusst,85  
 sp\_copy\_file-Systemprozedur  
     durch Aufrufer/Definierer-Einstellung für  
         Datenbank beeinflusst,85  
 sp\_create\_directory-Systemprozedur  
     durch Aufrufer/Definierer-Einstellung für  
         Datenbank beeinflusst,85  
 sp\_dboption-Systemprozedur  
     Transact-SQL,659  
 sp\_delete\_directory-Systemprozedur  
     durch Aufrufer/Definierer-Einstellung für  
         Datenbank beeinflusst,85  
 sp\_delete\_file-Systemprozedur  
     durch Aufrufer/Definierer-Einstellung für  
         Datenbank beeinflusst,85  
 sp\_dropgroup-Systemprozedur  
     Transact-SQL,654  
 sp\_droplogin-Systemprozedur  
     Transact-SQL,654  
 sp\_dropuser-Systemprozedur

- Transact-SQL,654
- sp\_forward\_to\_remote\_server-Systemprozedur
  - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sp\_get\_last\_synchronize\_result-Systemprozedur
  - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sp\_list\_directory-Systemprozedur
  - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sp\_move\_directory-Systemprozedur
  - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sp\_move\_file-Systemprozedur
  - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sp\_remote\_columns-Systemprozedur
  - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
  - verwenden,810
- sp\_remote\_exported\_keys-Systemprozedur
  - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sp\_remote\_imported\_keys-Systemprozedur
  - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sp\_remote\_primary\_keys-Systemprozedur
  - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sp\_remote\_procedures-Systemprozedur
  - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- sp\_remote\_tables-Systemprozedur
  - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
  - verwenden,794
- sp\_servercaps-Systemprozedur
  - verwenden,794
- Spalten
  - ändern mit Sybase Central,6
  - berechnet,301
  - breite,259
  - CHECK-Integritätsregeln,864
  - Daten einfügen,627,628
  - Datentypen und Domänen zuordnen,865
  - definierter Wert für Variable beginnend mit @,852
  - Domänen verwenden,866
  - GROUP BY-Klausel,474
  - IDENTITY,662
  - mit Standardwerten arbeiten,853
  - SELECT-Anweisungen,298
  - SELECT-Liste,297
  - Spalten-Integritätsregeln verwalten,862
  - Standardwerte,852
  - timestamp,661
- Spalten-Integritätsregeln
  - Tools für die Aufrechterhaltung der Datenintegrität,851
- UNIQUE,863
- Spalten-Standardwerte
  - ändern und löschen,853
- Spaltenattribute
  - AUTOINCREMENT,949
  - NEWID,949
  - Standardwerte generieren,949
- Spalteneindeutigkeit erzwingen
  - Info,40
- Spaltenreihenfolge
  - Ergebnisse spiegeln Reihenfolge in SELECT-Liste wider,298
  - Tabellen,259
  - Zusammengesetzte Indizes,29
- Spaltenstandardwerte
  - definierter Wert für Variable beginnend mit @,852
- Spaltenstatistiken
  - aktualisieren,229
  - Info,334
- Spaltenstatistiken aktualisieren
  - Info,229
- Speicher
  - Verbindungslimit,262
- Speichern
  - Ergebnismengen,783
  - Transaktionsergebnisse,882
- Speichernutzungswächter
  - Info über,221
- Speicherseiten, Statistik
  - Liste,215
- Sperren
  - Absichtssperren,910
  - Absichtssperren für Zeilen,910
  - Anzeigen von Deadlocks in Sybase Central,905
  - Auswirkungen durch Indizes vermindern,926
  - Auswirkungen von WITH HOLD,922
  - bei Abfragen,915
  - bei Einfügungen,918

---

- blockieren,902
- Cursorstabilität,922
- Dauer,907
- Deadlock,903
- durch Indizes reduzieren,40
- Einfügesperren,914
- exklusive Schemasperren,909
- exklusive Tabellensperren,912
- gemeinsame Schemasperren,909
- gemeinsame Tabellensperren,911
- Implementierung auf Isolationsstufe 0,915
- Implementierung auf Isolationsstufe 1,915
- Implementierung auf Isolationsstufe 2,916
- Implementierung auf Isolationsstufe 3,916
- in Sybase Central anzeigen,908
- Info,906
- Informationen anzeigen,908
- Inkonsistenzen versus typische Isolationsstufen,895
- INSERT,914
- Isolationsstufe,885
- Isolationsstufen, praktische Einführung,932
- Konfliktbehandlung,902
- Konflikte,914
- Lesevorgänge für Zeilen,909
- mit der sa\_locks-Systemprozedur anzeigen,908
- Phantom,913
- Phantomsperren,913
- Position,912
- Positionssperren,912
- praktische Einführung zu Phantomsperren,945
- Schema,908
- Schreibabsicht für Tabelle,912
- Schreibvorgänge,910
- Tabellen,911
- Tabellensperren für Schreibabsicht,912
- Transaktionen blockieren und Deadlock,901
- Typen,907
- Typen-Konflikte,911
- typische Transaktionen versus Isolationsstufen,924
- Verfahren bei Aktualisierungen,920
- Verfahren bei Einfügungen,918
- Verfahren bei Löschungen,921
- vorzeitige Freigabe von,925
- während des Löschens,921
- während einer Aktualisierung,920
- Waisen und referenzielle Integrität,919
- Zeilen,909
- Zeilenlesesperren,909
- Zeilenschreibsperren,910
- Sperren setzen
  - praktische Einführung zu Phantomsperren,945
- Sperrenanzahl, Statistik
  - Beschreibung,216
- SQL
  - eingeben,294
  - Unterschiede zu anderen SQL-Dialekten,643
- SQL Anywhere
  - Unterschiede zu anderen SQL-Dialekten,643
  - XML-Unterstützung,677
- SQL Flagger
  - aufrufen,642
  - Info,641
  - SQL-Konformität mit UltraLite SQL prüfen,641
  - Standards und Kompatibilität,643
- SQL Remote
  - Datenbanken neu aufbauen,767,769
  - für den Ferndatenzugriff nicht unterstützte Funktionen,822
- SQL SECURITY-Klausel
  - Syntax,82
- SQL Server
  - Datentypkonvertierungen,838
  - Ferndatenzugriff,838
- SQL Skriptdateien
  - SQL-Anweisungen, Fensterausschnitt,780
- SQL Standards
  - nicht-ANSI-Joins,499
- SQL, Standards
  - GROUP BY-Klausel,478
- SQL-Abfragen
  - Info,294
- SQL-Anweisungen
  - bei Transaktionen mit Snapshot-Isolation nicht erlaubt,889
  - in Interactive SQL ausführen,780
  - kompatible SQL-Anweisungen schreiben,663
  - Skriptdateien erstellen,780
- SQL-Ausdruck
  - Element in Ausführungsplänen,376
- SQL-Präprozessor, Dienstprogramm (sqlpp)
  - SQL Flagger, Verwendung,642
- SQL-Skriptdateien
  - ausführen,780
  - Ausgabe schreiben,783
  - erstellen,780

