



UltraLite®

Datenbankverwaltung

Version 16.0

Februar 2013

Version 16.0
Februar 2013

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

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

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

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

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

Inhalt

Über diese Dokumentation	vii
UltraLite-Überblick	1
UltraLite-Architektur	1
Funktionen des UltraLite-Synchronisationsclients	2
UltraLiteunterstützte Plattformen	3
Funktionsvergleiche von UltraLite, UltraLite Java Edition und SQL Anywhere	4
Einschränkungen für UltraLite- und UltraLite Java Edition-Datenbanken	9
Überblick über die CustDB-Beispielanwendung	15
Hinweise zu UltraLite-Lösungen für Windows Mobile	18
UltraLite-Datenbank erstellen	21
Erstellen einer UltraLite-Datenbank mit dem Assistenten Datenbank erstellen	21
UltraLite-Datenbank über die Eingabeaufforderung erstellen	22
UltraLite-Datenbank unter Verwendung eines MobiLink- Synchronisationsmodells erstellen	23
UltraLite-Datenbank über die zentrale Administration von entfernten Datenbanken erstellen	23
Erstellen einer UltraLite-Datenbank aus einer XML-Datei	23
UltraLite-Datenbank bei der ersten Verbindung erstellen	25
UltraLiteDatenbank-Erstellungsparameter	25
UltraLite-Zeichensätze	26
Datenbanksicherheit	29
Aus einer SQL Anywhere-Datenbank in eine UltraLite-Datenbank konvertieren	33
UltraLite-Datenbankverbindungen	35
UltraLite-Verbindungszeichenfolgen und Parameter	35

UltraLite-Verbindungsparameter und die Umgebungsvariable	
ULSQLCONNECT	37
UltraLite-Dateipfadformate in Verbindungsparametern	37
Aufgaben und Merkmale einer UltraLite-Datenbank	39
Lesen der Datenbankeigenschaften	39
Zugriff auf Datenbankoptionen	40
UltraLite-Ereignisbenachrichtigungen	41
Isolationsstufen	43
Eine UltraLite-Datenbank validieren	46
Sichern und Wiederherstellen einer UltraLite- und UltraLite Java	
Edition-Datenbank	47
UltraLite-Datenbankschemas	49
UltraLite-Tabellen und -Spalten	50
UltraLite-Indizes	56
UltraLite-Benutzer	61
UltraLite als MobiLink-Client	69
UltraLite-Clients	69
ActiveSync mit UltraLite unter Windows Mobile	88
UltraLite-Synchronisationsparameter	90
UltraLite-Netzwerkprotokolloptionen für dbmlsync	114
UltraLite-Deployment	117
Kompilierungs- und Deploymentspezifikationen für UltraLite-	
Anwendungen	117
Datenbank-Deploymenttechniken für UltraLite und UltraLite Java	
Edition	125
Deployment von UltraLite-Datenbankschema-Upgrades	125
Start der UltraLite-Engine	128
Deployment des ActiveSync-Providers für UltraLite	128
Anwendungen mit dem ActiveSync Manager registrieren	130

Praktische Einführung: Erstellen der Beispielanwendung	
CustDB	133
Lektion 1: CustDB-Anwendung erstellen und ausführen	133
Lektion 2: MobiLink-Server starten und eine erste Synchronisation durchführen	134
Lektion 3: Daten in der UltraLite-Datenbank aktualisieren	135
Lektion 4: UltraLite-Datenbank mit der konsolidierten Datenbank synchronisieren	137
Lektion 5: MobiLink-Synchronisationsskripten durchsuchen	138
UltraLite-Datenbankreferenz	141
UltraLite-Erstellungsparameter	141
UltraLite-Verbindungsparameter	168
UltraLite-Datenbankeigenschaften	191
UltraLite-Datenbankoptionen	195
UltraLite-Dienstprogramme	200
UltraLite-Systemtabellen	238
UltraLite Java Edition-Datenbankeigenschaften	245
UltraLite Java Edition-Datenbankoptionen	246
Dienstprogramme der UltraLite Java Edition	252
Systemtabellen der UltraLite Java Edition	263
UltraLite-SQL-Referenz	271
UltraLite-SQL-Sprachelemente	271
UltraLite, SQLDatentypen	296
UltraLite SQL-Funktionen	316
UltraLite-SQL-Anweisungen	421
UltraLite Performance-Tipps	469
Cachegrößenanpassung für eine UltraLite-Datenbank	469
Cachegrößen der UltraLite Java Edition-Datenbank	470
Tipps für die Abfrageperformance	471
Performance-Tipps für Einfügen und Aktualisieren	483

Benchmarktipps für UltraLite	488
UltraLite-Fehlerbehandlung	495
Starten der UltraLite-Engine nicht möglich	495
Verbinden mit Datenbanken nach Upgrade nicht möglich	495
UltraLite-Datenbankbeschädigung	496
Datenbankgröße nicht stabil	497
Importieren von ASCII-Daten in eine neue UltraLite-Datenbank	498
Dienstprogramme laufen weiterhin in der vorherigen Version	498
Ergebnismenge ändert sich unvorhersehbar	499
UltraLite-Engine-Client schlägt mit Fehler -764 fehl	499
Index	501

Über diese Dokumentation

Dieses Handbuch ist eine Einführung in die UltraLite-Datenbanksysteme für kleine Geräte.

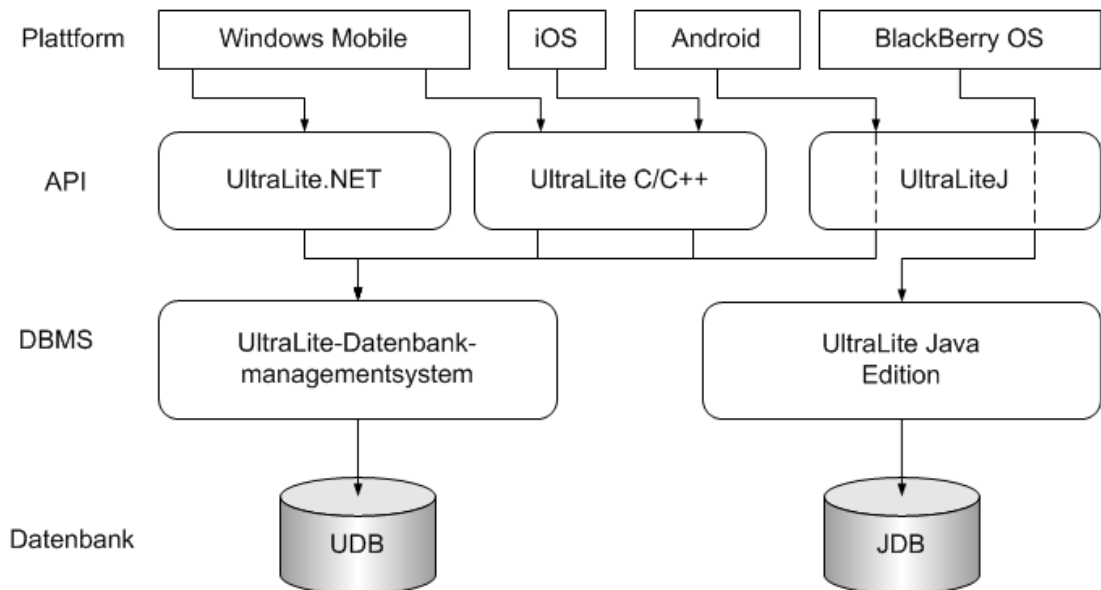
UltraLite-Überblick

UltraLite ist ein kompaktes relationale Datenbankmanagementsystem mit zahlreichen Features von SQL Anywhere. Sie kann verwendet werden, um mobile Datenbanken für Geräte mit geringem Speicherplatz wie z. B. Smartphones, Handhelds und Tablets zu entwickeln.

UltraLite enthält auch einen integrierten Synchronisationsclient, der Änderungen in UltraLite- und UltraLite Java Edition-Datenbanken protokolliert und Aktualisierungen über ein Netzwerk mit einem MobiLink-Server austauscht. Als MobiLink-Client stellt UltraLite sicher, dass mobile Anwendungen mit einer zentralen Datenbank und mit anderen UltraLite-Datenbanken synchron bleiben.

UltraLite-Architektur

UltraLite unterstützt eine Vielzahl von mobilen Plattformen und besteht aus API-Entwicklung, Datenbankverwaltung und Datenbankebenen, wie im folgenden Diagramm illustriert wird:



- **Mobile Plattformunterstützung** Ihre mobile Zielplattform bestimmt, welche UltraLite-API für die Entwicklung von Webanwendungen verfügbar ist. Weitere Hinweise zu unterstützten Plattformen und Geräten finden Sie unter <http://www.sybase.com/detail?id=1061806>.

- **API-Entwicklungsschicht** Dem obigen Diagramm können Sie entnehmen, welche API für Ihre mobile Zielplattform verwendet wird. Weitere Hinweise über die Entwicklung mit APIs finden Sie unter:
 - „UltraLite C++-Anwendungsentwicklung“ [[UltraLite - C- und C++-Programmierung](#)]
 - „UltraLite.NET-Anwendungsentwicklung“ [[UltraLite - .NET-Programmierung](#)]
 - „UltraLiteJ-Anwendungsentwicklung“ [[UltraLite® – Java-Programmierung](#)]
- **Datenbankverwaltungsebene und Synchronisationsclient** UltraLite bietet zwei Datenbankverwaltungssysteme, das native **UltraLite-Datenbank-Managementsystem** und die **UltraLite Java Edition**.
 - **UltraLite-Datenbank-Managementsystem** Die meisten mobilen Plattformen verwenden die UltraLite-APIs als Schnittstelle zum UltraLite-Datenbank-Managementsystem. Dieses System ermöglicht es Ihnen, eine **UltraLite-Datenbank** zu erstellen und Verbindung mit ihr aufzunehmen.

Eine umfassende Gruppe von Administrationstools wird bereitgestellt, um Sie bei der Verwaltung Ihres UltraLite-Projekts zu unterstützen. Sie können diese Tools entweder als Befehlszeilen-Dienstprogramm oder als Assistenten im UltraLite-Plug-In für Sybase Central ausführen. Weitere Hinweise finden Sie unter „[UltraLite-Dienstprogramme](#)“ auf Seite 200.
 - **UltraLite Java Edition** Die BlackBerry OS-Plattform erfordert die Verwendung der UltraLiteJ-API als Schnittstelle mit der UltraLite Java Edition, um eine **UltraLite Java Edition-Datenbank** zu erstellen und eine Verbindung mit ihr einzurichten. Diese Datenbanken sind nicht mit UltraLite-Datenbanken austauschbar und haben einige Einschränkungen. Weitere Hinweise finden Sie unter „[Einschränkungen für UltraLite- und UltraLite Java Edition-Datenbanken](#)“ auf Seite 9.

UltraLite stellt eine Reihe von Java Edition-Dienstprogrammen bereit, um zusätzliche Aufgaben in UltraLite Java Edition-Datenbanken zu übernehmen. Weitere Hinweise finden Sie unter „[Dienstprogramme der UltraLite Java Edition](#)“ auf Seite 252.

Weitere Hinweise zum UltraLite-Synchronisationsclient finden Sie unter „[Funktionen des UltraLite-Synchronisationsclients](#)“ auf Seite 2.
- **Datenbankebene** Diese Ebene ist das lokale Daten-Repository, das als Datei gespeichert wird. UltraLite-Datenbanken werden als UDB-Dateien gespeichert, UltraLite Java Edition-Datenbanken als JDB-Dateien. UDB-Dateien können auf alle mobilen Plattformen außer BlackBerry OS portiert werden.

Funktionen des UltraLite-Synchronisationsclients

UltraLite verfügt über einen integrierten bidirektionalen Synchronisationsclient, der bewirkt, dass alle Daten in einer UltraLite-Datenbank standardmäßig synchronisiert werden. Erstanwender der MobiLink-Synchronisation können dieses Standardverhalten zu Beginn nutzen und später, wenn die geschäftlichen Anforderungen eine benutzerspezifische Synchronisation erfordern, den Umfang und die Art der Synchronisation von UltraLite-Daten mit der konsolidierten Datenbank den jeweiligen Anforderungen

anpassen. Im Unterschied zu entfernten SQL Anywhere-Datenbanken brauchen Sie den UltraLite-Speicherplatz nicht zu erhöhen, um die Synchronisationsfunktionen einzubeziehen.

Zu den wichtigen Synchronisationsfunktionen, die in die UltraLite-Laufzeitbibliothek integriert sind, gehören die Protokollierung des Zeilenstatus und des Synchronisationsstatus.

Der Mechanismus zur Protokollierung des Zeilenstatus

Die Protokollierung des Tabellen- und Zeilenzustands ist für die Datensynchronisation besonders wichtig. Jede Zeile in einer UltraLite-Datenbank hat eine zugeordnete Zeilenstatusstruktur. Zusätzlich zur Synchronisation benutzt UltraLite diese Zeilenzustandsinformationen auch zur Steuerung der Transaktionsverarbeitung und der Datenwiederherstellung.

Synchronisationsstatus-Protokollierung

UltraLite verwendet einen Fortschrittszähler, um eine zuverlässige Synchronisation zu gewährleisten. Jeder Upload erhält einen eindeutigen Zähler, mit dem er identifiziert werden kann. Damit kann UltraLite im Falle eines Verbindungsfehlers feststellen, ob der Upload erfolgreich war.

Wenn Sie eine neue Datenbank erstellen, setzt UltraLite den Synchronisationsfortschritt zunächst immer auf Null. Ein Fortschrittszähler mit dem Wert 0 identifiziert die Datenbank als neue UltraLite-Datenbank, die den MobiLink-Server anweist, seine Statusinformation für diesen Client zurückzusetzen.

Vorsicht

Da in UltraLite der Fortschrittszähler bei jeder Synchronisation erhöht wird, können Sie eine UltraLite-Datenbank nicht mit verschiedenen konsolidierten Datenbanken synchronisieren. Ist der Wert des Fortschrittszählers ungleich Null und stimmt nicht mit der Nummer überein, die in der konsolidierten Datenbank gespeichert ist, meldet die MobiLink-Synchronisation einen Offset-Konflikt und die Synchronisation schlägt fehl. Sie können eine UltraLite-Datenbank nicht durch eine Sicherungskopie ersetzen, wenn der Fortschrittszähler älter ist als der aktuelle Wert.

Siehe auch

- „Verwaltung des Zeilenstatus in einer UltraLite-Datenbank“ auf Seite 485
- „MobiLink-Synchronisation“ [*MobiLink - Erste Orientierung*]
- „UltraLite als MobiLink-Client“ auf Seite 69
- „Praktische Einführung: MobiLink mit einer konsolidierten SQL Anywhere-Datenbank verwenden“ [*MobiLink - Erste Orientierung*]

UltraLiteunterstützte Plattformen

Weitere Hinweise zu den von UltraLite unterstützten Geräten, Plattformen und Netzwerkprotokollen finden Sie unter <http://www.sybase.com/detail?id=1061806>.

Siehe auch

- „UltraLite als MobiLink-Client“ auf Seite 69
- „UltraLite-Netzwerkprotokolloptionen für dbmlsync“ auf Seite 114
- „UltraLite-Synchronisationsdienstprogramm (ulsync)“ auf Seite 227

Funktionsvergleiche von UltraLite, UltraLite Java Edition und SQL Anywhere

Die folgende Tabelle verdeutlicht Funktionsvergleiche zwischen UltraLite und SQL Anywhere. UltraLite-Funktionen werden von der UltraLite Java Edition unterstützt, sofern nicht anders angegeben.

Hinweis

Das UltraLite-Datenbank-Managementsystem fügt der Größe Ihrer Anwendung 750-1500 KB hinzu, die UltraLite Java Edition 500 KB. Die Datenbank, der Datenbankserver und der Synchronisationsclient von SQL Anywhere fügen etwa 6 MB hinzu.

Funktion	SQL Anywhere	UltraLite	Hinweise
Transaktionsverarbeitung und Mehrtabellen-Joins	X	X	
Trigger, gespeicherte Prozeduren und Ansichten	X		
Externe gespeicherte Prozeduren (aufrufbare externe DLLs)	X		
Integrierte referenzielle Integrität und Entitätsintegrität	X	X ¹	UltraLite Java Edition erzwingt keine Fremdschlüssel-Integritätsregeln. Siehe Synchronisationsprobleme mit Fremdschlüsselzyklen vermeiden auf Seite 80 .
Kaskadierende Aktualisierungen und Löschungen	X	Eingeschränkt ¹	Die deklarative referenzielle Integrität, bei der Löschungen und Aktualisierungen kaskadierend sind, ist eine Funktion, die in UltraLite-Datenbanken nicht unterstützt wird – ausgenommen während der Synchronisation, wenn Löschungen für diesen Zweck kaskadiert werden.
Unterstützung mehrerer dynamischer Datenbanken	X	X	

Funktion	SQL Anywhere	UltraLite	Hinweise
Unterstützung von Anwendungen mit mehreren Threads	X	X	
Sperren auf Zeilenebene	X	X	
Dienstprogramme, um Daten in XML zu laden und zu entladen		X	<p>UltraLite verwendet separate Administrationstools für XML-Lade- und Entladevorgänge. Diese Funktionen sind nicht in die Laufzeitbibliothek integriert.</p> <p>Weitere Hinweise zu UltraLite finden Sie unter:</p> <ul style="list-style-type: none"> • „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222 • „UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload)“ auf Seite 233 <p>Weitere Hinweise zur UltraLite Java Edition finden Sie unter:</p> <ul style="list-style-type: none"> • „Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload)“ auf Seite 253 • „Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload)“ auf Seite 256
SQLX-Funktionalität	X		
SQL-Funktionen	X	X	<p>Nicht alle SQL-Funktionen sind zur Verwendung in UltraLite-Anwendungen verfügbar. Wenn Sie eine nicht-unterstützte Funktion verwenden, lösen Sie einen Fehler aus. Siehe „UltraLite SQL-Funktionen“ auf Seite 316.</p>

Funktion	SQL Anywhere	UltraLite	Hinweise
SQL-Anweisungen	X	X	Der Bereich der SQL-Anweisungen ist unterschiedlich im Vergleich mit SQL Anywhere. Siehe „ UltraLite-SQL-Anweisungen “ auf Seite 421.
Integrierter HTTP-Server	X		
Starke Verschlüsselung für Datenbankdateien und Netzwerk-kommunikation	X	X	
Ereignisplanung und Verarbeitung	X	X ¹	Ein UltraLite-Ereignismodell unterscheidet sich von SQL Anywhere.
Leistungsstarker, selbstoptimierender, kostenbasierter Abfrageoptimierer	X		UltraLite besitzt einen Abfrageoptimierer, der jedoch nicht so umfassend ist wie der von SQL Anywhere.
Wahl zwischen mehreren threadsicheren API-Schnittstellen	X	X	UltraLite stellt Anwendungsentwicklern eine außergewöhnlich flexible Architektur bereit, die es gestattet, Anwendungen für sich ändernde bzw. verschiedene Deploymentumgebungen zu erstellen. Siehe „ Vorteile der UltraLite APIs für Windows Mobile “ auf Seite 18.
Cursor-Unterstützung	X	X	Siehe „ Einschränkungen für UltraLite- und UltraLite Java Edition-Datenbanken “.

Funktion	SQL Anywhere	UltraLite	Hinweise
Dynamische Cachingdimensionierung	X	X ¹	<p>Mit UltraLite können Sie eine anfängliche, minimale oder maximale Cachegröße für eine Datenbank definieren. Die Größe des Caches wird von UltraLite laufend optimiert, bis die maximale Größe (falls angegeben) erreicht ist. Siehe:</p> <ul style="list-style-type: none"> • „UltraLite-Verbindungsparameter <code>CACHE_SIZE</code>“ auf Seite 172 • „UltraLite-Verbindungsparameter <code>CACHE_MIN_SIZE</code>“ auf Seite 171 • „UltraLite-Verbindungsparameter <code>CACHE_MAX_SIZE</code>“ auf Seite 170 <p>UltraLite Java Edition-Datenbanken unterstützen nur feste Cachegrößen.</p>
Datenbankwiederherstellung nach einem System- oder Anwendungsfehler	X	X	
BLOB-Unterstützung (Binary Large Object)	X	X	UltraLite kann BLOBs nicht indizieren oder vergleichen.
Integration des Windows-Systemmonitors	X		
Online-Defragmentierung von Tabellen und Indizes	X		
Online-Sicherung	X		

Funktion	SQL Anywhere	UltraLite	Hinweise
Direkte Geräteverbindungen zwischen PC und Windows Mobile-Gerät		X ¹	SQL Anywhere-Datenbanken brauchen einen Datenbankserver, damit sie PC-Verbindungen mit der Datenbank auf einem Windows Mobile-Gerät zulassen können. Unter UltraLite stellen Sie der Verbindungszeichenfolge WCE: \ voran. Siehe Windows Mobile auf Seite 38 .
Leistungsstarke Aktualisierungen und Abfragen durch die Verwendung von Indizes	X	X	UltraLite verwendet einen Mechanismus, um festzustellen, ob jede Tabelle mithilfe eines Indexes oder durch das direkte Überprüfen der Zeilen durchsucht werden soll. Außerdem können Sie die Hash-Methode auf Indizes anwenden, um die Datenabfrage zu beschleunigen. Siehe „ UltraLite-Erstellungsparameter max_hash_size “ auf Seite 151.
Synchronisation mit HANA, Oracle, DB2, Sybase Adaptive Server Enterprise, Microsoft SQL Server, MySQL oder SQL Anywhere	X	X	
Integrierte Synchronisation		X	Im Unterschied zu SQL Anywhere-Deployments benötigt UltraLite keinen Client-Agent für die Synchronisation. Die Synchronisation ist in die UltraLite-Laufzeitbibliothek integriert, um die Komponenten, für die das Deployment durchgeführt werden muss, zu minimieren. Siehe „ UltraLite-Clients “ auf Seite 69.
Prozessinterne Ausführung		X	
Berechnete Spalten	X		
Deklarierte temporäre Tabellen/globale temporäre Tabellen	X		

Funktion	SQL Anywhere	UltraLite	Hinweise
Systemfunktionen	X		
Zeitstempelspalten	X	X	<p>SQL Anywhere unterstützt den DEFAULT TIMESTAMP-Standardwert.</p> <p>UltraLite unterstützt nur den DEFAULT CURRENT TIME-STAMP-Standardwert. UltraLite kann daher den Zeitstempel nicht automatisch aktualisieren, wenn die Zeile aktualisiert wird.</p>
Benutzerbasiertes Berechtigungsschema zur Bestimmung objektbasierter Eigentümer und Zugriffsberechtigungen	X		<p>UltraLite wurde primär für Einzelbenutzer-Datenbanken konzipiert, in denen kein Autorisierungssystem erforderlich ist. Sie können jedoch bis zu vier Benutzer-IDs und Kennwörter festlegen, die ausschließlich für Authentifizierungszwecke verwendet werden. Die betreffenden Benutzer haben Zugriff auf alle Datenbankobjekte. Siehe „UltraLite-Benutzer“ auf Seite 61.</p>
Räumliche Daten	X	Eingeschränkt	UltraLite unterstützt nur Punktdaten.
Volltextdaten	X		

¹ Nicht verfügbar für UltraLite Java Edition.

Einschränkungen für UltraLite- und UltraLite Java Edition-Datenbanken

Die folgende Tabelle verdeutlicht feste Grenzen, die für UltraLite- und UltraLite Java Edition-Datenbanken gelten. In vielen Fällen liegen die Einschränkungen jenseits der Beschränkungen von mobilen Geräten. Performance-Überlegungen und Gerätefähigkeiten können striktere Einschränkungen bewirken.

Element	Einschränkungen der UltraLite-Datenbanken	Einschränkungen der UltraLite Java Edition-Datenbanken
Datenbank und Dateigröße	4 GB oder weniger, wenn das Betriebssystem die Dateigröße begrenzt.	Begrenzt durch das Betriebssystem. Die Anzahl der Datenbankseiten ist auf 64 kB begrenzt.
Größe der temporären Datei	Begrenzt durch das Betriebssystem.	Nicht anwendbar.
Cachegröße	Begrenzt durch den verfügbaren Speicher auf dem Gerät.	Begrenzt durch den verfügbaren Speicher auf dem Gerät.
Dynamische Cachedimensionierung	Mit UltraLite können Sie eine anfängliche, minimale oder maximale Cachegröße für eine Datenbank definieren. Die Größe des Caches wird von UltraLite laufend optimiert, bis die maximale Größe (falls angegeben) erreicht ist. Siehe: <ul style="list-style-type: none"> • „UltraLite-Verbindungsparameter <code>CACHE_SIZE</code>“ auf Seite 172 • „UltraLite-Verbindungsparameter <code>CACHE_MIN_SIZE</code>“ auf Seite 171 • „UltraLite-Verbindungsparameter <code>CACHE_MAX_SIZE</code>“ auf Seite 170 	UltraLite Java Edition-Datenbanken unterstützen nur feste Cachegrößen.
Maximale Anzahl von gleichzeitig geöffneten Verbindungen unterstützt von einer Datenbank	Bis zu 14.	Keine Grenze.
Maximale Anzahl von gleichzeitig geöffneten Verbindungen zu allen Datenbanken	Bis zu 16 auf einem mobilen Gerät und bis zu 64 auf einem PC.	Keine Grenze.
Maximale Anzahl der Datenbanken, die gleichzeitig laufen können	Bis zu 8 auf einem mobilen Gerät und bis zu 32 auf einem PC.	Keine Grenze.

Element	Einschränkungen der UltraLite-Datenbanken	Einschränkungen der UltraLite Java Edition-Datenbanken
Maximale Anzahl von Anwendungen, die gleichzeitig mit einer Datenbank verbunden sein können.	Verwenden Sie die UltraLite-Engine, um mehrere gleichzeitige Anwendungen zu handhaben, die eine Verbindung mit der Datenbank herstellen. Andernfalls kann jeweils nur eine Anwendung eine Verbindung mit einer Datenbank herstellen.	1
SQL-Kommunikationsbereiche (SQLCA)	Bis zu 63.	Nicht anwendbar.
Zeilen pro Tabelle	<p>Bis zu 16 Millionen</p> <p>Manchmal werden Änderungen der Zeile (Löschungen und Aktualisierungen) und andere Statusinformationen zusammen mit den Zeilendaten verwaltet. Dank dieser Informationen können diese Änderungen synchronisiert werden. Daher kann die aktuelle Zeilenbegrenzung unter 16 Millionen liegen, abhängig von der Anzahl der Transaktionen für eine Tabelle zwischen Synchronisationen oder davon, ob die Tabelle überhaupt synchronisiert wird. Siehe „UltraLite-Transaktionsverarbeitung“ auf Seite 486.</p>	Begrenzt durch die Seitengröße und die maximale Anzahl von Seiten pro Datenbank

Element	Einschränkungen der UltraLite-Datenbanken	Einschränkungen der UltraLite Java Edition-Datenbanken
Zeilengröße	<p>Die Länge einer gepackten Zeile darf die Seitengröße nicht übersteigen. Siehe Gepackte Zeilen und Tabellendefinitionen auf Seite 50.</p> <p>Zeichenfolgen, die kürzer als die Spaltengröße sind, werden gespeichert, ohne aufgefüllt zu werden. Diese Einschränkung gilt nicht für Spalten, die als LONG BINARY oder LONG VARCHAR deklariert sind, da diese Zeichenfolgen getrennt gespeichert werden.</p>	<p>Die Länge einer gepackten Zeile darf die Seitengröße nicht übersteigen. Siehe Gepackte Zeilen und Tabellendefinitionen auf Seite 50.</p> <p>Zeichenfolgen, die kürzer als die Spaltengröße sind, werden gespeichert, ohne aufgefüllt zu werden. Diese Einschränkung gilt nicht für Spalten, die als LONG BINARY oder LONG VARCHAR deklariert sind, da diese Zeichenfolgen getrennt gespeichert werden.</p>
Zeilen pro Datenbank	Begrenzt durch beständigen Speicher.	Begrenzt durch beständigen Speicher.
Tabellengröße	Begrenzt durch die Datenbankgröße.	Begrenzt durch die Datenbankgröße.
Tabellen pro Datenbank	Begrenzt durch die Datenbankgröße.	Bis zu 32000.
Spalten pro Tabelle	Die Zeilengröße ist durch die Seitengröße begrenzt. Die tatsächliche Grenze der Spaltenzahl pro Tabelle wird von dieser Größe abgeleitet. Gewöhnlich liegt die tatsächliche Grenze deutlich unter 4000.	Die Zeilengröße ist durch die Seitengröße begrenzt. Die tatsächliche Grenze der Spaltenzahl pro Tabelle wird von dieser Größe abgeleitet. Gewöhnlich liegt die tatsächliche Grenze deutlich unter 4000.
Indizes pro Tabelle	Begrenzt durch die Datenbankgröße.	Begrenzt durch die Datenbankgröße.
Anzahl von Publikationen	Bis zu 63.	Bis zu 63.
Seitengröße der Datenbank	Mindestens 1 kB und bis zu 16 kB.	Mindestens 256 Byte, bis zu 16 KB.

Element	Einschränkungen der UltraLite-Datenbanken	Einschränkungen der UltraLite Java Edition-Datenbanken
Cursor pro Verbindung	Auf einer bestimmten Verbindung mit einer UltraLite-Datenbank sind maximal 64 Cursor zulässig (alle Plattformen).	Auf einer bestimmten Verbindung mit einer UltraLite-Datenbank sind maximal 64 Cursor zulässig (alle Plattformen).
Zeichenfolgen	Die Zeile muss auf eine Seite passen.	Die Zeile muss auf eine Seite passen.
Binärdatentypen	Die Zeile muss auf eine Seite passen.	Die Zeile muss auf eine Seite passen.
Größe von Long Binary/Long Varchar	Nur begrenzt durch Datenbankgröße	Nur begrenzt durch Datenbankgröße
Blob-Größe	Begrenzt durch die Dateigröße.	Bis zu 2 ²⁴ Byte.
Verfügbare Isolationsstufen	0 (Nicht festgeschriebene Anweisungen lesen) oder 1 (Festgeschriebene Anweisungen lesen).	0 (Nicht festgeschriebene Anweisungen lesen).
Kaskadierende Aktualisierungen und Löschungen	Die deklarative referenzielle Integrität, bei der Löschungen und Aktualisierungen kaskadierend sind, ist eine Funktion, die in UltraLite-Datenbanken nicht unterstützt wird – ausgenommen während der Synchronisation, wenn Löschungen für diesen Zweck kaskadiert werden.	Nicht unterstützt.
Ereignisplanung und Verarbeitung	Ein UltraLite-Ereignismodell unterscheidet sich von SQL Anywhere.	Nicht unterstützt.

Element	Einschränkungen der UltraLite-Datenbanken	Einschränkungen der UltraLite Java Edition-Datenbanken
UltraLite Java Edition-Datenbankkompatibilität	<p>UltraLite-Datenbanken sind mit UltraLite Java Edition-Datenbanken nicht austauschbar. Eine UltraLite-Datenbank kann in eine UltraLite Java Edition-Datenbank konvertiert werden und umgekehrt. Verwenden Sie dazu die jeweiligen Dienstprogramme zum Laden und Entladen. Siehe:</p> <ul style="list-style-type: none"> • „UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload)“ auf Seite 233 • „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222 	<p>UltraLite-Datenbanken sind mit UltraLite Java Edition-Datenbanken nicht austauschbar. Eine UltraLite-Datenbank kann in eine UltraLite Java Edition-Datenbank konvertiert werden und umgekehrt. Verwenden Sie dazu die jeweiligen Dienstprogramme zum Laden und Entladen. Siehe:</p> <ul style="list-style-type: none"> • „Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload)“ auf Seite 256 • „Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload)“ auf Seite 253

Einschränkungen des Objektspeichers für BlackBerry

Bei einem BlackBerry-Smartphone wird die Größe des Datenbankspeichers durch die Anzahl der verfügbaren Objekt-Handles eingeschränkt. Die Anzahl der verfügbaren Objekt-Handles ist eine BlackBerry-Eigenschaft und wird durch die Größe des Flash-Speichers bestimmt.

UltraLite erfordert einen oder mehrere Objekt-Handles, abhängig vom Handle-Typ, um einen Datenbankwert im Speicher abzulegen. Eine Tabellenzeile mit zehn Spalten und zwei Indizes erfordert beispielsweise ein Minimum von zwölf Objekt-Handles.

Um größere Datenbankspeicher zu ermöglichen, gibt die UltraLiteJ-API den Benutzern die Möglichkeit, die Anzahl der im Speicher aufbewahrten Zeilen zu begrenzen. Gespeicherte Datenbankzeilen werden auf einer Datenbankseite kombiniert. Die UltraLiteJ-API erfordert nur einen beständigen Objekt-Handle für jede Datenbankseite.

Siehe auch

- „SQL Anywhere-Einschränkungen von Größe und Anzahl“ [[SQL Anywhere Server - Datenbankadministration](#)]

Überblick über die CustDB-Beispielanwendung

Das CustDB-Beispiel wird mit SQL Anywhere installiert. Es ist ein vielstufiges Datenbankverwaltungsbeispiel, das die MobiLink-Synchronisation mit einer konsolidierten SQL Anywhere-Datenbank implementiert.

CustDB hat folgende Komponenten:

- Eine **konsolidierte** SQL Anywhere-Datenbank. Diese Datenbank ist bereits mit Verkaufsdaten gefüllt.
- Eine **entfernte** UltraLite-Datenbank. Diese Datenbank ist anfangs leer.
- Eine UltraLite-Clientanwendung
- Synchronisationsskripten für den MobiLink-Server

Hinweis

Sie können gleichzeitig nur eine Instanz von CustDB ausführen. Der Versuch, mehr als eine Instanz auszuführen, bringt die erste Instanz in den Vordergrund.

CustDB gestattet es den Mitarbeitern, Transaktionen zu protokollieren und zu überwachen und Informationen von zwei Typen von Benutzern zu sammeln:

- Vertriebsmitarbeiter, die sich mit den Benutzer-IDs 51, 52 und 53 authentifizieren
- Mobile Manager, die sich mit der Benutzer-ID 50 authentifizieren

Die von diesen verschiedenen Benutzern gesammelten Informationen können mit der konsolidierten Datenbank synchronisiert werden.

Sowohl die konsolidierte als auch die entfernte Datenbank enthält eine Tabelle namens ULOrder. Während die konsolidierte Datenbank alle Bestellungen enthält (bestätigte und noch nicht bestätigte), zeigt die entfernte UltraLite-Datenbank nur einen Teil der Zeilen an, der abhängig vom angemeldeten Benutzer ist.

Spalten in der Tabelle werden als Felder in der Clientanwendung angezeigt. Wenn Sie eine Bestellung hinzufügen, müssen Sie die Felder "Customer", "Product", "Quantity", "Price" und "Discount" auffüllen. Sie können auch andere Detailinformationen anhängen, wie etwa den Status oder Notizen. In der Zeitstempelsspalte wird angegeben, ob die Zeile synchronisiert werden muss.

Die Synchronisationslogik für CustDB ist in der konsolidierten Datenbank in Form von MobiLink-Synchronisationsskripten gespeichert. Die Synchronisationslogik gestattet es Ihnen festzustellen, wie viel der konsolidierten Datenbank herunter- bzw. hochgeladen werden muss. Sie können mithilfe von Verfahren, wie etwa der zeitstempelbasierten Synchronisation oder der Snapshot-Synchronisation, vollständige Tabellen oder Teile von Tabellen (mit Zeilen- oder Spaltenteilmengen) herunterladen.

Sie können die Synchronisationsskripten, die in der konsolidierten Datenbank gespeichert sind, mit Sybase Central untersuchen. Sybase Central ist das vorrangige Tool zum Hinzufügen von Skripten zur Datenbank.

Die Datei *custdb.sql* fügt jedes Synchronisationsskript durch einen Aufruf von *ml_add_connection_script* bzw. *ml_add_table_script* zur konsolidierten Datenbank hinzu. Verbindungsskripten steuern Ereignisse höherer Ebenen, die keiner bestimmten Tabelle zugeordnet sind. Sie verwenden diese Ereignisse, um globale Aufgaben auszuführen, die bei jeder Synchronisation erforderlich sind. Tabellenskripten gestatten Aktionen bei bestimmten Ereignissen, die sich auf die Synchronisation einer bestimmten Tabelle beziehen, wie etwa den Start oder das Ende des Uploads von Zeilen, das Lösen von Konflikten oder das Auswählen von Zeilen, um einen Download durchzuführen.

SQL Anywhere-Datenbank CustDB

Dies ist die konsolidierte Datenbank. Bei der Installation wird für diese Datenbank die ODBC-Datenquelle SQL Anywhere 16 CustDB erstellt. Die Datenbankdatei befindet sich unter *%SQLANYSAMPI6%\UltraLite\CustDB*. Sie können diese Datenbank jederzeit mit *makedbs.cmd* (*makedbs.sh* für Mac OS X oder Linux) neu erstellen.

Sie können Änderungen löschen, die in die konsolidierte Datenbankdatei *CustDB.db* synchronisiert wurden, damit Sie eine saubere Version erhalten. Verwenden Sie dazu dieses Skript: *C:\Program Files\SQL Anywhere 16\UltraLite\CustDB\newdb.bat*.

Weitere Hinweise zum Schema dieser Datei finden Sie unter „[MobiLink-Beispiel CustDB](#)“ [[MobiLink - Erste Orientierung](#)].

UltraLite-Datenbank CustDB

Dies ist die entfernte Version der konsolidierten Datenbank, die nur eine Teilmenge der Informationen enthält, abhängig davon, welcher Benutzer die Datenbank synchronisiert.

Der Dateiname und der Speicherort können abhängig von Plattform, Programmiersprache oder auch Gerät variieren.

- Für UltraLite.NET: *C:\Program Files\SQL Anywhere 16\UltraLite.NET\CustDB\Common\custdb.udb*
- Für alle anderen Plattformen und APIs: *C:\Program Files\SQL Anywhere 16\UltraLite\CustDB\custdb.udb*

Die UltraLite-Datenbank wird auch mit dem Skript *makedbs.cmd* (*makedbs.sh* für Mac OS X oder Linux) neu erstellt.

RDBMS-spezifische Versionsskripten

Die SQL-Skripten bauen eine konsolidierte CustDB-Datenbank für die verschiedenen unterstützten RDBMS-Systeme neu auf.

Im Verzeichnis *C:\Program Files\SQL Anywhere 16\MobiLink\CustDB* können Sie die folgenden Dateien finden:

- Für SQL Anywhere: *custdb.sql*
- Für Adaptive Server Enterprise: *custase.sql*
- Für Microsoft SQL Server: *custmss.sql*
- Für Oracle: *custora.sql*
- Für IBM DB/2: *custdb2.sql*
- Für MySQL: *custmys.sql*

Weitere Hinweise zum Einrichten einer konsolidierten Datenbank finden Sie unter „[Eine konsolidierte CustDB-Datenbank einrichten](#)“ [[MobiLink - Erste Orientierung](#)].

UltraLite CustDB-Clientanwendungen und ReadMe-Dateien

Dies sind die Endbenutzeranwendungen, die eine komfortable Benutzeroberfläche für die entfernte UltraLite-Datenbank bereitstellen. Für jede unterstützte Plattform ist ein Beispiel-Client vorhanden.

Jede Clientanwendung enthält auch die Dateien *ReadMe.html* bzw. *ReadMe.txt*. Jede Datei enthält eine Beschreibung der Schritte, die zum Aufbauen und Ausführen des Beispiels erforderlich sind.

Der Speicherort der Anwendung und ihrer ReadMe-Datei hängt von Ihrer Entwicklungsumgebung ab. Siehe „[Lektion 1: CustDB-Anwendung erstellen und ausführen](#)“ auf Seite 133.

Synchronisationslogik

Die SQL-Anweisungen und Synchronisationsaufrufe der UltraLite-Datenbank befinden sich in *custdbcpp.cpp* für die C++-API und in *custdb.sql* für Embedded SQL.

Siehe auch

- „[MobiLink - konsolidierte Datenbanken](#)“ [[MobiLink - Serveradministration](#)]

CustDB-Dateispeicherorte

Die CustDB-Anwendung kann für zahlreiche Entwicklungsumgebungen erstellt werden.

UltraLite für 32-Bit-Windows-PCs

Es ist nicht erforderlich, die CustDB-Anwendung vor ihrer Ausführung zu erstellen.

Die CustDB-Programmdatei befindet sich im Verzeichnis *%SQLANY16%\UltraLite\Windows\x86*.

UltraLite für C/C++

- **Alle Versionen von C/C++** Aufgrund der zahlreichen verschiedenen C/C++-Entwicklungsumgebungen werden auch verschiedene Versionen der C/C++-CustDB-Projektdatei bereitgestellt. Die meisten Versionen verwenden die generischen Dateien. Diese Dateien befinden sich im Verzeichnis *C:\Program Files\SQL Anywhere 16\UltraLite\Custdb*.

Hinweise zu den verschiedenen Versionen der C/C++-CustDB-Anwendungen finden Sie unter *C:\Program Files\SQL Anywhere 16\UltraLite\Custdb\readme.txt*.

- **Visual Studio** Die Projektdateien befinden sich im Verzeichnis *%SQLANYAMP16%\UltraLite\CustDB*.
- **Xcode für iPhone** Weitere Hinweise zur Entwicklung für Mac OS X und iPhone finden Sie unter *samples/ultralite/custdb/iphone*.

UltraLite.NET

Die Projektdateien für Microsoft Visual Studio befinden sich im Verzeichnis *C:\Users\Public\Documents\SQL Anywhere 16\Samples\UltraLite.NET\CustDB*.