- Info,780
- SQL-Skriptdateien ausführen
  - ohne Laden,781
- SQL-Skripten
  - in Interactive SQL laden,783
- SQL-Standards
  - Entsprechung,641
  - Info,643
  - Konformität von SQL-Anweisungen prüfen,641
  - räumliche Daten,641
  - spezielle Funktionen von SQL Anywhere,643
- SQL/1999
  - Konformität von SQL-Anweisungen prüfen,641
- SQL/2003
  - Konformität von SQL-Anweisungen prüfen,641
- SQL/2008
  - Konformität von SQL-Anweisungen prüfen,641
  - spezielle Funktionen von SQL Anywhere,643
- SQL/XML
  - Info,687
- sql\_flagger\_error\_level, Option
  - SQL Flagger, Verwendung,642
- sql\_flagger\_warning\_level, Option
  - SQL Flagger, Verwendung,642
- SQL\_TXN\_ISOLATION
  - Info,899
- SQL\_TXN\_READ\_COMMITTED
  - ODBC-Isolationsstufe,898
- SQL\_TXN\_READ\_UNCOMMITTED
  - ODBC-Isolationsstufe,898
- SQL\_TXN\_REPEATABLE\_READ
  - ODBC-Isolationsstufe,898
- SQL\_TXT\_SERIALIZABLE
  - ODBC-Isolationsstufe,898
- SQLCA.lock
  - Isolationsstufen auswählen,898
  - versus Isolationsstufen,895
- SQLCODE, Variable
  - Einführung,131
- SQLFLAGGER-Funktion
  - SQL Flagger, Verwendung,642
- SQLSetConnectOption
  - Info,899
- SQLSTATE, Variable
  - Einführung,131
- SrtN, Planelement
  - Abkürzungen im Plan,356
- st\_geometry\_predefined\_srs-Systemprozedur
  - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- st\_geometry\_predefined\_uom-Systemprozedur
  - durch Aufrufer/Definierer-Einstellung für Datenbank beeinflusst,85
- Standardabweichungsfunktionen
  - OLAP,584
- Standardausgabe
  - umleiten an Datei,744,753,754
- Standardkonstantenausdruck
  - Info,859
- Standardverarbeitung von Warnungen in Prozeduren und Triggern
  - Info,135
- Standardwerte
  - aktuelles Datum und aktuelle Uhrzeit,854
  - AUTOINCREMENT,856
  - Benutzer-ID,855
  - definierter Wert für Variable beginnend mit @,852
  - erstellen,853
  - für die Datenintegrität,850
  - GLOBAL AUTOINCREMENT,856
  - in Sybase Central erstellen,853
  - INSERT-Anweisung,627
  - Konstantenausdruck,859
  - mit Transaktionen und Sperren,949
  - NEWID,858
  - NULL,859
  - Spalten,852
  - Transact-SQL,652
  - Zeichenfolgen und Zahlen,859
- Starten
  - Transaktionen,882
- statement-snapshot, Isolationsstufe
  - Info,923
  - SELECT-Anweisung, sperren,917
- statement-snapshot, schreibgeschützt, Isolationsstufe
  - SELECT-Anweisung, sperren,917
- Statistik
  - Statistik-Governor,230
- Statistik-Aufräumvorgang
  - Info,230
- Statistik-Governor
  - Info,230
- Statistiken
  - alphabetische Liste der Anforderungsstatistiken,216



- 
- alphabetische Liste der benutzerdefinierten Statistiken,217
  - alphabetische Liste der Cache-Statistiken,207
  - alphabetische Liste der Checkpoint- und Wiederherstellungsstatistiken,208
  - alphabetische Liste der Festplatten-I/O-Statistiken,211
  - alphabetische Liste der Indexstatistiken,213
  - alphabetische Liste der Kommunikationsstatistiken,209
  - alphabetische Liste der sonstigen Statistiken,218
  - alphabetische Liste der Speicherdiagnose-Statistiken,213
  - alphabetische Liste der Speicherseiten-Statistiken,215
  - alphabetische Liste der Statistiken über Festplatten-Lesezugriffe,211
  - alphabetische Liste der Statistiken über Festplatten-Schreibzugriffe,212
  - aus dem Systemmonitor entfernen,204
  - Ausführungspläne,341
  - benutzerdefinierte,217
  - Cache,207
  - Checkpoint- und Wiederherstellung,208
  - dem Systemmonitor hinzufügen,204
  - Festplatten-Lesezugriffe,211
  - Festplatten-Schreibzugriffe,212
  - Index,213
  - Kommunikation,209
  - Liste,207
  - mit dem Systemmonitor überwachen,203
  - Performance überwachen,207
  - Spaltenstatistiken aktualisieren,229
  - Speicherseiten,215
  - Statistik-Aufräumvorgang,230
  - über Festplatten-I/O-Vorgänge,211
  - überwachen,201
  - verschiedene,218
  - Zugriffspläne,370
  - Statistiken entfernen
    - Systemmonitor,204
  - Statistiken hinzufügen
    - Systemmonitor,204
  - Statistiken im Plan
    - Info,370
  - Status
    - von regulären Ansichten,48
  - Status und Eigenschaften von materialisierten Ansichten
    - Info,76
  - STDDEV-Funktion
    - äquivalente mathematische Formel,598
  - STDDEV\_POP-Funktion
    - äquivalente mathematische Formel,598
    - Beispiel,585
    - Verwendung,585
  - STDDEV\_SAMP-Funktion
    - äquivalente mathematische Formel,598
    - Beispiel,587
    - Verwendung,587
  - Stern-Joins
    - Info,513
  - Sternchen
    - bei Präfixsuche in Volltextsuchen verwendet,394
  - SELECT-Anweisung,296
  - Steueranweisungen
    - in Batches,112
    - Liste,118
  - STOPLIST, Einstellung
    - Definition,411
  - Stopplisten
    - Info,406
    - Verhalten bei der Suche nach Stopplistenbegriffen ,447
    - Volltextsuche,406
    - Vorsicht bei Anwendung,411
  - Strategien, zeitsparende
    - Daten importieren,723
  - Stücklistenproblem
    - Info,547
  - Suchbedingungen
    - Beispiel mit NOT-Schlüsselwort,311
    - Datumsangaben vergleichen,320
    - GROUP BY-Klausel,328
    - Musterübereinstimmung,313
    - Unterabfragen,600
    - Verwendungszweck,309
  - Suchen
    - chinesische, japanische und koreanische Daten (CJK),379
    - Volltextsuche,379
  - SUM-Funktion
    - äquivalente mathematische Formel,598
    - Verwendung,572
  - Summenwerte

- GROUP BY-Klausel,474
  - Info,470
  - Sybase Central
    - Datenbanken entladen,750
    - Inhalt von Systemobjekten anzeigen,3
    - Konvertierungsverfahren,671
    - Profilerstellung für Anwendungen,156
    - Systemobjekte anzeigen,3
    - Tabellen erstellen,5
    - Tabellen-Integritätsregeln,862
  - Sybase IQ
    - Fernzugriff,836
  - Symbole
    - Zeichenfolgenvergleiche,313
  - Synchronisation
    - Datenbanken neu aufbauen,767,769
  - Syntax-unabhängige Optimierung
    - Info,333
  - SYSCOLSTAT
    - Systemansicht, Spaltenstatistiken aktualisieren,229
  - SYSCOLUMNS
    - Transact-SQL, Namenskonflikte,656
  - SYSINDEXES
    - Transact-SQL, Namenskonflikte,656
  - SYSSERVER
    - Systemansicht, Fremdserver,788
  - SYSTEM PROCEDURE AS INVOKER-Klausel
    - Syntax,82
  - Systemadministrator
    - Adaptive Server Enterprise,653
  - Systemanbindungsprobleme
    - Ferndatenzugriff,823
  - Systemansichten
    - Indizes,37
    - Informationen zur referenziellen Integrität,877
    - Liste von Systemansichten nach Eigentümer abfragen,3
  - Systemausfall
    - Transaktionen,625
  - Systemfunktionen
    - TSEQUAL,662
  - Systemkatalog
    - Adaptive Server Enterprise,653
  - Systemmonitor
    - in Sybase Central öffnen,203
    - Info,203
    - Liste der unterstützten Statistiken,207
    - Statistiken hinzufügen und entfernen,204
  - Sybase Central,203
  - Überblick,203
  - Windows-Systemmonitor,205
  - Systemobjekte
    - Inhalt anzeigen,3
    - Liste von Systemobjekten nach Eigentümer abfragen,3
    - Listen von Objekten in Interactive SQL anzeigen,3
    - Listen von Systemobjekten in Interactive SQL anzeigen,3
    - Systemobjekte einer Datenbank anzeigen,3
  - Systemprozeduren
    - Ergebnisse der Profilerstellung generieren und prüfen,158
    - Prozedurprofilerstellung mit Systemprozeduren,197
  - Systemtabellen
    - Adaptive Server Enterprise,653
    - Eigentümer,653
    - Informationen zur referenziellen Integrität,877
    - Inhalt anzeigen,3
    - Liste von Systemtabellen nach Eigentümer abfragen,3
    - Transact-SQL, Namenskonflikte,656
  - Systemtrigger
    - Ergebnisse der Profilerstellung generieren und prüfen,158
    - referenzielle Integrität erzwingen,873
    - referenzielle Integritätsaktionen implementieren,874
- ## T
- Tabellen
    - Abhängigkeiten anzeigen,8
    - alternieren, wenn von materialisierter Ansicht referenziert,43
    - ändern mit Sybase Central,6
    - arbeiten mit,4
    - Arbeitstabellen,239
    - aus mehreren Datenbanken verknüpfen,812
    - Bitmaps,260
    - CHECK-Integritätsregeln,864
    - Daten exportieren,770
    - Daten in Sybase Central anzeigen und bearbeiten,9
    - Daten mit SQL anzeigen und bearbeiten,10
    - Defragmentierung,249
    - Einfügesperren,914