Hinweise zu UltraLite-Lösungen für Windows Mobile

In diesem Abschnitt werden die UltraLite-Designoptionen, die für die Entwicklung von Windows Mobile erforderlich sind, und die Implementierungsvorteile für jede Option beschrieben. Die folgenden Auswahlmöglichkeiten werden beschrieben:

- **UltraLite API-Auswahl** Die Vorteile der Verwendung jeder der folgenden APIs für die Windows Mobile-Entwicklung werden beschrieben:
 - UltraLite C/C++
 - UltraLite Embedded SQL
 - UltraLite.NET
- **Datenverwaltung-Komponentenauswahl** Die Vorteile der Verwendung jeder der folgenden Datenverwaltungskomponenten für die Windows Mobile-Entwicklung werden beschrieben:
 - Prozessintegrierte UltraLite-Laufzeitumgebung
 - UltraLite-Datenbank-Engine

Vorteile der UltraLite APIs für Windows Mobile

Die UltraLite C/C++, Embedded SQL- und .NET-APIs bieten einige Datenzugriffsmodelle, darunter eine einfache, tabellenbasierte Zugriffsschnittstelle und dynamisches SQL für komplexere Abfragen. Durch die Kombinationen dieser Vorteile bietet UltraLite Anwendungsentwicklern eine flexible Architektur für die Erstellung von Anwendungen für verschiedene Deployment-Umgebungen.

Vorteile der UltraLite.NET-API

Die UltraLite.NET-API wird in der Regel für die Windows Mobile-Entwicklung empfohlen, weil die SQL Anywhere .NET-API gemeinsame Programmiermodelle bereitstellt, die von UltraLite-Komponenten und SQL Anywhere gemeinsam genutzt werden können, und weil die .NET-Programmierung Vorteile gegenüber C/C++ bietet.

Vorteile UltraLite C/C++- und Embedded SQL-API

Während UltraLite hohe Performance in einer Vielzahl von Umgebungen und Anwendungsfällen bietet, stellen die Embedded SQL- und UltraLite-C++-API die unterste Ebene von APIs dar und bieten üblicherweise die höchste Performance.

Verwenden Sie die UltraLite C/C++-API, wenn Sie Anwendungen mit dem geringsten Speicherbedarf erstellen wollen. Diese Anwendungen liefern gewöhnlich die beste Performance und gleichzeitig kleine Anwendungsdateien.

Siehe auch

- „[NET-Anwendungsprogrammierung](#)“ [[SQL Anywhere Server - Programmierung](#)]
- „[UltraLite C++-Anwendungsentwicklung](#)“ [[UltraLite - C- und C++-Programmierung](#)]
- „[UltraLite C++-Anwendungsentwicklung mit Embedded SQL](#)“ [[UltraLite - C- und C++-Programmierung](#)]
- „[UltraLite.NET-Anwendungsentwicklung](#)“ [[UltraLite - .NET-Programmierung](#)]

UltraLite-Datenverwaltungskomponenten für Windows Mobile

Sie können mit UltraLite eine relationale Datenbanklösung mit geringem Speicherbedarf erstellen. Hierbei ist kein zusätzlicher Overhead durch das Einrichten eines separaten Datenbankservers erforderlich. Stattdessen verwenden die UltraLite-Programmierschnittstellen eine von zwei Ansätzen: Die **Prozessintegrierte UltraLite-Laufzeitbibliothek** und die **UltraLite-Datenbank-Engine**. Beide Ansätze steuern Verbindungs- und Datenzugriffsanforderungen.

Beide Komponenten enthalten einen bidirektionalen Synchronisationsclient, der UltraLite-Datenbanken mit dem MobiLink-Synchronisationsserver verknüpft. Weitere Hinweise zum UltraLite-Synchronisationsclient finden Sie unter „[Funktionen des UltraLite-Synchronisationsclients](#)“ auf [Seite 2](#).

Prozessintegrierte UltraLite-Laufzeitbibliothek

Die prozessintegrierte UltraLite-Laufzeitumgebung wird empfohlen, wenn jeweils nur eine Anwendung auf eine Datenbank zugreifen muss.

Die Laufzeitumgebung und die Anwendung sind Teil desselben Prozesses, wodurch die Datenbank spezifisch für die Anwendung generiert wird. Die Laufzeitumgebung verwaltet UltraLite-Datenbanken und integrierte Synchronisationsvorgänge.

Die Verknüpfung mit der Laufzeitbibliothek erfordert ein anderes Import-Bibliothek/DLL-Paar als das der Engine.

UltraLite unterstützt sowohl statische als auch dynamische Verknüpfung.

- **Statische Verknüpfung** Die statische Verknüpfung erfordert weniger Gerätespeicher und ist effizienter, wenn nur eine einzige UltraLite-Anwendung auf dem Gerät verwendet wird.
- **Dynamische Verknüpfung** Die dynamische Verknüpfung kann sparsamer mit dem Gerätespeicher umgehen, wenn mehrere UltraLite-Anwendungen auf dem Gerät verwendet werden.

UltraLite-Datenbank-Engine (Dienstprogramm *uleng16.exe*)

Die UltraLite-Engine steht nur für Windows-PC- und Windows Mobile-Plattformen zur Verfügung. Die Engine ist ein separates Programm, das den gleichzeitigen Zugriff von mehreren Anwendungen

unterstützt. Jede Anwendung muss zur Kommunikation mit der UltraLite-Engine eine Clientbibliothek verwenden.

Die UltraLite-Engine erfordert mehr Systemressourcen als die UltraLite-Laufzeitumgebung und bietet möglicherweise eine geringere Performance, wenn große Datenmengen zwischen dem Client und der Datenbank bewegt werden.

Die Verbindung mit der Engine erfordert, dass Sie ein anderes Import-Bibliothek/DLL-Paar als das der Laufzeitumgebung angeben.

Die UltraLite-Engine ist unter folgenden Bedingungen erforderlich:

- Mehrere Prozesse greifen möglicherweise gleichzeitig auf dieselbe Datenbankdatei zu (mit "gleichzeitig" ist gemeint, dass mehrere Prozesse zur gleichen Zeit mit derselben Datenbank verbunden sind).
- Zentrale Administration wird für die Verwaltung der UltraLite-Anwendungsdatenbank verwendet.

Siehe auch

- „Kompilierungs- und Deploymentspezifikationen für UltraLite-Anwendungen“ auf Seite 117
- „So erstellen und Sie UltraLite C++-Anwendungen und führen ein Deployment durch.“ [*UltraLite - C- und C++-Programmierung*]
- „Start der UltraLite-Engine“ auf Seite 128
- „UltraLite-Engine-Dienstprogramm (uleng16)“ auf Seite 210
- „UltraLite-Parallelität“ auf Seite 484
- „Funktionsvergleiche von UltraLite, UltraLite Java Edition und SQL Anywhere“ auf Seite 4
- „Einschränkungen für UltraLite- und UltraLite Java Edition-Datenbanken“ auf Seite 9

UltraLite-Datenbank erstellen

Es gibt drei gängige Methoden zur Datenbankerstellung:

- Desktop-Erstellungsmethoden mit speziellen UltraLite-Administrationstools für die Datenbankerstellung
- Erstellungsmethoden auf den Geräten mit UltraLite-API-Schnittstellen.

Die Methoden zur Erstellung auf den Geräten sind in den einzelnen API-spezifischen UltraLite-Programmierhandbüchern detailliert beschrieben.

- Eine entfernte Aufgabe der zentralen Administration, die zum Erstellen einer UltraLite-Datenbank auf einem Gerät konfiguriert wurde.

Sobald die Datenbank erstellt wurde, können Sie sich mit ihr verbinden und Tabellen und andere Datenbankobjekte aufbauen.

Siehe auch

- „UltraLiteDatenbank-Erstellungsparameter“ auf Seite 25
- „Zugriff auf Datenbankoptionen“ auf Seite 40

Erstellen einer UltraLite-Datenbank mit dem Assistenten Datenbank erstellen

Eine Datenbank mit Sybase Central erstellen

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Wählen Sie diese Methode, wenn Sie bei der Navigation durch die verfügbaren Parameter für die Datenbankerstellung unterstützt werden möchten. Dieser Assistent vereinfacht Ihre Auswahl, indem er Ihnen, basierend auf den gewählten Zielplattformen, nur die konfigurierbaren Elemente zur Auswahl stellt. Wenn die Datenbank erstellt wurde, wird die Befehlszeilensyntax angezeigt, die Sie aufzeichnen und mit dem Dienstprogramm ulinit verwenden können.

Aufgabe

1. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Sybase Central**.
2. Klicken Sie auf **Extras » UltraLite 16 » Datenbank erstellen**.
3. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen einer Datenbank**.

Ergebnisse

Die -Datenbank wird erstellt.

Nächste Schritte

Sie können nun eine Verbindung mit der Datenbank herstellen und Tabellen und andere Datenbankobjekte aufbauen.

Siehe auch

- [„UltraLite-Dienstprogramm zum Initialisieren einer Datenbank \(ulinit\)“ auf Seite 214](#)
- [„UltraLiteDatenbank-Erstellungsparameter“ auf Seite 25](#)

UltraLite-Datenbank über die Eingabeaufforderung erstellen

Wählen Sie eines der folgenden Dienstprogramme, um eine Datenbank an einer Eingabeaufforderung zu erstellen:

- Verwenden Sie das Dienstprogramm `ulinit`, um eine neue, leere UltraLite-Datenbank oder eine Datenbank zu erstellen, die auf dem Schema einer SQL Anywhere-Referenzdatenbank basiert. Mit diesem Dienstprogramm können Sie Dienstprogrammoptionen einbeziehen, um die Datenbank zu konfigurieren. Siehe [„Aus einer SQL Anywhere-Datenbank in eine UltraLite-Datenbank konvertieren“ auf Seite 33](#).
- Verwenden Sie das Dienstprogramm `ulload`, wenn Sie eine XML-Datei haben, die als Ausgangspunkt für das Schema bzw. die Daten der neuen UltraLite-Datenbank dienen soll. Siehe [„Erstellen einer UltraLite-Datenbank aus einer XML-Datei“ auf Seite 23](#).
- Zentrale Administration: Wählen Sie diese Methode, wenn für Ihr Deployment der MobiLink-Agent auf allen bereitgestellten Geräten konfiguriert ist oder wenn Sie die ursprüngliche UltraLite-Datenbank nicht mit Ihrer Anwendung bereitstellen können. Sie können eine entfernte Aufgabe zur Erstellung einer neuen UltraLite-Datenbank auf dem Gerät konfigurieren. Diese Datenbank kann dann zentral von einem Administrator verwaltet werden. Siehe [„Zentrale Administration von entfernten Datenbanken“ \[MobiLink - Serveradministration\]](#).

Erstellen einer neuen UltraLite-Datenbank (Befehlszeile)

Führen Sie das Dienstprogramm `ulinit` unter Angabe der neuen UltraLite-Datenbankdatei aus, um die Standardwerte zu übernehmen:

```
ulinit test.udb
```

Weitere Hinweise zu den unterstützten Optionen finden Sie unter [„UltraLite-Dienstprogramm zum Initialisieren einer Datenbank \(ulinit\)“ auf Seite 214](#).

UltraLite-Datenbank unter Verwendung eines MobiLink-Synchronisationsmodells erstellen

Zur Vereinfachung des Deployments umfasst MobiLink einen **Assistenten zum Erstellen eines Synchronisationsmodells**, mit dessen Hilfe eine UltraLite-Datenbank und serverseitige Synchronisationslogik erstellt werden können.

Wählen Sie diese Methode, wenn Sie ein Synchronisationssystem mit entfernten UltraLite-Datenbanken und einer zentralen konsolidierten Datenbank erstellen.

Wenn Sie das Modell erstellt haben, können Sie in Sybase Central im MobiLink-Modellmodus arbeiten, um das Synchronisationsmodell anzupassen, bevor Sie das Deployment durchführen. Wenn das Modell fertig ist, können Sie das Deployment durchführen, um die für die Synchronisationsanwendung erforderlichen Skripten und Tabellen zu erstellen.

Siehe auch

- „MobiLink-Plug-In für Sybase Central“ [[MobiLink - Erste Orientierung](#)]

UltraLite-Datenbank über die zentrale Administration von entfernten Datenbanken erstellen

MobiLink bietet einen Befehl zum Erstellen einer UltraLite-Datenbank.

Siehe auch

- „Der Befehl "Datenbank erstellen"“ [[MobiLink - Serveradministration](#)]
- „Zentrale Administration von entfernten Datenbanken“ [[MobiLink - Serveradministration](#)]

Erstellen einer UltraLite-Datenbank aus einer XML-Datei

XML als Zwischenformat für die Verwaltung Ihrer UltraLite-Datenbank erstellen

Voraussetzungen

UltraLite kann keine beliebige XML-Datei verwenden. Die Verzeichnisse `%SQLANY16%\Bin32` und `%SQLANY16%\Bin64` enthalten eine `usm.xsd`-Datei mit der Schemadefinition. Verwenden Sie diese Datei, um das XML-Format zu überprüfen.

Kontext und Bemerkungen

Sie können XML verwenden, um Folgendes zu tun:

- Laden Sie Daten in eine neue Datenbank mit unterschiedlichen Datenbankeigenschaften und -optionen.
- Führen Sie ein Upgrade des Datenbankschemas durch, das von einer früheren Version von UltraLite erstellt wurde.
- Erstellen Sie eine Textversion Ihrer UltraLite-Datenbank.

Aufgabe

1. Speichern Sie die XML-Datei in einem beliebigen Verzeichnis. Sie können eine der folgenden Vorgehensweisen wählen:
 - Exportieren bzw. entladen Sie eine Datenbank in eine XML-Datei. Wenn Sie eine SQL Anywhere-Datenbank entladen, verwenden Sie eine der unterstützten Exportmethoden.
 - Verwenden Sie die XML-Ausgabe von einer anderen Quelle. Diese Quelle kann eine andere relationale Datenbank oder auch eine Website sein, in der Transaktionen in einer Datei gespeichert werden. Stellen Sie sicher, dass das XML-Format die UltraLite-Anforderungen erfüllt.
2. Führen Sie das Dienstprogramm ulload mit den erforderlichen Optionen aus.

Ergebnisse

Die -Datenbank wird erstellt.

Nächste Schritte

Sie können eine Verbindung mit der Datenbank herstellen und Tabellen und andere Datenbankobjekte aufbauen.

Beispiel

Um eine neue UltraLite-Datenbank in der Datei *sample.udb* anhand der Tabellenformate und Daten in *sample.xml* zu erstellen, führen Sie folgenden Befehl aus:

```
ulload -c DBF=sample.udb sample.xml
```

Siehe auch

- „Als XML exportierte relationale Daten“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Erstellen einer UltraLite-Datenbank mit dem Assistenten Datenbank erstellen“ auf Seite 21
- „UltraLite-Datenbank über die Eingabeaufforderung erstellen“ auf Seite 22
- „Aus einer SQL Anywhere-Datenbank in eine UltraLite-Datenbank konvertieren“ auf Seite 33
- „UltraLite-Upgrades“ [*SQL Anywhere 16 - Änderungen und Upgrades*]
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- „UltraLiteDatenbank-Erstellungsparameter“ auf Seite 25

UltraLite-Datenbank bei der ersten Verbindung erstellen

Sie können die Anwendung so programmieren, dass eine neue UltraLite-Datenbank erstellt wird, wenn zum Zeitpunkt der Verbindung keine Datenbank gefunden wird. Die Anwendung kann dann SQL verwenden, um Tabellen, Indizes, Fremdschlüssel usw. zu erstellen. Um die Datenbank anzufüllen, führen Sie eine Synchronisation mit einer konsolidierten Datenbank durch.

Hinweise

Durch das Hinzufügen des gesamten zusätzlichen Datenbankerstellungs- und SQL-Codes kann die Größe der Anwendung erheblich anwachsen. Dennoch kann diese Vorgehensweise das Deployment vereinfachen, da Sie nur das Deployment der Anwendung auf dem Gerät durchführen müssen. In einigen frühen Entwicklungszyklen können Sie die Datenbank auf Ihrem Gerät für Testzwecke löschen und neu aufbauen.

Siehe auch

- [ULDatabaseManager.CreateDatabase-Methode \[UltraLite C++\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULDatabaseManager.CreateDatabase-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)
- [„UltraLite- und UltraLite Java Edition-Datenbank, Methoden zum Erstellen und Verbinden“ \[UltraLite® – Java-Programmierung\]](#)

UltraLiteDatenbank-Erstellungsparameter

Erstellungsparameter werden verwendet, um die UltraLite-Datenbank bei ihrer Erstellung zu konfigurieren. Sie können diese Einstellungen nur ändern, indem Sie die Datenbank neu erstellen.

Es gibt mehrere Optionen, die Sie bei der Erstellung Ihrer UltraLite-Datenbank verwenden können. Sie wurden für zahlreiche UltraLite-Verwendungszwecke entwickelt. Die meisten zum Zeitpunkt der Erstellung angegebenen Parameter können später nicht mehr geändert werden.

Sie können Erstellungsparameter beim Anlegen einer Datenbank mit den Dienstprogrammen ulinit oder ulload sowie von den unterstützten Client-Schnittstellen aus angeben.

Boolesche Parameter werden mit YES, Y, ON, TRUE, T oder 1 aktiviert und mit NO, N, OFF, FALSE, F oder 0 deaktiviert. Bei den Parametern wird die Groß-/Kleinschreibung nicht berücksichtigt.

UltraLite-Erstellungsparameter werden in einer durch Semikolons getrennten Zeichenfolge angegeben, wenn eine Datenbank mit einer Programmierschnittstelle erstellt wird, und als separater Befehlszeilenparameter, wenn ein Befehlszeilen-Dienstprogramm verwendet wird. Beispiel:

```
ulinit --case --utf8_encoding=1 test.udb
```

Alternativ dazu können Sie auch mehrere Optionen -c angeben.

Sie können weitere Aspekte der Datenbank über Datenbankoptionen und Verbindungsparameter konfigurieren.

Auf Erstellungsparameterwerte zugreifen

Es ist nicht möglich, Erstellungsparameterwerte nach der Erstellung einer Datenbank zu ändern. Sie können allerdings die entsprechenden Datenbankeigenschaften in Sybase Central anzeigen.

Für die UltraLiteJ-API können Sie die `getCreationString`-Methode verwenden, um die Erstellungszeichenfolge anzuzeigen, die mit der `setCreationString`-Methode registriert wurde.

Bei anderen UltraLite-APIs können Sie auf die Datenbankeigenschaften programmatisch von der UltraLite-Anwendung aus zugreifen, indem Sie die der UltraLite-API entsprechende `GetDatabaseProperty`-Funktion aufrufen.

Siehe auch

- „UltraLite-Datenbankoptionen“ auf Seite 195
- „UltraLite-Datenbankverbindungen“ auf Seite 35
- „UltraLite-Verbindungsparameter“ auf Seite 168
- „Lesen der Datenbankeigenschaften“ auf Seite 39
- `ULConnection.GetDatabaseProperty`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- `ULDatabaseSchema.GetDatabaseProperty`-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- `ConfigPersistent.getCreationString`-Methode [Android] [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- `ConfigPersistent.setCreationString`-Methode [Android] [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

UltraLite-Zeichensätze

Die Ergebnisse von Zeichenfolgenvergleichen und die Sortierreihenfolge von Zeichenfolgen hängen zum Teil vom Zeichensatz, von der Kollation und von den Kodierungseigenschaften der Datenbank ab.

Die Auswahl des richtigen Zeichensatzes, der Kollation und der Kodierungseigenschaften für Ihre Datenbank wird vorrangig von folgenden Komponenten bestimmt:

- Gewünschte Sortierreihenfolge Wählen Sie die Kollation, die am besten für die Zeichen geeignet ist, die Sie in der Datenbank speichern möchten.
- Plattform des Geräts. Die Anforderungen der verschiedenen unterstützten Geräte können variieren. Einige erfordern, dass Sie Ihre Zeichen mit UTF-8 kodieren. Wenn Sie mehrere Geräte unterstützen müssen, sollten Sie überprüfen, ob eine gemeinsame Datenbank verwendet werden kann.
- Wenn Sie Daten synchronisieren, welche Sprachen und Zeichensätze von der konsolidierten Datenbank unterstützt werden. Sie müssen sicherstellen, dass die von der UltraLite-Datenbank und der konsolidierten Datenbank verwendeten Zeichensätze kompatibel sind. Andernfalls können Daten verloren gehen oder auf unerwartete Weise verändert werden, wenn Zeichen aus dem Zeichensatz einer Datenbank im Zeichensatz der anderen Datenbank nicht vorhanden sind. Wenn die UltraLite-Anwendung in einer mehrsprachigen Umgebung eingesetzt wird, sollten Sie die UltraLite-Datenbank auch mit UTF-8 kodieren.

Beim Synchronisieren konvertiert der MobiLink-Server die Zeichen folgendermaßen:

1. Die Zeichen der UltraLite-Datenbank werden in Unicode konvertiert.
2. Die Unicode-Zeichen werden in den Zeichensatz der konsolidierten Datenbank konvertiert.

Zeichensätze und Kollationen für UltraLite Java Edition

UltraLite Java Edition-Datenbanken sind UTF-8-kodiert (Unicode). Die Kollation ist die Standardsortierreihenfolge für Java. Sie entspricht der UTF8BIN-Kollation, die von SQL Anywhere unterstützt wird. Während der Synchronisation mit einem MobiLink-Server wird MobiLink von UltraLite informiert, dass es UTF8-Zeichensatz und -Kollation verwendet.

Siehe auch

- „Plattformanforderungen für die Zeichensatzkodierung in UltraLite“ auf Seite 27
- „UltraLite-Erstellungsparameter collation“ auf Seite 145
- „UltraLite-Erstellungsparameter utf8_encoding“ auf Seite 167
- „Zeichensätze“ [*SQL Anywhere Server - Datenbankadministration*]
- „UltraLite-Verbindungsparameter“ auf Seite 168
- „Hinweise zum Zeichensatz“ [*MobiLink - Serveradministration*]
- „UltraLite-Erstellungsparameter case“ auf Seite 143
- „Datenbanksicherheit“ auf Seite 29

Plattformanforderungen für die Zeichensatzkodierung in UltraLite

Jede Plattform hat spezifische Zeichensatz- und Kodierungsanforderungen.

Windows-PCs und Windows Mobile

Wenn Sie eine UTF-8-kodierte Datenbank auf Windows verwenden, sollten Sie der Datenbank breite Zeichen (wide-characters) übergeben. Wenn Sie auf diesen Plattformen die UTF-8-Kodierung verwenden, erwartet UltraLite, dass Zeichenfolgenparameter, die nicht breite Zeichen verwenden (non-wide), mit UTF-8 kodiert sind. Dies ist kein natürlicher Zeichensatz für Windows. Eine Ausnahme gilt für Verbindungszeichenfolgen, bei denen von Zeichenfolgenparametern erwartet wird, dass sie in der aktiven Codepage enthalten sind. Mit der Verwendung von breiten Zeichen können Sie diese Komplikation jedoch vermeiden.

Siehe auch

- „UltraLite-Erstellungsparameter utf8_encoding“ auf Seite 167
- „Zeichensätze“ [*SQL Anywhere Server - Datenbankadministration*]
- „UltraLite-Verbindungsparameter“ auf Seite 168
- „Hinweise zum Zeichensatz“ [*MobiLink - Serveradministration*]
- „Datenbanksicherheit“ auf Seite 29

Unterstützte UltraLite-Kollationen

In der folgenden Tabelle sind die in UltraLite unterstützten CHAR-Kollationen aufgelistet. Sie können die Liste auch mit folgendem Befehl generieren:

```
ulinit -Z
```

UltraLite Java Edition-Datenbanken unterstützen nur die UTF8BIN-Kollation.

Kollationslabel	Beschreibung
1250LATIN2	Codepage 1250, Windows Latin 2, Zentral-/Osteuropäisch
1250POL	Codepage 1250, Windows Latin 2, Polnisch
1251CYR	Codepage 1251, Windows Kyrillisch
1252LATIN1	Codepage 1252, Windows Latin 1, Western
1252NOR	Codepage 1252, Windows Latin 1, Norwegisch
1252SPA	Codepage 1252, Windows Latin 1, Spanisch
1252SWEFIN	Codepage 1252, Windows Latin 1, Schwedisch/Finnisch
1253ELL	Codepage 1253, Windows Griechisch, ISO8859-7 mit Erweiterungen
1254TRK	Codepage 1254, Windows Türkisch, ISO8859-9 mit Erweiterungen
1254TRKALT	Codepage 1254, Windows Türkisch, ISO8859-9 mit Erweiterungen, I mit I-Punkt gleich I ohne I-Punkt
1255HEB	Codepage 1255, Windows Hebräisch, ISO8859-8 mit Erweiterungen
1256ARA	Codepage 1255, Windows Arabisch, ISO8859-6 mit Erweiterungen
1257LIT	Codepage 1257, Windows Baltische Staaten, Litauisch
874THAIBIN	Codepage 874, Windows Thailändisch, ISO8859-11, binäre Sortierung
932JPN	Codepage 932, Japanisch, Shift-JIS mit Microsoft-Erweiterungen
936ZHO	Codepage 936, Vereinfachtes Chinesisch, PRC GBK
949KOR	Codepage 949, Korean KS C 5601-1987-Codierung, Wansung
950ZHO_HK	Codepage 950, Traditionelles Chinesisch, Big 5-Kodierung mit HKSCS
950ZHO_TW	Codepage 950, Traditionelles Chinesisch, Big 5-Kodierung

Kollationslabel	Beschreibung
EUC_CHINA	Vereinfachtes Chinesisch, GB 2312-80-Zeichensatz
EUC_JAPAN	Japanische EUC JIS X 0208-1990 und JIS X 0212-1990-Kodierung
EUC_KOREA	Codepage 1361, Koreanisch KS C 5601-1992 8-Bit-Zeichensatz, Johab
EUC_TAIWAN	Codepage 964, EUC-TW-Kodierung
ISO1LATIN1	ISO8859-1, ISO Latin 1, Western, Latin 1-Sortierung
ISO9LATIN1	ISO8859-15, ISO Latin 9, Western, Latin 1-Sortierung
ISO_1	ISO8859-1, ISO Latin 1, Western
ISO_BINENG	Binäre Sortierung, Englisch ISO/ASCII 7-Bit Buchstaben-Fall-Zuordnungen
UTF8BIN	UTF-8, 8-Bit-Mehrbyte-Kodierung für Unicode, binäre Reihenfolge

Datenbanksicherheit

Sie haben die Möglichkeit, Ihre Datenbanken zu verschlüsseln oder zu verschleiern. Die Verschlüsselung sorgt für eine sichere Repräsentation der Daten in der Datenbank, während die Verschleierung nur verhindert, dass eine zufällig Beobachtung des Datenbankinhalts möglich ist.

Standardmäßig sind Datenbanken nicht verschlüsselt oder verschleiert. Text- und Binärspalten können gelesen werden, wenn Sie Tools zum Anzeigen wie z.B. einen Hex-Editor verwenden. Berücksichtigen Sie die folgenden Optionen, wenn Sie nicht möchten, dass Ihre Daten als normalen Text gespeichert werden:

- **Verschleierung** Diese Option bietet Schutz gegen unrechtmäßigen Zugriff auf Daten in der Datenbank, aber nicht so viel Sicherheit wie eine starke Verschlüsselung. Die Verschleierung beeinträchtigt die Performance nur geringfügig. Es ist keine spezielle Konfiguration erforderlich, um die einfache Verschleierung auf Ihrem Gerät zu verwenden.
- **AES 256-Bit-Verschlüsselung** Diese Option verschlüsselt Datenbanken mit einem AES 256-Bit-Algorithmus. Die starke Verschlüsselung bietet Sicherheit vor fachmännischen und zielgerichteten Versuchen, sich Zugriff auf die Daten zu verschaffen, hat jedoch erhebliche Auswirkungen auf die Performance. Es ist keine spezielle Konfiguration erforderlich, um die AES-Verschlüsselung auf Ihrem Gerät zu verwenden.
- **AES 256-Bit FIPS 140-2-zertifizierte Verschlüsselung (nur Windows und Windows Mobile)** Verschlüsselungsbibliotheken, die nach dem US-amerikanischen und kanadischen Standard FIPS 140-2 zertifiziert sind (unter Verwendung eines Certicom-zertifizierten Verschlüsselungsmoduls) werden bereitgestellt. Die AES FIPS-zertifizierte Verschlüsselung erfordert eine entsprechende Konfiguration Ihres Geräts.

Hinweis

Erforderliche getrennt lizenzierbare Komponenten.

FIPS-zertifizierte Verschlüsselung erfordert eine separate Lizenz. Alle Technologien für starke Verschlüsselungen unterliegen Exportbestimmungen.

Siehe „[Getrennt lizenzierbare Komponenten](#)“ [*SQL Anywhere 16 - Einführung*].

Berücksichtigen Sie die Auswirkungen der Datenbank-Cachegröße, wenn Sie entscheiden, ob Sie Datenbanken verschlüsseln oder verschleiern. Es gibt eine Overheadzunahme von 5 - 10 %. Dies kann zu geringerer Leistung führen. Die genauen Auswirkungen auf die Performance hängen von der Größe Ihres Caches ab. Wenn Ihr Cache zu klein ist, kann eine Verschlüsselung einen erheblichen Overhead bewirken. Wenn Ihr Cache jedoch groß genug ist, erkennen Sie möglicherweise nicht den geringsten Unterschied. Weitere Hinweise zur Cachegröße finden Sie unter „[Cachegrößenanpassung für eine UltraLite-Datenbank](#)“ auf Seite 469.

Datenbankverschleierung

Um Daten zu verschleiern, geben Sie `obfuscate=1` als Datenbankerstellungsparmeter an, wenn Sie die Datenbank erstellen. Endbenutzer müssen keinen entsprechenden Verbindungsparameter angeben.

Um Daten mit der UltraLiteJ API zu verschleiern, verwenden Sie die `ConfigPersistent.enableObfuscation`-Methode während der Erstellung der Datenbank.

Datenbankverschlüsselung

Chiffrierschlüssel sollten eine Kombination von Buchstaben, Ziffern und Sonderzeichen enthalten, um wirksam zu sein. Lange Chiffrierschlüssel reduzieren die Gefahr, dass andere den Schlüssel erraten.

Hinweis

Nachdem eine Datenbank verschlüsselt wurde, kann der Chiffrierschlüssel nicht mehr wiederhergestellt werden.

Mit den Assistenten in Sybase Central können Sie die UltraLite-Datenbankverschlüsselungsoptionen während der Erstellung festlegen, indem Sie auf die Option **Datenbank verschlüsseln** und dann auf **Starke Verschlüsselung und AES verwenden** klicken.

Mit dem Dienstprogramm `ulinit` können Sie den Chiffrierschlüssel mit dem `DBKEY`-Parameter festlegen. Verwenden Sie FIPS-Erstellungsparmeter, um festzulegen, ob FIPS-zertifizierte Verschlüsselung verwendet werden soll.

UltraLite API-Verschlüsselungsoptionen sind verfügbar, wenn Sie eine Datenbank erstellen. Geben Sie für UltraLite Java Edition-Datenbanken den Datenbankschlüssel an und verwenden Sie die `enableAesDBEncryption`- und die `setEncryptionKey`-Methode anstelle des `DBKEY`-Parameters.

Vorsicht

Sie können den Chiffrierschlüssel nach dem Erstellen der Datenbank ändern, jedoch nur mit äußerster Vorsicht.

Dieser Vorgang ist kostenträchtig und kann nicht wiederhergestellt werden. Sie können die gesamte Datenbank verlieren, wenn der Vorgang unterbrochen wird.

Bei stark verschlüsselten Datenbanken achten Sie darauf, eine Kopie des Schlüssels an einem sicheren Ort zu verwahren. Wenn Sie den Chiffrierschlüssel verlieren, gibt es keine Möglichkeit, auf die Daten zuzugreifen, auch nicht mit Unterstützung durch den technischen Support. Die Datenbank muss verworfen und eine neue Datenbank muss erstellt werden.

Weitere Hinweise finden Sie unter:

- [ULConnection.ChangeEncryptionKey-Methode \[UltraLite C++\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULConnection.ChangeEncryptionKey-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)
- [Connection.changeEncryptionKey-Methode \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)

Der DBKEY-Parameter muss übergeben werden, wenn Sie sich mit der Datenbank verbinden. Andernfalls schlägt die Verbindung fehl. Chiffrierschlüssel müssen als vertrauliche Daten behandelt werden.

Siehe auch

- „UltraLite-Erstellungsparameter obfuscate“ auf Seite 154
- „UltraLite-Verbindungsparameter DBKEY“ auf Seite 179
- „UltraLite-Erstellungsparameter fips“ auf Seite 150
- „UltraLite-Datenbank erstellen“ auf Seite 21
- „UltraLiteDatenbank-Erstellungsparameter“ auf Seite 25
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- [ConfigPersistent.enableObfuscation-Methode \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [ULDatabaseManager.CreateDatabase-Methode \[UltraLite C++\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULDatabaseManager.CreateDatabase-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)
- [ConfigPersistent.setCreationString-Methode \[Android\] \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [ConfigPersistent.enableAesDBEncryption-Methode \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [ConfigPersistent.getEncryptionKey-Methode \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [ConfigPersistent.setEncryptionKey-Methode \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- „Getrennt lizenzierbare Komponenten“ [*SQL Anywhere 16 - Einführung*]

Aus einer SQL Anywhere-Datenbank in eine UltraLite-Datenbank konvertieren

Erstellen Sie eine UltraLite-Datenbank aus einer SQL Anywhere-Referenzdatenbank, indem Sie das Dienstprogramm ulinit mit der Option -a ausführen. Die neue Datenbank wird mit den Einstellungen der Referenzdatenbank erstellt, soweit möglich.

Die SQL Anywhere-Referenzdatenbank dient als Datenbankvorlage und verwendet die folgenden Einstellungen, um ein UltraLite-Datenbankschema zu erstellen:

- Datenbankkonfiguration wie etwa die Kollationssequenz
- Tabellendefinitionen
- Synchronisationspublikationen

Sie können Daten einbeziehen und Spalten, Tabellen und Indizes als Teil einer Publikation in der Referenzdatenbank auswählen.

Hinweis

Um eine UltraLite-Datenbank von einem anderen RDBMS-System als SQL Anywhere zu initialisieren, verwenden Sie den **Assistenten zum Erstellen eines Synchronisationsmodells** in Sybase Central und stellen eine Verbindung zu einer konsolidierten Datenbank her, wenn Sie dazu aufgefordert werden, Schemadaten zu beziehen.

Hinweise zur Konvertierung

Prüfen Sie vor dem Ausführen des Dienstprogramms ulinit, ob die folgenden Referenzdatenbankaufgaben erforderlich sind:

- **Tabellen, Schlüssel, Indizes und Synchronisationspublikationen nach Bedarf hinzufügen** Tabellen hinzufügen und Primärschlüssel wie erforderlich einrichten Sie können innerhalb der UltraLite-Anwendung auch benötigte Fremdschlüsselbeziehungen zuweisen.

Indizes können die Performance erheblich steigern, vor allem auf langsamen Geräten. Primärschlüsselspalten werden automatisch indiziert, andere Spaltentypen dagegen nicht.

Tipp

Wenn Ihre UltraLite-Anwendung häufig Informationen in einer bestimmten Reihenfolge abruft, sollten Sie Ihrer Referenzdatenbank einen speziellen Index für diesen Zweck hinzufügen.

Verwenden Sie Synchronisationspublikationen, um verschiedene Tabellen zu unterschiedlichen Zeiten zu synchronisieren. Sie können mehrere Synchronisationspublikationen verwenden, um Teilmengen von Tabellen zu definieren und Synchronisationsprioritäten zu setzen.

- **Datenbankoptionen oder Tabellenschemata aktualisieren, die möglicherweise unerwünschte Auswirkungen haben** Wenn beispielsweise eine Spalte in der SQL Anywhere-Datenbank eine Klausel enthält, die UltraLite nicht unterstützt, wird der Standardwert ignoriert und der UltraLite-Standardwert wird für die neue Datenbank verwendet.

- **Kollationssequenz ändern, wenn sie von UltraLite nicht unterstützt wird** UltraLite verwendet den in der Referenzdatenbank definierten Namen der Kollationssequenz. Sie haben dennoch die Möglichkeit, die Datenbank mit UTF-8 zu kodieren, indem Sie die `utf8_encoding`-Eigenschaft setzen.

Um eine Liste von Kollationen und entsprechenden Codepages anzuzeigen, führen Sie `ulinit` mit der Option `-Z` von einer Eingabeaufforderung aus. Wenn die Referenzdatenbank eine nicht unterstützte Kollationssequenz verwendet, z.B. UCA für CHAR-Kollationssequenzen, ändern Sie die Kollationssequenz in eine unterstützte Sequenz, indem Sie die folgenden Schritte ausführen:

1. Verwenden Sie das Dienstprogramm zum Entladen, um die SQL Anywhere-Referenzdatenbank zu entladen.
2. Erstellen Sie eine neue SQL Anywhere-Datenbank mit einer anderen Kollation und führen Sie das `reload.sql`-Skript über Interactive SQL aus.

Siehe auch

- „Dienstprogramm zum Entladen (dbunload)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Tabellen und -Spalten“ auf Seite 50
- „Wann sollte ein Index erstellt werden“ auf Seite 58
- „Publizieren von Daten in UltraLite“ auf Seite 78
- „Index-Scan erstellen und verwalten“ auf Seite 471
- „UltraLite-Datenbank erstellen“ auf Seite 21

Beispiel

Der folgende Befehl erstellt eine neue UltraLite-Datenbank namens `customer.udb` aus einer vorhandenen SQL Anywhere-Referenzdatenbank, die in der Datenquelle `MySADb` definiert ist. Die Tabellen in der Referenzdatenbank werden in `TestPublication` definiert. Die erstellte UltraLite-Datenbank enthält dieselben Datenbankoptionen wie `TestPublication` und ist mit dem **mykey**-Chiffrierschlüssel verschlüsselt.

```
ulinit -a "DSN=MySADb;UID=JimmyB;PWD=secret" -n TestPublication -k mykey  
customer.udb
```

Der folgende Befehl erstellt eine neue UltraLite-Datenbank namens `customer.udb` aus einer vorhandenen SQL Anywhere-Datenbank namens `MySource.db`. Die Tabellen und Indizes in der erstellten Datenbank stimmen mit denjenigen in der Schemapublikation `Pub1` überein. Die Synchronisationspublikation `Pub2` wird in der UltraLite-Datenbank erstellt.

```
ulinit -a DBF=MySource.db;UID=JimmyB;PWD=secret customer.udb -n Pub1 -s Pub2
```

UltraLite-Datenbankverbindungen

Anwendungen, die eine Datenbank verwendet, muss eine Verbindung mit dieser Datenbank herstellen, bevor Transaktionen ausgeführt werden können. Eine Anwendung kann ein UltraLite-Dienstprogramm, ein Verbindungsfenster bzw. Ihre eigene benutzerdefinierte Anwendung sein.

Wenn Sie eine Verbindung mit einer UltraLite-Datenbank herstellen, bilden Sie einen Kanal, über den alle Aktivitäten der Anwendung ausgeführt werden. Jeder Verbindungsversuch erstellt eine datenbankspezifische SQL-Transaktion.

Siehe auch

- „UltraLite-Verbindungsparameter“ auf Seite 168
- UltraLite C/C++: „UltraLite-Datenbankverbindungen“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [[UltraLite - .NET-Programmierung](#)]
- UltraLiteJ: „UltraLite- und UltraLite Java Edition-Datenbank, Methoden zum Erstellen und Verbinden“ [[UltraLite® – Java-Programmierung](#)]

UltraLite-Verbindungszeichenfolgen und Parameter

Eine **Verbindungszeichenfolge** ist eine Gruppe von Verbindungsparametern, die von einer Anwendung an eine Datenbankverbindung übergeben werden, sodass eine Verbindung definiert und hergestellt werden kann. Einige Parameter sind immer erforderlich, um eine Verbindung zu öffnen, während andere verwendet werden, um Datenbankfunktionen für eine einzelne Verbindung einzustellen.

Verbindungszeichenfolgen werden definiert als *Schlüsselwort=Wert*-Paare in einer mit Semikola getrennten Liste. Das folgende Beispiel zeigt ein Fragment einer Verbindungszeichenfolge, das einen Datenbanknamen, eine Benutzer-ID und ein Kennwort angibt:

```
DBF=myULdb.udb;UID=JDoe;PWD=token
```

Die Methoden, mit denen Sie diese Parameter an eine Datenbank übergeben, können davon abhängig sein, ob Sie sich aus einer UltraLite-Dienstprogramm oder einer UltraLite-Anwendung verbinden.

Befehlszeilen-Dienstprogramme von UltraLite verwenden für gewöhnlich eine Verbindungszeichenfolge, wenn eine Verbindung zu einer Datenbank erforderlich ist.

UltraLite-Anwendungen können über eine UltraLite-API entwickelt werden, die Verbindungsparameterwerte aus einer gespeicherten Datei oder im Anwendungscode lesen. Sie können feste Verbindungszeichenfolgen übergeben, wenn keine Benutzerauthentifizierung erforderlich ist, oder Benutzer zur Übergabe der Parameterwerte bei der Verbindungsaufnahme definieren.

Wenn eine Verbindungszeichenfolge zusammengestellt wurde, wird sie an die UltraLite-Laufzeitbibliothek für die Verarbeitung übergeben. Die Verbindung mit der Datenbank wird erteilt, wenn der Verbindungsversuch validiert wird. Verbindungsfehler können auftreten, wenn die Datenbankdatei nicht existiert oder die Authentifizierung nicht erfolgreich war.

UltraLite generiert einen Fehler, wenn das Programm auf einen nicht erkannten Verbindungsparameter stößt.

Vorrang von Verbindungsparametern für UltraLite-Administrationstools

UltraLite-Administrationstools befolgen eine bestimmte Reihenfolge für den Vorrang von Verbindungsparametern:

- Betriebssystemspezifische Optionen haben Vorrang vor unspezifischen Optionen. Beispiel: CE_DBF hat Vorrang vor DBF auf Windows CE-Geräten.
- Wenn angegeben, haben die Parameter CE_FILE, desktop, device und NT_FILE immer Vorrang vor DBF.
- Wenn Sie in einer Verbindungszeichenfolge doppelte Parameter angeben, wird der letzte Parameter verwendet. Alle anderen werden ignoriert.
- Parameter in der Verbindungszeichenfolge haben Vorrang vor den in der Umgebungsvariablen ULSQLCONNECT oder in einem Verbindungsobjekt angegebenen Parametern.
- Wenn *weder* für die Benutzer-ID noch für das Kennwort in der Verbindungszeichenfolge oder in ULSQLCONNECT ein Wert angegeben wird, werden die Standardwerte **UID=DBA** und **PWD=sql** verwendet.