- 
- entfernte Tabellen auf einem Server auflisten,794
  - erstellen,5
  - Exklusivsperrern,912
  - exportieren,758
  - Ferndatenzugriff,785
  - Fragmentierung, praktische Einführung,281
  - Fremdschlüssel in Sybase Central erstellen,23
  - Fremdschlüssel mit SQL hinzufügen,24
  - Fremdschlüssel mit SQL verwalten,24
  - Fremdschlüssel verwalten,21
  - gemeinsame Sperren,911
  - Gruppenlesevorgänge,260
  - Hinweise zum Ändern,5
  - importieren,739
  - in Abfragen benennen,305
  - in Sybase Central entladen,752
  - Inhalt von Systemtabellen anzeigen,3
  - Integritätsregeln verwalten,862
  - Korrelationsnamen,305
  - kurze Gegenüberstellung von regulären und materialisierten Ansichten,42
  - löschen mit Sybase Central,8
  - mit Proxy-Tabellen arbeiten,805
  - Phantomsperrern,913
  - Positionssperrern,912
  - Primärschlüssel in Sybase Central anzeigen,18
  - Primärschlüssel in Sybase Central hinzufügen,18
  - Primärschlüssel in Sybase Central verwalten,18
  - Primärschlüssel mit SQL hinzufügen,19
  - Primärschlüssel mit SQL verwalten,19
  - Primärschlüssel verwalten,17
  - Proxy-Tabellen in Sybase Central erstellen,807
  - Proxy-Tabellen mit der CREATE EXISTING TABLE-Anweisung erstellen,807
  - Proxy-Tabellen mit der CREATE TABLE-Anweisung erstellen,807
  - qualifizierte Namen,1
  - Referenzen von anderen Tabellen anzeigen,23
  - Sperren,911
  - Sperren für Schreibabsicht,912
  - temporäre Tabellen erstellen,12
  - Transact-SQL-kompatible Tabellen erstellen,664
  - Zeilen kopieren,630
  - Tabellen exportieren
    - Schemata,771
    - UNLOAD-Anweisung,758
  - Tabellen importieren
    - DEFAULTS-Option,741
    - nicht übereinstimmende Tabellenstrukturen,741
    - NULL-Werte,741
    - Tabellenstrukturen zusammenführen,741
    - temporäre Tabellen,741
  - Tabellen neu organisieren
    - Tabellenfragmentierung reduzieren,249
  - Tabellen verknüpfen
    - mehr als zwei Tabellen mit einem Join,498
    - zwei Tabellen,497
  - Tabellen-Hints
    - entsprechende Isolationsstufen,885
  - Tabellen-Integritätsregeln
    - Tools für die Aufrechterhaltung der Datenintegrität,851
    - UNIQUE,863
  - Tabellenausdrücke
    - Schlüssel-Joins,528
    - verknüpfen,498
  - Tabellenfragmentierung
    - Fragmentierung, Registerkarte,251
    - Info,249
    - praktische Einführung,281
  - Tabellenfunktionen
    - SQL Anywhere-Implementierung,646
  - Tabellengröße
    - Hinweise zur Performance,260
    - Info,260
  - Tabellennamen
    - identifizieren,295
    - Prozeduren und Trigger,147
    - vollqualifizierte in Prozeduren,147
  - Tabellennamen nachschlagen, Fenster
    - Tabellenliste anzeigen,492
  - Tabellensperrern
    - Einfügesperrern,914
    - exklusive,912
    - gemeinsame,911
    - gemeinsame Tabellensperrern,911
    - Info,911
    - Konflikte,911
    - Phantomsperrern,913
    - Position,912
    - Schreibabsicht,912
  - Tabellensperrern für Schreibabsicht
    - Info,912
  - Tabellensperrern, Register
    - Sybase Central,908
  - Tabellenstrukturen

- für den Datenimport,741
- Tabellenstrukturen zusammenführen
  - Info,741
- Table-Scans
  - Plattenspeicherzuordnung und Performance,631
- TableScan, Planelement
  - Abkürzungen im Plan,356
- TCP/IP
  - Performance,264
- Temporäre Dateien
  - Arbeitstabellen,256
- Temporäre Prozeduren
  - erstellen,89
- Temporäre Tabellen
  - Arbeitstabellen in Abfrageverarbeitung,239
  - Daten importieren,741
  - erstellen,5
  - Hinweise zum Referenzieren aus Prozeduren,12
  - in Sybase Central erstellen,12
  - Indizes,29
  - Info,10
  - lokale und globale,10
  - mit temporären Tabellen arbeiten,10
  - nicht-transaktional machen,10
  - Tabellenstrukturen zusammenführen,741
  - Transact-SQL-Kompatibilität,665
  - Vorteile der nicht-transaktionalen Eigenschaft,10
- Temporäre Tabellenseiten, Statistik
  - Beschreibung,218
- TERM BREAKER-Einstellung
  - Definition,407
- Textindizes
  - aktualisieren,385
  - Aktualisierungstyp ändern,386
  - Aktualisierungstyp auswählen,424
  - Alterung und Aktualisierung,424
  - ändern,382,386
  - Ansichten abfragen,402
  - Auswirkung von Datenbankoptionen auf Erstellen und Aktualisierung,413
  - Begriffe und Einstellungen anzeigen (SQL),389
  - Begriffe und Einstellungen anzeigen (Sybase Central),388
  - Einstellungen in zugrunde liegenden Textkonfigurationsobjekten,406
  - erfordern Speicherplatz,379
  - erstellen,383
  - für Ansichten oder temporäre Tabellen nicht zulässig,423
  - GENERIC, Volltextsuche, praktische Einführung,426
  - Info,423
  - Nicht-Fuzzy, praktische Einführung,437
  - Textkonfigurationsobjekt nicht änderbar,382
  - umbenennen,386
  - Volltextsuche,423
  - Volltextsuche, Fuzzy, praktische Einführung,433
- Textindizes verwalten
  - Info,424
- Textkonfigurationsobjekte
  - ändern,382
  - Beispiele,415
  - Datum-, Zeit-, und Zeitstempelformate,382
  - default\_char-Einstellungen,406
  - default\_nchar-Einstellungen,406
  - Einstellungen für default\_char und default\_nchar,414
  - Einstellungen für Textkonfigurationsobjekte anzeigen,383
  - ermitteln, ob von Textindizes verwendet,383
  - erstellen,381
- Textpläne
  - lesen,342
- Thread-Deadlock
  - Beschreibung,903
  - Info,903
- Thread-Sicherheit
  - benutzerdefinierte Funktionen,93
- Threads
  - Deadlock, wenn keine verfügbar,903
- Tiefe
  - Element in Ausführungsplänen,374
- TIME-Format
  - Textindizes,413
- time\_format, Option
  - Auswirkung auf Textindizes,413
- time\_format-Option
  - ändern für Textkonfigurationsobjekte,382
- TIMESTAMP, Datentyp
  - Transact-SQL,661
- TIMESTAMP-Format
  - Textindizes,413
- timestamp\_format, Option
  - Auswirkung auf Textindizes,413
- timestamp\_format-Option

---

- ändern für Textkonfigurationsobjekte,382
- timestamp\_with\_time\_zone\_format-Option
  - ändern für Textkonfigurationsobjekte,382
- Tipps
  - Performance verbessern,220
- Tipps für Spitzen-Performance
  - Liste,219
- Tipps zum Schreiben von Prozeduren
  - Anweisungen müssen in der Prozedur immer getrennt werden,147
  - Voll qualifizierte Namen für Tabellen in Prozeduren verwenden,147
- Tipps zur Performance-Verbesserung
  - Abfrageperformance überwachen,245
  - Fragmentierung reduzieren,248
  - Tabellenfragmentierung reduzieren,249
- Token
  - Begriffsegmentierer und Volltextsuche,448
- Tools
  - Daten entladen,743
  - Daten importieren,723
  - Daten neu laden,763
  - Datenbanken neu aufbauen,763
  - Exportieren von Daten,743
- TOP-Klausel
  - verwenden,483
- Tr.Log-Gr.-Festschr./Sek., Statistik
  - Beschreibung,212
- TRACEBACK-Funktion
  - Info,133
- Transact-SQL
  - Adaptive Server Enterprise emulieren,657,658
  - Anweisung RAISERROR in Prozeduren verwenden,675
  - Batches,671
  - Datenbanken erstellen,657,658
  - Datenbanken für Transact-SQL-Kompatibilität konfigurieren,656
  - Einschränkungen bei Outer-Joins,509
  - Ergebnismengen,672
  - Ergebnismengen aus Transact-SQL-Prozeduren zurückgeben,672
  - Fehlerbehandlung in Transact-SQL-Prozeduren,673
  - IDENTITY-Spalte,662
  - Joins,668
  - Kompatibilität, Überblick,648
  - kompatible SQL-Anweisungen schreiben,663
  - nachgestellte Leerzeichen,656
  - nicht unterstützte Datei-Manipulationsanweisungen,651
  - NULL,664
  - NULL und Joins,510
  - Optionen für Transact-SQL-Kompatibilität festlegen,659
  - Outer-Joins,509
  - Outer-Joins und Ansichten,510
  - portierbare SQL-Anweisungen schreiben,663
  - Prozeduren,669
  - spezielle TIMESTAMP-Spalte und TIMESTAMP-Datentyp in Transact-SQL,661
  - Trigger,669
  - Überblick über Batches,671
  - Übersetzen von Prozeduren,671
  - Variable,673
  - WITH ROLLUP-Syntax,560
- Transact-SQL, Kompatibilität
  - Datenbanken,660
  - Datenbankoptionen einstellen,659
- Transact-SQL-Kompatibilität
  - SELECT-Anweisung,665
- Transact-SQL-Prozedursprache
  - Überblick ,669
- Transaktionen
  - abschließen,882
  - beenden,882
  - blockieren,902
  - Datenmanipulation,624
  - Datenwiederherstellung,625
  - Deadlock,903
  - Einführung in Snapshot-Isolation,890
  - Einschränkungen bei der Transaktionsverwaltung,818
  - Ferndatenzugriff,818
  - fertig stellen,882
  - Info,881
  - Interferenz zwischen,902
  - Interferenz zwischen Transaktionen,902
  - Isolationsstufen ändern,900
  - mehrere,884
  - Parallelität,884
  - praktische Einführung zur Blockierung,932
  - Prozeduren und Trigger,146
  - Reihenfolge von,924
  - Savepoints,884
  - starten,882

- typische Isolationsstufen,924
- überlappende,924
- Untertransaktionen und Savepoints,884
- verwenden,882
- Transaktionen blockieren
  - Info,901
- Transaktionen und Isolationsstufen
  - Info,881
- Transaktions-Festschreibungen, Statistik
  - Beschreibung,216
- Transaktions-Rollbacks, Statistik
  - Beschreibung,216
- Transaktionsergebnisse speichern
  - Info,882
- Transaktionslog
  - Datenwiederherstellung,722
  - dbmsync,767,769
  - Performance,884
  - Performance-Tipp,247
  - Replikation,767,769
- Transaktionslogs
  - Tipps zur Performance-Verbesserung,248
- Transaktionssperren
  - Dauer,907
- Transaktionsverarbeitung
  - Auswirkung von Zeitplänen,924
  - blockieren,902
  - Datenwiederherstellung,625
  - serialisierbare Zeitpläne,924
  - Zeitplan,924
- Transaktionsverwaltung
  - und entfernte Daten,817
- Transaktionszeitpläne
  - Auswirkungen,924
- Transformationen
  - Liste der Neuschreibungsoptimierungen,340
- trantest
  - Info,245
- Trennzeichen
  - einstellen,146
  - Konsistenz beim Abfragen von Verzeichniszugriff-Proxy-Tabellen,801
- Trigger
  - AFTER-Trigger,98
  - ändern,104
  - Anweisungsebene,669
  - Assistent zum Erstellen von Triggern,100
  - Aufbau,115
  - ausführen,104
  - Ausführungsberechtigungen,109
  - Auslöserang,109
  - Ausnahmeroutinen,136
  - BEFORE-Trigger,98
  - Cursor,128
  - Datumsangaben,147
  - Ergebnisse der Profilerstellung generieren und prüfen,158
  - erstellen (SQL),101
  - erstellen (Sybase Central),100
  - Fehlerbehandlung,131
  - Info,81
  - INPUT-Anweisung bewirkt Auslösen von INSERT-Trigger,721
  - INSTEAD OF-Trigger,110
  - löschen,106
  - Reihenfolge der Triggerauslösung,109
  - Rekursion,670
  - ROLLBACK-Anweisung,670
  - Savepoints,146
  - Tipps zum Schreiben,146
  - Transact-SQL,669
  - Transact-SQL-Kompatibilität und Benennung,661
  - Trennzeichen für Anweisungen,146
  - Typen,99
  - Überblick,81
  - UPDATE-Anweisung,634
  - verwenden,98
  - Vorgänge vorübergehend deaktivieren,107
  - Vorteile,82
  - Warnungen,135
  - Werkzeuge zur Aufrechterhaltung der Datenintegrität,851
  - Zeitangaben,147
  - zulässige Anweisungen,148
- Trigger auf Anweisungsebene
  - Transact-SQL,669
- Trigger erstellen, Assistent
  - verwenden,100
- Triggerbedingungen
  - Reihenfolge der Triggerauslösung,109
- TRUNCATE TABLE-Anweisung
  - mit Snapshot-Isolation verwenden,889
- trusted\_certificates\_file-Option
  - LDAP-Benutzerauthentifizierung aktivieren,958