Einschränkungen

Alle führenden bzw. nachgestellten Leerzeichen in Werten von Verbindungsparametern werden ignoriert. Werte von Verbindungsparametern dürfen keine führenden Apostrophe ('), Anführungszeichen (") oder Semikolons (;) enthalten.

Siehe auch

- „UltraLite-Verbindungsparameter“ auf Seite 168
- UltraLite C/C++: „UltraLite-Datenbankverbindungen“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [[UltraLite - .NET-Programmierung](#)]
- UltraLiteJ: „UltraLite- und UltraLite Java Edition-Datenbank, Methoden zum Erstellen und Verbinden“ [[UltraLite® – Java-Programmierung](#)]

Beispiel

Geben Sie eine Verbindungszeichenfolge in einer einzelnen Zeile ein, wobei die Parameternamen und Werte durch Semikolons getrennt sind:

```
parameter1=value1;parameter2=value2
```

Die UltraLite-Laufzeitbibliothek stellt sicher, dass die Parameter in einer Verbindungszeichenfolge zusammengestellt werden, die dann zum Herstellen einer Verbindung verwendet wird. Wenn Sie z.B. das Dienstprogramm ulload verwenden, wird die folgende Verbindungszeichenfolge verwendet, um die neuen XML-Daten in eine vorhandene Datenbank zu laden. Es ist erst dann möglich, eine Verbindung mit der Datenbankdatei herzustellen, wenn Sie diese Zeichenfolge bereitstellen:

```
ulload -c "DBF=sample.udb;UID=DBA;PWD=sql" sample.xml
```

UltraLite-Verbindungsparameter und die Umgebungsvariable ULSQLCONNECT

Verwenden Sie die ULSQLCONNECT-Verbindungsvariable, um zu vermeiden, dass Sie während der Entwicklung immer wieder dieselben Verbindungsparameter an Interactive SQL übergeben müssen. Interactive SQL ist das einzige Tool, das die Umgebungsvariable ULSQLCONNECT unterstützt. Sie können die Variable nicht für benutzerdefinierte Anwendungen verwenden.

Die Umgebungsvariable ULSQLCONNECT ist optional und wird standardmäßig nicht gesetzt. Sie können die Umgebungsvariable ULSQLCONNECT so einrichten, dass sie eine Liste von Parametern enthält, die in Form von *Schlüsselwort=Wert*-Paaren in einer durch Semikola getrennten Liste definiert werden.

Die angegebenen Werte werden zu Standardwerten für die UltraLite-Desktop-Administrationstools. Wenn Interactive SQL zusätzliche Parameter erfordert oder Sie die mit dieser Umgebungsvariablen eingestellten Standardwerte aufheben möchten, müssen Sie diese Werte festlegen. Vom Benutzer übergebene Werte haben immer Vorrang vor den Werten in diesen Umgebungsvariablen.

Vorsicht

Verwenden Sie nicht das Rautenzeichen (#) als Alternative zum Gleichheitszeichen, da es von dbisql ignoriert wird. Alle von UltraLite unterstützten Plattformen gestatten die Verwendung von = innerhalb eines Werts für eine Umgebungsvariable.

Siehe auch

- [Vorrang von Verbindungsparametern für UltraLite-Administrationstools auf Seite 36](#)

Beispiel

In diesem Beispiel verwenden Sie die Umgebungsvariable ULSQLCONNECT zum Verbinden mit einer Datei namens *c:\database\myfile.udb* und zum Authentifizieren des Benutzers **demo** mit dem Kennwort **test**.

Setzen Sie die folgende Variable in Ihre Umgebungsvariable ULSQLCONNECT:

```
set ULSQLCONNECT="DBF=c:\database\myfile.udb;UID=demo;PWD=test"
```

Wenn Sie diese Umgebungsvariable definieren, müssen Sie den dbisql-Verbindungsparameter -c nicht mehr für diese Standardwerte verwenden, außer Sie wollen diese Werte aufheben.

UltraLite-Dateipfadformate in Verbindungsparametern

Die physische Speicherung Ihres Geräts legt fest, ob die Datenbank als Datei gespeichert wird und welche Namenskonventionen Sie bei der Identifizierung Ihrer Datenbank beachten müssen.

Hinweis

Benutzen Sie absolute Dateipfade bei der Verwendung der UltraLite-Engine, um mehreren Prozessen den Zugang zur Datenbank zu ermöglichen, da die Engine eventuell an verschiedenen Orten gestartet wird.

Der DBF-Parameter ist am besten geeignet, wenn eine einzelne Entwicklungsplattform verwendet werden soll oder wenn UltraLite-Desktop-Administrationstools verwendet werden. Beispiel:

```
ulload -c DBF=sample.udb sample.xml
```

Tipp

Sie können mit den UltraLite-Administrationstools Datenbanken verwalten, für die bereits auf einem angeschlossenen Gerät ein Deployment durchgeführt wurde. Siehe [Windows Mobile auf Seite 38](#).

Wenn Sie dagegen eine Anwendung für mehrere Plattformen erstellen, sollten Sie die plattformspezifischen Datei-Verbindungsparameter (CE_DBF oder NT_DBF) verwenden, um eine universelle Verbindungszeichenfolge zu erstellen. Beispiel:

```
Connection = DatabaseManager::OpenConnection( "UID=JDoe;PWD=ULdb;CE_DBF=\database\MyCEDB.udb;NT_FILE=MyDB.udb" )
```

Desktop

Desktop-PCs gestatten absolute und relative Pfade.

Windows Mobile

Windows Mobile-Geräte erfordern absolute Pfadnamen.

Sie können eine Windows Mobile-Datenbank auf dem PC oder dem angeschlossenen Gerät verwalten. Um eine Datenbank auf einem Windows Mobile-Gerät zu verwalten, muss dem absoluten Pfad das Präfix **wce:** vorangestellt werden. Das folgende Beispiel zeigt die Verwendung des Dienstprogramms ulunload:

```
ulunload -c DBF=wce:\UltraLite\myULdb.udb c:\out\ce.xml
```

In diesem Beispiel entlädt UltraLite die Datenbank vom Windows Mobile-Gerät in die Datei *ce.xml* im Windows-PC-Ordner *c:\out*.

Wenn Sie die Dienstprogramme ulunloadold oder ulunload verwenden, um eine Datenbank auf dem Windows Mobile-Gerät direkt zu verwalten, kann UltraLite die Datenbank nicht sichern, bevor die Entladeaktion durchgeführt ist. Sie müssen die Aktion manuell ausführen, bevor Sie diese Dienstprogramme aufrufen.

Siehe auch

- „UltraLite-Verbindungsparameter DBF“ auf Seite 177
- „UltraLite-Verbindungsparameter NT_FILE“ auf Seite 185
- „UltraLite-Verbindungsparameter CE_FILE“ auf Seite 174

Aufgaben und Merkmale einer UltraLite-Datenbank

Lesen der Datenbankeigenschaften

Sie können die Einstellungen jeder Datenbankeigenschaft ändern, die nicht mit einem Datenbank-Erstellungsparameter übereinstimmt.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Jede UltraLite-API enthält eine GetDatabaseProperty-Methode, die Sie in Ihren Anwendungen für den Zugriff auf Datenbankeigenschaften benutzen können.

Sie können auch über den Aufruf der DB_PROPERTY SQL-Funktion auf Datenbankeigenschaften zugreifen.

Aufgabe

1. Stellen Sie mithilfe von Sybase Central eine Verbindung mit der Datenbank her.
2. Rechtsklicken Sie auf die Datenbank und klicken Sie auf **Eigenschaften**.

Ergebnisse

Im Fenster **Datenbankeigenschaften** werden auf den Registerkarten **Allgemein** und **Synchronisationsinformationen** die Datenbankeigenschaften aufgelistet. Auf der Registerkarte **Synchronisationsinformationen** sind die Datenbankeigenschaften alphabetisch anhand des Eigenschaftsnamens aufgelistet. Um die Datenbankeigenschaften nach ihrem Wert zu sortieren, klicken Sie auf die Spalte **Wert**.

Siehe auch

- „UltraLite-Datenbankeigenschaften“ auf Seite 191
- ULConnection.GetDatabaseProperty-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULConnection.GetDatabasePropertyInt-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseSchema.GetDatabaseProperty-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Connection.getDatabaseProperty-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- „DB_PROPERTY-Funktion [System]“ auf Seite 351

Zugriff auf Datenbankoptionen

Datenbankoptionen anzeigen und ändern, um das Datenbankverhalten zu konfigurieren.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Sie können die Einstellungen von permanenten Datenbankoptionen in Sybase Central anzeigen und ändern. Temporäre UltraLite-Datenbankoptionen können *nicht* in Sybase Central angezeigt oder geändert werden.

Datenbankoptionen können jederzeit eingestellt oder geändert werden. Temporäre Datenbankoptionen sind nur für den Zeitraum gültig, in dem die Datenbank läuft.

Optionswerte werden mit der SQL-Anweisung SET OPTION festgelegt.

Aufgabe

1. Stellen Sie mithilfe von Sybase Central eine Verbindung mit der Datenbank her.
2. Rechtsklicken Sie auf die Datenbank und klicken Sie auf **Optionen**.
3. Um eine Option festzulegen oder zurückzusetzen, geben Sie im Feld **Wert** einen neuen Wert ein.
4. Klicken Sie auf **Jetzt festlegen** oder **Jetzt zurücksetzen**, um die Änderung festzuschreiben.

Ergebnisse

Die Einstellung der Datenbankoption wurde geändert und gespeichert.

Siehe auch

- „UltraLite-Datenbankoptionen“ auf Seite 195
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- `ULConnection.SetDatabaseOption`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- `ULConnection.SetDatabaseOptionInt`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- `ULDatabaseSchema.SetDatabaseOption`-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- `Connection.setOption`-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- „Lesen der Datenbankeigenschaften“ auf Seite 39

UltraLite-Ereignisbenachrichtigungen

UltraLite unterstützt nun Ereignisse und Benachrichtigungen. Eine Benachrichtigung ist eine Meldung, die gesendet wird, wenn ein Ereignis eintritt, und die auch zusätzliche Parameterinformationen enthält. In UltraLite gibt es Systemereignisse und benutzerdefinierte Ereignisse.

Mit Ereignisbenachrichtigungen können Sie Verbindungen oder Anwendungen, die mit derselben Datenbank verbunden sind, koordinieren und zwischen ihnen Signale austauschen. Benachrichtigungen werden in Warteschlangen verwaltet: Entweder in der Standard-Warteschlange der Verbindung oder optional in Warteschlangen, die explizit erstellt und benannt werden. Wenn ein Ereignis eintritt, werden Benachrichtigungen an registrierte Warteschlangen (oder Verbindungen) gesendet.

Jede Verbindung verwaltet ihre eigene Benachrichtigungs-Warteschlange. Benannte Warteschlangen können für jede Verbindung erstellt werden.

Die Verwendung von vordefinierten Systemereignissen stellt auch "Trigger" für Datenänderungen bereit, wenn z.B. eine Tabelle geändert wird, oder es wird ein Signal gesendet, wenn eine Synchronisation stattgefunden hat. Vordefinierte Ereignisse sind die folgenden:

- Commit
- SyncComplete
- TableModified

Benutzerereignisse können auch von einer Anwendung definiert und ausgelöst werden.

APIs für Ereignisse und Benachrichtigungen werden in jeder unterstützten Sprache bereitgestellt. Außerdem wird eine SQL-Funktion für den Zugriff auf die API-Funktionalität bereitgestellt.

Ereignisse

Ereignis	Auftreten
Commit	Signalisiert den Abschluss einer Festschreibung
SyncComplete	Signalisiert den Abschluss einer Synchronisation
TableModified	<p>Wird ausgelöst, wenn Zeilen in einer Tabelle eingefügt, aktualisiert oder gelöscht werden. Ein Ereignis pro Anforderung wird signalisiert, unabhängig davon, wie viele Zeilen von der Anforderung bei der Registrierung für das Ereignis betroffen waren.</p> <p>Der Parameter <i>object_name</i> gibt die zu überwachende Tabelle an. Der Wert "*" steht für alle Tabellen in der Datenbank.</p> <p>Der Benachrichtigungsparameter <i>table_name</i> ist der Name der geänderten Tabelle.</p>

```
note_info.event_name = "SyncComplete";
note_info.event_name_len = 12;
note_info.parms_type = ul_ev_note_info::P_NONE;

note_info.event_name = "TableModified";
note_info.event_name_len = 13;
```

```
note_info.parms_type = ul_ev_note_info::P_TABLE_NAME;  
note_info.parms = table->name->data;  
note_info.parms_len = table->name->len;
```

Mit Warteschlangen arbeiten

Warteschlangen können erstellt und vernichtet werden.

CreateNotificationQueue erstellt eine Warteschlange für Ereignisbenachrichtigungen der aktuellen Verbindung. Der Bereich von Warteschlangennamen gilt für die jeweilige Verbindung, daher können verschiedene Verbindungen Warteschlangen mit demselben Namen erstellen. Wenn eine Ereignisbenachrichtigung gesendet wird, empfangen alle Warteschlangen in der Datenbank mit einem übereinstimmenden Namen eine separate Instanz der Benachrichtigung. Namen reagieren nicht auf Groß- und Kleinschreibung. Eine Standard-Warteschlange wird bei Bedarf für jede Verbindung erstellt, falls keine Warteschlange angegeben ist. Dieser Aufruf schlägt mit einem Fehler fehl, wenn der Name für die Verbindung bereits vorhanden oder nicht gültig ist.

DestroyNotificationQueue vernichtet die angegebene Ereignisbenachrichtigungs-Warteschlange. Eine Warnung wird signalisiert, wenn ungelesene Benachrichtigungen in der Warteschlange verbleiben. Ungelesene Benachrichtigungen werden verworfen. Die Standard-Ereigniswarteschlange einer Verbindung, falls erstellt, wird vernichtet, wenn die Verbindung geschlossen wird.

Mit Ereignissen arbeiten

DeclareEvent deklariert ein Ereignis, für das man sich anschließend registrieren kann und das ausgelöst werden kann. In UltraLite gibt es vordefinierte Systemereignisse, die von Vorgängen in der Datenbank oder in der Umgebung ausgelöst werden. Der Ereignisname muss eindeutig sein. Namen berücksichtigen nicht die Groß- und Kleinschreibung. Gibt TRUE zurück, wenn das Ereignis erfolgreich deklariert wurde, und FALSE, wenn der Name bereits verwendet wird oder ungültig ist.

RegisterForEvent registriert eine Warteschlange, um Benachrichtigungen für ein Ereignis zu empfangen. Wenn kein Warteschlangename geliefert wird, wird die Standard-Verbindungswarteschlange angenommen und ggf. erstellt. Bestimmte Systemereignisse lassen die Angabe eines Objektnamens zu, für den das Ereignis gilt. Das TableModified-Ereignis z.B. kann den Tabellennamen spezifizieren. Im Gegensatz zu SendNotification empfängt nur die spezifische registrierte Warteschlange Benachrichtigungen des Ereignisses, andere Warteschlangen mit demselben Namen bei anderen Verbindungen erhalten keine (außer sie sind auch explizit registriert). Gibt TRUE zurück, wenn die Registrierung erfolgreich war, und FALSE, wenn die Warteschlange oder das Ereignis nicht existiert.

TriggerEvent löst ein Ereignis aus und sendet eine Benachrichtigung an alle registrierten Warteschlangen. Gibt die Anzahl der gesendeten Ereignisbenachrichtigungen zurück. Parameter können als Name=Wert;-Paare übergeben werden.

Mit Benachrichtigungen arbeiten

SendNotification sendet eine Benachrichtigung an alle Warteschlangen in der Datenbank, die dem angegebenen Namen entsprechen (einschließlich etwaiger Warteschlangen für die aktuelle Verbindung). Dieser Aufruf bewirkt keine Blockierung. Verwenden Sie den speziellen Warteschlangennamen "*", um Benachrichtigungen an alle Warteschlangen zu senden. Gibt die Anzahl der gesendeten Benachrichtigungen zurück (die Anzahl der übereinstimmenden Warteschlangen). Parameter können als Name=Wert;-Paare übergeben werden.

GetNotification liest eine Ereignisbenachrichtigung. Dieser Aufruf wird blockiert, bis eine Benachrichtigung empfangen wird oder die angegebene Wartezeit abgelaufen ist. Um den Wartezustand zu beenden, senden Sie eine weitere Benachrichtigung an die Warteschlange oder verwenden Sie **CancelGetNotification**. Nach dem Lesen der Benachrichtigung verwenden Sie **ReadNotificationParameter**, um zusätzliche Parameter abzurufen. Gibt TRUE zurück, wenn ein Ereignis gelesen wurde, und FALSE, wenn die Wartezeit abgelaufen ist oder der Wartezustand beendet wurde.

GetNotificationParameter ruft einen benannten Parameter für die Ereignisbenachrichtigung ab, die gerade von **GetNotification** gelesen wurde. Es sind nur die Parameter aus der zuletzt gelesenen Benachrichtigung in der angegebenen Warteschlange verfügbar. Gibt TRUE zurück, wenn der Parameter gefunden wurde, und FALSE, wenn dies nicht der Fall war.

CancelGetNotification bricht ausstehende **GetNotification**-Aufrufe in allen Warteschlangen ab, die mit dem angegebenen Namen übereinstimmen. Gibt die Anzahl der betroffenen Warteschlangen (nicht notwendigerweise die Anzahl der blockierten Lesevorgänge) zurück.

Sonstige Hinweise

- Namen von Benachrichtigungswarteschlangen und Ereignissen sind auf 32 Zeichen beschränkt.
- Um Systemressourcen zu steuern, ist die Anzahl von Benachrichtigungen beschränkt. Wenn dieses Limit überschritten wird, wird `SQL_EVENT_NOTIFICATION_QUEUE_FULL` signalisiert und die ausstehenden Benachrichtigungen werden verworfen.

Isolationsstufen

Isolationsstufen legen fest, wie weit Vorgänge aus einer Transaktion für Vorgänge anderer paralleler Transaktionen sichtbar sind. UltraLite verwendet für Verbindungen die Standardisolationsstufe `read_committed`. Die UltraLite-Standardisolationsstufe unterstützt die Datenkonsistenz durch Isolieren nicht festgeschriebener Zeilen.

Isolationsstufe	Merkmale
0— <code>read_uncommitted</code> (nicht festgeschriebene Daten lesen)	<ul style="list-style-type: none"> • Lässt Dirty Reads, nicht wiederholbare Lesevorgänge und Phantomzeilen zu • Keine Garantie, dass gleichzeitige Transaktionen nicht die Zeile ändern oder Änderungen an Zeilen zurücksetzen
1 - <code>read_committed</code> (festgeschriebene Daten lesen)	<ul style="list-style-type: none"> • Lässt nicht wiederholbare Lesevorgänge und Phantomzeilen zu • Verhindert Dirty Reads • Keine Garantie, dass Abfrageergebnisse nicht während der Transaktion geändert werden

Sie können die Isolationsstufe wie folgt von `read_committed` zu `read_uncommitted` ändern:

- Verwenden Sie die SQL-Anweisung `SET OPTION` und die Datenbankoption `isolation_level`.

Die folgende Anweisung setzt beispielsweise die Isolationsstufe auf `read_uncommitted`.

```
SET OPTION isolation_level = 'READ_UNCOMMITTED'
```

- Für die UltraLite C++-API verwenden Sie die Methode `ULConnection.SetDatabaseOption`, um die Isolationsstufe zu ändern.

Für die UltraLite.NET-API verwenden Sie die Methoden `ULConnection.BeginTransaction` oder `ULDatabaseSchema.SetDatabaseOption` zum Erstellen einer Transaktion mit der Isolationsstufe `read_committed`.

Für die UltraLiteJ-API verwenden Sie die Methode `Connection.setOption`.

Hinweis

UltraLite Java Edition-Datenbanken unterstützen nur die Isolationsstufe `read_uncommitted`. Siehe „Datensynchronisation auf einem BlackBerry-Smartphone“ [[UltraLite® – Java-Programmierung](#)].

Parallelität und Sperren für UltraLite Java Edition-Datenbanken

- **Sperren** Zwei verschiedene Verbindungen können nicht gleichzeitig dieselbe Zeile ändern. Wenn zwei Verbindungen versuchen, auf dieselbe Zeile zuzugreifen, erhält die zweite Verbindung eine Fehlermeldung und darf die Zeile erst wieder ändern, wenn die erste Verbindung ihre aktuelle Transaktion festschreibt oder zurücksetzt.
- **Sichtbarkeit** Der Vorgang einer Verbindung in der Datenbank wird sofort für andere Verbindungen sichtbar.

Siehe auch

- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- „UltraLite-Option `isolation_level`“ auf Seite 199
- `ULConnection.SetDatabaseOption`-Methode [UltraLite C++] [[UltraLite - C- und C++-Programmierung](#)]
- `ULConnection.BeginTransaction`-Methode [UltraLite.NET] [[UltraLite - .NET-Programmierung](#)]
- `ULDatabaseSchema.SetDatabaseOption`-Methode [UltraLite.NET] [[UltraLite - .NET-Programmierung](#)]
- `Connection.setOption`-Methode [UltraLiteJ] [[UltraLite® – Java-Programmierung](#)]

Eigenschaften der `read_uncommitted`- Isolationsstufe

Die folgenden Nebenwirkungen sind möglich, wenn UltraLite auf der Isolationsstufe 0 (`read_uncommitted`) arbeitet:

- Anwendungen können nicht festgeschriebene Daten lesen (Dirty Reads). In diesem Szenario können Transaktionen auf Zeilen in der Datenbank zugreifen, die nicht festgeschrieben sind und von einer anderen Transaktion noch zurückgesetzt werden können. Phantomzeilen sind möglich (Zeilen, die nach der ursprünglichen Abfrage hinzugefügt werden, sodass bei einer Wiederholung der Abfrage eine andere Ergebnismenge zurückgegeben wird).

Eine praktische Einführung, in der die Nebenwirkungen von Dirty Reads veranschaulicht werden, finden Sie unter „[Praktische Einführung: Einführung in Dirty Reads](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Eine praktische Einführung mit einem Beispiel einer Phantomzeile finden Sie unter „[Praktische Einführung: Einführung in Phantomzeilen](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

- Anwendungen können nicht wiederholbare Lesevorgänge ausführen. In diesem Szenario liest eine Anwendung eine Zeile aus der Datenbank und führt dann andere Vorgänge aus. Anschließend aktualisiert bzw. löscht eine zweite Anwendung die Zeile und schreibt die Änderung fest. Wenn die erste Anwendung versucht, die ursprüngliche Zeile erneut zu lesen, erhält sie entweder die aktualisierten Informationen oder stellt fest, dass die ursprüngliche Zeile gelöscht wurde.

Eine praktische Einführung, in der Auswirkungen von nicht wiederholbaren Lesevorgängen gezeigt werden, finden Sie unter „[Praktische Einführung: Einführung in nicht-wiederholbare Lesevorgänge](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- [ULConnection.SetDatabaseOption-Methode](#) [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- [ULConnection.BeginTransaction-Methode](#) [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- [ULDatabaseSchema.SetDatabaseOption-Methode](#) [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- [Connection.setOption-Methode](#) [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- „[SET OPTION-Anweisung](#) [UltraLite]“ auf Seite 459
- „[UltraLite-Option isolation_level](#)“ auf Seite 199

Beispiel

Die beiden Verbindungen A und B haben jeweils eine eigene Transaktion.

1. Wenn die Verbindung A mit der Ergebnismenge einer Abfrage arbeitet, **ruft** UltraLite eine Kopie der aktuellen Zeile in einen Puffer ab.

Hinweis

Durch das Lesen oder Abrufen einer Zeile wird diese Zeile nicht festgeschrieben. Wenn Verbindung A eine Zeile abrufen, aber nicht ändert, kann diese Zeile von Verbindung B noch geändert werden.

2. Wenn A die aktuelle Zeile bearbeitet, wird die Kopie im Puffer geändert. Die Kopie im Puffer wird in die Datenbank zurückgeschrieben, wenn die Verbindung A eine Aktualisierungsmethode aufruft oder die Ergebnismenge schließt.
3. Es wird eine Schreibsperrung für die Zeile gesetzt, damit andere Transaktionen diese Zeile nicht ändern können. Diese Änderung wird erst festgeschrieben, wenn die Verbindung A einen Commit-Vorgang ausführt.
4. Abhängig von der Änderung stellt die Verbindung B beim Abrufen der aktuellen Zeile möglicherweise Folgendes fest:

Änderung der Verbindung A	Ergebnis ¹
Zeile wurde gelöscht.	Verbindung B ruft die nächste Zeile in der Ergebnismenge ab.
Zeile wurde geändert.	Verbindung B ruft die letzte Kopie der Zeile ab.

¹ Von Verbindung A und B verwendete Abfragen enthalten keine temporären Tabellen. Temporäre Tabellen können andere Auswirkungen haben.

Eine UltraLite-Datenbank validieren

Sie sollten daher regelmäßig die Gültigkeit Ihrer Datenbank prüfen, indem Sie Tools wie den **Assistenten zum Validieren einer Datenbank** in Sybase Central oder das UltraLite-Dienstprogramm zum Validieren von Datenbanken oder die ValidateDatabase-Methode in der UltraLite API verwenden.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Die Beschädigung einer Datenbankdatei wird möglicherweise erst erkannt, wenn der Datenbankserver versucht, auf den betroffenen Teil der Datenbank zuzugreifen.

Vorsicht

Eine Datenbankvalidierung sollte nur durchgeführt werden, wenn keine Verbindung eine Änderung an der Datenbank durchführt, da sonst eine Datenbankbeschädigung gemeldet wird, obwohl eine solche nicht vorhanden ist.

Sie können eine UltraLite-Datenbank mit einer der folgenden Methoden validieren:

- Mit dem **Assistenten zum Validieren einer Datenbank** in Sybase Central.
- Mit dem Befehlszeilendienstprogramm ulvalid.
- Mit der ValidateDatabase-Methode in der UltraLite API.

Die UltraLite Java Edition-Datenbankvalidierung wird nicht unterstützt.

Aufgabe

1. Klicken Sie im linken Fensterausschnitt von Sybase Central auf die UltraLite-Datenbank.
2. Klicken Sie auf **Datei » Datenbank validieren**.
3. Befolgen Sie die Anweisungen im **Assistenten zum Validieren einer Datenbank**.

Ergebnisse

Die Datenbank wird validiert.

Siehe auch

- [ULConnection.ValidateDatabase-Methode \[UltraLite C++\]](#) [*UltraLite - C- und C++-Programmierung*]
- [ULConnection.ValidateDatabase-Methode \[UltraLite.NET\]](#) [*UltraLite - .NET-Programmierung*]
- [Connection.validateDatabase-Methode \[Android\]](#) [*UltraLiteJ*] [*UltraLite® – Java-Programmierung*]
- „UltraLite-Dienstprogramm zum Validieren von Datenbanken (ulvalid)“ auf Seite 237

Sichern und Wiederherstellen einer UltraLite- und UltraLite Java Edition-Datenbank

Wenn eine Anwendung, die eine UltraLite oder UltraLite Java Edition-Datenbank verwendet, unerwartet ausfällt, wird die Datenbank beim nächsten Start automatisch in einem konsistenten Zustand wiederhergestellt. Alle festgeschriebenen Transaktionen, die vor dem unerwarteten Ausfall in den Speicher geschrieben wurden, sind in der Datenbank enthalten. Alle Transaktionen, die zum Zeitpunkt des Ausfalls noch nicht in den Speicher geschrieben wurden, werden zurückgesetzt.

Eine UltraLite verwendet kein Transaktionslog für die Wiederherstellung. Stattdessen speichert UltraLite für jede Zeile Zustandsdaten, um die Behandlung einer Zeile bei der Wiederherstellung zu bestimmen.

Sicherungen

UltraLite bietet Schutz gegen Systemausfälle, aber nicht gegen Datenträgerfehler. Die einfachste Möglichkeit, eine Sicherung einer UltraLite-Anwendung zu erstellen, besteht darin, sie mit einer konsolidierten Datenbank zu synchronisieren. Um eine UltraLite- oder UltraLite Java Edition-Datenbank wiederherzustellen, starten Sie mit einer leeren Datenbank und füllen sie über die Synchronisation mit Daten aus der konsolidierten Datenbank.

Siehe auch

- „Verwaltung des Zeilenstatus in einer UltraLite-Datenbank“ auf Seite 485
- „Bereinigen einzelner oder gruppierter Transaktionen“ auf Seite 487
- „UltraLite als MobiLink-Client“ auf Seite 69

UltraLite-Datenbankschemas

Der logische Rahmen der Datenbank wird als **Schema** bezeichnet.

UltraLite-Datenbankschemas

Sie können ein Upgrade des Schemas einer UltraLite-Datenbank mithilfe der entsprechenden DDL-Anweisungen (Data Definition Language) oder unter Verwendung der ALTER DATABASE SCHEMA FROM FILE-Anweisung zur Änderung der Schemadefinition mittels eines SQL-Skripts durchführen.

Schemaänderungen können erhebliche Zeit in Anspruch nehmen. Beispiel: Wenn der Spaltentyp geändert wird, müssen alle Zeilen in der zugeordneten Tabelle aktualisiert werden. DDL-Anweisungen werden nur bei Abwesenheit der folgenden Elemente erfolgreich ausgeführt:

- Nicht festgeschriebene Transaktionen
- Andere aktive Verwendungen der Datenbank (z.B. Synchronisation, vorbereitete, aber noch nicht freigegebene Anweisungen oder laufende Datenbankvorgänge)

Wenn die DDL-Anweisung ausgeführt wird, wird jeder andere Versuch, die Datenbank zu verwenden, blockiert, bis die DDL-Anweisung die Schemaänderung abgeschlossen hat.

UltraLite Java Edition-Datenbankschemas

Das UltraLite Java Edition-Datenbankschema wird als Katalog von Systemtabellen geführt, der die Metadaten für die UltraLite Java Edition-Datenbank enthält. Systemtabelle-Metadaten sind:

- **Tabellendefinitionen** Gespeichert in der Systemtabelle systable.
- **Spaltendefinitionen** Gespeichert in der Systemtabelle syscolumn.
- **Indexdefinitionen** Gespeichert in den Systemtabellen sysindex und sysindexcolumn.
- **Publikationsdefinitionen** Gespeichert in den Systemtabellen syspublications und sysarticles.
- **Fremdschlüsseldefinitionen** Gespeichert in den Systemtabellen sysforeignkey und sysfkcol.
- **Benutzernamen und Kennwörter** Gespeichert in der Systemtabelle sysuldata.

Siehe auch

- „ALTER DATABASE SCHEMA FROM FILE-Anweisung [UltraLite]“ auf Seite 423
- „Deployment von UltraLite-Datenbankschema-Upgrades“ auf Seite 125
- „systable-Systemtabelle“ auf Seite 268
- „syscolumn-Systemtabelle“ auf Seite 264
- „sysindex-Systemtabelle“ auf Seite 266
- „sysindexcolumn-Systemtabelle“ auf Seite 267
- „syspublications-Systemtabelle“ auf Seite 268
- „sysarticles-Systemtabelle“ auf Seite 263
- „sysforeignkey-Systemtabelle“ auf Seite 265
- „sysfkcol-Systemtabelle“ auf Seite 266
- „sysuldata-Systemtabelle“ auf Seite 269

UltraLite-Tabellen und -Spalten

Tabellen werden zum Speichern von Daten verwendet. Sie legen die Beziehungen der in ihnen enthaltenen Daten fest. Tabellen bestehen aus Zeilen und Spalten. Jede Spalte enthält eine spezielle Art von Informationen, z.B. Telefonnummern oder Namen, während jede Zeile einen speziellen Eintrag enthält.

Wenn Sie eine UltraLite-Datenbanken erstellen, sehen Sie zunächst nur die Systemtabellen. Systemtabellen enthalten das UltraLite-Schema. Sie können diese Tabellen in Sybase Central nach Bedarf anzeigen oder ausblenden.

Sie können der Anwendung dann erforderliche neue Tabellen hinzufügen. Sie können diese Tabellen auch durchsuchen und Daten zwischen vorhandenen Tabellen in der Quelldatenbank oder anderen offenen Zieldatenbanken kopieren und einfügen.

In UltraLite können Sie nur Basistabellen erstellen, die Sie für dauerhafte Daten deklarieren. UltraLite unterstützt keine globalen temporären oder deklarierte temporären Tabellen.

Gepackte Zeilen und Tabellendefinitionen

UltraLite verwendet Zeilen in zwei Formaten:

- **Entpackte Zeilen** sind das unkomprimierte Format. Jede Zeile muss entpackt werden, damit einzelne Spaltenwerte gelesen oder geschrieben werden können.
- **Gepackte Zeilen** sind die komprimierte Darstellung der entpackten Zeile, wobei jeder Spaltenwert komprimiert wird, sodass die gesamte Zeile möglichst wenig Platz einnimmt. Die Größe einer gepackten Zeile hängt von den Werten in den einzelnen Spalten ab. Die Größe von zwei gepackten Zeilen einer Tabelle kann völlig unterschiedlich sein. Beachten Sie, dass LONG BINARY- und LONG VARCHAR-Spalten getrennt von der gepackten Zeile gespeichert werden.

In UltraLite muss eine gepackte Zeile auf eine Datenbankseite passen. Da LONG BINARY- und LONG VARCHAR-Spalten nicht zusammen mit der gepackten Zeile gespeichert werden, können sie die Seitengröße übersteigen.

Beachten Sie, dass Tabellendefinitionen die Zeile *vor* dem Packen der Daten durch die UltraLite-Laufzeitdatenbank beschreiben. Da die Größe der gepackten Zeile von den Werten in den einzelnen Spalten abhängt, ist es nicht möglich, anhand der Tabellendefinition vorab festzustellen, ob die Größenanforderung von den gepackten Zeilen erfüllt wird. Es ist somit in UltraLite möglich, eine Tabelle zu definieren, in der eine entpackte Zeile nicht auf eine Seite passt. Um zu ermitteln, ob eine Zeile auf eine Seite passt, müssen Sie versuchen, die Zeile einzufügen oder zu aktualisieren. Wenn sie nicht auf die Seite passt, erkennt UltraLite dies und meldet einen Fehler.

Hinweis

Sie können Tabellen nicht mit beliebiger Größe deklarieren. UltraLite hält ein deklariertes Tabellenzeilen-Größenlimit von 64 kB aufrecht. Wenn Sie versuchen, eine Tabelle zu definieren, in der eine entpackte Zeile diese Obergrenze übersteigen kann, generiert UltraLite den SQL-Fehler `SQL_MAX_ROW_SIZE_EXCEEDED (-1132)`.

Siehe auch

- „UltraLite-Erstellungsparameter `page_size`“ auf Seite 155
- „Datenbanktabellen“ [*SQL Anywhere 16 - Einführung*]
- „Datenbankerstellung“ [*SQL Anywhere Server - Datenbankadministration*]
- „UltraLite-Systemtabellen“ auf Seite 238

UltraLite-Tabellen erstellen

Sie können Basistabellen erstellen, in der die beständigen relationalen Daten gespeichert werden.

Voraussetzungen

Tabellen in UltraLite-Anwendungen müssen einen Primärschlüssel enthalten. Während der MobiLink-Synchronisation sind ebenfalls Primärschlüssel erforderlich, um Zeilen in der UltraLite-Datenbank mit Zeilen in der konsolidierten Datenbank zu verknüpfen.

Aufgabe

1. Stellen Sie mithilfe von Sybase Central eine Verbindung mit der UltraLite-Datenbank her.
2. Rechtsklicken Sie im linken Fensterausschnitt auf **Tabellen** und klicken Sie auf **Neu » Tabelle**.
3. Im Feld **Wie lautet der Name der neuen Tabelle?** geben Sie den Namen der neuen Tabelle ein.
4. Klicken Sie auf **Fertig stellen**.
5. Klicken Sie im Menü **Datei** auf die Option **Speichern**.

Ergebnisse

Die Tabelle wird erstellt. Die Tabelle und die darin enthaltenen Daten bleiben so lange bestehen, bis Sie die Daten oder die Tabelle ausdrücklich löschen.

Nächste Schritte

Spalten hinzufügen oder Indizes erstellen

Siehe auch

- „CREATE TABLE-Anweisung [UltraLite]“ auf Seite 438
- „Eine Spalte einer UltraLite-Tabelle hinzufügen“ auf Seite 52

Eine Spalte einer UltraLite-Tabelle hinzufügen

Spalten können einer UltraLite-Tabelle hinzugefügt werden, nachdem sie erstellt wurde.

Voraussetzungen

Wenn die Tabelle bereits Daten enthält, können Sie nur dann eine Spalte hinzufügen, wenn die Spaltendefinition einen Standardwert enthält oder NULL zulässt.

Aufgabe

1. Stellen Sie mithilfe von Sybase Central eine Verbindung mit der UltraLite-Datenbank her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Tabellen**.
3. Doppelklicken Sie auf eine Tabelle.
4. Klicken Sie auf die Registerkarte **Spalten**, rechtsklicken Sie auf den leeren Bereich unterhalb der Tabelle und klicken Sie auf **Neu » Spalte**.
5. Legen Sie die Attribute für die neue Spalte fest und speichern Sie Ihre Änderungen.

Ergebnisse

Die Spalte wird der Tabelle hinzugefügt.

Siehe auch

- „Hinweise zu Objektnamen“ [*SQL Anywhere Server - Datenbankadministration*]
- „UltraLite, SQLDatentypen“ auf Seite 296
- „Hinweise zu Spalten-Datentypen“ [*SQL Anywhere Server - Datenbankadministration*]
- „CREATE TABLE-Anweisung [UltraLite]“ auf Seite 438
- „ALTER TABLE-Anweisung [UltraLite]“ auf Seite 427

UltraLite-Spaltendefinitionen ändern

Sie ändern die Struktur von Spaltendefinitionen für eine Tabelle, indem Sie verschiedene Spaltenattribute ändern oder auch ganze Spalten löschen.

Voraussetzungen

Die geänderte Spaltendefinition muss die Anforderungen aller Daten erfüllen, die bereits in der betreffenden Spalte gespeichert sind. Sie können z.B. nicht für eine Spalte festlegen, dass sie den Wert NULL nicht enthalten darf, wenn die Spalte einen NULL-Eintrag enthält.

Aufgabe

1. Stellen Sie mithilfe von Sybase Central eine Verbindung mit der UltraLite-Datenbank her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Tabellen**.
3. Doppelklicken Sie auf eine Tabelle.
4. Klicken Sie auf die Registerkarte **Spalten** und ändern Sie die Spaltenattribute.
5. Klicken Sie im Menü **Datei** auf die Option **Tabelle speichern**.

Ergebnisse

Die Tabelle wird mit den neuen Spaltenattributen gespeichert.

Siehe auch

- „Hinweise zu Objektnamen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „UltraLite, SQLDatentypen“ auf Seite 296
- „Hinweise zu Spalten-Datentypen“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „ALTER TABLE-Anweisung [UltraLite]“ auf Seite 427

UltraLite-Tabellen löschen

Sie können Tabellen löschen, wenn Sie sie nicht mehr brauchen.

Voraussetzungen

Sie können jede Tabelle löschen, wenn Folgendes zutrifft:

- Die Tabelle wird nicht als Artikel in einer Publikation verwendet.
- Die Tabelle hat keine Spalten, die vom Fremdschlüssel einer anderen Tabelle referenziert werden.

In diesen Fällen müssen Sie die Publikation ändern oder den Fremdschlüssel löschen, *bevor* Sie die Tabelle erfolgreich löschen können.

Aufgabe

1. Stellen Sie mithilfe von Sybase Central eine Verbindung mit der UltraLite-Datenbank her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Tabellen**.

3. Rechtsklicken Sie auf die Tabelle und klicken Sie auf **Löschen**.
4. Klicken Sie auf **Ja**.

Ergebnisse

Die Tabelle wird gelöscht.

Siehe auch

- [„DROP TABLE-Anweisung \[UltraLite\]“ auf Seite 448](#)

Informationen in UltraLite-Tabellen durchsuchen

Daten anzeigen, die sich in den Tabellen einer UltraLite-Datenbank befinden.

Voraussetzungen

Die Datenbank muss verbunden und ausgewählt sein.

Kontext und Bemerkungen

Tabellen können Benutzertabellen oder Systemtabellen sein. Sie können Tabellen filtern, indem Sie Systemtabellen in Ihrer aktuellen Ansicht der Datenbank ein- bzw. ausblenden. Da UltraLite das Konzept der Eigentümerschaft nicht kennt, können alle Benutzer alle Tabellen durchsuchen.

Aufgabe

1. Stellen Sie mithilfe von Sybase Central eine Verbindung mit der UltraLite-Datenbank her.
2. Um eine Liste der Tabellen anzuzeigen, doppelklicken Sie auf **Tabellen**.
3. Um Tabellendaten anzuzeigen, doppelklicken Sie auf die Tabelle und klicken auf die Registerkarte **Daten** im rechten Fensterausschnitt.

Ergebnisse

Die Tabellen und Daten werden angezeigt.

Siehe auch

- [„UltraLite-Systemtabellen“ auf Seite 238](#)

Kopieren und Einfügen von Daten aus bzw. in UltraLite-Datenbanken

In Sybase Central können Sie sowohl kopieren und einfügen als auch ziehen und ablegen. Diese Datenübertragung ermöglicht es Ihnen, Objekte in einer oder mehreren Datenbanken gemeinsam zu

verwenden oder zu verschieben. Wenn Sie Objekte kopieren und einfügen oder ziehen und ablegen, können Sie Daten wie in der folgenden Tabelle beschrieben gemeinsam verwenden.

Ziel	Ergebnis
Eine weitere UltraLite- oder SQL Anywhere-Datenbank	Es wird ein neues Objekt erstellt, und der Code des ursprünglichen Objekts wird auf das neue Objekt kopiert.
Dieselbe UltraLite-Datenbank	Eine Kopie des Objekts wird erstellt. Sie müssen das neue Objekt umbenennen.

Hinweis

Sie können Daten aus einer in MobiLink geöffneten Datenbank kopieren und in eine UltraLite-Datenbank einfügen. Es ist jedoch nicht möglich, UltraLite-Daten in eine in MobiLink geöffnete Datenbank einzufügen.

Sybase Central

Wenn Sie eines der folgenden Objekte im UltraLite-Plug-In kopieren, wird auch die SQL-Anweisung für das Objekt in die Zwischenablage kopiert. Sie können diese SQL-Anweisung in andere Anwendungen, wie z.B. Interactive SQL oder einen Texteditor, einfügen. Wenn Sie beispielsweise einen Index in Sybase Central kopieren und in einen Texteditor einfügen, erscheint die Anweisung CREATE INDEX für diesen Index. Sie können die folgenden Objekte im UltraLite-Plug-In kopieren:

- Artikel
- Spalten
- Fremdschlüssel
- Indizes
- Publikationen
- Tabellen
- Eindeutigkeits-Integritätsregeln

Interactive SQL

In Interactive SQL können Sie auch Daten aus einer Ergebnismenge in ein anderes Objekt kopieren.

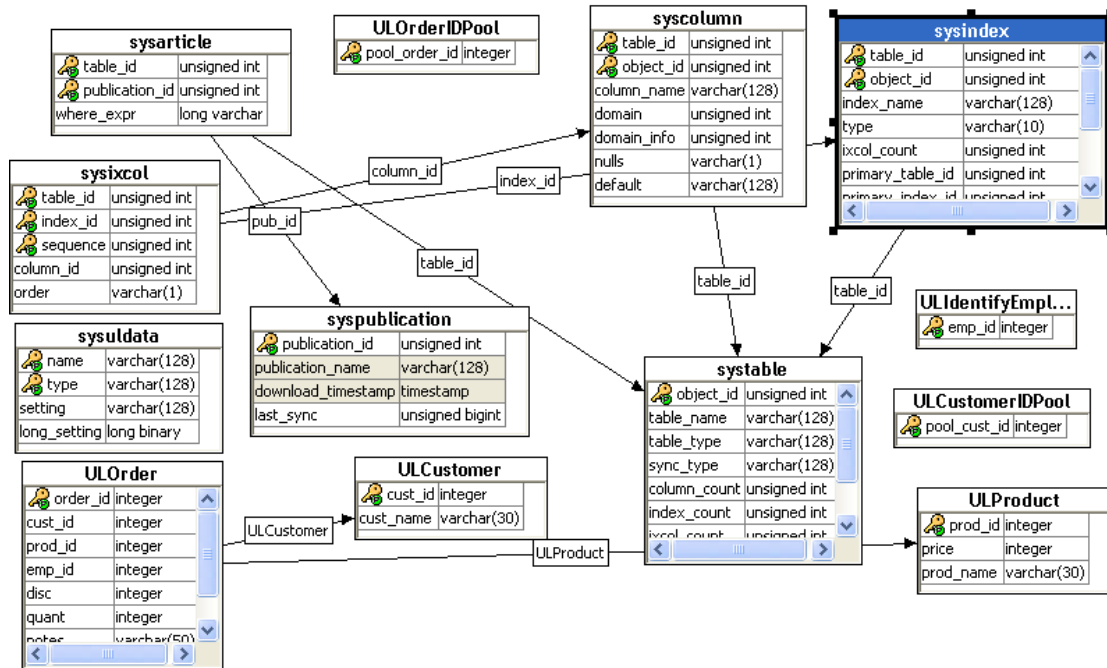
- Verwenden Sie die SELECT-Anweisung, um Ergebnisse in ein benanntes Objekt zu kopieren.
- Verwenden Sie die INSERT-Anweisung, um eine Zeile oder eine Auswahl von Zeilen an anderen Stellen in der Datenbank in eine Tabelle einzufügen.

Siehe auch

- „INSERT-Anweisung [UltraLite]“ auf Seite 452
- „SELECT-Anweisung [UltraLite]“ auf Seite 457

Anzeigen von Entity-Relationship-Diagrammen aus dem UltraLite-Plug-In

Wenn Sie über das UltraLite-Plug-In mit einer Datenbank verbunden sind, können Sie ein Entity-Relationship-Diagramm der Tabellen in der Datenbank anzeigen. Wenn Sie eine Datenbank ausgewählt haben, klicken Sie auf die Registerkarte **ER-Diagramm** im rechten Fensterausschnitt, um das Diagramm anzuzeigen.



Wenn Sie Objekte im Diagramm neu anordnen, gelten die Änderungen auch für die nachfolgenden Sybase Central-Sitzungen. Wenn Sie auf eine Tabelle doppelklicken, werden die Spaltendefinitionen für diese Tabelle angezeigt.

Siehe auch

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

UltraLite-Indizes

Ein Index ist eine Gruppe von Zeigern auf Zeilen in einer Tabelle, basierend auf der Reihenfolge der Werte von Daten in einer oder mehreren Tabellenspalten. Der Index ist ein Datenbankobjekt, das von UltraLite nach seiner Erstellung automatisch verwaltet wird. Wenn UltraLite eine Abfrage optimiert, werden vorhandene Indizes analysiert, um festzustellen, ob ein Index für die in der Abfrage genannten Tabellen vorhanden ist. Wenn der Index dazu beitragen kann, dass UltraLite Zeilen schneller zurückgibt,

wird er verwendet. Wenn Sie die UltraLite-Tabellen-API in Ihrer Anwendung verwenden, können Sie einen Index angeben, um die Reihenfolge zu ermitteln, in der die Zeilen durchsucht werden.

Tipp

Indizes können die Performance einer Abfrage verbessern, v.a. für große Tabellen. Um festzustellen, ob eine Abfrage einen bestimmten Index verwendet, können Sie den Ausführungsplan mit Interactive SQL überprüfen.

Alternativ dazu kann die UltraLite-Anwendung PreparedStatement-Objekte enthalten, die eine Methode zur Zurückgabe von Plänen besitzen.

UltraLite unterstützt folgende Indizes. Die Indizes können eine oder mehrere Spalten (so genannte zusammengesetzte Indizes) umfassen. Es ist nicht möglich, LONG VARCHAR- oder LONG BINARY-Spalten zu indizieren.

Index	Merkmale
Primärschlüssel	Erforderlich. Eine Instanz eines eindeutigen Schlüssels. Es kann nur einen Primärschlüssel geben. Werte in den indizierten Spalten müssen eindeutig sein und dürfen nicht NULL sein.
Fremdschlüssel ¹	Optional. Werte in den indizierten Spalten können dupliziert werden. Die Nullwertfähigkeit hängt davon ab, ob bei der Spaltenerstellung NULL als Wert zugelassen wurde. Werte in den Fremdschlüsselspalten müssen in der referenzierten Tabelle vorhanden sein.
Eindeutiger Schlüssel ²	Optional. Werte in den indizierten Spalten müssen eindeutig sein und dürfen nicht NULL sein.
Nicht eindeutiger Index	Optional. Werte in den indizierten Spalten können dupliziert werden und NULL sein.
Eindeutiger Index	Optional. Werte in den indizierten Spalten dürfen nicht dupliziert werden und können NULL sein.

¹ Ein Fremdschlüssel kann einen Primärschlüssel oder einen eindeutigen Schlüssel referenzieren.

² Auch bekannt als Eindeutigkeits-Integritätsregel.

Zusammengesetzte Indizes

Mehrsaltige Indizes werden auch als zusammengesetzte Indizes bezeichnet. Durch zusätzliche Spalten in einem Index können Sie zwar Ihre Suche eingrenzen, doch ist ein zweispaltiger Index nicht dasselbe wie zwei separate Indizes. 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.

Siehe auch

- „Index-Scan erstellen und verwalten“ auf Seite 471
- „Ausführungspläne in UltraLite“ auf Seite 477
- „Zusammengesetzte Indizes“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- UltraLite.NET: „Datenerstellung und -änderung mit der UTable-Klasse“ [*UltraLite - .NET-Programmierung*]
- ULCommand.Prepare-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- UltraLite für C++: „Datenerstellung und -änderung mit der UTable-Klasse“ [*UltraLite - C- und C++-Programmierung*]
- ULPreparedStatement-Klasse [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]

Wann sollte ein Index erstellt werden

Verwenden Sie in folgenden Fällen einen Index:

- **Die referenzielle Integrität von UltraLite soll bewahrt werden** Ein Index bietet UltraLite darüber hinaus eine Möglichkeit, eine Eindeutigkeits-Integritätsregel auf die Zeilen der Tabelle zu erzwingen. Es ist nicht erforderlich, für Daten, die sehr ähnlich sind, einen Index hinzuzufügen.
- **Die Performance einer bestimmten Abfrage ist wichtig für Ihre Anwendung** Wenn ein Index die Performance einer Abfrage verbessert, die Performance dieser Abfrage wichtig für die Anwendung ist und die Abfrage häufig verwendet wird, sollten Sie diesen Index verwalten. Außer wenn die betreffende Tabelle sehr klein ist, können Indizes die Performance bei Suchen erheblich verbessern. Indizes werden daher gewöhnlich empfohlen, wenn Daten häufig gesucht werden.
- **Sie haben komplexe Abfragen** Komplexere Abfragen (z.B. mit JOIN-, GROUP BY- und ORDER BY-Klauseln) können erhebliche Verbesserungen erzielen, wenn ein Index verwendet wird, wenngleich es schwieriger sein kann, den Grad der Performance-Verbesserung zu ermitteln. Testen Sie daher Ihre Abfragen sowohl mit als auch ohne Indizes, um festzustellen, welche Version eine bessere Performance erzielt.
- **Eine UltraLite-Tabelle ist sehr groß** Die durchschnittliche Dauer zum Auffinden einer Zeile nimmt mit der Größe der Tabelle zu. Um die Suchvorgänge in sehr großen Tabellen zu beschleunigen, sollten Sie daher einen Index verwenden. Ein Index gestattet es UltraLite, Zeilen schnell zu finden. Dies gilt jedoch nur für indizierte Spalten. Andernfalls muss UltraLite jede Zeile in der Tabelle durchsuchen, um festzustellen, ob die Zeile die Suchbedingung erfüllt. Dieser Vorgang kann in einer großen Tabelle sehr zeitaufwändig sein.
- **Die UltraLite-Clientanwendung führt nicht viele Insert-, Update- oder Delete-Vorgänge aus** Da UltraLite Indizes zusammen mit den Daten verwaltet, hat ein Index in diesem Kontext eine gegenteilige Wirkung auf die Performance von Datenbankvorgängen. Aus diesem Grund sollten Sie die Verwendung von Indizes auf Daten begrenzen, die regelmäßig abgefragt werden, wie oben

beschrieben. Die Verwaltung von UltraLite-Standardindizes (Indizes für Primärschlüssel und für Eindeutigkeits-Integritätsregeln) kann ausreichend sein.

- **Indizes für Spalten verwenden, die in WHERE- bzw. ORDER BY-Klauseln verwendet werden** Diese Indizes können die Auswertung dieser Spalten beschleunigen. Ein Index trägt dazu bei, eine mehrspaltige ORDER BY-Klausel zu optimieren, jedoch nur wenn die Positionierung von Spalten im Index und in den ORDER BY-Klauseln genau übereinstimmen.

Indextypen

UltraLite unterstützt verschiedene Typen von Indizes: eindeutige Schlüssel, eindeutige Indizes und nicht eindeutige Indizes. Diese Indextypen unterscheiden sich dadurch, was in den einzelnen Indizes erlaubt ist.

Indexmerkmale	Eindeutige Schlüssel	Eindeutige Indizes	Nicht eindeutige Indizes
Erlaubt doppelte Indexeinträge für Zeilen, die in indizierten Spalten dieselben Werte enthalten	no	no	yes
Erlaubt NULL in Indexspalten	no	yes	Ja

Hinweis

Sie können Fremdschlüssel für eindeutige Schlüssel, aber nicht für eindeutige Indizes erstellen.

Die manuelle Erstellung eines Indexes für Schlüsselspalten ist nicht nötig und wird in der Regel nicht empfohlen. UltraLite erstellt und verwaltet Indizes für eindeutige Schlüssel automatisch.

Siehe auch

- „Hinzufügen eines UltraLite-Indexes“ auf Seite 59

Hinzufügen eines UltraLite-Indexes

Das Hinzufügen von Indizes beschleunigt die Suche.

Voraussetzungen

Die Datenbank muss verbunden sein.

Kontext und Bemerkungen

Hinweis

UltraLite erkennt keine doppelten oder redundanten Indizes. Da Indizes mit den Daten in Ihrer Datenbank verwaltet werden müssen, sollten Sie Indizes stets mit Umsicht hinzufügen.

Aufgabe

1. Stellen Sie mithilfe von Sybase Central eine Verbindung mit der UltraLite-Datenbank her.
2. Rechtsklicken Sie auf **Indizes** und klicken Sie auf **Neu » Index**.
3. Befolgen Sie die Anweisungen des Assistenten.

Ergebnisse

Der Index wird erstellt.

Beispiel

Um die Suche nach Nachnamen von Mitarbeitern in einer Datenbank, die Mitarbeiterinformationen speichert, zu beschleunigen und die Performance von Abfragen mit diesem Index zu optimieren, können Sie einen Index namens EmployeeNames erstellen und die Hash-Größe auf 20 Byte setzen. Verwenden Sie dazu die folgende Anweisung:

```
CREATE INDEX EmployeeNames  
ON Employees (Surname, GivenName)  
WITH MAX HASH SIZE 20
```

Diese Anweisung erstellt einen Index, der die maximale, von Ihnen konfigurierte Standard-Hash-Größe verwendet. Sie können einen Index erstellen, der die Standardeinstellung außer Kraft setzt, indem Sie die Klausel `WITH MAX HASH SIZE Wert` verwenden, um einen neuen Wert für diese Indexinstanz festzulegen.

Siehe auch

- „CREATE INDEX-Anweisung [UltraLite]“ auf Seite 433

Löschen eines UltraLite-Indexes

Wenn Sie einen Index entfernen, wird er aus der Datenbank gelöscht.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. Stellen Sie mithilfe von Sybase Central eine Verbindung mit der UltraLite-Datenbank her.
2. Doppelklicken Sie im linken Fensterausschnitt auf **Indizes**.
3. Rechtsklicken Sie auf einen Index und klicken Sie auf **Löschen**.
4. Klicken Sie auf **Ja**.

Ergebnisse

Der Index wird aus der Datenbank entfernt.

Siehe auch

- „DROP INDEX-Anweisung [UltraLite]“ auf Seite 446

UltraLite-Benutzer

Eine typische UltraLite-Datenbank enthält eine Benutzer-ID und ein Kennwort. UltraLite-Datenbanken werden mit der Standardbenutzer-ID **DBA** und dem Standardkennwort **sql** erstellt, sofern nicht anders angegeben. UltraLite Java Edition-Datenbanken werden mit dem Standardkennwort **DBA** erstellt.

Das Ändern des Benutzerschemas ist optional und **nicht** erforderlich. Viele Anwendungen benötigen keine Authentifizierung auf Datenbankebene und gehen davon aus, dass ein Kennwort auf Geräteebe für den Zugriff auf eine Anwendung und ihre Daten ausreicht.

Mögliche Gründe, Benutzer nicht zu authentifizieren, können sein, dass das Deployment für ein Einzelbenutzer-Gerät erfolgt oder dass es zu umständlich wäre, die Benutzer jedes Mal zu einer Eingabe aufzufordern, wenn sie die Anwendung starten.

Es ist nicht erforderlich, eine Benutzer-ID oder ein Kennwort in der Verbindungszeichenfolge für die Datenbank einzufügen, wenn Sie keine Authentifizierung auf Datenbankebene benötigen. Die einfachste UltraLite Verbindungszeichenfolge ist **DBF=Dateiname**. Im restlichen Teil dieses Abschnitts wird erklärt, wie UltraLite Benutzer-IDs implementiert, und beschrieben, wie Sie sie verwenden, wenn Sie eine explizite Benutzerauthentifizierung benötigen.

Bei der Entwicklung einer UltraLite-Anwendung mit angepasster Schnittstelle für die Benutzerauthentifizierung können Sie die in einer UltraLite-Datenbank gespeicherten UltraLite-Benutzer-IDs und Kennwort-Hashwerte effektiv dazu verwenden, vom Benutzer übergebene Anmeldeinformationen zu validieren, und so die Erstellung eines eigenen Kennwort-Hash-Algorithmus vermeiden. Durch Hinzufügen von Benutzern zu Ihrer UltraLite-Datenbank speichern Sie ihre Benutzer-IDs und Kennwort-Hashwerte. Anschließend können Sie die vom Benutzer angegebenen Anmeldeinformationen in Ihrer Anwendung validieren, indem Sie versuchen, eine Verbindung mit der Datenbank mit den Verbindungsparametern **UID** und **PWD** herzustellen und dabei **UID=Benutzername** und **PWD=Kennwort** zu verwenden. Eine erfolgreiche UltraLite-Datenbankverbindung zeigt an, dass der Benutzer authentisch ist.

Vorsicht

Im Gegensatz zu SQL Anywhere-Benutzern werden UltraLite-Datenbankbenutzer nur für den Zweck der Authentifizierung erstellt und verwaltet und nicht als Eigentümer von Objekten oder für bestimmte Datenbankrollen und -privilegien. Nachdem Benutzer authentifiziert wurden, erhalten sie vollen Zugriff auf die Datenbank.

Durch das Erstellen von Benutzer-IDs und Kennwörtern kontrollieren Sie Verbindungen mit der UltraLite-Datenbank, aber sichern keine Daten in der Datenbankdatei. Die Inhalte werden als normaler Text gespeichert und können direkt gelesen werden.

Um den Datenbankinhalt zu sichern, wird empfohlen, die Datei zu verschlüsseln. Beim Verschlüsseln der Datei können Sie Benutzer mit einem Chiffrierschlüssel und nicht einer Benutzer-ID und einem Kennwort authentifizieren.

Sie können die Datei verschleiern, um die Speicherung zu ändern, damit Daten nicht als normaler Text gespeichert werden. Diese Methode führt aber nicht zu sicheren Daten.

Weitere Hinweise finden Sie unter „Datenbanksicherheit“ auf Seite 29 und „UltraLite-Verbindungsparameter DBKEY“ auf Seite 179.

Hinweis

UltraLite-Benutzer-IDs unterscheiden sich von MobiLink-Benutzernamen.

Einschränkungen

Benutzer-IDs und Kennwörter werden in UltraLite Java Edition-Datenbanken nicht gespeichert. Diese Datenbanken werden mit einem einzigen Kennwort gesichert und erfordern keine Benutzererstellung oder Authentifizierung. Weitere Hinweise finden Sie unter „UltraLite- und UltraLite Java Edition-Datenbank, Methoden zum Erstellen und Verbinden“ [*UltraLite® – Java-Programmierung*].

Die folgenden Einschränkungen gelten für UltraLite-Benutzer-IDs:

- UltraLite unterstützt bis zu vier eindeutige Benutzer-IDs pro Datenbank.
- Benutzer-IDs und Kennwörter können mithilfe von Sybase Central, SQL-Anweisungen oder UltraLite-API-Methoden in Ihrer Anwendung geändert werden.
- Benutzer-IDs sind auf 31 Zeichen begrenzt.
- Die Benutzer-IDs dürfen keine führenden Apostrophe ('), Anführungszeichen (") oder Semikolons (;) enthalten.
- Benutzer-IDs berücksichtigen nie die Groß- und Kleinschreibung, während Kennwörter die Groß- und Kleinschreibung immer berücksichtigen.
- Benutzer-IDs können nicht umbenannt werden. Sie können in einer bestehenden Datenbankverbindung nur neue Benutzer-IDs hinzufügen und vorhandene löschen.
- Benutzer können mit den UltraLite-APIs nicht programmatisch aufgelistet werden. Sie können nur Datenbanktools verwenden, um bestehende Benutzer der Datenbank aufzulisten.

- Wenn Sie erstmalig eine Verbindung zu einer UltraLite-Datenbank verwenden, sind die Werte für **UID** und **PWD** dieselben Werte, die eingestellt wurden, als die Datenbank erstellt wurde. UltraLite versucht, eine Verbindung mit der Benutzer-ID **DBA** und dem Kennwort **sql** einzurichten, wenn diese Verbindungsparameter nicht angegeben sind. Es ist nicht erforderlich, bei der Verbindungsaufnahme mit der Datenbank einen Benutzernamen oder ein Kennwort einzugeben, wenn Sie bei der Erstellung Benutzernamen und Kennwort nicht explizit eingerichtet haben.

Siehe auch

- „UltraLite-Verbindungsparameter UID“ auf Seite 190
- „UltraLite-Verbindungsparameter PWD“ auf Seite 186

Verbindungsparameter zum Verwalten von UltraLite-Benutzern

Sie können mit den Verbindungsparametern **UID** und **PWD** Benutzer in einer UltraLite-Datenbank erstellen oder authentifizieren.

Hinweis

Als Alternative zu Verbindungsparametern können Sie die folgenden UltraLite-API-Methoden in Ihrer Anwendung verwenden, um den Benutzerzugriff auf eine UltraLite-Datenbank zu ermöglichen oder zu sperren:

- [ULConnection.GrantConnectTo-Methode \[UltraLite C++\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULConnection.RevokeConnectFrom-Methode \[UltraLite C++\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULGrantConnectTo-Methode \[UltraLite Embedded SQL\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULRevokeConnectFrom-Methode \[UltraLite Embedded SQL\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULConnection.GrantConnectTo-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)
- [ULConnection.RevokeConnectFrom-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)

Methoden zum Erteilen und Entziehen sind in der UltraLiteJ-API nicht verfügbar. Für Android-Smartphones stellen Sie die Verbindungsparameter UID und PWD mit der `ConfigPersistent.setConnectionString`-Methode ein.

Verbindungsparameter mit UltraLite-Datenbanken verwenden

Für die meisten UltraLite-APIs kann die `createDatabase`-Methode eines `DatabaseManager`-Objekts dazu verwendet werden, eine neue UltraLite-Datenbank mit den angegebenen Verbindungs- und Erstellungsparametern zu erstellen.

Das folgende Beispiel zeigt die Vorgehensweise zum Erstellen eines Standardbenutzers für eine neue UltraLite-Datenbank durch Übergabe der Parameter **UID** und **PWD** an die `CreateDatabase`-Methode in der UltraLite C++-API:

```
ULConnection * conn;
ULError ulerr;
```

```
ULDatabaseManager::CreateDatabase("dbf=sample.udb;uid=default-  
name;pwd=default-password", &ulerr);
```

Das folgende Beispiel zeigt die Vorgehensweise, um einen Benutzer in einer vorhandenen UltraLite-Datenbank zu authentifizieren, indem Sie die Parameter **UID** und **PWD** der OpenConnection-Methode in der UltraLite C++-API übergeben:

```
ULConnection * conn;  
ULError ulerr;  
ULDatabaseManager::OpenConnection("dbf=sample.udb;uid=test-name;pwd=test-  
password", &ulerr);
```

Verbindungsparameter mit UltraLite-Datenbanken auf Android-Smartphones verwenden

Für die UltraLiteJ-API können Sie die setConnectionString-Methode eines Configuration-Objekts in der UltraLiteJ-API verwenden, um Benutzer zu erstellen oder zu authentifizieren.

Das folgende Beispiel zeigt die Vorgehensweise zum Erstellen eines Standardbenutzers für eine neue UltraLite-Datenbank durch Übergabe der Parameter **UID** und **PWD** an die createDatabase-Methode in der UltraLiteJ-API:

```
ConfigFile config =  
    DatabaseManager.createConfigurationFileAndroid("DBname.udb",  
    getApplicationContext());  
config.setConnectionString("uid=default-name;pwd=default-password");  
Connection conn = DatabaseManager.createDatabase(config);
```

Das folgende Beispiel zeigt die Vorgehensweise, um einen Benutzer in einer vorhandenen UltraLite-Datenbank zu authentifizieren, indem Sie die Parameter **UID** und **PWD** an die connect-Methode in der UltraLiteJ-API übergeben:

```
ConfigFile config =  
    DatabaseManager.createConfigurationFileAndroid("DBname.udb",  
    getApplicationContext());  
config.setConnectionString("uid=test-name;pwd=test-password");  
Connection conn = DatabaseManager.connect(config);
```

Als Alternative zur setConnectionString-Methode können Sie setPassword oder setUsername verwenden, um einen Benutzer zu erstellen oder zu authentifizieren.

Erstellen oder Authentifizieren von Benutzern in einer UltraLite Java Edition-Datenbank

UltraLite Java Edition-Datenbanken unterstützen keine Benutzerauthentifizierung. Stattdessen richten Sie mit der Configuration.setPassword-Methode ein Datenbankkennwort ein.

Das folgende Beispiel veranschaulicht, wie Sie das Kennwort für eine vorhandene UltraLite Java Edition-Datenbank einrichten:

```
ConfigNonPersistent config =  
    DatabaseManager.createConfigurationNonPersistent("DBname.ulj");  
config.setPassword("my_password");  
Connection conn = DatabaseManager.connect(config);
```

Siehe auch

- [ConfigPersistent.setUserName-Methode \[Android\] \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Configuration.setPassword-Methode \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- „UltraLite-Verbindungsparameter UID“ auf Seite 190
- „UltraLite-Verbindungsparameter PWD“ auf Seite 186
- [ULDatabaseManager.CreateDatabase-Methode \[UltraLite C++\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULDatabaseManager.CreateDatabase-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)
- [ConfigPersistent.setConnectionString-Methode \[Android\] \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [DatabaseManager.createDatabase-Methode \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)

SQL-Anweisungen für die Verwaltung von UltraLite-Benutzern

Mit den Anweisungen CREATE USER, ALTER USER und DROP USER können Sie Benutzer in einer UltraLite-Datenbank verwalten.

Es ist nicht möglich, diese SQL-Anweisungen mit UltraLite Java Edition-Datenbanken zu verwenden.

Hinweis

Als Alternative zu SQL-Anweisungen können Sie die folgenden UltraLite-API-Methoden in Ihrer Anwendung verwenden, um den Benutzerzugriff auf eine UltraLite-Datenbank zu ermöglichen oder zu sperren:

- [ULConnection.GrantConnectTo-Methode \[UltraLite C++\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULConnection.RevokeConnectFrom-Methode \[UltraLite C++\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULGrantConnectTo-Methode \[UltraLite Embedded SQL\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULRevokeConnectFrom-Methode \[UltraLite Embedded SQL\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULConnection.GrantConnectTo-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)
- [ULConnection.RevokeConnectFrom-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)

Methoden zum Erteilen und Entziehen sind in der UltraLiteJ-API nicht verfügbar. Für Android-Smartphones erstellen Sie eine CREATE USER-, ALTER USER- oder DROP USER-Anweisung als Zeichenfolgenvariable und übergeben sie an die Connection.prepareStatement-Methode.

Beispiel

Im folgenden Beispiel wird gezeigt, wie Sie auf einem Android-Smartphone eine Verbindung zu einer vorhandenen UltraLite-Datenbank mithilfe der UltraLiteJ API herstellen und die CREATE USER-Anweisung verwenden, um einen neuen Benutzer zu erstellen:

```
ConfigFile config =
    DatabaseManager.createConfigurationFileAndroid("DBname.udb",
    getApplicationContext());
```

```
Connection conn = DatabaseManager.connect(config);

String sql_string = "CREATE USER test-user IDENTIFIED BY test-password";
PreparedStatement authenticator = conn.prepareStatement(sql_string);
authenticator.execute();
authenticator.close();
```

Siehe auch

- „CREATE USER-Anweisung [UltraLite]“ auf Seite 444
- „ALTER USER-Anweisung [UltraLite]“ auf Seite 431
- „DROP USER-Anweisung [UltraLite]“ auf Seite 449
- ULPreparedStatement-Klasse [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULConnection.PrepareStatement-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULCommand-Klasse [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- ULCommand.Prepare-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- PreparedStatement-Schnittstelle [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- Connection.prepareStatement-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

UltraLite-Benutzer mit Sybase Central erstellen

Verwenden Sie Sybase Central, um Benutzer für eine UltraLite-Datenbank zu erstellen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Sybase Central ist nicht kompatibel mit UltraLite Java Edition-Datenbanken.

Hinweis

Als Alternative zu Sybase Central können Sie die folgenden UltraLite-API-Methoden in Ihrer Anwendung verwenden, um den Benutzerzugriff auf eine UltraLite-Datenbank zu erteilen:

- ULConnection.GrantConnectTo-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULGrantConnectTo-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULConnection.GrantConnectTo-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Methoden zum Erteilen sind in der UltraLiteJ API nicht verfügbar. Für Android-Smartphones erstellen Sie eine CREATE USER-Anweisung als Zeichenfolgenvariable und übergeben sie an die Connection.prepareStatement-Methode.

Aufgabe

1. Stellen Sie mithilfe von Sybase Central eine Verbindung mit der UltraLite-Datenbank her.
2. Rechtsklicken Sie auf den Ordner **Benutzer** und klicken Sie auf **Neu » Benutzer**.

3. Befolgen Sie die Anweisungen des Assistenten.

Ergebnisse

Der neue Benutzer wird erstellt.

Siehe auch

- [„CREATE USER-Anweisung \[UltraLite\]“ auf Seite 444](#)

Einen UltraLite-Benutzer mit Sybase Central löschen

Verwenden Sie Sybase Central, um Benutzer explizit aus einer UltraLite-Datenbank zu löschen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Sybase Central ist nicht kompatibel mit UltraLite Java Edition-Datenbanken.

Hinweis

Als Alternative zu Sybase Central können Sie die folgenden UltraLite-API-Methoden in Ihrer Anwendung verwenden, um den Benutzerzugriff auf eine UltraLite-Datenbank zu entziehen:

- [ULConnection.RevokeConnectFrom-Methode \[UltraLite C++\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULRevokeConnectFrom-Methode \[UltraLite Embedded SQL\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULConnection.RevokeConnectFrom-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)

Methoden zum Entziehen sind in der UltraLiteJ API nicht verfügbar.

Aufgabe

1. Stellen Sie mithilfe von Sybase Central eine Verbindung mit der UltraLite-Datenbank her.
2. Doppelklicken Sie im linken Fensterausschnitt auf den Ordner **Benutzer**.
3. Rechtsklicken Sie auf den Benutzer und klicken Sie auf **Löschen**.

Ergebnisse

Der Benutzer wird aus der Datenbank gelöscht.

Siehe auch

- [„DROP USER-Anweisung \[UltraLite\]“ auf Seite 449](#)

UltraLite als MobiLink-Client

Dieser Abschnitt beschreibt, wie Sie UltraLite-Clients für die MobiLink-Synchronisation einrichten und verwenden.

UltraLite-Clients

Zum Synchronisieren einer UltraLite-Datenbank muss die Anwendung die Synchronisationsparameter einstellen, die die Adresse des MobiLink-Servers sowie andere erforderliche Informationen angeben und eine Synchronisationsfunktion aufrufen bzw. die SQL-Anweisung SYNCHRONIZE ausführen. Die Option, die Sie wählen, richtet sich nach der von Ihnen verwendeten API.

Anpassen des Verhaltens der UltraLite-Clientsynchronisation

Wenn Sie UltraLite für eine benutzerdefinierte Synchronisation einrichten möchten, müssen Sie mehrere wichtige Grundsätze beachten:

- **Eindeutigkeit der Primärschlüssel in Synchronisationsmodellen bewahren, die mehr als einen entfernten Client enthalten** Erforderlich. In einem Synchronisationssystem können identische Zeilen in unterschiedlichen Datenbanken (entfernt und konsolidiert) nur über den Primärschlüssel ermittelt werden. Zudem bietet er die einzige Möglichkeit zur Konflikterkennung. Aus diesem Grund müssen bei mehreren Clients folgende Regeln eingehalten werden:
 - Jede zu synchronisierende Tabelle muss einen Primärschlüssel haben.
 - Aktualisieren Sie nie die Werte von Primärschlüsseln.
 - Primärschlüssel müssen für alle synchronisierten Datenbanken eindeutig sein.
- **Datenspalten so einrichten, dass Teildaten nicht verloren gehen** Für eine konsolidierte SQL Anywhere-Datenbank ist dies normalerweise kein Problem. Bei Datenbanken wie beispielsweise Oracle können jedoch Kompatibilitätsprobleme auftreten, die berücksichtigt werden müssen. UltraLite- und Oracle-Datenbanken müssen beispielsweise die gleiche Zeitstempelpräzision haben. Außerdem sollten Sie der Oracle-Datenbank einen TIMESTAMP hinzufügen, damit keine Sekundenbruchteile verloren gehen, wenn die entfernte UltraLite-Datenbank einen Upload von Daten in die konsolidierte Datenbank durchführt.
- **Beschreiben, welche Datenteilmengen in die konsolidierte Datenbank übertragen werden sollen** Optional: Sie brauchen dies nur dann durchzuführen, wenn Sie nicht standardmäßig alle Daten synchronisieren wollen. Um die zu synchronisierenden Daten einzugrenzen, können Sie eine der folgenden Techniken für die Erstellung von Teilmengen verwenden.

Sie könnten z.B. eine Publikation für Daten mit hoher Priorität erstellen. Die Anwendung kann dann diese Daten über drahtlose Netzwerke synchronisieren. Da die Nutzung drahtloser Netzwerke oft sehr teuer ist, können Sie die Kosten niedrig halten, indem Sie nur geschäftskritische Daten übertragen. Später synchronisieren Sie dann weniger dringende Daten über eine Dockingstation.

- **Synchronisation von der UltraLite-Anwendung einleiten und Parameter übertragen, die die Sitzung beschreiben** Erforderlich. Die Programmierung der Synchronisation erfolgt in zwei Teilen: Beschreibung der Sitzung und danach Beginn des Synchronisationsvorgangs.

Die Beschreibung der Sitzung umfasst im Wesentlichen die Auswahl eines Kommunikationsdatenstroms für die Synchronisation (auch als Netzwerkprotokoll bezeichnet) und der zugehörigen Parameter. Hierzu konfigurieren Sie Ihre Synchronisationsskripten und geben den MobiLink-Benutzer an. Allerdings können Sie weitere Parameter konfigurieren. Dazu gehören beispielsweise die `upload_only`- und `download_only`-Parameter, um die vorgegebene Zwei-Wege-Synchronisation in eine Einweg-Synchronisation umzustellen.

Alle anderen wichtigen Synchronisationseinstellungen werden über den MobiLink-Server mit MobiLink-Synchronisationsskripten gesteuert. Es handelt sich dabei um die Folgenden:

- Welche Daten im Download als Aktualisierungen oder Einfügungen in die Tabellen des entfernten UltraLite-Clients übertragen werden
- Welche Verarbeitung für im Upload übertragene Änderungen aus einer entfernten Datenbank erforderlich ist

Sie können Ihre Synchronisationsskripten so schreiben, dass die Daten auf geeignete Weise zwischen den entfernten Datenbanken verteilt werden.

Siehe auch

- „Eindeutige Primärschlüssel“ [[MobiLink - Serveradministration](#)]
- „Eindeutigkeit des Primärschlüssels in UltraLite“ auf Seite 70
- „Konsolidierte Oracle-Datenbank“ [[MobiLink - Serveradministration](#)]
- „UltraLite-Erstellungsparameter `precision`“ auf Seite 157
- „UltraLite-Clientsynchronisationsplanung“ auf Seite 74
- „Synchronisationseinrichtung für Ihre UltraLite-Anwendung“ auf Seite 81
- „MobiLink - konsolidierte Datenbanken“ [[MobiLink - Serveradministration](#)]
- „Schreiben von Synchronisationsskripten“ [[MobiLink - Serveradministration](#)]
- „Direkte Zeilenbehandlung“ [[MobiLink - Serveradministration](#)]
- „Partitionierte Zeilen in entfernten Datenbanken“ [[MobiLink - Serveradministration](#)]

Eindeutigkeit des Primärschlüssels in UltraLite

UltraLite kann die Eindeutigkeit des Primärschlüssels durch eine der von MobiLink unterstützten Techniken bewahren.

Eine dieser Methoden besteht in der Verwendung einer GLOBAL AUTOINCREMENT-Spalte. GLOBAL AUTOINCREMENT ähnelt AUTOINCREMENT, außer dass die Domäne partitioniert ist. UltraLite liefert Spaltenwerte nur aus dem Wertebereich, der der globalen Datenbank-ID der Datenbank zugewiesen ist. Jeder UltraLite-Datenbank ist eine eindeutige Ganzzahl als globale Datenbank-ID zugeordnet.

Eine zweite Methode ist die Verwendung einer UUID-Primärschlüsselspalte. Eine UUID erfordert mehr Daten, aber bedarf keines unterschiedlichen Datenbankbezeichners.

Siehe auch

- „Eindeutige Primärschlüssel“ [[MobiLink - Serveradministration](#)]
- „UltraLite-Option `global_database_id`“ auf Seite 198

GLOBAL AUTOINCREMENT-Spalten in UltraLite deklarieren

Sie können den Standardwert einer Spalte in einer UltraLite-Datenbank mit dem Typ GLOBAL AUTOINCREMENT deklarieren. Bevor Sie diese Spalten-IDs autoinkrementieren können, müssen Sie die globale Datenbank-ID für die entfernte UltraLite-Datenbank einrichten.

Vorsicht

Global autoincrement-Spaltenwerte, die über die MobiLink-Synchronisation heruntergeladen wurden, aktualisieren den global autoincrement-Wertzähler nicht. Daher kann ein Fehler auftreten, wenn ein MobiLink-Client einen Wert einfügt, der im Wertebereich eines anderen Clients liegt. Um dieses Problem zu vermeiden, stellen Sie sicher, dass jede Kopie Ihrer UltraLite-Anwendung nur Werte in ihrem eigenen Wertebereich einfügt.

Die Datenbankoption `global_database_id` erlaubt Ihnen, den Wert in Ihrer UltraLite-Datenbank einzustellen. Wenn Sie das Deployment von UltraLite durchführen, müssen Sie jeder Datenbank eine andere Identifizierungsnummer zuordnen.

UltraLite stellt dann Standardwerte für die Spalte bereit, die aus dem Wertebereich stammen, der durch die Nummer der UltraLite-Datenbank bestimmt wird.

UltraLite folgt diesen Regeln:

- Wenn die Spalte keine Werte aus dem aktuellen Wertebereich enthält, ist der erste Standardwert $pn + 1$, wobei p für die Partitionsgröße und n für den Wert der globalen ID-Nummer steht.
- Wenn die Spalte bereits Werte im aktuellen Wertebereich enthält, die alle kleiner als $p(n + 1)$ sind, wird der nächste Standardwert um eins größer als der bisherige Höchstwert in diesem Bereich.
- Standardspaltenwerte sind von Werten, die sich außerhalb der aktuellen Partition befinden, nicht betroffen, d.h. von Nummern kleiner als $pn + 1$ oder größer als $p(n + 1)$. Solche Werte können vorhanden sein, wenn sie von einer anderen Datenbank über MobiLink-Synchronisation repliziert wurden.

Wenn Sie Ihrer UltraLite-Datenbank beispielsweise die globale ID 1 zugeordnet haben und die Partitionsgröße 1000 ist, werden die Standardwerte in dieser Datenbank aus dem Bereich 1001–2000 ausgewählt. Eine andere Kopie der Datenbank mit der Kennnummer 2 liefert Standardwerte für dieselbe Spalte im Bereich 2001–3000.

- Da Sie die globale ID-Nummer nicht auf einen negativen Wert setzen können, sind die Werte in der GLOBAL AUTOINCREMENT-Spalte immer positiv. Die maximale Identifizierungsnummer wird nur durch den Spaltentyp und die Partitionsgröße beschränkt.
- Wenn Sie keinen globalen ID-Wert einrichten oder die Werte aus der Partition erschöpft sind, wird NULL in die Spalte eingefügt. Falls NULL nicht zulässig ist, bewirkt der Versuch einer Einfügung in die Zeile einen Fehler.

Wenn die verfügbaren Werte für die als GLOBAL AUTOINCREMENT deklarierten Spalten fast oder vollständig aufgebraucht sind, müssen Sie eine neue globale Datenbank-ID festlegen. UltraLite wählt GLOBAL AUTOINCREMENT-Werte aus der Partition, die durch die globale ID-Nummer angegeben wird, aber nur bis der maximale Wert erreicht ist. Wenn die Werte erschöpft sind, beginnt UltraLite mit der Generierung von NULL. Durch die Zuweisung einer neuen globalen Datenbank-ID-Nummer ermöglichen Sie es UltraLite, geeignete Werte aus einem anderen Wertebereich zu benutzen.

Eine Methode für die Auswahl einer neuen globalen Datenbank-ID besteht darin, einen Pool von nicht verwendeten globalen Datenbank-ID-Werten einzurichten. Dieser Pool wird auf dieselbe Weise wie ein Pool von Primärschlüsseln verwaltet.

Tipp

Die UltraLite-APIs bieten die Möglichkeit, den Anteil der bereits verwendeten Nummern festzustellen. Der Rückgabewert ist ein SHORT-Datentyp im Bereich 0–100, der die Prozentwerte der bislang verwendeten Anzahl von Nummern angibt. Beispiel: Der Wert "99" zeigt an, dass nur sehr wenige noch nicht verwendete Nummern übrig sind und der Datenbank eine neue Identifizierungsnummer zugewiesen werden sollte. Die Methode, diese Identifizierungsnummer festzulegen, hängt davon ab, welche Programmierschnittstelle Sie verwenden.