---

## U

- UA, Planelement
  - Abkürzungen im Plan,356
- Überlaufterfehler
  - arithmetische Vorgänge,303
- Übernehmen
  - CROSS APPLY- und OUTER APPLY-Joins,517
- Übersetzen
  - Prozeduren,671
- ULODBC-Serverklasse
  - Info,827
- UltraLite
  - Konformität von SQL-Anweisungen prüfen,641
  - Serverklasse,827
- UltraLite SQL
  - prüfen, ob eine SQL Anywhere-Anweisung mit UltraLite SQL kompatibel ist,641
- Umleiten
  - Ausgabe an Datei,744,753,754
- Unbekannte Werte
  - Info,317
- Ungleichheiten
  - auf Ungleichheiten überprüfen,320
- Ungültige Daten
  - Info,849
- UNION-Klausel
  - Abfragen kombinieren,486
  - NULL,489
  - Regeln,487
  - Transact-SQL-Kompatibilität,665
- Union-Liste
  - Element in Ausführungsplänen,375
- UnionAll, Planelement
  - Abkürzungen im Plan,356
- UNIQUE-Integritätsregeln
  - Info,863
- Unix
  - anfängliche Cachegröße,223
  - maximale Cachegröße,223
  - minimale Cachegröße,223
- UNKNOWN
  - NULL,318
- Unnötige Distinct-Eliminierung
  - Info,241
- Unterabfrage-Tests
  - Info,609
- Unterabfragen
  - ALL-Test,613
  - als Joins umschreiben,616
  - ANY-Operator,612
  - ANY-Test,612
  - äußere Referenzen,608
  - Caching,339
  - Einführung,600
  - einzeilige Unterabfragen,600
  - entschachteln,241
  - Existenztest,614
  - GROUP BY,607
  - HAVING-Klausel,607
  - in einer WHERE-Klausel in Joins umwandeln,616
  - Info,600
  - Kategorisierung,600
  - korreliert,603
  - korrelierte Unterabfragen,603
  - mehrzeilige Unterabfragen,600
  - oder Joins,605
  - quantifizierter Vergleichstest,609
  - Schlüsselwort IN,313
  - Suchbedingungen,609
  - Test der Zugehörigkeit zu einer Gruppe,610
  - Typen von Operatoren,609
  - Vergleichsoperatoren,618
  - Vergleichstest,609
  - verschachtelt,604
  - WHERE-Klausel,606
  - WHERE-Klausel und Optimierer-Verhalten,616
  - Zeilenauswahl,606
  - Zeilengruppenauswahl,607
  - zu Joins konvertieren,616
- Unterabfragen entschachteln
  - Info,241
- Unterabfragen und Joins
  - Info,616
- Unterbrechungsbedingungen
  - einstellen,978
- Untertransaktionen
  - Prozeduren und Trigger,146
  - Savepoints,884
- Unzulässige XML-Namen kodieren
  - Info,689
- UPDATE, Konflikte
  - Snapshot-Isolation,894
- UPDATE-Anweisung
  - Beispiele,876
  - Fehler,876

- Joins aktualisieren,634
- sperren während,920
- SQL Anywhere-Implementierung,646
- Trigger auslösen,634
- Verletzungen von Integritätsregeln,636
- Verwendung,633
- Upgrade
  - Datenbankdateiformat,764
  - Datenbanken,764
- user\_estimates, Option
  - Optimiererstatistiken-Feldbeschreibungen,362
- USING CLIENT FILE-Klausel
  - importieren von und exportieren auf Clientcomputer,760
- USING VALUE-Klausel
  - importieren von und exportieren auf Clientcomputer,761
- UUID
  - generieren,949
  - Standard-Spaltenwert,858
  - verglichen mit globalem Autoincrement,858

## V

- VALIDATE LDAP SERVER-Anweisung
  - LDAP-Server zur Benutzerauthentifizierung verwenden,958
- Validieren
  - Indizes,34
  - WITH EXPRESS CHECK für Tabellen,238
  - XML,704
- Validierung
  - XML,704
- ValuePtr, Parameter
  - Info,899
- VAR\_POP-Funktion
  - äquivalente mathematische Formel,598
  - Beispiel,587
  - Verwendung,587
- VAR\_SAMP-Funktion
  - äquivalente mathematische Formel,598
  - Beispiel,588
  - Verwendung,588
- Variable
  - lokale,667
  - SELECT-Anweisung,667
  - SET-Anweisung,667
  - Transact-SQL,673

- zuordnen,667
- Variable prüfen
  - Debugger,981
- Variablen
  - Werte im Debugger anzeigen,981
- VARIANCE-Funktion
  - äquivalente mathematische Formel,598
- Varianzfunktionen
  - OLAP,584
- Vektoraggreatfunktionen
  - Info,477
- Veraltete Daten
  - manuelle Ansichten aktualisieren,60
- Veraltung
  - Einstellungen für materialisierte Ansichten,79
  - manuelle Ansichten,59
- Verbessern der Performance
  - Reihenfolge der Spalten in Tabellen,259
- Verbindungen
  - Debugger,983
  - entfernte,818
  - Ermitteln der Isolationsstufe,901
  - für Ferndatenzugriff verwalten,823
  - Loopback,812
  - materialisierte Ansichten als Kandidaten,237
- Verbindungen löschen
  - Ferndatenzugriff,823
- Verbindungsanzahl, Statistik
  - Beschreibung,218
- Verbindungsoptionen
  - Auswirkungen auf materialisierte Ansichten,61
- Verbindungssperren
  - Dauer,907
- Verfügbare I/O, Statistik
  - Beschreibung,218
- Vergleiche
  - Einführung,320
  - nachgestellte Leerzeichen,310
  - NULL,318
  - Sortierreihenfolge,310
- Vergleichsoperatoren
  - NULL,318
  - Unterabfrage,618
- Vergleichstest
  - Unterabfragen,609
- Verkettungszeichenfolgen
  - NULL,319
- Verschachtelt