Siehe auch

- „UltraLite-Option `global_database_id`“ auf Seite 198
- „Primärschlüsselpools“ [*SQL Remote*]
- „Partitionsgrößen“ auf Seite 73
- „UltraLite-Option `global_database_id`“ auf Seite 198
- `ULCommand.Connection`-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- `ULConnection.Synchronize`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- `ULSetDatabaseID`-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- `ULConnection.GlobalAutoIncrementUsage`-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- `ULGlobalAutoincUsage`-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- `Connection.setDatabaseId`-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- `Connection.getDatabaseId`-Methode [BlackBerry] [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Methoden zum Finden des zuletzt zugewiesenen GLOBAL AUTOINCREMENT-Werts

Sie können den global autoincrement-Wert ermitteln, der beim letzten Einfügevorgang gewählt wurde. Da diese Werte häufig für Primärschlüssel verwendet werden, ist es günstig zu wissen, welcher Wert generiert wurde, wenn Sie Zeilen einfügen, die den Primärschlüssel der ersten Zeile referenzieren. Sie können den Wert folgendermaßen überprüfen:

- **UltraLite für C/C++** Rufen Sie die Funktion `GetLastIdentity` für das Objekt `ULConnection` auf.
- **UltraLite.NET** Rufen Sie die Eigenschaft `LastIdentity` für das Objekt `ULConnection` auf.

- **API UltraLiteJ** Verwenden Sie die Methode `getLastIdentity` für die Connection-Schnittstelle.

Der Rückgabewert ist eine 64-Bit-Ganzzahl ohne Vorzeichen vom Datenbanktyp UNSIGNED BIGINT. Da Sie mit dieser Anweisung nur den zuletzt zugewiesenen Standardwert erfahren, sollten Sie diesen Wert möglichst bald nach der Ausführung der INSERT-Anweisung abrufen, um falsche Ergebnisse zu vermeiden.

Hinweis

Eine einzelne INSERT-Anweisung kann gegebenenfalls mehrere Spalten vom Typ GLOBAL AUTOINCREMENT umfassen. In diesem Fall ist der Rückgabewert einer der generierten Standardwerte, doch es gibt keine zuverlässige Methode, um festzustellen, welcher der Werte es ist. Aus diesem Grund sollten Sie Ihre Datenbank so planen und Ihre INSERT-Anweisungen so schreiben, dass diese Situation vermieden wird.

Siehe auch

- [ULConnection.GetLastIdentity-Methode \[UltraLite C++\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULConnection.LastIdentity-Eigenschaft \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)
- [Connection.getLastIdentity-Methode \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)

Partitionsgrößen

Die Partitionsgröße kann jede positive Ganzzahl sein, obwohl dieser Wert normalerweise so eingeteilt wird, dass seine Größe kaum jemals überschritten werden kann.

Bei Spalten vom Typ INT oder UNSIGNED INT beträgt die Standard-Partitionsgröße $2^{16} = 65536$; bei Spalten anderen Typs ist die Standard-Partitionsgröße $2^{32} = 4294967296$. Da diese Standardwerte nicht immer sinnvoll sind, empfiehlt es sich, die Partitionsgröße explizit festzulegen.

Standard-Partitionsgrößen für einige Datentypen von UltraLite-Anwendungen unterscheiden sich von SQL Anywhere-Datenbanken. Deklarieren Sie die Partitionsgröße explizit, wenn Sie die Konsistenz unterschiedlicher Datenbanken erhalten möchten.

Siehe auch

- [„CREATE TABLE-Anweisung \[UltraLite\]“ auf Seite 438](#)
- [„ALTER TABLE-Anweisung \[UltraLite\]“ auf Seite 427](#)
- [ULTableSchema.GetColumnPartitionSize-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)
- [ULTableSchema.GetGlobalAutoincPartitionSize-Methode \[UltraLite C++\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULGlobalAutoincUsage-Methode \[UltraLite Embedded SQL\] \[UltraLite - C- und C++-Programmierung\]](#)
- [Connection.getLastIdentity-Methode \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)

Partitionsgröße für eine GLOBAL AUTOINCREMENT-Spalte überschreiben

Das Erhöhen der Partitionsgröße für eine GLOBAL AUTOINCREMENT-Spalte stellt sicher, dass der Vorrat an Zahlen in der Partition kaum aufgebraucht wird.

Voraussetzungen

Sie müssen mit einer UltraLite-Datenbank verbunden sein.

Aufgabe

1. Erstellen Sie eine Tabelle mit einer GLOBAL AUTOINCREMENT-Spalte, deren Partitionsgröße in Klammern angegeben wird.

Führen Sie den folgenden SQL-Code aus:

```
CREATE TABLE customer (  
  id    INT          DEFAULT GLOBAL AUTOINCREMENT (5000),  
  name  VARCHAR(128) NOT NULL,  
  PRIMARY KEY (id)  
)
```

Eine einfache Referenztabelle mit zwei Spalten wird erstellt: einer Ganzzahl, die eine Kunden-Identifizierungsnummer angibt, und einer Zeichenfolge, die den Namen des Kunden enthält. Die ID hat eine Partitionsgröße von 5000.

2. Stellen Sie in Sybase Central eine Verbindung mit der UltraLite-Datenbank her.
3. Rechtsklicken Sie auf die ID-Spalte der customer-Tabelle und klicken Sie auf **Eigenschaften**.
4. Klicken Sie auf die Registerkarte **Wert**.
5. Geben Sie eine positive Ganzzahl, die größer als 5000 ist, im Feld **Partitionsgröße** ein.

Ergebnisse

Die Partitionsgröße der ID-Spalte wird entsprechend des Werts aktualisiert, der im Feld **Partitionsgröße** eingegeben wurde.

UltraLite-Clientsynchronisationsplanung

Alle Daten in einer UltraLite-Datenbank werden standardmäßig synchronisiert. Wenn Sie mit dem Deployment von UltraLite als entfernte MobiLink-Datenbank noch nicht vertraut sind, sollten Sie zunächst das Standardverhalten verwenden.

Wenn Sie mit dem Synchronisationsprozess besser vertraut sind, können Sie entscheiden, das Verhalten des Synchronisationsvorgangs anzupassen, damit eine komplexere Geschäftslogik verarbeitet werden kann. Für die Planung eines benutzerdefinierten Synchronisationsverhaltens müssen Sie zunächst folgende Fragen klären: Wenn Ihre Geschäftsanforderungen einfach sind, brauchen Sie möglicherweise nur ein einzelnes Synchronisationsmerkmal. Bei komplexen Deployments müssen Sie indessen mehrere Synchronisationsfunktionen verwenden, um das gewünschte Synchronisationsverhalten zu konfigurieren.

Designfragen	Wenn Ihre Antwort "Ja" lautet, gehen Sie wie folgt vor:
Wollen Sie einen Download der Änderungen aus der konsolidierten Datenbank durchführen, aber es soll kein Upload lokaler Änderungen in die Datenbank erfolgen?	Das Suffix <code>download_only</code> im Tabellennamen ermöglicht die Identifizierung von Tabellen, für die eine Synchronisation nur per Download erfolgen soll. An der lokalen Tabelle vorgenommene Änderungen werden nicht in die konsolidierte Datenbank übertragen.
Wollen Sie Tabellen aus der Synchronisation ausschließen?	Das Tabellennamensuffix <code>"nosync"</code> ermöglicht die Identifizierung von Tabellen, die nicht synchronisiert werden sollen.
Wollen Sie nur komplette Tabellen synchronisieren, auch wenn sich die Daten nicht geändert haben?	Das Tabellennamensuffix <code>"allsync"</code> ermöglicht Ihnen die Synchronisation der kompletten Tabelle, auch wenn keine Änderungen erkannt werden.
Wollen Sie eine komplette Tabelle oder nur Zeilen synchronisieren, auf die bestimmte Bedingungen zutreffen? Benötigen bestimmte Daten eine Synchronisationspriorität wegen ihrer Bedeutung oder Dringlichkeit?	<p>Eine Publikation enthält Artikel, in denen die Tabellen aufgelistet werden, für die eine Synchronisation erforderlich ist. Ein Artikel kann eine WHERE-Klausel enthalten, die die Zeilen für den Upload nach Maßgabe bestimmter Kriterien angibt.</p> <p>Mehrere Publikationen können die Prioritätsvorgaben steuern, sodass bestimmte UltraLite-Daten vor den anderen im Upload übertragen werden.</p>
Benötigen Sie eine Tabellenreihenfolge für die Synchronisation, weil Sie mit Fremdschlüsselschleifen arbeiten?	Mit dem Synchronisationsparameter <code>TableOrder</code> können Sie die Reihenfolge der Synchronisationsvorgänge festlegen, wenn Sie mit zyklischen Fremdschlüsseln arbeiten. Für UltraLite werden aber zyklische Fremdschlüssel grundsätzlich nicht empfohlen.
Wollen Sie das Synchronisationsverhalten steuern? Zum Beispiel: Wollen Sie, dass Downloads zu demselben Zeitpunkt vorgenommen werden wie Uploads? Oder möchten Sie die Zwei-Wege-Synchronisation in eine Einweg-Synchronisation verwandeln?	<p>Benutzen Sie den geeigneten Synchronisationsparameter in folgenden Bereichen:</p> <ul style="list-style-type: none"> • In der Synchronisationsstruktur Ihrer Anwendung (oder in der Synchronisations-Enumeration) • Mit der Option <code>-e</code> des <code>ulsync</code>-Dienstprogramms

Designfragen	Wenn Ihre Antwort "Ja" lautet, gehen Sie wie folgt vor:
Soll der UltraLite-Client TLS-fähig sein?	Von dem von Ihnen gewählten Verschlüsselungsalgorithmus ist es abhängig, wie Ihr Gerät entsprechend der Plattform eingerichtet werden muss, die auf dem Gerät läuft.

Siehe auch

- „UltraLite-Tabellen für reinen Download“ auf Seite 77
- „Nicht synchronisierende Tabellen in UltraLite“ auf Seite 76
- „UltraLite-Tabellen für Allsync (Alles synchronisieren)“ auf Seite 78
- „Publizieren von Daten in UltraLite“ auf Seite 78
- „Tabellenreihenfolge in UltraLite“ auf Seite 80
- „UltraLite-Synchronisationsparameter“ auf Seite 90
- „UltraLite-Netzwerkprotokolloptionen für dbmlsync“ auf Seite 114
- „Der Synchronisationsprozess“ [*MobiLink - Erste Orientierung*]
- Upload und Download [*MobiLink - Erste Orientierung*]

Nicht synchronisierende Tabellen in UltraLite

Durch Erstellung der Tabelle unter Verwendung von SYNCHRONIZE OFF steuern Sie den Zeitpunkt zum Ausschluss der gesamten Tabelle aus dem Upload-Vorgang. Sie können diese nicht synchronisierten Tabellen für kundenspezifische dauerhafte Daten verwenden, die in der konsolidierten Datenbank nicht erforderlich sind. Abgesehen davon, dass sie von der Synchronisation ausgeschlossen sind, können Sie sie in genau der gleichen Weise wie andere Tabellen in der UltraLite-Datenbank benutzen.

Hinweis

Der Synchronisationstyp für eine Tabelle kann nur geändert werden, wenn keine nicht synchronisierten Änderungen vorhanden sind, die hochgeladen werden müssen.

Wenn Sie eine Tabelle mit dem Suffix `_nosync` erstellen, muss diese Tabelle beim Umbenennen das Suffix `_nosync` beibehalten. Die folgende Anweisung `ALTER TABLE` mit einer Klausel zum Umbenennen ist beispielsweise nicht zulässig, da der neue Name nicht auf `_nosync` endet:

```
ALTER TABLE purchase_comments_nosync
RENAME comments
```

Diese Anweisung muss umgeschrieben werden, sodass das Suffix enthalten bleibt:

```
ALTER TABLE purchase_comments_nosync
RENAME comments_nosync
```

Alternativ dazu können Sie Publikationen verwenden, um den gleichen Effekt zu erreichen.

Hinweis

Als Alternative zum Erstellen oder Ändern einer Tabelle mit der SYNCHRONIZE OFF-Klausel können Sie die Phrase **_nosync** auf den Tabellennamen anwenden, um die Tabelle in eine nicht synchronisierte Tabelle zu verwandeln.

Siehe auch

- „Synchronisationsparameter Download Only“ auf Seite 96
- „Synchronisationsparameter Upload Only“ auf Seite 110
- „Synchronisationsparameter Additional Parameters“ auf Seite 91
- „CREATE TABLE-Anweisung [UltraLite]“ auf Seite 438
- „ALTER TABLE-Anweisung [UltraLite]“ auf Seite 427
- „Publikationen“ [*MobiLink - Clientadministration*]
- „UltraLite-Publikationen“ auf Seite 85
- „CREATE PUBLICATION-Anweisung [UltraLite]“ auf Seite 435
- „Schreiben von Synchronisationsskripten“ [*MobiLink - Serveradministration*]

UltraLite-Tabellen für reinen Download

Durch Erstellung der Tabelle unter Verwendung von SYNCHRONIZE DOWNLOAD schließen Sie ganze Tabelle aus dem Upload-Vorgang aus. Sie können diese Tabellen für Daten verwenden, die nicht in der konsolidierten Datenbank synchronisiert werden sollen. Abgesehen davon, dass sie von der Synchronisation ausgeschlossen sind, können Sie sie in genau der gleichen Weise wie andere Tabellen in der UltraLite-Datenbank benutzen.

Alternativ dazu können Sie Publikationen verwenden, um den gleichen Effekt zu erreichen.

Hinweis

Der Synchronisationstyp für eine Tabelle kann nur geändert werden, wenn keine nicht synchronisierten Änderungen vorhanden sind, die hochgeladen werden müssen.

Hinweis

Als Alternative zum Erstellen oder Ändern einer Tabelle mit der SYNCHRONIZE DOWNLOAD-Klausel können Sie die Phrase **_download_only** auf den Tabellennamen anwenden, um die Tabelle in eine reine Download-Tabelle zu verwandeln.

Siehe auch

- „Synchronisationsparameter Download Only“ auf Seite 96
- „Synchronisationsparameter Upload Only“ auf Seite 110
- „Synchronisationsparameter Additional Parameters“ auf Seite 91
- „CREATE TABLE-Anweisung [UltraLite]“ auf Seite 438
- „ALTER TABLE-Anweisung [UltraLite]“ auf Seite 427
- „Publikationen“ [*MobiLink - Clientadministration*]
- „UltraLite-Publikationen“ auf Seite 85
- „CREATE PUBLICATION-Anweisung [UltraLite]“ auf Seite 435
- „Schreiben von Synchronisationsskripten“ [*MobiLink - Serveradministration*]

UltraLite-Tabellen für Allsync (Alles synchronisieren)

Durch Erstellung oder Ändern der Tabelle unter Verwendung von SYNCHRONIZE OFF steuern Sie, ob das Synchronisationsverhalten während des Uploads geändert wird, sodass alle Tabellendaten synchronisiert werden, auch wenn sich seit der letzten Synchronisationssitzung nichts geändert hat.

Hinweis

Der Synchronisationstyp für eine Tabelle kann nur geändert werden, wenn keine nicht synchronisierten Änderungen vorhanden sind, die hochgeladen werden müssen.

Einige UltraLite-Anwendungen erfordern anwender- oder clientspezifische Daten, die in SYNCHRONIZE ALL-Tabellen gespeichert werden können. Sie können die Daten in der Tabelle in eine temporäre Tabelle der konsolidierten Datenbank hochladen und die Daten zum Steuern der Synchronisation durch Ihre anderen Skripten verwenden, ohne dass die Daten in der konsolidierten Datenbank verwaltet werden müssen. Sie könnten zum Beispiel verlangen, dass Ihre UltraLite-Anwendungen anzeigen, an welchen Kanälen oder Themen sie interessiert sind, und diese Informationen zum Download der entsprechenden Zeilen verwenden.

Hinweis

Als Alternative zum Erstellen oder Ändern einer Tabelle mit der SYNCHRONIZE ALL-Klausel können Sie die Phrase **_allsync** auf den Tabellennamen anwenden, um die Tabelle in eine "Alles synchronisieren"-Tabelle zu verwandeln.

Siehe auch

- „Synchronisationsparameter Download Only“ auf Seite 96
- „Synchronisationsparameter Upload Only“ auf Seite 110
- „Synchronisationsparameter Additional Parameters“ auf Seite 91
- „CREATE TABLE-Anweisung [UltraLite]“ auf Seite 438
- „ALTER TABLE-Anweisung [UltraLite]“ auf Seite 427
- „Publikationen“ [*MobiLink - Clientadministration*]
- „UltraLite-Publikationen“ auf Seite 85
- „CREATE PUBLICATION-Anweisung [UltraLite]“ auf Seite 435
- „Schreiben von Synchronisationsskripten“ [*MobiLink - Serveradministration*]

Publizieren von Daten in UltraLite

Publikationen legen eine Gruppe von Artikeln fest, die die zu synchronisierenden Daten beschreiben. Sie können Publikationen sowohl mit Sybase Central als auch mit SQL zu einer UltraLite-Datenbank hinzufügen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Publikationsartikel können aus einer kompletten Tabelle oder einer definierten Untermenge der Daten in einer Tabelle bestehen. Sie können ein optionales Prädikat (eine WHERE-Klausel) einfügen, um eine

Teilmenge von Zeilen in einer gegebenen Tabelle zu definieren. Publikationen sind flexibler als mit SYNCHRONIZE OFF erstellte Tabellen. Um Datenteilmengen einer UltraLite-Datenbank *getrennt* zu synchronisieren, benötigen Sie mehrere Publikationen. Sie können Publikationen mit den Synchronisationsparametern für den reinen Upload oder den reinen Download kombinieren, um Änderungen mit hoher Priorität effektiv zu synchronisieren.

Hinweis

Die maximale Anzahl von Benutzerpublikationen in UltraLite ist 63.

UltraLite-Publikationen unterstützen weder die Definition von Spaltenteilmengen noch die Klausel SUBSCRIBE BY, die in SQL Anywhere zur Verfügung steht. Wenn Spalten in einer UltraLite-Tabelle nicht genau mit Tabellen in einer konsolidierten Datenbank übereinstimmen, können Sie diese Unterschiede mit MobiLink-Skripten beseitigen.

Es ist nicht erforderlich, in einer Publikation die Reihenfolge für die Tabellensynchronisation festzulegen. Wenn die Tabellenfolge für Ihr Deployment wichtig ist, können Sie diese bei der Synchronisation der UltraLite-Datenbank mit dem Synchronisationsparameter TableOrder festlegen.

Aufgabe

1. Stellen Sie mithilfe des UltraLite-Plug-Ins eine Verbindung zur UltraLite-Datenbank her.
2. Rechtsklicken Sie auf den Ordner **Publikationen** und klicken Sie auf **Neu » Publikation**.
3. Geben Sie einen Namen für die neue Publikation ein. Klicken Sie auf **Weiter**.
4. Wählen Sie auf der Registerkarte **Tabellen** aus der Liste **Verfügbare Tabellen** eine Tabelle aus. Klicken Sie auf **Hinzufügen**.

Die Tabelle erscheint in der Liste **Ausgewählte Tabellen** auf der rechten Seite.

5. Fügen Sie weitere Tabellen hinzu.
6. Falls erforderlich, klicken Sie auf die Registerkarte **WHERE-Klauseln**, um die Zeilen anzugeben, die in die Publikation aufgenommen werden sollen. Sie können keine Spalten-Teilmengen angeben.
7. Klicken Sie auf **Fertig stellen**.

Ergebnisse

Die neue Publikation wird erstellt.

Siehe auch

- „UltraLite-Publikationen“ auf Seite 85
- „Synchronisationsparameter Download Only“ auf Seite 96
- „Synchronisationsparameter Upload Only“ auf Seite 110
- „Synchronisationsparameter Additional Parameters“ auf Seite 91
- „Publikationen“ [*MobiLink - Clientadministration*]
- „CREATE PUBLICATION-Anweisung [UltraLite]“ auf Seite 435
- „Schreiben von Synchronisationsskripten“ [*MobiLink - Serveradministration*]

Tabellenreihenfolge in UltraLite

Mit dem Synchronisationsparameter `TableOrder` können Sie die Reihenfolge der Synchronisationsvorgänge steuern. Um die Tabellenreihenfolge für die Synchronisation festzulegen, können Sie den Parameter `TableOrder` in Ihren Programmcode aufnehmen oder als Teil des Dienstprogramms `ulsync` während des Testens verwenden. Der Parameter `TableOrder` legt die Reihenfolge der Tabellen für den Upload fest.

Die Tabellenfolge muss nur dann explizit eingestellt werden, wenn Ihre UltraLite-Datenbank folgende Charakteristika aufweist:

- Fremdschlüsselzyklen. Sie müssen dann alle Tabellen auflisten, die Teil eines Zyklus sind.
- Fremdschlüsselbeziehungen, die sich von den Beziehungen in der konsolidierten Datenbank unterscheiden.

Synchronisationsprobleme mit Fremdschlüsselzyklen vermeiden

Die Tabellenreihenfolge ist besonders wichtig bei UltraLite-Datenbanken mit Fremdschlüsselzyklen. Sie bilden einen Zyklus, indem Sie mehrere Tabellen so miteinander verknüpfen, dass ein Kreis entsteht. Wenn sich Zyklen in der konsolidierten Datenbank und der entfernten UltraLite-Datenbank unterscheiden, führt dies jedoch zu komplexen Strukturen. Daher sind Fremdschlüsselzyklen nicht zu empfehlen.

Wenn Sie Fremdschlüsselzyklen verwenden, müssen Sie Ihre Tabellen in die richtige Reihenfolge bringen, damit Vorgänge für eine Primärtabelle Vorrang vor der zugehörigen Fremdtabelle haben. Ein `TableOrder`-Parameter stellt sicher, dass beim Einfügen in der Fremdtabelle die referenzielle Integrität für den Fremdschlüssel erfüllt ist (und ebenso für andere Vorgänge, wie etwa Löschungen).

Sie können zusätzlich zur Tabellenreihenfolge auch eine andere Methode verwenden, um Synchronisationsprobleme zu vermeiden: Warten Sie mit der Prüfung der referenziellen Integrität, bis die Transaktion festgeschrieben ist. Wenn Ihre konsolidierte Datenbank eine SQL Anywhere-Datenbank ist, setzen Sie einen der Fremdschlüssel auf **check on commit**. Damit stellen Sie sicher, dass die referenzielle Integrität des Fremdschlüssels während des Festschreibens und nicht beim Einleiten des Vorgangs geprüft wird. Beispiel:

```
CREATE TABLE c (  
    id INTEGER NOT NULL PRIMARY KEY,  
    c_pk INTEGER NOT NULL  
);  
CREATE TABLE p (  
    p_pk INTEGER NOT NULL PRIMARY KEY,  
    p_fk INTEGER NOT NULL  
);
```

```

    pk INTEGER NOT NULL PRIMARY KEY,
    c_id INTEGER NOT NULL,
    FOREIGN KEY p_to_c (c_id) REFERENCES c(id)
);
ALTER TABLE c
ADD FOREIGN KEY c_to_p (c_pk)
REFERENCES p(pk)
CHECK ON COMMIT;

```

Wenn Sie als konsolidierte Datenbank die Datenbank eines anderen Herstellers verwenden, überprüfen Sie, ob die Datenbank über ähnliche Methoden zur Überprüfung der referenziellen Integrität verfügt. Ist dies der Fall, sollten Sie diese Methode implementieren. Anderfalls müssen Sie die Tabellenbeziehungen umgestalten, um alle Fremdschlüsselzyklen zu entfernen.

Siehe auch

- „Synchronisationsparameter Additional Parameters“ auf Seite 91
- „Referenzielle Integrität und Synchronisation“ [*MobiLink - Erste Orientierung*]

Synchronisationseinrichtung für Ihre UltraLite-Anwendung

In UltraLite beginnt die Synchronisation, indem eine bestimmte Verbindung mit dem MobiLink-Server über den konfigurierten Kommunikationsdatenstrom (auch als Netzwerkprotokoll bezeichnet) aufgenommen wird. Zusätzlich zur Synchronisationsunterstützung für direkte Netzwerkverbindungen unterstützen Windows Mobile-Geräte auch die ActiveSync-Synchronisation.

Verbindung definieren

Jede entfernte UltraLite-Datenbank synchronisiert mit einem MobiLink-Server über ein Netzwerkprotokoll. Das Netzwerkprotokoll wird mit dem Parameter für den Synchronisationsdatenstrom eingestellt. Als Netzwerkprotokolle werden TCP/IP, HTTP, HTTPS und TLS unterstützt. Für das gewählte Protokoll müssen Sie auch Datenstromparameter übergeben, die andere erforderliche Verbindungsinformationen wie z.B. Host und Port des MobiLink-Servers definieren. Sie müssen auch MobiLink- Benutzerdaten und die Synchronisationsskriptversion angeben.

Synchronisationsverhalten definieren

Sie können das Synchronisationsverhalten steuern, indem Sie diverse Synchronisationsparameter einstellen. Wie die Parameter gesetzt werden, hängt von der jeweils verwendeten UltraLite-Schnittstelle ab.

Beachten Sie folgende wichtige Elemente des Synchronisationsverhaltens:

- **Synchronisationsrichtung** Standardmäßig wird eine Zwei-Wege-Synchronisation ausgeführt. Wenn Sie nur Einweg-Synchronisationen benötigen, müssen Sie die entsprechenden Parameter `upload_only` oder `download_only` verwenden. Mithilfe von Einweg-Synchronisationen können Sie die Zeit für Synchronisationen reduzieren. Außerdem müssen Sie bei reinen Download-Synchronisationen vor der Synchronisation nicht alle Änderungen der UltraLite-Datenbank festschreiben. Nicht festgeschriebene Änderungen von Tabellen, die nicht synchronisiert werden, verursachen durch die unvollständigen Transaktionen keine Probleme, da für sie kein Upload erfolgt.

Bei der reinen Download-Synchronisation müssen Sie sicher stellen, dass die Zeilen, die mit dem Download überlappen, nicht lokal geändert werden. Wenn Daten lokal geändert werden, schlägt die Synchronisation in der UltraLite-Anwendung fehl, und es kommt zu einem SQLE_DOWNLOAD_CONFLICT-Fehler.

- **Parallele Änderungen während der Synchronisation** Während der Upload-Phase haben UltraLite-Anwendungen nur Lesezugriff auf UltraLite-Datenbanken. Während der Download-Phase ist der Lese- und Schreibzugriff erlaubt. Wenn eine Anwendung eine Zeile ändert, die anschließend auch vom Download geändert werden soll, schlägt der Download jedoch fehl und wird zurückgesetzt. Diesen parallelen Datenzugriff während der Synchronisation können Sie deaktivieren, indem Sie den Synchronisationsparameter `Disable_Concurrency` verwenden.

Die folgende Vorgehensweise wird üblicherweise verwendet, um Ihrer Anwendung Synchronisationsfunktionen hinzuzufügen:

1. Übergeben Sie die für die Sitzung notwendigen Synchronisationsparameter und Protokolloptionen als Felder einer Synchronisationsinformationsstruktur.

Beispiel: Mit der UltraLite C/C++-API können Sie einer Ihrer Anwendungen Synchronisationsfunktionen hinzufügen, indem Sie die geeigneten Werte in der `ul_sync_info`-Struktur einstellen:

```
ul_sync_info info;
// define a sync structure named "info"
ULEnableTcpiipSynchronization( &sqlca );
// use a TCP/IP stream
conn->InitSynchInfo( &info );
// initialize the structure
info.stream = ULStream();
// specify the Socket Stream
info.stream_parms= UL_TEXT( "host=myMLserver;port=2439" );
// set the MobiLink host information
info.version = UL_TEXT( "custdb 11.0" );
// set the MobiLink version information
info.user_name = UL_TEXT( "50" );
// set the MobiLink user name
info.download_only = ul_true;
// make the synchronization download-only
```

2. Initialisieren Sie die Synchronisation.

Für eine direkte Synchronisation rufen Sie eine API-spezifische Synchronisationsmethode auf. Diese Methoden geben einen booleschen Wert zurück, der Erfolg oder Fehlschlag des Synchronisationsvorgangs bekanntgibt. Wenn die Synchronisation fehlschlägt, können Sie detaillierte Fehlerstatusfelder in einer anderen Struktur prüfen, um zusätzliche Informationen zur Fehlerbedingung zu erhalten.

Bei der ActiveSync-Synchronisation müssen Sie die Synchronisationsnachricht vom ActiveSync-Provider abfangen und die `DoSync`-Methode benutzen, um die `ULSynchronize`-Methode aufzurufen.

3. Benutzen Sie eine Beobachter-Callbackfunktion, um den Fortschritt der Synchronisation an den Benutzer zu melden.

Tipp

Wenn Sie eine Umgebung verwenden, in der DLLs fehlschlagen, weil die DLL sehr groß oder die Netzwerkverbindung nicht zuverlässig ist, können Sie wiederaufnehmbare Downloads einrichten. Siehe „Fehlgeschlagene Downloads“ [[MobiLink - Serveradministration](#)] und „Wiederaufnahme fehlgeschlagener Downloads“ [[MobiLink - Serveradministration](#)].

Siehe auch

- „ActiveSync mit UltraLite unter Windows Mobile“ auf Seite 88
- „Reine Upload- und reine Download-Synchronisation“ [[MobiLink - Serveradministration](#)]
- Upload und Download [[MobiLink - Erste Orientierung](#)]
- „UltraLite-Synchronisationsparameter“ auf Seite 90
- „UltraLite-Netzwerkprotokolloptionen für dbmlsync“ auf Seite 114
- UltraLite.NET: „MobiLink-Datensynchronisation“ [[UltraLite - .NET-Programmierung](#)]
- UltraLite C/C++: „MobiLink-Datensynchronisation“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite C/C++: „ActiveSync-Synchronisation einrichten“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite für Embedded SQL: „Synchronisation für eine Embedded SQL-Anwendung einrichten“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLiteJ-API: „MobiLink-Datensynchronisation“ [[UltraLite® – Java-Programmierung](#)]

MobiLink-Dateiübertragungen

UltraLite unterstützt die Möglichkeit der Übertragung von Dateien mit dem MobiLink-Server.

Bei allen anderen APIs benutzen Sie den MobiLink-Dateiübertragungsmechanismus unter folgenden Bedingungen:

- Mehrere Dateien müssen für ein Deployment auf mehreren Geräten übertragen werden, insbesondere wenn Unternehmensfirewalls als Schutzmechanismus verwendet werden. MobiLink ist bereits so konfiguriert, dass Synchronisationen über firmeninterne Firewalls möglich sind. Dies erleichtert die Dateiübertragung im Rahmen von Updates und für andere Zwecke dank der MLFileTransfer-Methode.
- Sie haben Dateien, die für eine bestimmte MobiLink-Benutzer-ID gedacht sind. Dazu ist es erforderlich, dass Sie für jede gewünschte Benutzer-ID ein oder mehrere benutzerspezifische Verzeichnisse auf dem MobiLink-Server erstellen. Wenn nur eine Version der Datei vorhanden ist, können Sie ein Standardverzeichnis verwenden.

Arbeitsweise bei der Dateiübertragung

Sie können eine der beiden von MobiLink initiierten Dateiübertragungsmethoden verwenden, um den Download von Dateien auf ein Gerät durchzuführen: Führen Sie das Dienstprogramm mlfiletransfer für Übertragungen vom PC aus oder rufen Sie die entsprechende Funktion für die API auf, die Sie für die Entwicklung Ihrer UltraLite-Anwendung benutzen. Für beide Ansätze gelten folgende Voraussetzungen:

1. Beschreiben Sie das Übertragungsziel.

Ob Sie nun das mlfiletransfer-Dienstprogramm vom PC oder eine geeignete Funktion Ihrer API verwenden, in beiden Fällen müssen Sie den lokalen Pfad und Dateinamen der Datei auf dem

Zielgerät oder PC angeben. Wenn diese Angaben von der Anwendung oder vom Endbenutzer nicht übergeben werden, wird angenommen, dass der Name der Quelldatei verwendet wird, um die Datei im aktuellen Verzeichnis zu speichern.

Das Zielverzeichnis kann je nach Betriebssystem des Geräts unterschiedlich sein:

- Unter Windows Mobile: Wenn das Ziel NULL ist, wird die Datei im Stammverzeichnis gespeichert (\).
Der Dateiname muss die Dateinamenskonventionen für Windows Mobile einhalten. Siehe [Windows Mobile auf Seite 38](#).
- Auf dem PC: Wenn das Ziel NULL ist, wird die Datei im aktuellen Verzeichnis gespeichert.
Der Dateiname muss die Dateinamenskonventionen für das PC-Betriebssystem einhalten. Siehe [Desktop auf Seite 38](#).
- Auf iPhone sollten Sie Dateien im Dokumentverzeichnis Ihrer Anwendung speichern. Sie erhalten den Speicherort des Dokumentverzeichnis durch den Aufruf der NSSearchPathForDirectories/uDomains mit dem NSDocumentDirectory-Parameter.
- Auf einem BlackBerry legen Sie den Speicherort mit dem FileTransfer-Objekt fest.

2. Legen Sie die MobiLink-Anmeldeinformationen fest, mit deren Hilfe der Benutzer identifiziert werden kann und die richtigen Dateien heruntergeladen werden können.

Der Benutzername und das Kennwort unterscheiden sich von der ID und dem Kennwort des Datenbankbenutzers und dienen der Kennzeichnung und Authentifizierung der Anwendung beim MobiLink-Server.

3. Legen Sie den gewünschten Datenstromtyp fest und definieren Sie die Parameter für den gewünschten Datenstrom. Es handelt sich um dieselben Parameter, die von UltraLite für die MobiLink-Synchronisation unterstützt werden.

Die meisten Synchronisationsdatenströme benötigen Parameter, um die Adresse des MobiLink-Servers zu bestimmen und anderes Verhalten zu steuern. Wenn der Datenstromtyp auf einen für die Plattform unzulässigen Wert gesetzt wird, erhält der Datenstromparameter den Wert TCP/IP. Ausgenommen davon sind UltraLite Java Edition-Datenbanken (nur Unterstützung für HTTP).

4. Beschreiben Sie das gewünschte Verhalten für den Übertragungsmechanismus.

Beispiel: Sie können Eigenschaften festlegen, die es diesem Mechanismus erlauben, einen Download zu erzwingen, auch wenn die Datei auf dem Zielgerät bereits vorhanden ist und nicht geändert wurde, oder zulassen, dass ein Teil-Download wieder aufgenommen wird. Sie können auch festlegen, ob der Fortschritt überwacht und gemeldet werden soll.

5. Vergewissern Sie sich, dass der MobiLink-Server läuft und mit der Option -ftr gestartet wurde.
6. Starten Sie die Übertragung und überwachen Sie gegebenenfalls den Download-Fortschritt.

Durch Anzeigen des Download-Fortschritts kann der Benutzer den Download abbrechen und zu einem späteren Zeitpunkt fortsetzen.

Siehe auch

- „UltraLite-Synchronisationsparameter“ auf Seite 90
- „UltraLite-Netzwerkprotokolloptionen für dbmsync“ auf Seite 114
- „mlsrv16-Option -ftr“ [*MobiLink - Serveradministration*]
- „MobiLink-Dienstprogramm für die Dateiübertragung (mlfiletransfer)“ [*MobiLink - Clientadministration*]
- MLFileDownload-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- MLFileUpload-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULFileTransfer-Klasse [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- FileTransfer-Schnittstelle [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

UltraLite-Publikationen

Eine Publikation ist ein Datenbankobjekt, das die zu synchronisierenden Daten identifiziert. Um alle Tabellen und alle Zeilen dieser Tabellen in Ihrer UltraLite-Datenbank zu synchronisieren, erstellen Sie keine Publikationen.

Eine Publikation besteht aus einer Reihe von Artikeln. Jeder Artikel kann aus einer ganzen Tabelle oder aus Zeilen einer Tabelle bestehen. Sie können diese Gruppe von Zeilen mit einer WHERE-Klausel definieren.

Jede Datenbank kann mehrere Publikationen haben, abhängig von der gewünschten Synchronisationslogik. Sie könnten z.B. eine Publikation für Daten mit hoher Priorität erstellen. Der Benutzer kann diese Daten über drahtlose Hochgeschwindigkeitsnetzwerke synchronisieren. Da drahtlose Netzwerke hohe Nutzungskosten haben können, sollten sie auf geschäftskritische Daten beschränkt werden. Alle anderen weniger zeitsensitiven Daten könnten zu einem späteren Zeitpunkt über eine Dockingstation ausgeführt werden.

Sie können Publikationen mithilfe von Sybase Central oder mit der Anweisung CREATE PUBLICATION erstellen. In Sybase Central erscheinen alle Publikationen und Artikel im Ordner **Publikationen**.

Verwendungshinweise

- UltraLite-Publikationen unterstützen weder die Definition von Spaltenteilmengen noch die Klausel SUBSCRIBE BY. Wenn Spalten in einer UltraLite-Tabelle nicht genau mit Tabellen in einer konsolidierten SQL Anywhere-Datenbank übereinstimmen, können Sie diese Unterschiede mit MobiLink-Skripten beseitigen.
- Spalten werden immer in der Reihenfolge gesendet, in der sie in der CREATE TABLE-Anweisung definiert wurden.
- Es ist nicht erforderlich, in einer Publikation die Reihenfolge für die Tabellensynchronisation festzulegen. Wenn die Tabellenfolge für Ihr Deployment wichtig ist, können Sie diese bei der Synchronisation der UltraLite-Datenbank mit dem Synchronisationsparameter TableOrder festlegen.
- Da die Objekteigentümerschaft in UltraLite nicht unterstützt wird, kann jeder Benutzer eine Publikation löschen.

Siehe auch

- „Tabellenreihenfolge in UltraLite“ auf Seite 80
- „Publikationen“ [*MobiLink - Clientadministration*]
- „UltraLite-Clientsynchronisationsplanung“ auf Seite 74
- „Schreiben von Synchronisationsskripten“ [*MobiLink - Serveradministration*]

Ganze Tabellen in UltraLite publizieren

Eine Publikation besteht aus einer Reihe von Artikeln. Die einfachste Publikation, die Sie erstellen können, besteht aus einem einzelnen Artikel, der alle Zeilen und Spalten einer Tabelle enthält.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Diese Aufgabe kann in Sybase Central oder in Interactive SQL durchgeführt werden.

Aufgabe

1. Stellen Sie mithilfe von Sybase Central eine Verbindung mit der UltraLite-Datenbank her.
2. Rechtsklicken Sie auf den Ordner **Publikationen** und klicken Sie auf **Neu » Publikation**.
3. Geben Sie in das Feld **Wie lautet der Name der neuen Publikation?** einen Namen für die neue Publikation ein. Klicken Sie auf **Weiter**.
4. Klicken Sie auf der Registerkarte **Tabellen** in der Liste **Verfügbare Tabellen** auf Tabellen. Klicken Sie auf **Hinzufügen**.
5. Klicken Sie auf **Fertig stellen**.

Ergebnisse

Die Publikation wird erstellt.

Nächste Schritte

Keine.

Siehe auch

- „CREATE PUBLICATION-Anweisung [UltraLite]“ auf Seite 435
- „UltraLite-Clients“ auf Seite 69

Teilmenge von Zeilen aus einer UltraLite-Tabelle publizieren

Eine Publikation kann nur bestimmte Tabellenzeilen enthalten. Eine WHERE-Klausel begrenzt die Zeilen, die hochgeladen werden, auf die Zeilen, die geändert wurden und die eine Suchbedingung in der WHERE-Klausel erfüllen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Geben Sie keine WHERE-Klausel an, wenn Sie alle geänderten Zeilen übertragen wollen.

Aufgabe

1. Stellen Sie mithilfe von Sybase Central eine Verbindung mit der UltraLite-Datenbank her.
2. Rechtsklicken Sie auf den Ordner **Publikationen** und klicken Sie auf **Neu » Publikation**.
3. Geben Sie in das Feld **Wie lautet der Name der neuen Publikation?** einen Namen für die neue Publikation ein.
4. Klicken Sie auf **Weiter**.
5. In der Liste **Verfügbare Tabellen** wählen Sie eine Tabelle aus und klicken auf **Hinzufügen**.
6. Klicken Sie auf das Register **WHERE-Klauseln** und wählen Sie die Tabelle in der Liste **Artikel** aus. Sie können auch das Fenster **Einfügen** verwenden, das Sie bei der Erstellung der Suchbedingungen unterstützt.
7. Klicken Sie auf **Fertig stellen**.

Ergebnisse

Die Zeilen, die hochgeladen werden, sind nun auf die Zeilen begrenzt, die geändert wurden und die eine Suchbedingung in der WHERE-Klausel erfüllen.

Nächste Schritte

Keine.

Siehe auch

- „CREATE PUBLICATION-Anweisung [UltraLite]“ auf Seite 435
- „UltraLite-Clients“ auf Seite 69

Publikation für UltraLite löschen

Das Löschen der Publikationen einer Tabelle ermöglicht Ihnen, alle Tabellen und Zeilen dieser Tabelle in der UltraLite-Datenbank zu synchronisieren.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Sie können eine Publikation mithilfe von Sybase Central oder Interactive SQL löschen.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung mit der UltraLite-Datenbank her.
2. Doppelklicken Sie im linken Fensterausschnitt auf den Ordner **Publikationen**.
3. Rechtsklicken Sie auf die Publikation und klicken Sie auf **Löschen**.
4. Klicken Sie auf **Ja**.

Ergebnisse

Die Publikation wird gelöscht.

Siehe auch

- „DROP PUBLICATION-Anweisung [UltraLite]“ auf Seite 447
- „UltraLite-Clients“ auf Seite 69

ActiveSync mit UltraLite unter Windows Mobile

Auch wenn Sie Daten auf einem Windows Mobile-Gerät über eine Ethernet- oder Wi-Fi-Verbindung synchronisieren können, wird in diesem Abschnitt beschrieben, wie Sie Ihren PC und das Gerät für die ActiveSync-Synchronisation konfigurieren, damit Ihre UltraLite-Datenbank zur gleichen Zeit wie andere ActiveSync-Vorgänge synchronisiert wird. Um mit einer der anderen alternativen Methoden direkt zu synchronisieren, müssen Sie Ihre Anwendung mit einer entsprechenden Synchronisationsfunktion speziell dafür programmieren.

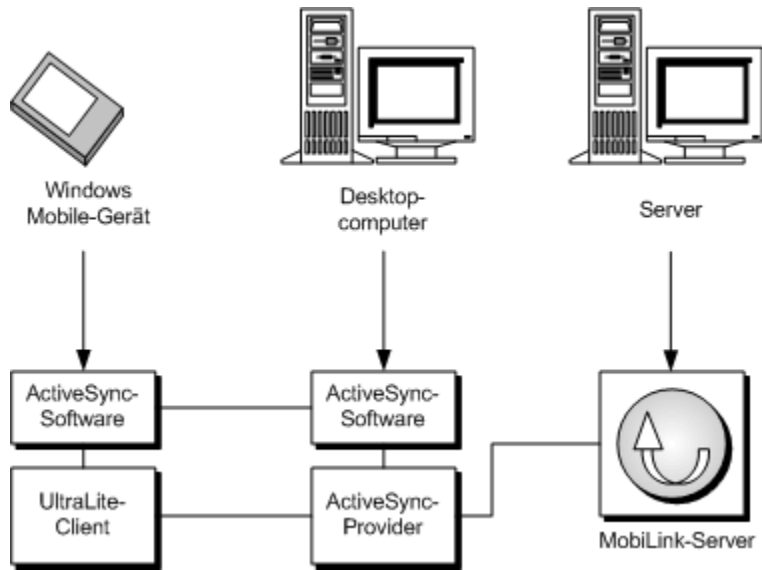
Um ActiveSync zu verwenden, sind folgende Voraussetzungen zu erfüllen:

- Registrieren Sie alle Anwendungen, die eine durch ActiveSync initiierte Synchronisation mit ActiveSync erfordern.
- Der ActiveSync-Provider muss auf dem PC installiert sein und per Deployment auf Ihrem Gerät bereitgestellt werden.

Weitere Hinweise dazu, auf welchen Plattformen der Provider unterstützt wird, finden Sie unter <http://www.sybase.com/detail?id=1002288>.

Die ActiveSync-Architektur

Im folgenden Diagramm werden die Berechnungsschichten dargestellt, die von der ActiveSync-Architektur benötigt werden.



Sie müssen den ActiveSync-Provider nicht nur auf Ihrem PC, sondern auch auf Ihrem Gerät installieren. Auf einem Computer kann sich nur ein ActiveSync-Provider befinden. Wenn Sie mehr als eine UltraLite-Anwendung auf einem Windows Mobile-Gerät installiert haben, können Sie diese mit demselben Provider registrieren, sodass sie gleichzeitig synchronisiert werden.

Siehe auch

- UltraLite C/C++: „ActiveSync-Synchronisation einrichten“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite für Embedded SQL: „Synchronisation für eine Embedded SQL-Anwendung einrichten“ [[UltraLite - C- und C++-Programmierung](#)]

Übersicht über die ActiveSync-Synchronisation

1. ActiveSync startet eine Synchronisationsitzung.
2. Der ActiveSync-Provider sendet eine Synchronisationsmeldung an die erste registrierte Anwendung auf dem Gerät. Wenn die Anwendung noch nicht läuft, wird sie jetzt gestartet.
3. WndProc wird für jede registrierte Anwendung aufgerufen.
4. Sobald die Anwendung festgestellt hat, dass es sich um eine Synchronisationsmeldung von ActiveSync handelt, ruft die Anwendung `ULIsSynchronizeMessage` auf, um den Datenbank-Synchronisationsvorgang zu starten.
5. Sobald die Synchronisation abgeschlossen ist, ruft die Anwendung `ULSignalSyncIsComplete` auf, um dem Provider dies mitzuteilen.

6. Die Schritte Zwei bis Fünf werden für jede Anwendung wiederholt, die mit dem Provider registriert wurde.

UltraLite-Synchronisationsparameter

Die Synchronisationsparameter steuern die Synchronisation zwischen einer UltraLite-Datenbank und dem MobiLink-Server. Wie die Parameter gesetzt werden, hängt von der jeweils verwendeten UltraLite-Schnittstelle ab. Dieser Abschnitt beschreibt die Auswirkungen der Parameter und verweist auf andere Kapitel, in denen Sie Hinweise zur Einstellung erhalten.

Hinweis

Die in diesem Abschnitt beschriebenen Parameter gelten nur für entfernte UltraLite-Datenbanken. Hinweise zur Synchronisation von entfernten SQL Anywhere-Datenbanken finden Sie unter „[MobiLink SQL Anywhere Client-Dienstprogramm \(dbmlsync\)](#)“ [*MobiLink - Clientadministration*].

Erforderliche Parameter

Folgende Parameter sind erforderlich:

- Stream Type
- User name
- Version

Die Synchronisationsfunktion löst einen Ausnahmefehler aus, wie z. B. `SQLCode.SQLE_SYNC_INFO_INVALID` oder gleichwertig, wenn Sie diese Parameter nicht einstellen.

Parameterkonflikte

Sie können nur höchstens einen dieser Parameter angeben:

- Download Only
- Ping
- Upload Only

Die Synchronisationsfunktion löst einen Ausnahmefehler aus, wie z. B. `SQLCode.SQLE_SYNC_INFO_INVALID` oder gleichwertig, wenn Sie mehr als einen dieser Parameter auf TRUE einstellen.

Siehe auch

- [ULSynchronize-Methode \[UltraLite Embedded SQL\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULSyncParms-Klasse \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)
- [ULConnection.Synchronize-Methode \[UltraLite C++\] \[UltraLite - C- und C++-Programmierung\]](#)
- [SyncParms-Klasse \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [„Synchronisationsparameter Stream Type“ auf Seite 107](#)
- [„Synchronisationsparameter User Name“ auf Seite 112](#)
- [„Synchronisationsparameter Version“ auf Seite 113](#)
- [„Synchronisationsparameter Download Only“ auf Seite 96](#)
- [„Synchronisationsparameter Ping“ auf Seite 102](#)
- [„Synchronisationsparameter Upload Only“ auf Seite 110](#)

Synchronisationsparameter Additional Parameters

Dieser Synchronisationsparameter ermöglicht es einer Anwendung, zusätzliche Parameter zu liefern, die nicht ohne Weiteres mit anderen vordefinierten Parameter angegeben werden können. Einige selten verwendete Parameter werden in diesem Parameterfeld angegeben.

Diese zusätzlichen Parameter werden als Zeichenfolge von Schlüsselwort=Wert-Einstellungen, durch ein Semikolon getrennt, angegeben.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Nicht verfügbar in UltraLiteJ.

Zulässige Werte

Die folgenden Eigenschaften können als Teil der zusätzlichen Parametereinstellungen festgelegt werden:

Eigenschaftsname	Beschreibung
AllowDownloadDupRows	<p>Verhindert, dass Fehler ausgegeben werden, wenn eine Synchronisation auf heruntergeladene Zeilen mit mehrfach vorhandenen Primärschlüsseln stößt.</p> <p>Setzen Sie diese Eigenschaft auf 0, wenn Fehler ausgegeben und der Download zurückgesetzt werden sollen. Um Warnungen auszugeben und den Download fortzusetzen, setzen Sie sie auf 1.</p> <p>Diese Eigenschaft ist nur in UltraLite C/C++ verfügbar.</p>
CheckpointStore	<p>Fügt während der Synchronisation in die Datenbank zusätzliche Checkpoints ein, um das Anwachsen der Datenbank während des Synchronisationsprozesses zu verhindern.</p> <p>Setzen Sie diese Eigenschaft auf 1, um diese Funktion zu aktivieren. Dies ist vorteilhaft bei großen Downloads mit vielen Updates, verlangsamt jedoch die Synchronisation. Andernfalls wählen Sie für die Eigenschaft die Standardeinstellung 0.</p>

Eigenschaftsname	Beschreibung
DisableConcurrency	<p>Lässt keinen Datenbankzugriff durch andere Threads während der Synchronisation während der Upload-Phase zu.</p> <p>Setzen Sie diese Eigenschaft auf 0, um gleichzeitigen Datenbankzugriff zuzulassen. Andernfalls legen Sie für die Eigenschaft 1 fest. Standardmäßig ist diese Eigenschaft auf 0 gesetzt.</p>
TableOrder	<p>Stellen Sie die für die Prioritätensynchronisation erforderliche Tabellenfolge ein, wenn die Standardtabellenfolge von UltraLite für Ihr Deployment unpassend ist.</p> <p>Setzen Sie diese Eigenschaft auf eine Liste von Tabellennamen, die in der gewünschten Reihenfolge für den Upload angeordnet sind. Für UltraLite verwenden Sie eine durch Kommas getrennte Liste. Für ulsync verwenden Sie eine durch Semikolons getrennte Liste. Standardmäßig basiert die Reihenfolge auf Fremdschlüsselbeziehungen. Normalerweise ist die Standardeinstellung problemlos, wenn die Fremdschlüssel in der konsolidierten Datenbank mit der entfernten UltraLite-Datenbank übereinstimmen und keine Fremdschlüsselzyklen vorliegen.</p> <p>Geben Sie Tabellennamen in Apostrophen oder Anführungszeichen an. Beispiel "Customer,Sales" und 'Customer,Sales' werden in UltraLite unterstützt.</p> <p>Wenn Sie Tabellen einbeziehen, die in der Synchronisation nicht enthalten sind, werden diese ignoriert. Alle nicht angegebenen Tabellen werden basierend auf den in der entfernten Datenbank definierten Fremdschlüsseln entsprechend sortiert.</p> <p>Die Tabellenfolge des Downloads entspricht der des Uploads.</p> <p>Nur für folgende UltraLite-Tabellen muss die Tabellenfolge explizit eingestellt werden:</p> <ul style="list-style-type: none"> • Sie sind Teil von Fremdschlüsselzyklen. Sie müssen dann alle Tabellen auflisten, die Teil eines Zyklus sind. • Sie haben unterschiedliche Fremdschlüsselbeziehungen in der konsolidierten Datenbank.

Siehe auch

- [ul_sync_info-Struktur \[UltraLite C- und Embedded SQL-Datentypen\] \[UltraLite - C- und C++-Programmierung\]](#)
- [ULSyncParms.AdditionalParms-Eigenschaft \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#)

Beispiel

Für Anwendungen in UltraLite für C/C++ können zusätzliche Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;
// ...
info.additional_parms = UL_TEXT(
    "AllowDownloadDupRows=1;
    CheckpointStore=1;
    DisableConcurrency=1;
    TableOrder=Customer,Sales"
);
```

Synchronisationsparameter Authentication Parameters

Stellt Authentifizierungsparametern in MobiLink-Ereignissen Parameter bereit.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API.

Bemerkungen

Parameter können z.B. ein Benutzername und ein Kennwort sein.

Wenn Sie diesen Parameter verwenden, müssen Sie auch die Anzahl der Parameter angeben.

Zulässige Werte

Ein Array von Zeichenfolgen. NULL ist als Wert in keiner der Zeichenfolgen zulässig. Sie können aber eine leere Zeichenfolge verwenden.

Siehe auch

- „Synchronisationsparameter Number of Authentication Parameters“ auf Seite 99
- „Authentifizierungsparameter“ [*MobiLink - Serveradministration*]
- „authenticate_parameters (Verbindungsereignis)“ [*MobiLink - Serveradministration*]
- ul_sync_info-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- ULSyncParms.AuthenticationParms-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- SyncParms.setAuthenticationParms-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

Für Anwendungen in UltraLite für C/C++ können die Parameter folgendermaßen eingestellt werden:

```
ul_char * Params[ 3 ] = { UL_TEXT( "parm1" ),
                          UL_TEXT( "parm2" ),
                          UL_TEXT( "parm3" ) };

// ...
info.num_auth_parms = 3;
info.auth_parms = Params;
```

Synchronisationsparameter Authentication Status

Dieses Feld wird von einer Synchronisation eingestellt, um den Status einer MobiLink-Benutzerauthentifizierung zu melden. Der MobiLink-Server gibt diese Information an den Client weiter.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API.

Zulässige Werte

Die zulässigen Werte werden in einer schnittstellenspezifischen Enumeration gehalten. Für C/C++-Anwendungen sieht die Enumeration beispielsweise folgendermaßen aus:

Konstante	Wert	Beschreibung
UL_AUTH_STATUS_UNKNOWN	0	Autorisierungsstatus ist unbekannt, möglicherweise weil die Verbindung noch nicht synchronisiert wurde.
UL_AUTH_STATUS_VALID	1	Benutzer-ID und Kennwort waren zur Zeit der Synchronisation gültig.
UL_AUTH_STATUS_VALID_BUT_EXPIRES_SOON	2	Benutzer-ID und Kennwort waren zur Zeit der Synchronisation gültig, laufen aber bald ab.
UL_AUTH_STATUS_EXPIRED	3	Autorisierung fehlgeschlagen: Benutzer-ID oder Kennwort ist abgelaufen.
UL_AUTH_STATUS_INVALID	4	Autorisierung fehlgeschlagen: Benutzer-ID oder Kennwort ist ungültig.
UL_AUTH_STATUS_IN_USE	5	Autorisierung fehlgeschlagen: Benutzer-ID wird bereits verwendet.

Bemerkungen

Wenn ein benutzerdefiniertes **authenticate_user**-Synchronisationsskript in der konsolidierten Datenbank einen anderen Wert zurückgibt, wird der Wert gemäß den Regeln interpretiert, die in einem `authenticate_user`-Verbindungsereignis enthalten sind.

Wenn Sie ein individuelles Authentifizierungsschema benutzen, muss das Synchronisationsskript `authenticate_user` oder `authenticate_user_hashed` einen der zulässigen Werte dieses Parameters zurückgeben.

Dieser Parameter wird vom MobiLink-Server gesetzt und ist damit schreibgeschützt.

Siehe auch

- „authenticate_user (Verbindungsereignis)“ [*MobiLink - Serveradministration*]
- „MobiLink-Benutzer“ [*MobiLink - Clientadministration*]
- ul_sync_info-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- ULSyncResult.AuthStatus-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- SyncResult.getAuthStatus-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

Anwendungen in UltraLite für C/C++ können folgendermaßen auf diese Parameter zugreifen:

```
ul_sync_info info;  
// ...  
returncode = info.auth_status;
```

Authentication Value-Synchronisationsparameter

Dieses Feld wird von einer Synchronisation eingestellt, um Ergebnisse eines angepassten MobiLink-Benutzerauthentifizierungsskripts zu melden. Der MobiLink-Server gibt diese Information an den Client weiter.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Nicht verfügbar in UltraLiteJ.

Bemerkungen

Die durch das Standardverfahren für die MobiLink-Benutzerauthentifizierung eingestellten Werte werden im Verbindungsereignis "authenticate_user" und im Synchronisationsparameter "Authentication Status" beschrieben.

Dieser Parameter wird vom MobiLink-Server gesetzt und ist damit schreibgeschützt.

Siehe auch

- „authenticate_user (Verbindungsereignis)“ [*MobiLink - Serveradministration*]
- „authenticate_user_hashed (Verbindungsereignis)“ [*MobiLink - Serveradministration*]
- „Synchronisationsparameter Authentication Status“ auf Seite 94
- ul_sync_info-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- ULSyncResult.AuthValue-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- SyncResult.getAuthValue-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

Anwendungen in UltraLite für C/C++ können folgendermaßen auf diese Parameter zugreifen:

```
ul_sync_info info;  
// ...  
returncode = info.auth_value;
```

Synchronisationsparameter Download Only

Verhindert, dass während der Synchronisation ein Upload von Änderungen aus der UltraLite-Datenbank erfolgt.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API.

Standardwert

FALSE

Zulässige Werte

Boolescher Wert

Konflikte mit

Ping und reinem Upload

Bemerkungen

Datenänderungen werden nicht hochgeladen, wenn eine reine Download-Synchronisation vorgenommen wird. Informationen über das Schema und den Wert, der im Fortschrittszähler gespeichert ist, werden hochgeladen. Wenn die eingelesenen Daten Konflikte mit Änderungen in der entfernten Datenbank, die noch nicht hochgeladen wurden, verursachen, schlägt die Synchronisation fehl und wird zurückgesetzt.

Wenn Sie eine reine Downloadsynchronisation für UltraLite Java Edition-Datenbanken verwenden, sollten Sie regelmäßig eine vollständige Synchronisation durchführen, um die Größe des Transaktionslogs zu reduzieren, das beim Synchronisieren von entfernten Datenbanken gescannt wird. Andernfalls nehmen die Synchronisationen zunehmend mehr Zeit in Anspruch.

Siehe auch

- „Synchronisationsparameter Upload Only“ auf Seite 110
- Synchronisationsstatus-Protokollierung auf Seite 3
- „UltraLite-Synchronisationsdienstprogramm (ulsync)“ auf Seite 227
- `ul_sync_info`-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- `ULSyncParms.DownloadOnly`-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- `SyncParms.setDownloadOnly`-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

Das folgende Beispiel zeigt, wie Sie den `DownloadOnly`-Synchronisationsparameter mithilfe des `ulsync`-Dienstprogramms einstellen:

```
ulsync -c DBF=myuldb.udb  
      "MobiLinkUid=remoteA;ScriptVersion=2;DownloadOnly=ON;Stream=http"
```

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.download_only = ul_true;
```

Synchronisationsparameter Ignored Rows

Dieses Feld wird von einer Synchronisation definiert, um anzugeben, dass Zeilen während der Synchronisation vom MobiLink-Server aufgrund fehlender Skripten ignoriert wurden.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API.

Zulässige Werte

Boolescher Wert

Bemerkungen

Der Parameter ist schreibgeschützt.

Siehe auch

- [ul_sync_info-Struktur](#) [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- [ULSyncResult.IgnoredRows-Eigenschaft](#) [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- [SyncResult.getIgnoredRows-Methode](#) [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

Anwendungen in UltraLite für C/C++ können folgendermaßen auf diese Parameter zugreifen:

```
ul_sync_info info;  
// ...  
res = info.ignored_rows;
```

Synchronisationsparameter Keep Partial Download

Wenn ein Download wegen eines Kommunikationsfehlers während der Synchronisation fehlschlägt, steuert dieser Parameter, ob UltraLite den teilweisen Download fortsetzt oder die Änderungen zurücksetzt.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Nicht verfügbar in UltraLiteJ.

Standardwert

FALSE, womit angegeben wird, dass UltraLite nach einem fehlgeschlagenen Download alle Änderungen zurücksetzt

Zulässige Werte

Boolescher Wert

Siehe auch

- „Wiederaufnahme fehlgeschlagener Downloads“ [*MobiLink - Serveradministration*]
- „Synchronisationsparameter Resume Partial Download“ auf Seite 104
- `ul_sync_info`-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- `ULSyncParms.KeepPartialDownload`-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.keep_partial_download = ul_true;
```

Synchronisationsparameter New Password

Legt ein neues MobiLink-Kennwort für den entsprechenden Benutzernamen fest

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Sie können diesen Parameter auch mit `ulsync` einstellen.

Zulässige Werte

Zeichenfolge

Bemerkungen

Der Parameter ist optional.

Siehe auch

- „MobiLink-Benutzer“ [*MobiLink - Clientadministration*]
- „UltraLite-Synchronisationsdienstprogramm (`ulsync`)“ auf Seite 227
- `ul_sync_info`-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- `ULSyncParms.NewPassword`-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- `SyncParms.setPassword`-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

`ulsync` kann diesen Parameter folgendermaßen als erweiterten Synchronisationsparameter einstellen:

```
ulsync -c DBF=myuldb.ldb  
"MobiLinkId=remoteA;ScriptVersion=2;NewMobiLinkPwd=mynewpassword;Stream=http  
"
```

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.new_password = UL_TEXT( "mlnewpass" );
```

Synchronisationsparameter Number of Authentication Parameters

Liefert die Anzahl von Parametern, die an Authentifizierungsparameter in MobiLink-Ereignissen übergeben werden

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Nicht erforderlich für UltraLiteJ.

Standardwert

Keine Parameter an ein benutzerdefiniertes Authentifizierungsskript übergeben

Bemerkungen

Der Parameter wird gemeinsam mit Authentifizierungsparametern verwendet, um Informationen an benutzerdefinierte Authentifizierungsskripten zu übergeben.

Siehe auch

- „Synchronisationsparameter Authentication Parameters“ auf Seite 93
- „authenticate_parameters (Verbindungsereignis)“ [*MobiLink - Serveradministration*]
- „Authentifizierungsparameter“ [*MobiLink - Serveradministration*]
- ul_sync_info-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- ULSyncParms.AuthenticationParms-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.num_auth_parms = 3;
```

Synchronisationsparameter Observer

Gibt einen Zeiger auf eine Callback-Funktion oder einen Event-Handler an, der die Synchronisation überwacht. Die Signatur der Callback-Funktion, die Sie implementieren müssen, ist vom Typ

ul_sync_observer_fn:

```
typedef void(UL_CALLBACK_FN *ul_sync_observer_fn)( ul_sync_status * status );
```

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API.

Siehe auch

- „Synchronisationsparameter User Data“ auf Seite 111
- `ul_sync_info`-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- `ULSyncProgressListener`-Schnittstelle [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- `SyncParms.setSyncObserver`-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.observer=callfunction;
```

Synchronisationsparameter Partial Download Retained

Dieses Feld wird von einer Synchronisation definiert, um anzugeben, ob UltraLite diese Download-Änderungen anwendet, anstatt die Änderungen zurückzusetzen, wenn ein Download auf Grund eines Kommunikationsfehlers während der Synchronisation fehlschlägt.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Nicht unterstützt von UltraLite Java Edition-Datenbanken.

Zulässige Werte

Boolescher Wert

Bemerkungen

Der Parameter wird während der Synchronisation gesetzt, wenn ein Download-Fehler eintritt und ein Teil-Download einbehalten wurde.

Partielle geladene Änderungen werden nur übernommen, wenn "Keep Partial Download" auf TRUE gesetzt ist.

Siehe auch

- „Synchronisationsparameter Keep Partial Download“ auf Seite 97
- „Wiederaufnahme fehlgeschlagener Downloads“ [*MobiLink - Serveradministration*]
- „Synchronisationsparameter Resume Partial Download“ auf Seite 104
- `ul_sync_info`-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- `ULSyncResult.PartialDownloadRetained`-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Greifen Sie wie folgt auf den Parameter zu:

```
ul_sync_info info;  
// ...  
returncode=info.partial_download_retained;
```

Synchronisationsparameter Password

Legt das MobiLink-Kennwort fest, das dem Benutzernamen zugeordnet ist

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Sie können diesen Parameter auch mit ulsync einstellen.

Zulässige Werte

Zeichenfolge

Bemerkungen

Der Parameter ist optional.

Dieser MobiLink-Benutzername und das Kennwort unterscheiden sich von der ID und dem Kennwort des Datenbankbenutzers und dienen der Kennzeichnung und Authentifizierung der Anwendung beim MobiLink-Server.

Wenn der MobiLink-Client bereits ein Kennwort hat, benutzen Sie den Parameter New Password, um es zu ändern.

Siehe auch

- „Synchronisationsparameter User Name“ auf Seite 112
- „Synchronisationsparameter New Password“ auf Seite 98
- „MobiLink-Benutzer“ [*MobiLink - Clientadministration*]
- „UltraLite-Synchronisationsdienstprogramm (ulsync)“ auf Seite 227
- ul_sync_info-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- ULSyncParms.Password-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- SyncParms.setPassword-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

ulsync kann diesen Parameter folgendermaßen als erweiterten Synchronisationsparameter einstellen:

```
ulsync -c DBF=myuldb.udb  
"MobiLinkUid=remoteA;ScriptVersion=2;MobiLinkPwd=mypassword;Stream=http"
```

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.password = UL_TEXT( "mypassword" );
```

Synchronisationsparameter Ping

Bestätigt Kommunikationen zwischen dem UltraLite-Client und dem MobiLink-Synchronisationsserver. Wenn dieser Parameter auf TRUE gesetzt wird, findet keine Synchronisation statt.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Sie können diesen Parameter auch mit `ulsync` einstellen.

Standardwert

FALSE

Zulässige Werte

Boolescher Wert

Bemerkungen

Wenn der MobiLink-Server eine ping-Anfrage erhält, verbindet er sich mit der konsolidierten Datenbank, authentifiziert den Benutzer und sendet dann den Status des authentifizierten Benutzers und den Wert an den Client zurück.

Wird der ping-Parameter erfolgreich ausgeführt, gibt der MobiLink-Server eine Informationsmeldung aus. Wird der ping-Parameter erfolglos ausgeführt, gibt der Server eine Fehlermeldung aus.

Wenn die MobiLink-Benutzer-ID nicht in der Systemtabelle `ml_user` gefunden werden kann und der MobiLink-Server mit der Befehlszeilenoption `-zu+` ausgeführt wird, fügt er den Benutzer der Tabelle `ml_user` hinzu.

Der MobiLink-Server kann die folgenden Skripten für eine ping-Anfrage verwenden, sofern sie vorhanden sind:

- `begin_connection`
- `authenticate_user`
- `authenticate_user_hashed`
- `authenticate_parameters`
- `end_connection`

Siehe auch

- „`dbmlsync`-Option `-pi`“ [*MobiLink - Clientadministration*]
- „UltraLite-Synchronisationsdienstprogramm (`ulsync`)“ auf Seite 227
- `ul_sync_info`-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- `ULSyncParms.PingOnly`-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- `SyncParms.setPingOnly`-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

`ulsync` kann diesen Parameter folgendermaßen als erweiterten Synchronisationsparameter einstellen:

```
ulsync -c DBF=myuldb.udb  
"MobiLinkId=remoteA;ScriptVersion=2;Ping=True;Stream=http"
```

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.ping = ul_true;
```

Synchronisationsparameter Publications

Legt die zu synchronisierenden Publikationen fest.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Sie können diesen Parameter auch mit ulsync benutzen.

Standardwert

Alle Publikationen werden synchronisiert.

Bemerkungen

Wenn Sie in C/C++ synchronisieren, setzen Sie den Publications-Synchronisationsparameter auf eine **Publikationsliste**: eine durch Kommas getrennte Liste von Publikationsnamen.

Siehe auch

- „Publizieren von Daten in UltraLite“ auf Seite 78
- „UltraLite-Publikationen“ auf Seite 85
- „UltraLite-Synchronisationsdienstprogramm (ulsync)“ auf Seite 227
- ul_sync_info-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- ULConnection.SYNC_ALL_DB-Feld [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- ULConnection.SYNC_ALL_PUBS-Feld [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- SyncParms.setPublications-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

ulsync kann diesen Parameter folgendermaßen als erweiterten Synchronisationsparameter einstellen:

```
ulsync -c DBF=myuldb.udb  
"MobiLinkId=remoteA;ScriptVersion=2;Publications=UL_PUB_MYPUB1,UL_PUB_MYPUB2  
;Stream=http"
```

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.publications = UL_TEXT( "Pubs1,Pubs3" );
```

Synchronisationsparameter Resume Partial Download

Setzt einen fehlgeschlagenen Download fort.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Sie können diesen Parameter auch mit `ulsync` einstellen. Nicht unterstützt von UltraLite Java Edition-Datenbanken.

Standardwert

FALSE

Zulässige Werte

Boolescher Wert

Bemerkungen

Die Synchronisation sendet keine Änderungen beim Upload und lädt nur die Änderungen, die beim fehlgeschlagenen Download geladen werden sollten.

Siehe auch

- „Wiederaufnahme fehlgeschlagener Downloads“ [*MobiLink - Serveradministration*]
- „Synchronisationsparameter Keep Partial Download“ auf Seite 97
- „Synchronisationsparameter Partial Download Retained“ auf Seite 100
- „UltraLite-Synchronisationsdienstprogramm (ulsync)“ auf Seite 227
- `ul_sync_info`-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- `ULSyncParms.ResumePartialDownload`-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.resume_partial_download = ul_true;
```

Synchronisationsparameter Send Download Acknowledgement

Informiert den MobiLink-Server, dass der Client eine Downloadbestätigung sendet.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Sie können diesen Parameter auch mit `ulsync` einstellen.

Standardwert

FALSE

Zulässige Werte

Boolescher Wert

Siehe auch

- „UltraLite-Synchronisationsdienstprogramm (ulsync)“ auf Seite 227
- ul_sync_info-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- ULSyncParms.SendDownloadAck-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- SyncParms.setAcknowledgeDownload-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

ulsync kann diesen Parameter folgendermaßen als erweiterten Synchronisationsparameter einstellen:

```
ulsync -c DBF=myuldb.udb  
"MobiLinkId=remoteA;ScriptVersion=2;SendDownloadACK=true;Stream=http"
```

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.send_download_ack = ul_true;
```

Synchronisationsparameter Stream Error

Bietet eine Struktur, die Berichtsinformationen zu Kommunikationsfehlern enthält

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API.

Gilt für

Dieser Parameter gilt nur für C/C++-Schnittstellen.

Zulässige Werte

Der Parameter hat keinen Standardwert und muss mit einem der bereitgestellten Felder explizit eingestellt werden. Die Felder ul_stream_error lauten wie folgt:

- **stream_error_code** Hinweise zu Fehlercode-Suffixen finden Sie unter *%SQLANY16%\SDK\Include\sserror.h*.
- **system_error_code** Ein systemspezifischer Fehlercode. Weitere Hinweise zum Fehlercode finden Sie in Ihrer Plattformdokumentation. Für Windows-Plattformen ist dies die Microsoft Developer Network-Dokumentation.

Nachstehend finden Sie häufig vorkommende Systemfehler unter Windows:

- **10048 (WSAADDRINUSE)** Adresse wird bereits verwendet
- **10053 (WSAECONNABORTED)** Software hat Verbindungsabbruch verursacht
- **10054 (WSAECONNRESET)** Kommunikations-Gegenstelle hat den Socket geschlossen
- **10060 (WSAETIMEDOUT)** Zeit für Verbindung abgelaufen.
- **10061 (WSAECONNREFUSED)** Verbindung abgelehnt. Im Allgemeinen läuft der MobiLink-Server nicht oder er wartet auf dem angegebenen Port nicht auf Verbindungen. Siehe <http://msdn2.microsoft.com/en-us/library/ms740668.aspx>.
- **error_string** Eine Zeichenfolge mit zusätzlichen Informationen, sofern verfügbar, für stream_error_code. Die Zeichenfolge kann auch leer sein. Wenn error_string nicht leer ist, werden zusätzlich zum stream_error_code-Wert weitere Informationen übergeben. Bei einem Schreibfehler (Fehlercode 9) ist beispielsweise die Fehlerzeichenfolge eine Zahl, die angibt, wie viele Byte geschrieben werden sollten.

Bemerkungen

Andere UltraLite-Anwendungen als die UltraLite C++-Komponente empfangen Informationen über Kommunikationsfehler als Teil des Parameters Sync Result.

Das Feld stream_error ist eine Struktur vom Typ ul_stream_error.

```
typedef struct {
    ss_error_code stream_error_code;
    asa_uint16    alignment;
    asa_int32     system_error_code;
    char          error_string[UL_STREAM_ERROR_STRING_SIZE];
} ul_stream_error, * p_ul_stream_error;
```

Die Struktur ist definiert in %SQLANY16%\SDK\Include\sserror.h.

Überprüfen Sie, ob ein SQLE_MOBILINK_COMMUNICATIONS_ERROR vorliegt:

```
ULConnection * conn;
ul_sync_info info;
...
conn->InitSynchInfo( &info );
if( !conn->Synchronize( &info ) ) {
    ULError const * error = conn->GetLastError();
    char buf[256];
    if( error->GetSQLCode() == SQLE_MOBILINK_COMMUNICATIONS_ERROR ) {
        error->GetString( buf, sizeof(buf) );
        printf( "%s\n", buf );
        // more handling for communication error
    }
}
```

Siehe auch

- „Fehlermeldungen zur MobiLink-Kommunikation“ [*Fehlermeldungen*]
- „Synchronisationsparameter Sync Result“ auf Seite 109
- ul_sync_info-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]

Synchronisationsparameter Stream Type

Legt das MobiLink-Netzwerkprotokoll zur Verwendung in der Synchronisation fest

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Sie können diesen Parameter auch mit `ulsync` einstellen.

Bemerkungen

Dieser Parameter ist erforderlich. Er hat keinen Standardwert.

Die meisten Netzwerkprotokolle erfordern Protokolloptionen, um die Adresse des MobiLink-Servers und anderes Verhalten zu ermitteln. Diese Optionen werden in den Datenstromparametern übergeben.

Wenn das Netzwerkprotokoll eine Option erfordert, müssen Sie diese mit dem Parameter `Stream Parameters` übergeben. Andernfalls setzen Sie den Parameter `Stream Parameters` auf `NULL`.

Die folgenden Datenstromtypen sind verfügbar, allerdings sind nicht alle auf allen Zielplattformen verfügbar:

Netzwerkprotokoll	Beschreibung
HTTP	Synchronisieren über HTTP
HTTPS	Synchronisieren über HTTP Das HTTPS-Protokoll verwendet TLS als zugrunde liegende Sicherheitsschicht. Es arbeitet über TCP/IP.
TCP/IP	Synchronisieren über TCP/IP Dieses Protokoll wird von UltraLite Java Edition-Datenbanken nicht unterstützt.
TLS	Synchronisieren über TCP/IP mit Transport-Layer-Sicherheit (TLS). TLS sichert die Client-Server-Kommunikation durch digitale Zertifikate und Verschlüsselung mit öffentlichen Schlüsseln.

Eine Liste der unterstützten Plattformen finden Sie unter <http://www.sybase.com/detail?id=1061806>.

Siehe auch

- „Synchronisationsparameter Stream Parameters“ auf Seite 108
- „Dienstprogramm zum Erstellen von Zertifikaten [createcert]“ [*SQL Anywhere Server - Datenbankadministration*]
- „Dienstprogramm zum Anzeigen von Zertifikaten (viewcert)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Transportschichtsicherheit“ [*SQL Anywhere Server - Datenbankadministration*]
- „UltraLite-Synchronisationsdienstprogramm (ulsync)“ auf Seite 227
- „UltraLite-Netzwerkprotokolloptionen für dbmlsync“ auf Seite 114
- ul_sync_info-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- ULSyncParms.Stream-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Stellen Sie für Anwendungen in UltraLite für C/C++ Parameter folgendermaßen ein:

```
Connection conn;  
ul_sync_info info;  
...  
conn.InitSynchInfo( &info );  
info.stream = "http";
```

Synchronisationsparameter Stream Parameters

Setzt Optionen zum Konfigurieren des Netzwerkprotokolls

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Sie können diesen Parameter auch mit ulsync einstellen.

Standardwert

Null

Zulässige Werte

Zeichenfolge

Bemerkungen

Dieser Parameter ist optional. Er akzeptiert eine durch Semikolons getrennte Liste von Netzwerkprotokolloptionen. Jede Option hat die Form *Schlüsselwort=Wert*, wobei die zulässige Gruppe der Schlüsselwörter vom Netzwerkprotokoll abhängt.

Siehe auch

- „UltraLite-Synchronisationsdienstprogramm (ulsync)“ auf Seite 227
- „UltraLite-Netzwerkprotokolloptionen für dbmlsync“ auf Seite 114
- ul_sync_info-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- ULSyncParms.StreamParms-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- StreamHTTPParms-Schnittstelle [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.stream_parms= UL_TEXT( "host=myserver;port=2439" );
```

Synchronisationsparameter Sync Result

Berichtet über den Status einer Synchronisation

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API.

Bemerkungen

Dieser Parameter wird von UltraLite gesetzt und ist schreibgeschützt.

Die Schnittstelle für C/C++ erhält diese Information in separaten Parametern als Teil einer ul_sync_info-Struktur. Ansonsten wird diese Information als zusammengesetzter Parameter definiert, der eine Reihe von Informationen in separaten Feldern enthält:

- **Authentication Status** Berichtet über Erfolg oder Misserfolg der Authentifizierung.
- **Ignored Rows** Nennt die Anzahl der ignorierten Zeilen.
- **Stream Error-Informationen** Die Stream Error-Informationen umfassen den Stream Error-Code, den Stream Error-Kontext, die Stream Error-ID und das Stream Error-System.
- **Upload OK** Berichtet über Erfolg oder Misserfolg in der Upload-Phase.

Siehe auch

- „Synchronisationsparameter Authentication Status“ auf Seite 94
- „Synchronisationsparameter Ignored Rows“ auf Seite 97
- „Synchronisationsparameter Stream Error“ auf Seite 105
- „Synchronisationsparameter Upload OK“ auf Seite 110
- ULSyncParms-Klasse [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- ul_sync_result-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- ULGetSyncResult-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- SyncResult-Klasse [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Synchronisationsparameter Upload OK

Dieses Feld wird von einer Synchronisation verwendet, um den Status der auf den MobiLink-Server hochgeladenen Daten zu melden.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API.

Bemerkungen

Dieser Parameter wird von UltraLite gesetzt und ist damit schreibgeschützt.

Nach der Synchronisation enthält der Parameter **true**, wenn der Upload erfolgreich war, und **false** wenn nicht. Wenn es zu einem Synchronisationsfehler gekommen ist, können Sie diesen Parameter überprüfen, um festzustellen, ob die Daten erfolgreich hochgeladen wurden, bevor der Fehler aufgetreten ist.

Siehe auch

- ul_sync_info-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- ULSyncResult.UploadOK-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- SyncResult.isUploadOK-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

Anwendungen in UltraLite für C/C++ können folgendermaßen auf diese Parameter zugreifen:

```
ul_sync_info info;  
// ...  
returncode = info.upload_ok;
```

Synchronisationsparameter Upload Only

Weist darauf hin, dass bei der aktuellen Synchronisation keine Daten eingelesen werden sollen. Dies kann Verbindungszeit sparen, vor allem bei langsamen Kommunikationsverbindungen.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Sie können diesen Parameter auch mit `ulsync` einstellen.

Standardwert

FALSE

Zulässige Werte

Boolescher Wert

Konflikte mit

Download Only, Ping und Resume Partial Download

Bemerkungen

Wenn dieser Parameter den Wert TRUE hat, wartet der Client auf die Uploadbestätigung vom MobiLink-Server. Danach beendet er die Synchronisationssitzung erfolgreich.

Siehe auch

- „UltraLite-Clientsynchronisationsplanung“ auf Seite 74
- „Synchronisationsparameter Download Only“ auf Seite 96
- „UltraLite-Synchronisationsdienstprogramm (ulsync)“ auf Seite 227
- `ul_sync_info`-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- `ULSyncParms.UploadOnly`-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- `SyncParms.isUploadOnly`-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

`ulsync` kann diesen Parameter folgendermaßen als erweiterten Synchronisationsparameter einstellen:

```
ulsync -c DBF=myuldb.udb  
"MobiLinkUid=remoteA;ScriptVersion=2;UploadOnly=True;Stream=http"
```

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.upload_only = ul_true;
```

Synchronisationsparameter User Data

Übergibt anwendungsspezifische Informationen an die Synchronisations-Beobachtungsfunktion

Gilt für

Nur C/C++-Anwendungen. Andere Komponenten, wie z.B. UltraLite.NET, erfordern keinen separaten Parameter, um Benutzerdaten zu verarbeiten, und haben daher keinen Parameter User Data.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API.

Bemerkungen

Bei der Implementierung der Callback-Funktion oder des Event-Handlers innerhalb der Synchronisations-Beobachtungsfunktion können Sie anwendungsspezifische Informationen über den Parameter User Data bereitstellen.

Siehe auch

- „Synchronisationsparameter Observer“ auf Seite 99
- `ul_sync_info`-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]

Synchronisationsparameter User Name

Erforderlich. Eine Zeichenfolge, die der MobiLink-Server zur Authentifizierung verwendet

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Sie können diesen Parameter auch mit `ulsync` einstellen.

Bemerkungen

Dieser Parameter ist erforderlich. Leere Zeichenfolgen und NULL-Zeichenfolgen werden generell verworfen.

Der Parameter hat keinen Standardwert und muss explizit festgelegt werden.

Der Benutzername muss bei der Verwendung einer entfernten ID nicht eindeutig sein.

Dieser MobiLink-Benutzername und das Kennwort unterscheiden sich von der ID und dem Kennwort des Datenbankbenutzers und dienen nur der Kennzeichnung und Authentifizierung der Anwendung beim MobiLink-Server.

Damit ein Benutzer Teil eines Synchronisationssystems werden kann, müssen Sie den Benutzernamen im MobiLink-Server registrieren. Der Benutzername wird in der Spalte `name` in der MobiLink-Systemtabelle `ml_user` in der konsolidierten Datenbank gespeichert.

Siehe auch

- „Entfernte IDs“ [*MobiLink - Clientadministration*]
- „Synchronisationsparameter Password“ auf Seite 101
- „MobiLink-Benutzer“ [*MobiLink - Clientadministration*]
- UltraLite-Benutzerauthentifizierung [*MobiLink - Clientadministration*]
- „UltraLite-Synchronisationsdienstprogramm (ulsync)“ auf Seite 227
- ul_sync_info-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- ULSyncParms.UserName-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- SyncParms.setUserName-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

ulsync kann diesen Parameter folgendermaßen als erweiterten Synchronisationsparameter einstellen:

```
ulsync -c DBF=myuldb.udb "MobiLinkId=remoteA;ScriptVersion=2;Stream=http"
```

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.user_name= UL_TEXT( "remoteA" );
```

Synchronisationsparameter Version

Legt die Version der konsolidierten Datenbank fest.

Syntax

Die Syntax variiert je nach der von Ihnen verwendeten API. Sie können diesen Parameter auch mit ulsync einstellen.

Zulässige Werte

Zeichenfolge

Bemerkungen

Dieser Parameter ist erforderlich. Leere Zeichenfolgen und NULL-Zeichenfolgen werden generell verworfen.

Jedes Synchronisationsskript in der konsolidierten Datenbank wird mit einer Versionszeichenfolge markiert. Es können z.B. zwei verschiedene download_cursor-Skripten vorhanden sein, die durch unterschiedliche Versionszeichenfolgen gekennzeichnet werden.

Siehe auch

- „Skriptversionen“ [*MobiLink - Serveradministration*]
- „UltraLite-Synchronisationsdienstprogramm (ulsync)“ auf Seite 227
- ul_sync_info-Struktur [UltraLite C- und Embedded SQL-Datentypen] [*UltraLite - C- und C++-Programmierung*]
- ULSyncParms.Version-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- SyncParms.setVersion-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Beispiel

ulsync kann diesen Parameter folgendermaßen als erweiterten Synchronisationsparameter einstellen:

```
ulsync -c DBF=myuldb.udb "MobiLinkUid=remoteA;ScriptVersion=2;Stream=http"
```

Für Anwendungen in UltraLite für C/C++ kann der Parameter folgendermaßen eingestellt werden:

```
ul_sync_info info;  
// ...  
info.version = UL_TEXT( "default" );
```

UltraLite-Netzwerkprotokolloptionen für dbmlsync

Das Netzwerkprotokoll muss in der Anwendung festgelegt werden. Jede UltraLite-Datenbank, die mit einem MobiLink-Server synchronisiert, verwendet ein Netzwerkprotokoll. Als Netzwerkprotokolle sind TCP/IP, HTTP, HTTPS und TLS verfügbar. Außerdem wird die ActiveSync-Benachrichtigung unter Windows Mobile unterstützt.