- 
- Joins,498
    - Outer-Joins,507
  - Verschachtelte Unterabfragen
    - Info,604
  - Verschachtelte zusammengesetzte Anweisungen und Ausnahmeroutinen
    - Info,138
  - Verschachtelung
    - abgeleitete Tabellen in Joins,517
  - Verschiedene Statistiken
    - Liste,218
  - Verschlüsseln
    - materialisierte Ansichten mit Sybase Central,71
  - Verschlüsselung
    - Cachegröße,220
    - materialisierte Ansichten,71
    - Objekte verbergen,149
  - Versionsspeicherseiten, Statistik
    - Beschreibung,218
  - VersionStorePages, Eigenschaft
    - verwenden,890
  - Verzeichniszugriffsserver
    - ändern,802
    - Ergebnismenge,795
    - erstellen,796
    - Info,795
    - löschen,802
    - Proxy-Tabellen abfragen,800
    - Proxy-Tabellen löschen,802
    - Trennzeichen,801
  - Verzeichniszugriffsserver erstellen, Assistent
    - Info,796
  - Virtuelle Indizes
    - Indexberater,167
    - Info,167
  - Virtueller Speicher
    - wertvolle Ressource,220
  - Vollständige Outer-Joins
    - Info,505
  - Volltextsuche
    - Arten der Volltextsuche,390
    - Auswirkung von Datenbankoptionen auf Textindizes,413
    - Begriff-Höchstlänge,406
    - Begriff-Mindestlänge,406
    - Begriffe und Ausdrücke gruppieren,390
    - Begriffs- und Phrasensuche,390
    - Begriffsegmentierungsalgorithmus,406
    - Beispiel-Textkonfigurationsobjekte,415
    - Beispielzeichenfolgen-Interpretationen,417
    - Boolesche Suche,399
    - Callbacks aus externen Bibliotheken,456
    - chinesische, japanische und koreanische Daten (CJK),379
    - Dokumente,449
    - Eintrittspunktfunktion für Begriffsegmentierer,469
    - Eintrittspunktfunktion für Vorfilter,468
    - externe Begriffsegmentierer-Bibliothek deklarieren,454
    - externe Vorfilter-Bibliothek deklarieren,451
    - Fuzzy, praktische Einführung,433
    - GENERIC, praktische Einführung,426
    - Indizierung von Dateien wie Word, PDF und HTML,449
    - Info,379
    - mehrere Spalten durchsuchen,390,401
    - Nachbarschaftssuche,397
    - Nicht-Fuzzy, praktische Einführung,437
    - Phrasensuche,393
    - Präfixsuche,394
    - Punktwerte für Suchergebnisse erhalten,403
    - Stopplisten,406
    - Suche nach nicht-indizierten Begriffen,447
    - Textindizes,423
    - Textindizes ändern,382
    - Textindizes auflisten,388
    - Textindizes auflisten (SQL),389
    - Textindizes verwalten,423
    - Textkonfigurationsobjekte,406
    - Textkonfigurationsobjekte auflisten,383
    - Verbotene Schlüsselwörter und Platzhalter,390
    - Volltextabfrage erstellen,380
  - Vollvergleiche
    - Info,40
    - Statistiken in Zugriffsplänen,370
  - Voraussetzungen
    - SQL Anywhere-Debugger,971
  - Vorbereitete Anweisungen
    - Verbindungsbegrenzungen,262
  - Vorfilter
    - Ablauflogik für externe Vorfilter-Bibliothek,452
    - Beispiel für externen Vorfilter,451
    - Einstellungen für Textkonfigurationsobjekte,412
    - Volltextsuche, externe Vorfilter-Bibliothek definieren,451
  - Vorfilter-Bibliotheken

- Callbacks aus externen Bibliotheken,456
- Vorfiltern
  - Einstellungen für Textkonfigurationsobjekte,412
- Vorgänge mit Batches
  - Interactive SQL,782
- Vorphase der Optimierung
  - Abfrageverarbeitung,330
- Vorteile
  - Indexberater, Ergebnisse,168
- Vorwärmen
  - Cache,227
- Vorzeitige Freigabe von Sperren
  - Transaktionen,925

## W

- Waisen und referenzielle Integrität
  - Info,919
- wait\_for\_commit, Option
  - verwenden,919
- wait\_for\_commit-Option
  - UPDATE-Anweisungen mit Verletzungen von Integritätsregeln verarbeiten,636
- Warten
  - referenzielle Integrität überprüfen,919
- Warten, um auf gesperrte Zeilen zuzugreifen
  - Deadlock,902
- Wartende Abfragen, Statistik
  - Beschreibung,213
- Wartung
  - Performance,219
- Watcom SQL
  - Dialekt,649
  - Info,648
  - kompatible SQL-Anweisungen schreiben,663
  - Übersetzen von Prozeduren,671
- Werte speichern
  - allgemeine Tabellenausdrücke,542
- WHERE-Klausel
  - Einführung in Datumsvergleiche,320
  - GROUP BY-Klausel,476
  - Info,309
  - Joins,502
  - Musterübereinstimmung,313
  - NULL,318
  - Performance,292
  - und HAVING-Klausel,328
  - Unterabfragen,606
  - verglichen mit HAVING,480
  - Zeichenfolgenvergleiche,313
  - Zeilen in einer Tabelle ändern,633
- WHILE-Anweisung
  - Steueranweisungen,118
- Wiederherstellung
  - clientseitige Daten laden,762
  - Import/Export,722
- Wiederherstellung, Statistik
  - Liste,208
- Wiederherstellungs-Dringlichkeit, Statistik
  - Beschreibung,208
- Wiederherstellungs-I/O-Schätzung, Statistik
  - Beschreibung,208
- Wiederholbare Lesevorgänge
  - Arten von Inkonsistenz,895
  - Einführung,885
  - Einstellung für ODBC,898
  - Parallelität verbessern,925
  - SELECT-Anweisungen,915
- Window, Planelement
  - Abkürzungen im Plan,356
- WINDOW-Klausel
  - in der SELECT-Anweisung verwenden,564
  - Inlining und die WINDOW-Klausel,568
- Windows
  - Anfängliche Cachegröße,223
  - Maximale Cachegröße,223
  - Minimale Cachegröße,223
- Windows Mobile
  - Hinweise zur Cache- und Seitengröße,260
- Windows-Systemmonitor
  - Info,205
  - mehrere Kopien ausführen,205
  - Starten,205
- WITH CHECK OPTION-Klausel
  - mit der CREATE VIEW-Anweisung verwenden,47
- WITH CUBE-Klausel
  - Info,562
- WITH EXPRESS CHECK
  - Performance,238
- WITH HOLD-Cursor
  - Cursorstabilität,922
- WITH HOLD-Klausel
  - Cursorstabilität,922
- WITH RESULT SET-Klausel
  - EXECUTE IMMEDIATE in einer Prozedur verwenden,143