Für das verwendete Netzwerkprotokoll können Sie verschiedene Protokolloptionen wählen, um sicherzustellen, dass die UltraLite-Anwendung den MobiLink-Server lokalisieren und eine korrekte Verbindung mit ihm aufbauen kann. Die Netzwerkprotokolloptionen für MobiLink-Client bieten Informationen wie beispielsweise Adressinformationen (Host und Port) und protokollspezifische Informationen.

Siehe auch

- „Netzwerkprotokolloptionen des MobiLink-Clients“ [*MobiLink - Clientadministration*]
- „UltraLite-Clients für die Verwendung von Transportschichtssicherheit konfigurieren“ [*SQL Anywhere Server - Datenbankadministration*]
- „Netzwerkprotokolloptionen des MobiLink-Clients“ [*MobiLink - Clientadministration*]
- „Synchronisationsparameter Stream Parameters“ auf Seite 108
- Option -x unter „UltraLite-Synchronisationsdienstprogramm (ulsync)“ auf Seite 227

Optionen für den Synchronisationsdatenstrom

Sie können die Informationen, die zur Lokalisierung des MobiLink-Servers benötigt werden, in Ihrer Anwendung durch Einstellen des Parameters StreamParameter angeben.

Siehe auch

- „Synchronisationsparameter Stream Parameters“ auf Seite 108
- ULSyncParms.StreamParms-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

UltraLite-Deployment

In den meisten Fällen findet die UltraLite-Entwicklung für das mobile Gerät auf einem Windows-PC oder unter Mac OS X statt. Abhängig von Ihrer Deployment-Umgebung können Sie allerdings verschiedene Deployment-Verfahren für die Installation von UltraLite verwenden.

UltraLite-Anwendungsprojekte können zu verschiedenen Erscheinungsformen derselben UltraLite-Datenbank führen: einer Entwicklungsdatenbank, einer Testdatenbank und einer bereitgestellten Produktionsdatenbank. Im Lebenszyklus einer per Deployment bereitgestellten Datenbankanwendung werden Änderungen und Verbesserungen zuerst in der Entwicklungsdatenbank durchgeführt und abschließend von der Testdatenbank übernommen, um schließlich an die Produktionsdatenbank verteilt zu werden.

Die Module, die Sie für Ihre UltraLite-Anwendung verwenden müssen, hängen von der Plattform ab, für die Sie entwickeln, von der benutzten Schnittstelle und den gewünschten Funktionen.

Siehe auch

- „UltraLite-Datenverwaltungskomponenten für Windows Mobile“ auf Seite 19
- „So erstellen und Sie UltraLite C++-Anwendungen und führen ein Deployment durch.“ [*UltraLite - C- und C++-Programmierung*]
- „UltraLite.NET-Anwendungsentwicklung“ [*UltraLite - .NET-Programmierung*]
- „So erstellen Sie UltraLiteJ-Anwendungen und stellen sie per Deployment bereit“ [*UltraLite® – Java-Programmierung*]

Kompilierungs- und Deploymentspezifikationen für UltraLite-Anwendungen

Dieser Abschnitt enthält Tabellen mit den Kompilierungs- und Deploymentanforderungen für UltraLite-Anwendungen.

Hinweis

Es gibt Versionen der UltraLite Engine, die sich in Verzeichnissen mit dem Suffix **_dev** befinden, beispielsweise im Verzeichnis *x86_dev*. Diese Versionen enthalten Protokollierungsfunktionen für den Entwicklungsvorgang, die verwendet werden können, um Probleme auf Plattformen für Debugging-Zwecke zu diagnostizieren. Die Logausgabe ist für Sybase-Techniker und nicht für die Verwendung von Kunden vorgesehen. Bei Produktionssystemen verwenden Sie die Version der Engine, die sich NICHT in einem **_dev**-Verzeichnis befindet.

Kompilierungs- und Deploymentanforderungen für UltraLite-Anwendungen und Verschlüsselung

Die folgende Tabelle beschreibt die Mindestanforderungen für die Kompilierung und das Deployment einer UltraLite-Anwendung für alle unterstützten Plattformen und Geräte, einschließlich der Anforderungen für die UltraLite-Datenbankverschlüsselung.

Plattform oder Gerät	Mindestanforderungen	AES- Verschlüsselungsanforderungen	FIPS 140-2 AES- Verschlüsselungsanforderungen
Windows Mobile und Desktop (UltraLite C/C++ mit statischer Verknüpfung)	Verknüpfung mit: <ul style="list-style-type: none"> • <i>ulrt.lib</i>¹ • <i>ulbase.lib</i>¹ 	Verwenden Sie den DBKEY-Erstellungsparameter zum Festlegen des Chiffrierschlüssels, wenn Sie die Datenbank erstellen oder eine Verbindung mit der Datenbank herstellen. Rufen Sie die EnableAesDBEncryption-Methode auf.	Legen Sie beim Erstellen und Verbinden mit der Datenbank den DBKEY-Verbindungsparameter mit dem Chiffrierschlüssel fest. Legen Sie den Erstellungsparameter mit fips=yes fest, wenn Sie die Datenbank erstellen. Rufen Sie die EnableAesFipsDBEncryption-Methode auf. Deployment vornehmen: <ul style="list-style-type: none"> • <i>ulfips16.dll</i>² • <i>sbgse2.dll</i>²
Windows Mobile und Desktop (UltraLite C/C++ mit dynamischer Verknüpfung)	Verknüpfung mit: <ul style="list-style-type: none"> • <i>ulimp.lib</i>^{1, 10} • <i>ulbase.lib</i>¹ Deployment vornehmen: <ul style="list-style-type: none"> • <i>ulrt16.dll</i>¹ 	Verwenden Sie den DBKEY-Erstellungsparameter zum Festlegen des Chiffrierschlüssels, wenn Sie die Datenbank erstellen oder eine Verbindung mit der Datenbank herstellen. Rufen Sie die EnableAesDBEncryption-Methode auf.	Legen Sie beim Erstellen und Verbinden mit der Datenbank den DBKEY-Verbindungsparameter mit dem Chiffrierschlüssel fest. Legen Sie den Erstellungsparameter mit fips=yes fest, wenn Sie die Datenbank erstellen. Rufen Sie die EnableAesFipsDBEncryption-Methode auf. Deployment vornehmen: <ul style="list-style-type: none"> • <i>ulfips16.dll</i>² • <i>sbgse2.dll</i>²

Plattform oder Gerät	Mindestanforderungen	AES- Verschlüsselungsanforderungen	FIPS 140-2 AES- Verschlüsselungsanforderungen
Windows Mobile und Desktop (UltraLite C/C++ mit der UltraLite-Engine)	<p>Verknüpfung mit:</p> <ul style="list-style-type: none"> • <i>ulrtc.lib</i>¹ • <i>ulbase.lib</i>¹ <p>Deployment vornehmen:</p> <ul style="list-style-type: none"> • <i>uleng16.exe</i>² 	<p>Verwenden Sie den DBKEY-Erstellungsparameter zum Festlegen des Chiffrierschlüssels, wenn Sie die Datenbank erstellen oder eine Verbindung mit der Datenbank herstellen.</p>	<p>Legen Sie beim Erstellen und Verbinden mit der Datenbank den DBKEY-Verbindungsparameter mit dem Chiffrierschlüssel fest.</p> <p>Legen Sie den Erstellungsparameter mit fips=yes fest, wenn Sie die Datenbank erstellen.</p> <p>Deployment vornehmen:</p> <ul style="list-style-type: none"> • <i>ulfips16.dll</i>² • <i>sbgse2.dll</i>²
Windows Mobile- und Desktop (UltraLite.NET)	<p>Hinzufügen der Referenzen zu:</p> <ul style="list-style-type: none"> • iAnywhere.Data.UltraLite • iAnywhere.Data.UltraLite.resources <p>Deployment vornehmen:</p> <ul style="list-style-type: none"> • <i>iAnywhere.Data.UltraLite.dll</i>⁷ • <i>iAnywhere.Data.UltraLite.resources.dll</i>⁸ • <i>ulnet16.dll</i>⁶ 	<p>Verwenden Sie den DBKEY-Erstellungsparameter zum Festlegen des Chiffrierschlüssels, wenn Sie die Datenbank erstellen oder eine Verbindung mit der Datenbank herstellen.</p>	<p>Legen Sie beim Erstellen und Verbinden mit der Datenbank den DBKEY-Verbindungsparameter mit dem Chiffrierschlüssel fest.</p> <p>Legen Sie den Erstellungsparameter mit fips=yes fest, wenn Sie die Datenbank erstellen.</p> <p>Deployment vornehmen:</p> <ul style="list-style-type: none"> • <i>ulfips16.dll</i>² • <i>sbgse2.dll</i>²

Plattform oder Gerät	Mindestanforderungen	AES-Verschlüsselungsanforderungen	FIPS 140-2 AES-Verschlüsselungsanforderungen
Windows Mobile und Desktop (UltraLite.NET mit der UltraLite-Engine)	<p>Hinzufügen der Referenzen zu:</p> <ul style="list-style-type: none"> • iAnywhere.Data.UltraLite • iAnywhere.Data.UltraLite.resources <p>Deployment vornehmen:</p> <ul style="list-style-type: none"> • <i>iAnywhere.Data.UltraLite.dll</i>⁷ • <i>iAnywhere.Data.UltraLite.resources.dll</i>⁸ • <i>ulnetclient16.dll</i>⁶ • <i>uleng16.exe</i>² 	Verwenden Sie den DBKEY-Erstellungsparameter zum Festlegen des Chiffrierschlüssels, wenn Sie die Datenbank erstellen oder eine Verbindung mit der Datenbank herstellen.	<p>Legen Sie beim Erstellen und Verbinden mit der Datenbank den DBKEY-Verbindungsparameter mit dem Chiffrierschlüssel fest.</p> <p>Legen Sie den Erstellungsparameter mit fips=yes fest, wenn Sie die Datenbank erstellen.</p> <p>Deployment vornehmen:</p> <ul style="list-style-type: none"> • <i>ulfips16.dll</i>² • <i>sbgse2.dll</i>²
Mac OS X und iOS (UltraLite C/C++)	<p>Hinzufügen zu Ihrem Xcode-Projekt:</p> <ul style="list-style-type: none"> • <i>libulrt.a</i>⁹ • <i>libulbase.a</i>⁹ (nur Mac OS X) 	<p>Verwenden Sie den DBKEY-Erstellungsparameter zum Festlegen des Chiffrierschlüssels, wenn Sie die Datenbank erstellen oder eine Verbindung mit der Datenbank herstellen.</p> <p>Rufen Sie die EnableAesDBEncryption-Methode auf.</p>	Nicht anwendbar
Linux (UltraLite C/C++)	<p>Verknüpfung mit:</p> <ul style="list-style-type: none"> • <i>libulrt.a</i>³ • <i>libulbase.a</i>³ 	<p>Verwenden Sie den DBKEY-Erstellungsparameter zum Festlegen des Chiffrierschlüssels, wenn Sie die Datenbank erstellen oder eine Verbindung mit der Datenbank herstellen.</p> <p>Rufen Sie die EnableAesDBEncryption-Methode auf.</p>	Nicht anwendbar

Plattform oder Gerät	Mindestanforderungen	AES-Verschlüsselungsanforderungen	FIPS 140-2 AES-Verschlüsselungsanforderungen
Android (UltraLiteJ)	<p>Hinzufügen zu Ihrem Android-Projekt:</p> <ul style="list-style-type: none"> • <i>UltraLiteJNI16.jar</i>⁵ • <i>libultralitej16.so</i>⁴ 	<p>Verwenden Sie den DBKEY-Erstellungsparameter oder die setEncryptionKey-Methode, um den Chiffrierschlüssel festzulegen, wenn Sie die Datenbank erstellen oder eine Verbindung mit der Datenbank herstellen.</p> <p>Rufen Sie die EnableAesDBEncryption-Methode auf.</p>	Nicht anwendbar
Black-Berry (UltraLiteJ)	<p>Deployment vornehmen:</p> <ul style="list-style-type: none"> • <i>UltraLiteJ16.cod</i>¹¹ • <i>UltraLiteJ16.jad</i>¹¹¹² 	<p>Verwenden Sie die setEncryptionKey-Methode, um den Chiffrierschlüssel festzulegen, wenn Sie die Datenbank erstellen oder eine Verbindung mit der Datenbank herstellen.</p> <p>Rufen Sie die EnableAesDBEncryption-Methode auf.</p>	Nicht anwendbar

¹ Unter Windows Mobile befindet sich diese Datei in %SQLANY16%\UltraLite\CE\Arm.50\Lib. Unter Windows befindet sie sich in %SQLANY16%\UltraLite\Windows\x64\Lib\VS9 oder %SQLANY16%\UltraLite\Windows\x86\Lib\VS9.

² Unter Windows Mobile befindet sich diese Datei in %SQLANY16%\UltraLite\CE\Arm.50. Unter Windows befindet sie sich in %SQLANY16%\UltraLite\Windows\x64 oder %SQLANY16%\UltraLite\Windows\x86.

³ Diese Datei befindet sich in /opt/sqlanywhere16/ultralite/linux/x64/lib.

⁴ Diese Datei befindet sich in %SQLANY16%\UltraLite\UltraLiteJ\Android\ARM.

⁵ Diese Datei befindet sich in %SQLANY16%\UltraLite\UltraLiteJ\Android.

⁶ Unter Windows Mobile befindet sich diese Datei in %SQLANY16%\UltraLite\UltraLite.NET\CE\Arm.50. Unter Windows Mobile befindet sich diese Datei in %SQLANY16%\UltraLite\UltraLite.NET\x64 oder %SQLANY16%\UltraLite\UltraLite.NET\win32.

⁷ Diese Datei befindet sich in %SQLANY16%\UltraLite\UltraLite.NET\Assembly\V2.

⁸ Diese Datei befindet sich in %SQLANY16%\UltraLite\UltraLite.NET\Assembly\V2\de.

⁹ Unter Mac OS X befindet sich diese Datei in /Applications/SQLAnywhere16/System/ultralite/macosx/x86_64. Unter iOS müssen die UltraLite-Laufzeitdateien nach der Installation erstellt werden. Folgen Sie den Anweisungen in install-dir/ultralite/iphone/readme.txt.

¹⁰ Wenn eine Verknüpfung mit dieser Bibliothek erstellt wird, definieren Sie den UL_USE_DLL-Präprozessormakro beim Kompilieren. Beispielsweise kann die Eingabe so lauten:

```
-DUL_USE_DLL
```

¹¹ Diese Datei befindet sich in %SQLANY16%\UltraLite\UltraLiteJ\BlackBerry4.2.

¹² Nur für OTA-Deployment (Over-The-Air) erforderlich. Als Alternative können Sie Ihre eigene JAD-Datei erstellen, die das Deployment von UltraLiteJ mit Ihrer Anwendung vornimmt.

Hinweis

Um die UltraLite-Datenbankverschleierung für jede Plattform und jedes Gerät zu starten, müssen Sie bei der Erstellung der Datenbank den Erstellungsparameter **obfuscate=1** angeben.

Zusätzliche Kompilierungs- und Deploymentanforderungen für die Synchronisation und Komprimierung

Die folgende Tabelle zeigt die Datenstrom-, Protokolloptions- und Codeanforderungen für die Kompilierung und Bereitstellung einer UltraLite-Anwendung, die die Synchronisation verwendet.

Hinweis

Die HTTPS-Datenstromoption kann in der UltraLiteJ API durch Übergabe der SyncParms.HTTPS_STREAM-Konstante an die Connection.createSyncParms-Methode aktiviert werden.

Synchronisationstyp	Spezifikation der Datenstromoption	Anforderungen für die Protokolloption	Methodenaufrieforderungen für UltraLite C und C++
TCP/IP	"tcpip"	Keine	<ul style="list-style-type: none">• EnableTcpipSynchronization
HTTP	"http"	Keine	<ul style="list-style-type: none">• EnableHttpSynchronization
RSA_TLS	"tls"	Keine	<ul style="list-style-type: none">• EnableTlsSynchronization• EnableRsaSyncEncryption
RSA HTTPS	"https"	Keine	<ul style="list-style-type: none">• EnableHttpsSynchronization• EnableRsaSyncEncryption
RSA FIPS 140-2 TLS	"tls"	fips=yes	<ul style="list-style-type: none">• EnableTlsSynchronization• EnableRsaFipsSyncEncryption
RSA FIPS 140-2 HTTPS	"https"	fips=yes	<ul style="list-style-type: none">• EnableHttpsSynchronization• EnableRsaFipsSyncEncryption

Die folgende Tabelle zeigt die Protokolloptions- und Codeanforderungen für die Kompilierung und Bereitstellung einer UltraLite-Anwendung, die die Komprimierung oder Ende-zu-Ende-Verschlüsselung verwendet:

Komprimierungs- und Datenstrom-Verschlüsselungsoptionen	Anforderungen für die Protokolloption	Methodenaufananforderungen für UltraLite C/C++ und UltraLiteJ
ZLIB-Komprimierung	<ul style="list-style-type: none"> compression=zlib 	<ul style="list-style-type: none"> C/C++: EnableZlibSyncCompression Java: setZlibCompression
RSA E2EE	<ul style="list-style-type: none"> e2ee_public_key=Schlüssel-datei 	<ul style="list-style-type: none"> C/C++: EnableRsaE2ee Java: setE2eePublicKey
RSA FIPS 140-2 E2EE	<ul style="list-style-type: none"> e2ee_public_key=Schlüssel-datei fips=yes 	<ul style="list-style-type: none"> C/C++: EnableRsaFipsE2ee Java: Nicht anwendbar

Die folgende Tabelle zeigt zusätzliche Kompilierungs- und Deploymentanforderungen für Komprimierung und verschlüsselte Synchronisation.

Hinweis

Es gibt keine zusätzlichen Kompilierungs- und Deploymentanforderungen für TCP/IP- und HTTP-Synchronisation.

Plattform oder Gerät	Anforderungen für die ZLIB-Komprimierung	RSA TLS-, RSA HTTPS- und RSA E2EE-Anforderungen	RSA FIPS 140-2 TLS, RSA FIPS 140-2 HTTPS, und RSA FIPS 140-2 E2EE-Anforderungen
Windows Mobile und Desktop (UltraLite C/C++ mit statischer Verknüpfung)	Keine	Verknüpfung mit: <ul style="list-style-type: none"> <i>ulrsa.lib</i>¹ 	Deployment vornehmen: <ul style="list-style-type: none"> <i>mlcrsafips16.dll</i>² <i>sbgs2.dll</i>²
Windows Mobile und Desktop (UltraLite C/C++ mit dynamischer Verknüpfung)	Deployment vornehmen: <ul style="list-style-type: none"> <i>mlczlib16.dll</i>² 	Deployment vornehmen: <ul style="list-style-type: none"> <i>mlcrsa16.dll</i>² 	Deployment vornehmen: <ul style="list-style-type: none"> <i>mlcrsafips16.dll</i>² <i>sbgs2.dll</i>²
Windows Mobile und Desktop (UltraLite C/C++ mit der UltraLite-Engine)	Deployment vornehmen: <ul style="list-style-type: none"> <i>mlczlib16.dll</i>² 	Deployment vornehmen: <ul style="list-style-type: none"> <i>mlcrsa16.dll</i>² 	Deployment vornehmen: <ul style="list-style-type: none"> <i>mlcrsafips16.dll</i>² <i>sbgs2.dll</i>²

Plattform oder Gerät	Anforderungen für die ZLIB-Komprimierung	RSA TLS-, RSA HTTPS- und RSA E2EE-Anforderungen	RSA FIPS 140-2 TLS, RSA FIPS 140-2 HTTPS, und RSA FIPS 140-2 E2EE-Anforderungen
Windows Mobile- und Desktop (UltraLite.NET)	Deployment vornehmen: <ul style="list-style-type: none"> <i>mlczlib16.dll</i>² 	Deployment vornehmen: <ul style="list-style-type: none"> <i>mlcrsa16.dll</i>² 	Deployment vornehmen: <ul style="list-style-type: none"> <i>mlcrsafips16.dll</i>² <i>sbgse2.dll</i>²
Windows Mobile und Desktop (UltraLite.NET mit der UltraLite-Engine)	Deployment vornehmen: <ul style="list-style-type: none"> <i>mlczlib16.dll</i>² 	Deployment vornehmen: <ul style="list-style-type: none"> <i>mlcrsa16.dll</i>² 	Deployment vornehmen: <ul style="list-style-type: none"> <i>mlcrsafips16.dll</i>² <i>sbgse2.dll</i>²
Mac OS X und iOS (UltraLite C/C++)	Keine	Keine	Nicht anwendbar
Linux (UltraLite C/C++)	Keine	Verknüpfung mit: <ul style="list-style-type: none"> <i>libulrsa.a</i>³ 	Nicht anwendbar
Android (UltraLiteJ)	Keine	Deployment vornehmen: <ul style="list-style-type: none"> <i>libmlcrsa16.so</i>⁴ 	Nicht anwendbar
BlackBerry (UltraLiteJ)	Keine	Übertragen Sie die DER-kodierte Datei mithilfe der FileTransfer-Schnittstelle oder speichern Sie sie auf einer SD-Karte.	Nicht anwendbar

¹ Unter Windows Mobile befindet sich diese Datei in %SQLANY16%\UltraLite\CE\Arm.50\Lib. Unter Windows befindet sie sich in %SQLANY16%\UltraLite\Windows\x64\Lib\VS9 oder %SQLANY16%\UltraLite\Windows\x86\Lib\VS9.

² Unter Windows Mobile befindet sich diese Datei in %SQLANY16%\CE\Arm.50. Unter Windows befindet sie sich in %SQLANY16%\Windows\x64 oder %SQLANY16%\Windows\x86.

³ Diese Datei befindet sich in /opt/sqlanywhere16/ultralite/linux/x64/lib.

⁴ Diese Datei befindet sich in %SQLANY16%\UltraLite\UltraLiteJ\Android\ARM.

Siehe auch

- „So erstellen und Sie UltraLite C++-Anwendungen und führen ein Deployment durch.“ [[UltraLite - C- und C++-Programmierung](#)]
- „UltraLite.NET-Anwendungen erstellen und das Deployment durchführen“ [[UltraLite - .NET-Programmierung](#)]
- „So erstellen Sie UltraLiteJ-Anwendungen und stellen sie per Deployment bereit“ [[UltraLite® – Java-Programmierung](#)]
- „Netzwerkprotokolloptionen des MobiLink-Clients“ [[MobiLink - Clientadministration](#)]

Datenbank-Deploymenttechniken für UltraLite und UltraLite Java Edition

Sie können eine der folgenden Methoden verwenden, um die Ausgangsdatenbankdatei auf das Gerät zu übertragen:

- Verwenden Sie die UltraLite-API in Ihrer Anwendung, um die Ausgangsdatenbankdatei zu erstellen.
- Erstellen Sie eine Schemadatei aus einem SQL-Skript und verwenden Sie die Anweisung ALTER DATABASE SCHEMA FROM FILE.¹
- Verwenden Sie die UltraLite FileTransfer-Methoden zum Download der ursprünglichen Datenbankdatei, falls sie auf dem Gerät nicht bereits existiert.
- Bündeln Sie die ursprüngliche Datenbank mit der Anwendung.¹
- Wenn Sie das Deployment auf einem Windows- oder Windows Mobile-Gerät vornehmen, verwenden Sie die zentrale Administration, um die UDB-Datei herunterzuladen, oder senden Sie einen Befehl, um die Ausgangsdatenbankdatei zu erstellen und führen Sie eine SQL-Anweisung aus, um sie mit einem Schema zu versorgen.¹

¹ Nicht unterstützt für UltraLite Java Edition-Datenbanken.

Siehe auch

- „Deployment von UltraLite-Datenbankschema-Upgrades“ auf Seite 125
- „ALTER DATABASE SCHEMA FROM FILE-Anweisung [UltraLite] “ auf Seite 423
- „Zentrale Administration von entfernten Datenbanken“ [[MobiLink - Serveradministration](#)]

Deployment von UltraLite-Datenbankschema-Upgrades

Führen Sie ein Schema-Upgrade durch.

Voraussetzungen

Die verwendete SQL-Datei muss das komplette neue Schema enthalten.

Kontext und Bemerkungen

UltraLite-Datenbankschema-Upgrades können mit einer der folgenden Methoden bereitgestellt werden:

- **Einzelne DDL-Anweisungen** Beispiel: In UltraLite C können Sie die folgende Anweisung ausführen, um eine neue Publikation zu erstellen:

```
dbconnection->ExecuteStatement("CREATE PUBLICATION p (table t)");
```

- **ALTER DATABASE SCHEMA FROM FILE-Anweisung** Diese Anweisung kann für die Durchführung von Schema-Upgrades verwendet werden, wenn Sie die DDL-Anweisungsanforderungen nicht kennen oder die einzelnen DDL-Anweisungen nicht angeben möchten.

Vorsicht

Setzen Sie ein Gerät nicht während des Schema-Upgrades zurück. Wenn Sie das Gerät während eines Schema-Upgrades zurücksetzen, gehen Daten verloren und die UltraLite-Datenbank wird als "beschädigt" gekennzeichnet.

UltraLite führt die folgenden Schritte aus, wenn Sie ein Upgrade eines UltraLite-Datenbankschemas mit der ALTER DATABASE SCHEMA FROM FILE-Anweisung verwenden:

1. Das neue und das bestehende Datenbankschema werden miteinander verglichen, um die Unterschiede zu ermitteln.
2. Das Schema der bestehenden Datenbank wird geändert.
3. Zeilen, die nicht dem neuen Schema entsprechen, werden gelöscht. Beispiel:
 - Wenn Sie eine Eindeutigkeits-Integritätsregel einer Tabelle hinzufügen und mehrere Zeilen mit denselben Werten vorhanden sind, werden alle Zeilen bis auf eine gelöscht.
 - Wenn bei dem Versuch, eine Spaltendomäne zu ändern, ein Konvertierungsfehler auftritt, wird die betreffende Zeile gelöscht. Beispiel: Wenn Sie eine VARCHAR-Spalte haben, sie in eine INT-Spalte konvertieren und der Wert einer Zeile ABCD ist, wird die Zeile gelöscht.
 - Wenn Ihr neues Schema neue Fremdschlüssel aufweist und die Fremdschlüsselzeile keine passende Primärschlüsselzeile hat, werden diese Zeilen gelöscht.
4. Wenn Zeilen gelöscht werden, wird eine Warnung `SQL_ROW_DROPPED_DURING_SCHEMA_UPGRADE (130)` ausgegeben.

Aufgabe

1. Erstellen Sie ein SQL-Skript der DDL-Anweisungen zum Erstellen eines völlig neuen Schemas.

Sie können ein Masterschema auf Ihrem Computer aufbewahren und das Schema aktualisieren, wenn sich Ihre Anwendung ändert.

Verwenden Sie die Dienstprogramme `ulinit` oder `ulunload` zum Extrahieren der für Ihr Skript erforderlichen DDL-Anweisungen. Indem Sie diese Dienstprogramme mit den folgenden Optionen verwenden, stellen Sie sicher, dass die DDL-Anweisungen syntaktisch korrekt sind:

- Für eine UltraLite-Datenbank verwenden Sie das Dienstprogramm `ulunload` mit den Optionen `-n` und `-s [Schemadatei]`. Beispiel:

```
ulunload -c dbf=mydatabase.udb -n -s MySchema.sql
```

- Für eine SQL Anywhere-Datenbank verwenden Sie das Dienstprogramm ulinit mit den Optionen -a und -l [Schemadatei]. Beispiel:

```
ulinit -a "dsn=mysqlanywheredatabase" -l MySchema.sql
```

Wenn Sie weder ulunload noch ulinit verwenden, überprüfen Sie das Skript und stellen Sie Folgendes sicher:

- Das Skript deklariert das gesamte gewünschte Schema mit CREATE-Anweisungen.
 - Tabellen, Spalten und Publikationen werden nicht umbenannt. Der RENAME-Vorgang wird nicht unterstützt. Umbenannte Tabellen werden als DROP TABLE- und CREATE TABLE-Vorgang verarbeitet.
 - Es gibt keine Non-DDL Anweisungen, auch keine Non-DDL Anweisungen, die möglicherweise nicht den erwarteten Effekt haben.
 - Wörter in der SQL-Anweisung werden durch Leerstellen getrennt.
 - Nur eine SQL-Anweisung darf in der jeweiligen Zeile erscheinen.
 - Kommentaren wird ein doppelter Bindestrich (-) vorangestellt, und sie treten nur am Anfang der Zeile auf.
 - Jede Anweisung wird durch eine Zeile getrennt, die genau dem Wort **GO** entspricht.
2. Nehmen Sie das Deployment der neuen SQL-Skriptdatei vor.
 3. Vergewissern Sie sich, dass die Datenbank synchronisiert ist.
 4. Führen Sie die neue Anweisung auf dem Gerät aus. Beispiel:

```
ALTER DATABASE SCHEMA FROM FILE 'MySchema.sql '
```

Ergebnisse

Das Schema wird aktualisiert.

Fehlerbenachrichtigung

Da das UltraLite-Fehler-Callback während des Upgrade-Vorgangs aktiv ist, werden Sie über Fehler während des Konvertierungsprozesses benachrichtigt. SQLE_CONVERSION_ERROR z.B. gibt in seinen Parametern Auskunft über alle Werte, die nicht konvertiert werden konnten. Fehler bedeuten *nicht*, dass der Vorgang fehlgeschlagen ist. Der letzte SQL-Code nach der Anweisung gibt in diesem Fall eine 130-Warnung zurück. Diese Warnungen beschreiben Vorgänge des Konvertierungsprozesses und stoppen nicht den Upgrade-Prozess.

Siehe auch

- „UltraLite-Datenbankschemas“ auf Seite 49
- „ALTER DATABASE SCHEMA FROM FILE-Anweisung [UltraLite]“ auf Seite 423
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload)“ auf Seite 233

Start der UltraLite-Engine

Bei der Verwendung der UltraLite-Engine zur Verwaltung von Daten auf einem Windows- oder Windows Mobile-Gerät startet Ihre UltraLite-Anwendung die Engine automatisch, außer wenn die Anwendung explizit den Speicherort des Verzeichnisses der Engine übergeben muss.

Wenn eine UltraLite-Anwendung versucht, die UltraLite-Engine zu starten, durchsucht die Anwendung folgende Verzeichnisse:

Clientplattform	Verzeichnisspeicherorte
Windows-Desktop	<ol style="list-style-type: none">1. Verzeichnis der Anwendung, die sie startet2. Aktuelles Arbeitsverzeichnis3. Systempfad4. SQL Anywhere-Installationsverzeichnis (entweder unter <i>bin32</i> oder <i>bin64</i>), abhängig davon, ob ein 32-Bit- oder 64-Bit-Client vorliegt
Windows Mobile/CE	<ol style="list-style-type: none">1. \Windows\2. \ (das Stammverzeichnis)3. \UltraLiteDB\
Linux	<ol style="list-style-type: none">1. Verzeichnis der Anwendung, die sie automatisch startet2. %SQLANY16%/bin32

Wenn die UltraLite-Engine an einer anderen Stelle gespeichert wird, starten Sie die Engine, indem Sie den Verbindungsparameter START angeben.

Beispiel: Eine Verbindungszeichenfolge für die Datenbank oder ein Verbindungscode für eine Windows Mobile-Clientanwendung könnte diesen START-Parameterwert verwenden:

```
"START=\Program Files\MyApp\ulengl6.exe"
```

Siehe auch

- „Kompilierungs- und Deploymentspezifikationen für UltraLite-Anwendungen“ auf Seite 117
- „UltraLite-Verbindungsparameter START“ auf Seite 189

Deployment des ActiveSync-Providers für UltraLite

Wenn Sie das Deployment von UltraLite für einen Endbenutzer durchführen, müssen Sie den ActiveSync-Provider manuell auf dem Computer des Endbenutzers installieren und registrieren. Diese Anforderung stellt sicher, dass ActiveSync weiß, wann eine bestimmte Instanz eines Providers für eine bestimmte Anwendung aufgerufen werden muss.

Voraussetzungen

Stellen Sie sicher, dass der Benutzer über Folgendes verfügt:

- Der ActiveSync Manager ist installiert.
- Die ActiveSync-Providerdateien wurden von einem Entwicklercomputer auf die Festplatte des Benutzers kopiert.

Kontext und Bemerkungen

Der UltraLite-ActiveSync-Provider ist ein Softwaremodul, über das Benutzer vom PC aus Zugriff auf ihre Geräte erhalten. Wie bei anderen Softwarekomponenten müssen Sie ein Deployment der erforderlichen Dateien auf das Gerät vornehmen, um sicherzustellen, dass UltraLite Windows Mobile ActiveSync verwendet:

- **mlasinst.exe** Installiert den ActiveSync-Provider und registriert ihn mit dem ActiveSync Manager. Dieses Dienstprogramm registriert außerdem Anwendungen mit dem ActiveSync-Provider für die Synchronisation.
- **mlasdesk.dll** Die DLL, die vom ActiveSync Manager auf den PC geladen wird. *mlasinst.exe* registriert die Position dieser Datei mit dem ActiveSync Manager.
- **mlasdev.dll** Die DLL, die vom ActiveSync Manager auf das Gerät geladen wird. *mlasinst.exe* führt das Deployment dieser Datei an der richtigen Position auf dem Gerät durch.
- **dbigen16.dll** Die Sprachenbibliothek.

Eine Liste der unterstützten Provider-Plattformen finden Sie unter <http://www.sybase.com/detail?id=1002288>.

Aufgabe

1. Führen Sie *mlasinst* aus, um einen Provider für ActiveSync zu installieren. Sie können hiermit auch Registrierung und Deployment der UltraLite-Anwendung auf dem Windows Mobile-Gerät des Benutzers durchführen - abhängig von der verwendeten Befehlszeilensyntax. Wenn die UltraLite-Anwendung mehrere Dateien verwendet, müssen Sie die erforderlichen Dateien manuell kopieren.

Im folgenden Beispiel wird angenommen, dass sich sowohl *mlasdesk.dll* als auch *mlasdev.dll* im aktuellen Verzeichnis befinden. Die Optionen -k und -v werden verwendet. Die Optionen -p und -x sind Befehlszeilenoptionen für die Anwendung, wenn sie von ActiveSync gestartet wird.

```
mlasinst "C:\My Files\myULapp.exe" "\Program Files\myULapp.exe"  
"My Application" MYAPP -p -x -v -k
```

Wenn Sie dieses Dienstprogramm für ein Deployment einer im voraus kompilierten CustDB für den ARM 5.0-Prozessor verwenden, würde die Befehlszeile der folgenden ähnlich sein:

```
mlasinst -v "%SQLANY16%\UltraLite\ce\arm.50"  
"%SQLANY16%\UltraLite\ce\arm.50\custdb.exe" custdb.exe CustDB CUSTDBDEMO
```

Hinweis

Sie können auch ActiveSync verwenden, um Ihre UltraLite-Anwendung zu einem späteren Zeitpunkt zu registrieren. Siehe „[Anwendungen mit dem ActiveSync Manager registrieren](#)“ auf Seite 130.

2. Starten Sie Ihren Computer neu, damit ActiveSync den neuen Provider erkennen kann.
3. Aktivieren Sie den MobiLink-Provider.
 - a. Klicken Sie im ActiveSync-Fenster auf **Optionen**.
 - b. Prüfen Sie die **MobiLink-Clients** in der Liste und klicken Sie auf **OK**, damit der Provider aktiviert wird.
 - c. Sie können eine Liste der registrierten Anwendungen einsehen, wenn Sie auf **Optionen** klicken, **MobiLink Clients** klicken und auf **Einstellungen** klicken.

Ergebnisse

Die Dateien werden auf dem Gerät bereitgestellt, sodass UltraLite mit Windows Mobile ActiveSync arbeiten kann.

Siehe auch

- „[Anwendungen mit dem ActiveSync Manager registrieren](#)“ auf Seite 130
- „[Dienstprogramm zur Installation des Microsoft ActiveSync-Providers \(mlasinst\)](#)“ [*MobiLink - Clientadministration*]

Anwendungen mit dem ActiveSync Manager registrieren

Anwendungen, die ActiveSync-Synchronisation verwenden, müssen sowohl bei ActiveSync registriert als auch auf das Gerät kopiert werden.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Sie können Ihre Anwendung für die Verbindung mit ActiveSync mithilfe des Installationsdienstprogramms des ActiveSync-Providers oder mit dem ActiveSync Manager selbst registrieren. In diesem Abschnitt wird beschrieben, wie der ActiveSync Manager verwendet wird.

Aufgabe

1. Starten Sie ActiveSync.
2. Klicken Sie im ActiveSync-Fenster auf **Optionen**.

3. Klicken Sie in der Liste der Informationstypen auf **MobiLink-Clients** und anschließend auf **Einstellungen**.
4. Klicken Sie im Dialogfeld **MobiLink-Synchronisation** auf **Neu**.
5. Geben Sie die folgenden Daten für Ihre Anwendung ein:
 - **Anwendungsname** Ein Name, der die in der Benutzerschnittstelle von ActiveSync angezeigte Anwendung kennzeichnet
 - **Klassenname** Der registrierte Klassenname für die Anwendung
 - **Pfad** Der Speicherort der Anwendung auf dem Gerät
 - **Argumente** Alle Befehlszeilenargumente, die verwendet werden sollen, wenn ActiveSync die Anwendung startet
6. Klicken Sie auf **OK**, damit die Anwendung registriert wird.

Ergebnisse

Die Anwendung wird bei ActiveSync registriert.

Nächste Schritte

Kopieren Sie die Anwendung auf das Zielgerät.

Siehe auch

- „Klassennamen für Anwendungen zuordnen“ [*UltraLite - C- und C++-Programmierung*]
- „Dienstprogramm zur Installation des Microsoft ActiveSync-Providers (mlasinst)“ [*MobiLink - Clientadministration*]

Praktische Einführung: Erstellen der Beispielanwendung CustDB

In dieser praktischen Einführung erfahren Sie, wie Sie Folgendes durchführen:

- Den MobiLink-Server ausführen, um Daten zwischen der konsolidierten Datenbank und der entfernten UltraLite-Datenbank zu synchronisieren
- Mit Sybase Central die Daten in der entfernten UltraLite-Datenbank durchsuchen
- UltraLite-Datenbanken mit UltraLite-Dienstprogrammen verwalten

Es gibt verschiedene Versionen des Anwendungscodes für jede unterstützte Programmierschnittstelle und Plattform. Diese praktische Einführung bezieht sich allerdings nur auf die kompilierte Version der Anwendung für Windows-PCs. Jede Version variiert abhängig von den Konventionen der einzelnen Plattformen.

Siehe auch

- „Überblick über die CustDB-Beispielanwendung“ auf Seite 15
- CustDB-Szenario [*MobiLink - Erste Orientierung*]
- „Benutzer im Beispiel CustDB“ [*MobiLink - Erste Orientierung*]
- „Tabellen in den CustDB-Datenbanken“ [*MobiLink - Erste Orientierung*]

Lektion 1: CustDB-Anwendung erstellen und ausführen

In dieser Lektion erstellen Sie die CustDB-Anwendung und führen sie aus.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. In Nicht-Windows-Umgebungen erstellen Sie die CustDB-Anwendung.
 - a. Öffnen Sie eine CustDB-Projektdaten in der betreffenden Umgebung.
 - b. Kompilieren Sie den Quellcode.
2. Führen Sie die CustDB-Anwendung aus.

In Windows 32-Bit-Umgebungen führen Sie `%SQLANY16%\UltraLite\Windows\x86\custdb.exe` aus.

Ergebnisse

Die CustDB-Anwendung wird ausgeführt.

Nächste Schritte

Gehen Sie weiter zu „[Lektion 2: MobiLink-Server starten und eine erste Synchronisation durchführen](#)“ auf Seite 134.

Siehe auch

- „[CustDB-Dateispeicherorte](#)“ auf Seite 17

Lektion 2: MobiLink-Server starten und eine erste Synchronisation durchführen

In dieser Lektion starten Sie den MobiLink-Server und synchronisieren die CustDB-Datenbank mithilfe der CustDB-Anwendung mit der UltraLite-Datenbank.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe „[Lektion 1: CustDB-Anwendung erstellen und ausführen](#)“ auf Seite 133.

Aufgabe

1. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » MobiLink » Synchronisationsserver-Beispiel**. Oder führen Sie den folgenden Befehl aus:

```
mlsrv16 -c "DSN=SQL Anywhere 16 CustDB" -vcrs
```

Verwenden Sie *mobilink.sh* unter Mac OS X oder Linux.

Das Fenster zeigt Meldungen zum Status des MobiLink-Servers an.

2. Klicken Sie auf **Start » Programme » SQL Anywhere » UltraLite » Windows-Beispielanwendung**.
3. Klicken Sie im Menü **Datei** auf die Option **Synchronisieren**.

Die Anwendung wird synchronisiert und im MobiLink-Server-Meldungsfenster werden Meldungen angezeigt, die Auskunft über den Fortschritt der Synchronisation geben.

Das Synchronisationsskript legt fest, welcher Teil der Kunden, Produkte und Bestellungen in die Anwendung heruntergeladen werden, wenn Benutzer 50 sich anmeldet. In diesem Fall werden nur Bestellungen, die noch nicht bestätigt wurden, heruntergeladen.

4. Überprüfen Sie, ob der Firmenname und eine Beispielbestellung im Anwendungsfenster angezeigt werden.

Ergebnisse

Die CustDB-Anwendung wird mit der konsolidierten Datenbank synchronisiert.