---

WITH ROLLUP-Klausel

Info,560

WITH-Klausel

allgemeine Tabellenausdrücke,536

Word-Dateien

Unterstützung für externen Vorfilter und  
Begriffsegmentierer-Bibliothek,449

WRITE CLIENT FILE-Systemprivileg

auf Clientcomputer exportieren,760

WRITE\_CLIENT\_FILE-Funktion

importieren von und exportieren auf  
Clientcomputer,760

## X

XML

Abfrageergebnisse als XML aus relationalen Daten  
erhalten,687

Abfrageergebnisse als XML erhalten,688

Daten mit dem DataSet-Objekt exportieren,679

Daten von Interactive SQL exportieren,678

Definition,677

FOR XML AUTO verwenden,693

FOR XML EXPLICIT verwenden,696

FOR XML RAW verwenden,691

in relationalen Datenbanken speichern,677

Info,677

Kodierung,677

mit dem DataSet-Objekt importieren,679,685

relationale Daten als XML exportieren,678

relationale Daten als XML importieren,679

Standard-Namespaces,686

Verwendung mit SQL Anywhere,677

XML importieren

Info,679

XML mit dem DataSet-Objekt,679,685

XML, Datentyp

verwenden,677

xml, Direktive

anwenden,703

XML-Dokument

wohlgeformt,705

XMLAGG-Funktion

verwenden,706

XMLCONCAT-Funktion

verwenden,707

XMLELEMENT-Funktion

verwenden,708

XMLFOREST-Funktion

verwenden,710

XMLGEN-Funktion

verwenden,711

xp\_cmdshell-Systemprozedur

durch Aufrufer/Definierer-Einstellung für  
Datenbank beeinflusst,85

xp\_getenv-Systemprozedur

durch Aufrufer/Definierer-Einstellung für  
Datenbank beeinflusst,85

xp\_read\_file-Systemprozedur

durch Aufrufer/Definierer-Einstellung für  
Datenbank beeinflusst,85

XML importieren,683

xp\_sendmail-Systemprozedur

durch Aufrufer/Definierer-Einstellung für  
Datenbank beeinflusst,85

xp\_startmail-Systemprozedur

durch Aufrufer/Definierer-Einstellung für  
Datenbank beeinflusst,85

xp\_startsmtp-Systemprozedur

durch Aufrufer/Definierer-Einstellung für  
Datenbank beeinflusst,85

xp\_stopmail-Systemprozedur

durch Aufrufer/Definierer-Einstellung für  
Datenbank beeinflusst,85

xp\_stopsmtmp-Systemprozedur

durch Aufrufer/Definierer-Einstellung für  
Datenbank beeinflusst,85

xp\_write\_file-Systemprozedur

durch Aufrufer/Definierer-Einstellung für  
Datenbank beeinflusst,85

XPath

verwenden,679

## Z

Zeichendaten

suchen,316

Zeichenfolgen

Anführungszeichen und Zeichenfolgen,316

Apostrophe,316

Datenbank mit Volltextsuche durchsuchen,379

quoted\_identifizier und Anführungszeichen,316

SELECT-Liste mit,301

übereinstimmende,313

und Zahlen, Standardwerte,859

verwenden,316

- Zeichensatzkonvertierung
  - Zugriff auf entfernte Daten,787
- Zeilen
  - Absichtssperren,910
  - auswählen,309
  - Auswirkungen des Löschens,632
  - Daten in alle Spalten einfügen,627
  - Daten in bestimmte Spalten einfügen,628
  - kopieren mit INSERT,630
  - Lesesperren,909
  - löschen,638
  - Schreibsperren,910
  - Sperren,909
- Zeilen beschränken
  - FIRST-Klausel,483
  - TOP-Klausel,483
- Zeilenbeschränkungsklauseln
  - verwenden,483
- Zeilenlimit gesamt
  - Element in Ausführungsplänen,376
- Zeilennummerierungsfunktionen mit Fenstern
  - Fenster-Rangfunktionen,590
  - Info,597
- Zeilensperren
  - Absichtssperren,910
  - Info,909
  - Lesevorgänge,909
  - Schreibvorgänge,910
  - Typen,909
- Zeilenumbruch
  - SQL,294
- Zeilenversionen
  - Info,890
- Zeitangaben
  - Prozeduren und Trigger,147
  - SQL Anywhere-Implementierung,644
- Zeitfolgen-Dienstprogramme
  - Info,200
- Zeitpläne
  - Auswirkungen von nicht-serialisierbaren,924
  - Auswirkungen von Serialisierung,924
  - Sperrendauer,922
- Zeitstempel
  - SQL Anywhere-Implementierung,644
- Zu lang laufende Abfragen
  - Abfrageperformance überwachen,245
  - Performance-Probleme beheben,189
  - praktische Einführung,272
- Zufällige Transitionen
  - Element in Ausführungsplänen,374
- Zugriffspläne
  - Erklärung der Statistiken,370
  - Info,333
- Zuordnen
  - Datentypen zu Spalten,865
  - Domänen zu Spalten,865
- Zusammenfassung, Registerkarte
  - Indexberater, Ergebnisse,167
- Zusammenführung
  - Verhalten mit Trigger,733
- Zusammenführungsverhalten
  - Info,732
- Zusammengesetzte Anweisungen
  - atomare,120
  - deklarieren,119
  - verwenden,119
- Zusammengesetzte Indizes
  - Auswirkung der Spaltenreihenfolge,29
  - Info,29
  - ORDER BY-Klausel,30
- Zusammenhängende Speicherung von Zeilen
  - Info,631
- Zusätzl. verfügbar, Statistik
  - Beschreibung,213
- Zuweisung von Mehrfachseiten, Statistik
  - Beschreibung,213
- Zwischenergebnisse
  - CUBE-Klausel,560
  - ROLLUP-Klausel,558
  - WITH CUBE-Klausel,562
  - WITH ROLLUP-Klausel,560
- Zyklischer Blockierkonflikt
  - Info,903