Nächste Schritte

Gehen Sie weiter zu „[Lektion 3: Daten in der UltraLite-Datenbank aktualisieren](#)“ auf Seite 135.

Lektion 3: Daten in der UltraLite-Datenbank aktualisieren

In dieser Lektion verwenden Sie die CustDB-Anwendung, um Daten in der entfernten Datenbank hinzuzufügen, zu aktualisieren oder zu löschen.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe „[Lektion 1: CustDB-Anwendung erstellen und ausführen](#)“ auf Seite 133.

Aufgabe

1. Durchsuchen der Bestellungen.

Bestellungen werden in allen Versionen der CustDB-Anwendung auf ähnliche Weise durchsucht. Wenn Sie eine Bestellung durchsuchen, blättern Sie durch die Daten in Ihrer lokalen UltraLite-Datenbank. Da Kunden alphabetisch sortiert sind, können Sie durch die Liste blättern und einen Kunden anhand seines Namens finden.

- Um die Liste der Kunden nach unten abzurollen, klicken Sie auf **Weiter**.
- Um die Liste der Kunden nach oben abzurollen, klicken Sie auf **Zurück**.

2. Bestellung hinzufügen.

Bestellungen werden in allen Versionen der CustDB-Anwendung auf ähnliche Weise hinzugefügt. Durch das Hinzufügen einer Bestellung ändern Sie die Daten in Ihrer lokalen UltraLite-Datenbank. Diese Daten werden erst in die konsolidierte Datenbank hochgeladen, wenn Sie synchronisieren.

- Klicken Sie auf **Order (Bestellung) » New (Neu)**.
- Klicken Sie in der Liste **Customer** auf **Basements R Us**.
- Klicken Sie in der Liste **Product** auf **Screwmaster Drill**. Der Preis dieses Artikels wird automatisch in das Feld **Price** (Preis) eingetragen.
- Im Feld **Quantity (Menge)** geben Sie **20** ein.
- Geben Sie in das Feld **Discount** (Rabatt) den Wert **5** (Prozent) ein und klicken Sie auf **OK**.

3. Bestätigen, Ablehnen und Löschen von Bestellungen.

Da Sie Ihre Identität als Benutzer-ID 50 authentifiziert haben, sind Sie ein Manager, der dieselben Aufgaben wie ein Vertriebsmitarbeiter ausführen kann. Sie haben jedoch außerdem die Berechtigung, Bestellungen zu akzeptieren oder abzulehnen. Wenn Sie eine Bestellung akzeptieren oder ablehnen, ändern Sie ihren Status und fügen außerdem einen Hinweis für den Vertriebsmitarbeiter zur

Überprüfung hinzu. Die Daten in der konsolidierten Datenbank werden jedoch bis zur Synchronisation nicht verändert.

- a. Genehmigen Sie die Bestellung für **Apple Street Builders**.
 - i. Um den Kunden zu lokalisieren, klicken Sie auf **Previous** (Zurück).
 - ii. Um die Bestellung zu genehmigen, klicken Sie auf **Order** (Bestellung) und dann auf **Approve** (Genehmigen).
 - iii. Klicken Sie in der Liste **Name** auf **Good**.
 - iv. Klicken Sie auf **OK**.

Die Bestellung erscheint mit dem Status **Approved** (Bestätigt).

- b. Lehnen Sie die Bestellung für **Art's Renovations** ab.
 - i. Gehen Sie zur nächsten Bestellung in der Liste, die von **Art's Renovations** stammt.
 - ii. Um die Bestellung abzulehnen, klicken Sie auf **Order** (Bestellung) und dann auf **Deny** (Ablehnen).
 - iii. Klicken Sie in der Liste **Note** (Hinweis) auf **Discount Is Too High** (Rabatt ist zu hoch).
 - iv. Klicken Sie auf **OK**.

Die Bestellung erscheint mit dem Status "Denied" (Abgelehnt).

- c. Löschen Sie die Bestellung für **Awnings R Us**.
 - i. Gehen Sie zur nächsten Bestellung in der Liste, die von **Awnings R Us** stammt.
 - ii. Löschen Sie diese Bestellung, indem Sie **Order** » **Delete** (Auftrag > Löschen) wählen.
 - iii. Klicken Sie auf **OK**, um das Löschen zu bestätigen.

Die Bestellung wird als gelöscht markiert. Die aktuellen Daten bleiben jedoch in der entfernten UltraLite-Datenbank enthalten, bis Sie die Änderungen in die konsolidierte Datenbank synchronisieren.

Ergebnisse

Änderungen der Daten in der UltraLite-Datenbank werden gespeichert, aber nicht mit der Datenbank CustDB synchronisiert.

Nächste Schritte

Gehen Sie weiter zu „[Lektion 4: UltraLite-Datenbank mit der konsolidierten Datenbank synchronisieren](#)“ auf Seite 137.

Siehe auch

- „[Tabellen in den CustDB-Datenbanken](#)“ [[MobiLink - Erste Orientierung](#)]

Lektion 4: UltraLite-Datenbank mit der konsolidierten Datenbank synchronisieren

In dieser Lektion synchronisieren Sie Datenbanken und verwenden Interactive SQL oder Sybase Central, um eine Verbindung mit der konsolidierten Datenbank herzustellen und zu überprüfen, ob Ihre Änderungen synchronisiert wurden.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe „[Lektion 1: CustDB-Anwendung erstellen und ausführen](#)“ auf Seite 133.

Kontext und Bemerkungen

Der Synchronisationsprozess für die CustDB-Anwendung löscht bestätigte Bestellungen aus Ihrer Datenbank.

Aufgabe

1. UltraLite-Datenbank synchronisieren

Klicken Sie im Menü **Datei** auf die Option **Datenbank synchronisieren**.

2. Stellen Sie sicher, dass die Synchronisation stattgefunden hat.

In der entfernten Datenbank können Sie kontrollieren, ob alle erforderlichen Transaktionen ausgeführt wurden, indem Sie überprüfen, ob die Bestellung für **Awnings R Us** gelöscht wurde. Hierzu durchsuchen Sie die Bestellungen, um festzustellen, ob der betreffende Eintrag fehlt.

In der konsolidierten Datenbank können Sie auch kontrollieren, ob alle erforderlichen Aktionen ausgeführt wurden, indem Sie die Daten überprüfen.

- Stellen Sie mithilfe von Sybase Central sicher, dass die Synchronisation stattgefunden hat.
 - a. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Sybase Central**.
 - b. Klicken Sie auf **Verbindungen » Verbinden mit SQL Anywhere 16**.
 - c. Klicken Sie im Dropdown-Menü **Aktion** auf **Mit einer ODBC-Datenquelle verbinden**.
 - d. Klicken Sie auf **ODBC-Datenquellenname**.
 - e. Klicken Sie auf **Durchsuchen** und auf **SQL Anywhere 16 CustDB**.
 - f. Klicken Sie auf **OK**.
 - g. Klicken Sie auf **Verbinden**.
 - h. Doppelklicken Sie auf **Tabellen**.
 - i. Doppelklicken Sie auf **ULOrder**.
 - j. Klicken Sie auf die Registerkarte **Daten** und überprüfen Sie, dass Bestellung 5100 genehmigt, Bestellung 5101 abgelehnt und Bestellung 5102 gelöscht ist.

- Stellen Sie mithilfe von Interactive SQL sicher, dass die Synchronisation stattgefunden hat.
 - a. Verbinden Sie sich mithilfe von Interactive SQL mit der konsolidierten Datenbank.
 - i. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Interactive SQL**.
 - ii. Klicken Sie in der Dropdown-Liste **Aktion** auf **Mit einer ODBC-Datenquelle verbinden**.
 - iii. Klicken Sie auf **ODBC Data Source Name (ODBC-Datenquellenname)** und auf **SQL Anywhere 16 CustDB**.
 - b. Um zu gewährleisten, dass die Annahmen und Ablehnungen synchronisiert wurden, führen Sie folgende Anweisung aus:

```
SELECT order_id, status
FROM ULOrder
WHERE status IS NOT NULL
```

Die Ergebnisse zeigen, dass Bestellung 5100 bestätigt und Bestellung 5101 abgelehnt wurde.

- c. Die gelöschte Bestellung hat die Bestell-ID (order_id) 5102. Die folgende Abfrage gibt keine Zeilen zurück, wodurch belegt wird, dass die Bestellung aus dem System gelöscht wurde:

```
SELECT *
FROM ULOrder
WHERE order_id = 5102
```

Ergebnisse

Die bestätigten Bestellungen werden aus der Datenbank entfernt und Sie bestätigen das Entfernen.

Nächste Schritte

Gehen Sie weiter zu „[Lektion 5: MobiLink-Synchronisationsskripten durchsuchen](#)“ auf Seite 138.

Lektion 5: MobiLink-Synchronisationsskripten durchsuchen

In dieser Lektion durchsuchen Sie Synchronisationsskripten, um ein besseres Verständnis für die Synchronisationslogik von CustDB zu erhalten.

Voraussetzungen

In dieser Lektion wird davon ausgegangen, dass Sie bereits alle vorherigen Lektionen abgeschlossen haben. Siehe „[Lektion 1: CustDB-Anwendung erstellen und ausführen](#)“ auf Seite 133.

Aufgabe

1. Klicken Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Sybase Central**.

2. Rechtsklicken Sie im linken Fensterausschnitt von **Sybase Central** auf **MobiLink 16** und klicken Sie dann auf **Projekt öffnen**.
3. Navigieren Sie zu `%SQLANYSAMPI6%\MobiLink\CustDB` und klicken Sie auf `project.mlp`.
4. Doppelklicken Sie auf **Konsolidierte Datenbanken** und auf die konsolidierte CustDB-Datenbank.

Ihre Verbindung mit der konsolidierten Datenbank basiert auf den Verbindungsinformationen, die bereitgestellt wurden, als Sie die konsolidierte Datenbank zu Ihrem Projekt hinzugefügt haben.

5. Doppelklicken Sie auf **Verbindungsskripten**.

Im rechten Fensterausschnitt wird eine Liste der Synchronisationsskripten sowie eine Gruppe von Ereignissen, die diesen Skripten zugeordnet sind, aufgeführt. Während der MobiLink-Server den Synchronisationsprozess ausführt, löst er eine Sequenz von Ereignissen aus. Alle Synchronisationsskripten, die den betreffenden Ereignissen zugeordnet sind, werden zu den jeweiligen Zeitpunkten ausgeführt. Durch das Schreiben von Synchronisationsskripten und deren Zuweisung zu Synchronisationsereignissen können Sie die Aktionen steuern, die während der Synchronisation ausgeführt werden.

6. Klicken Sie auf **Synchronisierte Tabellen**.
7. Im rechten Fensterausschnitt doppelklicken Sie auf **ULCustomer**.

Eine Reihe von spezifischen Skripten für diese Tabelle und ihre entsprechenden Ereignisse werden angezeigt. Diese Skripten steuern die Art und Weise, wie Daten in der Tabelle ULCustomer mit den entfernten Datenbanken synchronisiert werden.

Ergebnisse

Sie haben die Synchronisationsskripten überprüft.

Nächste Schritte

Keine.

Siehe auch

- „Schreiben von Synchronisationsskripten“ [[MobiLink - Serveradministration](#)]
- „Quellcode der Synchronisationslogik“ [[MobiLink - Erste Orientierung](#)]
- Synchronisationsplanung [[MobiLink - Erste Orientierung](#)]
- „UltraLite-Clients“ auf Seite 69
- „Verbindungsskripten“ [[MobiLink - Serveradministration](#)]
- „Tabellenskripten“ [[MobiLink - Serveradministration](#)]

UltraLite-Datenbankreferenz

Dieser Abschnitt enthält Referenzinformationen zu UltraLite-Datenbankeigenschaften, Optionen, Verbindungsparametern und Dienstprogrammen.

UltraLite-Erstellungsparameter

Dieser Abschnitt beschreibt die UltraLite-Erstellungsparameter, die zur Verfügung stehen, wenn Sie eine neue UltraLite-Datenbank erstellen.

Zusätzlich zu Erstellungsparametern können die folgenden UltraLite-Verbindungsparameter als Erstellungsparameter verwendet werden:

- [„UltraLite-Verbindungsparameter CE_FILE“ auf Seite 174](#)
- [„UltraLite-Verbindungsparameter DBF“](#)
- [„UltraLite-Verbindungsparameter DBKEY“ auf Seite 179](#)
- [„UltraLite-Verbindungsparameter desktop“ auf Seite 181](#)
- [„UltraLite-Verbindungsparameter device“ auf Seite 182](#)
- [„UltraLite-Verbindungsparameter MIRROR_FILE“ auf Seite 184](#)
- [„UltraLite-Verbindungsparameter NT_FILE“ auf Seite 185](#)
- [„UltraLite-Verbindungsparameter PWD“ auf Seite 186](#)
- [„UltraLite-Verbindungsparameter RESERVE_SIZE“ auf Seite 187](#)
- [„UltraLite-Verbindungsparameter UID“ auf Seite 190](#)

Mitglieder

UltraLite unterstützt die folgenden Verbindungsparameter:

Name	Beschreibung
case	Legt die Berücksichtigung von Groß- und Kleinschreibung für Zeichenfolgenvergleiche in der UltraLite-Datenbank fest. Siehe „UltraLite-Erstellungsparameter case“ auf Seite 143 .
checksum_level	Legt die Ebene der Prüfsummenvalidierung in der Datenbank fest. Siehe „UltraLite-Erstellungsparameter checksum_level“ auf Seite 144 .

Name	Beschreibung
collation	Legt die von der UltraLite-Datenbank verwendete Kollationssequenz fest. Durch das Festlegen dieser Eigenschaft mit oder ohne UTF-8-Eigenschaft wird der Zeichensatz der Datenbank definiert. Siehe „ UltraLite-Zeichensätze “ auf Seite 26 und „ UltraLite-Erstellungsparameter collation “ auf Seite 145 und „ UltraLite-Erstellungsparameter utf8_encoding “ auf Seite 167.
date_format	Setzt das standardmäßige Zeichenfolgenformat, in dem Datumsangaben aus der Datenbank abgerufen werden. Siehe „ UltraLite-Erstellungsparameter date_format “ auf Seite 146.
date_order	Steuert die Interpretation der Reihenfolge von Monat, Tag und Jahr bei Datumsangaben. Siehe „ UltraLite-Erstellungsparameter date_order “ auf Seite 149.
fips	Steuert die AES FIPS-zertifizierte Verschlüsselung mithilfe eines Certicom-zertifizierten Verschlüsselungsalgorithmus. Siehe „ Datenbank-sicherheit “ auf Seite 29 und „ UltraLite-Erstellungsparameter fips “ auf Seite 150.
max_hash_size	Legt die Standardgröße für den Index-Hash in Byte fest. Siehe „ UltraLite-Erstellungsparameter max_hash_size “ auf Seite 151.
nearest_century	Steuert die Interpretation von zweistelligen Jahresangaben bei Konvertierungen von Zeichenfolgen in Datumsangaben. Siehe „ UltraLite-Erstellungsparameter nearest_century “ auf Seite 153.
obfuscate	Steuert, ob Daten in der Datenbank verschleiert werden. Die Verschleierung ist eine vereinfachte Form der Verschlüsselung. Siehe „ Datenbank-sicherheit “ auf Seite 29 und „ UltraLite-Erstellungsparameter obfuscate “ auf Seite 154.
page_size	Definiert die Seitengröße in der Datenbank. Siehe „ UltraLite-Erstellungsparameter page_size “ auf Seite 155.
precision	Legt die maximale Anzahl von Stellen im Ergebnis aller Berechnungen mit Dezimaltrennzeichen fest. Siehe „ UltraLite-Erstellungsparameter precision “ auf Seite 157.
scale	Legt die Mindestanzahl der Stellen nach dem Dezimalzeichen fest, wenn ein arithmetisches Ergebnis auf die maximale Gesamtstellenanzahl gekürzt wird. Siehe „ UltraLite-Erstellungsparameter scale “ auf Seite 158.
time_format	Setzt das Format für Zeitwerte, die aus der Datenbank abgerufen werden. Siehe „ UltraLite-Erstellungsparameter time_format “ auf Seite 160.

Name	Beschreibung
timestamp_format	Legt das Format für Zeitstempelwerte fest, die aus der Datenbank abgerufen werden. Siehe „ UltraLite-Erstellungsparameter timestamp_format “ auf Seite 162.
timestamp_increment	Legt fest, wie der Zeitstempel in UltraLite gekürzt wird. Siehe „ UltraLite-Erstellungsparameter timestamp_increment “ auf Seite 164.
timestamp_with_time_zone_format	Diese Option stellt das Format für von der Datenbank abgerufene TIME-STAMP WITH TIME ZONE-Werte ein. Siehe „ UltraLite-Erstellungsparameter timestamp_with_time_zone_format “ auf Seite 166.
utf8_encoding	Kodiert Daten im UTF-8-Format, der 8-Bit-Mehrbyte-Kodierung für Unicode. Siehe „ UltraLite-Zeichensätze “ auf Seite 26 und „ UltraLite-Erstellungsparameter utf8_encoding “ auf Seite 167.

Siehe auch

- [Auf Erstellungsparameterwerte zugreifen auf Seite 26](#)
- [„UltraLite-Dienstprogramm zum Initialisieren einer Datenbank \(ulinit\)“ auf Seite 214](#)
- [„UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien \(ulload\)“ auf Seite 222](#)

UltraLite-Erstellungsparameter case

Legt die Berücksichtigung von Groß- und Kleinschreibung für Zeichenfolgenvergleiche in der UltraLite-Datenbank fest. Übergeben Sie **case=respect** an den Erstellungszeichenfolge-Parameter der CreateDatabase-Methode in der Programmierschnittstelle (oder **case=ignore** für eine Datenbank, in der nicht zwischen Groß- und Kleinschreibung unterschieden wird).

Syntax

```
ulinit --case=value database.udb
```

Zulässige Werte

Ignore, Respect

Standardwert

Ignore

Bemerkungen

Die Berücksichtigung der Groß- und Kleinschreibung bei Daten wirkt sich auf Tabellen, Indizes etc. aus. Standardmäßig berücksichtigen UltraLite-Datenbanken in Vergleichen die Groß- und Kleinschreibung nicht, obwohl Daten stets in der Schreibweise abgelegt werden, in der sie eingegeben wurden. Bezeichner (wie Tabellen- und Spaltennamen) und Benutzer-IDs berücksichtigen nie die Groß- und Kleinschreibung, unabhängig von der entsprechenden Einstellung in der Datenbank. Kennwörter berücksichtigen immer die Groß- und Kleinschreibung, unabhängig davon, ob die Datenbank die Schreibweise berücksichtigt.

Die Ergebnisse von Zeichenfolgenvergleichen und die Sortierreihenfolge von Zeichenfolgen hängen zum Teil von der Berücksichtigung der Groß- und Kleinschreibung in der Datenbank ab.

Es gibt Kollationen, bei denen Sie mit der Annahme der Nichtberücksichtigung von Groß-/Kleinschreibung bei den Bezeichnern vorsichtig sein müssen. Insbesondere türkische Kollationen haben ein Groß-/Kleinbuchstaben-Konvertierungsverhalten, das unerwartete und subtile Fehler zur Folge haben kann. Der häufigste Fehler ist, dass ein Systemobjekt, das den Buchstaben I oder i enthält, nicht auffindbar ist.

Es ist nicht möglich, die Berücksichtigung von Groß- und Kleinschreibung einer vorhandenen Datenbank zu ändern. Sie müssen stattdessen eine neue Datenbank erstellen.

In Sybase Central können Sie die Berücksichtigung von Groß- und Kleinschreibung in jedem Assistenten einstellen, der eine Datenbank erstellt. Klicken Sie auf der Seite **Kollatierung und Zeichensatz der neuen Datenbank** auf die Option **Groß- und Kleinschreibung bei Zeichenfolgenvergleichen berücksichtigen**.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- „Zeichenfolgen in UltraLite“ auf Seite 272
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- ULCreateDatabase-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26

UltraLite-Erstellungsparameter checksum_level

Legt die Ebene der Prüfsummenvalidierung bei der Datenbank fest.

Syntax

```
ulinit --checksum_level=value
```

Zulässige Werte

0, 1, 2

Standardwert

0

Bemerkungen

Prüfsummen werden verwendet, um Offline-Beschädigungen von Seiten, die auf dem Datenträger, im Flash-Speicher oder im Speicher abgelegt sind, zu ermitteln, was die Wahrscheinlichkeit verringert, dass andere Daten aufgrund einer beschädigten kritischen Seite in Mitleidenschaft gezogen werden. Abhängig

von der gewählten Ebene berechnet UltraLite eine Prüfsumme für jede Datenbankseite und protokolliert sie, bevor die Seite in den Speicher geschrieben wird.

Wenn die berechnete Prüfsumme nicht mit der gespeicherten Prüfsumme für einen Lesevorgang aus dem Speicher übereinstimmt, wurde die Seite verändert oder während der Speicherung bzw. der Abfrage beschädigt. Wenn eine Prüfsummenvalidierung beim Laden einer Seite fehlschlägt, stoppt UltraLite die Datenbank und meldet einen schwerwiegenden Fehler. Dieser Fehler kann nicht behoben werden. Sie müssen Ihre UltraLite-Datenbank neu erstellen und den Datenbankfehler an iAnywhere melden.

Wenn Sie eine UltraLite-Datenbank entladen und mit aktivierten Prüfsummen neu laden, wird die Stufe der Prüfsummenvalidierung beibehalten und wiederhergestellt.

Die folgenden Werte werden bei checksum_level unterstützt:

- **0** Datenbankseiten keine Prüfsummen hinzufügen
- **1** Wichtigen Datenbankseiten Prüfsummen hinzufügen (wie z.B. Indizes und Synchronisationsstatusseiten), aber nicht zu Zeilenseiten
- **2** Allen Datenbankseiten Prüfsummen hinzufügen

In Sybase Central können Sie die Verwendung von Prüfsummen in jedem Assistenten konfigurieren, der eine Datenbank erstellt. Klicken Sie auf der Seite **Einstellungen für die Speicherung der neuen Datenbank** auf die Option **Prüfsummenstufe für Datenbankseiten**.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- Datenbankeigenschaft checksum_level: „UltraLite-Datenbankeigenschaften“ auf Seite 191
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- ULCreateDatabase-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- „UltraLite Performance-Tipps“ auf Seite 469
- „UltraLite-Erstellungsparameter page_size“ auf Seite 155
- „UltraLite-Datenbankverbindungen“ auf Seite 35

UltraLite-Erstellungsparameter collation

Legt die Datenbankkollation fest.

Syntax

ulinit --collation=value

Zulässige Werte

Zeichenfolge

Standardwert

1252Latin1

Bemerkungen

Sie können auch eine Liste der unterstützten Kollationen in UltraLite anzeigen, indem Sie den folgenden Befehl ausführen:

```
ulinit -Z
```

In Sybase Central können Sie die Kollation in jedem Assistenten einstellen, der eine Datenbank erstellt. Klicken Sie auf der Seite **Kollatierung und Zeichensatz der neuen Datenbank** entweder die Standardkollation (d.h. 1252Latin1) oder eine Alternative aus der bereitgestellten Liste.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- „Unterstützte UltraLite-Kollationen“ auf Seite 28
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- „UltraLite-Zeichensätze“ auf Seite 26
- ULCreateDatabase-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26

UltraLite-Erstellungsparameter date_format

Legt das Format für Datumsangaben fest, die aus der Datenbank abgerufen werden

Für Android-Smartphones können Sie Connection.setOption(OPTION_DATE_FORMAT, Wert) als Alternative zum Einstellen dieses Erstellungsparameters verwenden. Siehe [Connection.setOption-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*] und [Connection.OPTION_DATE_FORMAT-Variable \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*].

Syntax

```
ulinit --date_format=value
```

Zulässige Werte

Zeichenfolge

Standardwert

YYYY-MM-DD (dies entspricht den ISO-Datumsformat-Spezifikationen)

Bemerkungen

DATE-Datentypwerte werden in einem Format dargestellt, das vom Erstellungsparameter `date_format` festgelegt wird. Datumswerte können jedoch auch durch Zeichenfolgen dargestellt werden. Bevor der Wert abgerufen werden kann, muss er einer Zeichenfolge zugeordnet werden.

UltraLite baut ein Datum anhand von Datumsteilen auf. Datumsteile können das Jahr, den Monat, den Tag, den Wochentag, den Tag des Jahres, die Stunde, die Minute und die Sekunde (und Teile von diesen Angaben) enthalten.

ISO (YYYY-MM-DD) ist das Standardformat für das Datum und seine Reihenfolge. Das Datum "07. Januar 06" wird in diesem internationalen Format z.B. wie folgt geschrieben: 2006-01-07. Wenn Sie das ISO-Standardformat und die ISO-Standardreihenfolge nicht verwenden wollen, müssen Sie ein anderes Format und eine andere Reihenfolge für diese Datumsteile festlegen.

Das Format ist eine Zeichenfolge mit folgenden Symbolen:

Symbol	Beschreibung
YY	Jahr zweistellig.
YYYY	Jahr vierstellig.
MM	Monat zweistellig oder Minuten zweistellig, falls nach einem Doppelpunkt (wie 'HH:MM')
MMM[m...]	Zeichenkurzform für Monate. Es werden so viele Zeichen angezeigt, wie "m" angegeben werden. Durch ein großes "M" werden Großbuchstaben ausgegeben.
D	Einzelzeichen für Wochentag (0 = Sonntag, 6 = Samstag).
DD	Monatstag zweistellig. Es ist keine vorangestellte Null erforderlich.
DDD[d...]	Zeichenkurzform für Wochentag. Durch ein großes "D" werden Großbuchstaben ausgegeben.
HH	Stunde zweistellig. Es ist keine vorangestellte Null erforderlich.
NN	Minuten zweistellig. Es ist keine vorangestellte Null erforderlich.
SS[s...]	Sekunden und Sekundenbruchteile
AA	12-Stundentakt verwenden. Uhrzeiten am Vormittag werden durch AM gekennzeichnet.
PP	12-Stundentakt verwenden. Uhrzeiten am Nachmittag werden durch PM gekennzeichnet.
JJJ	Jahrestag von 1 bis 366.

Es ist nicht möglich, das Datumsformat einer vorhandenen Datenbank zu ändern. Sie müssen stattdessen eine neue Datenbank erstellen.

Zulässige Werte werden aus den in der vorangehenden Tabelle aufgelisteten Symbolen erstellt. Jedes Symbol wird durch die entsprechenden Daten für das Datum ersetzt, das formatiert wird.

Bei den Zeichenkurzformen wird die Anzahl der angegebenen Buchstaben gezählt. Die Angabe A.M. oder P.M. (die lokalisiert sein könnte) wird gegebenenfalls auf die Anzahl der Byte gekürzt, die der angegebenen Anzahl von Zeichen entspricht.

Groß- und Kleinschreibung der Ausgabe steuern Für Symbole, die Zeichendaten darstellen (wie z.B. MMM), können Sie die Groß- und Kleinschreibung in der Ausgabe wie folgt steuern:

- Geben Sie das Symbol in Großbuchstaben ein, wenn das Format in Großbuchstaben erscheinen soll. Beispiel: MMM ergibt JAN.
- Geben Sie das Symbol in Kleinbuchstaben ein, wenn das Format in Kleinbuchstaben erscheinen soll. Beispiel: mmm ergibt jan.
- Geben Sie das Symbol in Groß- und Kleinbuchstaben ein, damit UltraLite für die jeweils benutzte Sprache die richtige Schreibweise auswählt. In Deutsch ergibt z.B. die Schreibweise Mmm die Ausgabe Mai, während dieselbe Schreibweise in Französisch die Ausgabe mai ergibt.

Auffüllen mit Nullen steuern Für Symbole, die numerische Daten darstellen, können Sie das Auffüllen mit Nullen durch die Schreibweise der Symbole steuern:

- Geben Sie das Symbol in einheitlicher Schreibung (z.B. MM oder mm) ein, um das Auffüllen mit Nullen zu gestatten. Beispiel: yyyy/mm/dd ergibt 2002/01/01.
- Geben Sie das Symbol in Groß- und Kleinschreibung (z.B. Mm) ein, wenn das Auffüllen mit Nullen nicht gestattet werden soll. Beispiel: yyyy/Mm/Dd ergibt 2002/1/1.

In Sybase Central können Sie das Datumsformat in jedem Assistenten einstellen, der eine Datenbank erstellt. Klicken Sie auf der Seite **Parameter für die Erstellung der neuen Datenbank** auf die Option **Datumsformat**.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- „UltraLite-Erstellungsparameter date_order“ auf Seite 149
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- ULCreateDatabase-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26

Beispiel

Die folgende Tabelle veranschaulicht date_format-Einstellungen anhand der Ausgabe einer SELECT CURRENT DATE-Anweisung, die für Donnerstag, den 21.05.01, ausgeführt wurde.

Verwendete date_format-Syntax	Zurückgegebenes Ergebnis
YYYY/MM/DD/dd	2001/05/21/Don
JJJ	141
mmm YYYY	Mai 2001
MM-YYYY	05-2001

UltraLite-Erstellungsparameter date_order

Steuert die Interpretation von Datumsformaten

Für Android-Smartphones können Sie Connection.setOption(OPTION_DATE_ORDER, Wert) als Alternative zum Einstellen dieses Erstellungsparameters verwenden. Siehe [Connection.setOption-Methode \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#) und [Connection.OPTION_DATE_ORDER-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#).

Syntax

```
ulinit --date_order=value
```

Zulässige Werte

MDY, YMD, DMY

Standardwert

YMD (dies entspricht den ISO-Datumsformat-Spezifikationen)

Bemerkungen

DATE-Datentypwerte werden in einem Format dargestellt, das vom Erstellungsparameter date_format festgelegt wird. Datumswerte können jedoch auch durch Zeichenfolgen dargestellt werden. Bevor der Wert abgerufen werden kann, muss er einer Zeichenfolge zugeordnet werden.

UltraLite baut ein Datum anhand von Datumsteilen auf. Datumsteile können das Jahr, den Monat, den Tag, den Wochentag, den Tag des Jahres, die Stunde, die Minute und die Sekunde (und Teile von diesen Angaben) enthalten.

ISO (YYYY-MM-DD) ist das Standardformat für das Datum und seine Reihenfolge. Das Datum "07. Januar 06" wird in diesem internationalen Format z.B. wie folgt geschrieben: 2006-01-07. Wenn Sie das ISO-Standardformat und die ISO-Standardreihenfolge nicht verwenden wollen, müssen Sie ein anderes Format und eine andere Reihenfolge für diese Datumsteile festlegen.

Es ist nicht möglich, die Datumsreihenfolge einer vorhandenen Datenbank zu ändern. Sie müssen stattdessen eine neue Datenbank erstellen.

In Sybase Central können Sie die Datumsreihenfolge in jedem Assistenten einstellen, der eine Datenbank erstellt. Klicken Sie auf der Seite **Parameter für die Erstellung der neuen Datenbank** auf die Option **Datumsreihenfolge**.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- „UltraLite-Erstellungsparameter date_format“ auf Seite 146
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- ULCreateDatabase-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26

Beispiel

Verschiedene Werte legen fest, wie das Datum 10/11/12 konvertiert wird:

Verwendete Syntax	Konvertierung
MDY	Okt 11 1912
YMD	Nov 12 1910
DMY	Nov 10 1912

UltraLite-Erstellungsparameter fips

Steuert, ob die neue Datenbank mit starker AES- oder AES FIPS-Verschlüsselung verschlüsselt werden soll. Dieser Parameter wird von UltraLiteJ und UltraLite für iPhone nicht unterstützt.

Syntax

```
{ulinit -a | ulload -c } --fips=value --key=value
```

Zulässige Werte

Yes (verwendet AES_FIPS), No (verwendet AES)

Standardwert

Ja

Bemerkungen

Die einzige Möglichkeit, den Typ der Datenbankverschlüsselung zu ändern, besteht darin, die Datenbank mit dem entsprechenden Erstellungsparameter `fips` oder `obfuscate` neu zu erstellen. Sie können den Chiffrierschlüssel der Datenbank ändern, indem Sie im Connection-Objekt einen neuen Chiffrierschlüssel erstellen. Benutzer müssen jedes Mal, wenn sie eine Verbindung mit der Datenbank herstellen, den Schlüssel angeben.

In Sybase Central können Sie die Verschlüsselung in jedem Assistenten konfigurieren, der eine Datenbank erstellt. Auf der Seite **Einstellungen für die Speicherung der neuen Datenbank** klicken Sie auf die Option **AES FIPS-Algorithmus**. Sie müssen außerdem den Chiffrierschlüssel festlegen und ihn bestätigen.

Legen Sie diese Eigenschaft als einen der Erstellungsparameter für die `CreateDatabase`-Methode einer API in der Datenbankmanager-Klasse fest.

Um eine Datenbank mit der aktivierten Option bereitzustellen, kopieren Sie alle erforderlichen Bibliotheken für Ihre Plattform.

Siehe auch

- „Einfache Verschlüsselung und starke Verschlüsselung“ [*SQL Anywhere Server - Datenbankadministration*]
- „Datenbanksicherheit“ auf Seite 29
- „UltraLite-Erstellungsparameter `obfuscate`“ auf Seite 154
- „UltraLite-Verbindungsparameter `DBKEY`“ auf Seite 179
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (`ulinit`)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (`ulload`)“ auf Seite 222
- `ULCreateDatabase`-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- `ULChangeEncryptionKey`-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- `ULDatabaseManager.CreateDatabase`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- `ULDatabaseManager.CreateDatabase`-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- `ULConnection.ChangeEncryptionKey`-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26

UltraLite-Erstellungsparameter `max_hash_size`

Legt die maximale Standardgröße für den Index-Hash in Byte fest.

Syntax

```
ulinit --max_hash_size=value
```

Zulässige Werte

0 bis 32 Byte

Standardwert

4 Byte

Bemerkungen

Ein Hash ist ein optionaler Teil eines Indexeintrags, der auf der Indexseite gespeichert wird. Der Hash-Code wandelt die tatsächlichen Zeilenwerte der indizierten Spalten in numerische Entsprechungen (einen Schlüssel) um, wobei die Reihenfolge für den Index beibehalten wird. Die Größe des Schlüssels und der Umfang des tatsächlichen Werts, der von UltraLite in den Hash-Wert einbezogen wird, wird von der festgelegten Hash-Größe bestimmt.

Eine Zeilen-ID gestattet es UltraLite, die Zeile für die tatsächlichen Daten in der Tabelle zu finden. Eine Zeilen-ID ist immer Teil eines Indexeintrags. Wenn Sie die Hash-Größe auf 0 setzen (d.h. den Index-Hash deaktivieren), enthält der Indexeintrag nur diese Zeilen-ID. Für alle anderen Hash-Größen wird der Hash-Schlüssel, der alle umgewandelten Daten in der Zeile oder einen Teil davon enthalten kann, zusammen mit der Zeilen-ID auf der Indexseite gespeichert. Sie können die Abfrageperformance für diese indizierten Spalten verbessern, da UltraLite dann nicht immer die Daten suchen, laden und entpacken muss, bevor die tatsächlichen Zeilenwerte gefunden wurden.

Um die Hash-Standardgröße für die Datenbank festzulegen, müssen Sie jedoch die Vor- und Nachteile der Abfrageeffizienz einerseits und der Datenbankgröße andererseits abwägen: Je höher der maximale Hash-Wert ist, desto größer wird die Datenbank.

UltraLite verwendet nur so viele Byte, wie für den Datentyp der Spalte erforderlich sind, bis zu dem in diesem Parameter angegebenen Maximum. Die Standard-Hash-Größe wird nur verwendet, wenn Sie bei der Erstellung des Indexes keine Größe festlegen. Wenn Sie die Standard-Hash-Größe auf 0 setzen, wendet UltraLite die Index-Hash-Methode nicht auf Zeilenwerte an.

Sie können die Hash-Größe bei einem bestehenden Index nicht ändern. Wenn Sie einen neuen Index erstellen, können Sie den Standardwert mit dem **UltraLite-Assistenten zum Erstellen von Indizes** in Sybase Central oder mit der WITH MAX SIZE-Klausel einer Anweisung CREATE INDEX oder CREATE TABLE aufheben.

Wenn Sie Ihre Spalten als DOUBLE, FLOAT oder REAL deklarieren, wird die Hash-Methode nicht verwendet. Die Hash-Größe wird immer ignoriert.

In Sybase Central können Sie die maximale Hash-Größe in jedem Assistenten einstellen, der eine Datenbank erstellt. Klicken Sie auf der Seite **Einstellungen für die Speicherung der neuen Datenbank** auf die Option **Maximale Hashgröße für Indizes**.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- „UltraLite Performance-Tipps“ auf Seite 469
- „UltraLite-Indizes“ auf Seite 56
- „Optimale Hash-Größengrenze“ auf Seite 474
- „CREATE INDEX-Anweisung [UltraLite]“ auf Seite 433
- „CREATE TABLE-Anweisung [UltraLite]“ auf Seite 438
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- ULCreateDatabase-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26

UltraLite-Erstellungsparameter nearest_century

Steuert die Interpretation von zweistelligen Jahresangaben in Umwandlungen aus Zeichenfolgen in Datumswerte

Für Android-Smartphones können Sie `Connection.setOption(OPTION_NEAREST_CENTURY, Wert)` als Alternative zum Einstellen dieses Erstellungsparameters verwenden. Siehe [Connection.setOption-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*] und [Connection.OPTION_NEAREST_CENTURY-Variable \[\]](#) [*UltraLiteJ*] [*UltraLite® – Java-Programmierung*].

Syntax

```
ulinit --nearest_century=value
```

Zulässige Werte

Ganzzahl zwischen 0 und 100 (inklusive)

Standardwert

50

Bemerkungen

UltraLite konvertiert automatisch eine Zeichenfolge in ein Datum, wenn ein Datumswert erwartet wird, auch wenn das Jahr in der Zeichenfolge nur durch zwei Stellen dargestellt ist. Bei einer zweistelligen Datumsangabe müssen Sie den geeigneten Überschreitungswert angeben. Zweistellige Jahresangaben, die kleiner als der festgelegte Wert sind, werden in *20JJ* konvertiert, während Jahresangaben, die größer oder gleich dem betreffenden Wert sind, in *19JJ* konvertiert werden.

Die Auswahl eines geeigneten Überschreitungswerts wird gewöhnlich durch folgende Faktoren bestimmt:

- **Verwendung von zweistelligen Datumsangaben** Andernfalls ist die Konvertierung in das nächste Jahrhundert nicht anwendbar. Zweistellige Jahresangaben, die kleiner als der von `nearest_century` festgelegte Wert sind, werden in *20JJ* konvertiert, während Jahresangaben, die größer oder gleich dem betreffenden Wert sind, in *19JJ* konvertiert werden.

Es wird empfohlen, vierstellige Datumsangaben zu speichern, um Probleme mit unkorrekten Konvertierungen zu vermeiden.

- **Kompatibilität mit konsolidierten Datenbanken** Das bisherige Verhalten von SQL Anywhere besteht darin, 1900 zum Jahr hinzuzufügen. Adaptive Server Enterprise verhält sich so, dass er das nächste Jahrhundert verwendet und so bei jedem Jahrwert *jj*, *der kleiner als 50 ist, das Jahr auf 20jj setzt*.
- **Repräsentation des Datums: vergangenes oder zukünftiges Ereignis** Geburtsjahre erfordern gewöhnlich einen niedrigeren Überschreitungswert, da sie in der Vergangenheit liegen. Für alle Jahresangaben, bei denen *JJ* kleiner als 20 ist, sollte das Jahr auf *20JJ* gesetzt werden. Wenn das Datum jedoch als Ablaufdatum verwendet wird, wäre ein höherer Wert eine logische Wahl, da das Datum in der Zukunft liegt.

Es ist nicht möglich, das nächste Jahrhundert für eine vorhandene Datenbank zu ändern. Sie müssen stattdessen eine neue Datenbank erstellen.

In Sybase Central können Sie die Einstellung für das nächste Jahrhundert in jedem Assistenten konfigurieren, der eine Datenbank erstellt. Klicken Sie auf der Seite **Parameter für die Erstellung der neuen Datenbank** auf die Option **Nächstes Jahrhundert**.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- ULCreateDatabase-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26

UltraLite-Erstellungsparameter obfuscate

Steuert die Verschleierung von Daten in der Datenbank.

Syntax

{ulinit -a | ulload -c } obfuscate=*value*

Zulässige Werte

Boolescher Wert.

Standardwert

0 (Datenbanken werden nicht verschleiert)

Bemerkungen

Verschleiern ist gleichbedeutend mit einer einfachen Verschlüsselung und erschwert das Dechiffrieren der Daten in Ihrer Datenbank mithilfe eines Festplatten-Dienstprogramms. Bei der einfachen Verschlüsselung wird kein Schlüssel benötigt, um die Datenbank zu chiffrieren.

Sie müssen starke Verschlüsselung verwenden, damit ohne den richtigen Chiffrierschlüssel nicht auf die Datenbank zugegriffen werden kann.

In Sybase Central können Sie die Verschleierung in jedem Assistenten einstellen, der eine Datenbank erstellt. Klicken Sie auf der Seite **Einstellungen für die Speicherung der neuen Datenbank** auf die Option **Einfache Verschlüsselung (Verschleierung) verwenden**.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- „Datenbanksicherheit“ auf Seite 29
- „UltraLite-Erstellungsparameter fips“ auf Seite 150
- „UltraLite-Verbindungsparameter DBKEY“ auf Seite 179
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- ULCreateDatabase-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26

UltraLite-Erstellungsparameter page_size

Definiert die Seitengröße in KByte.

Syntax

```
{ulinit | ulload } -c --page_size=sizeK
```

Zulässige Werte

1K, 2K, 4K, 8K, 16K

Standardwert

4K

Bemerkungen

Die Seitengröße muss mit "K" oder "k" nach der Zahl angegeben werden. Alternativ kann eine Bytezahl (1024, 2048, 4096, 8192 oder 16384) angegeben werden.

UltraLite-Datenbanken werden in Seiten gespeichert und alle I/O-Vorgänge werden seitenweise ausgeführt. Die gewählte Seitengröße kann die Performance bzw. die Größe der Datenbank beeinflussen.

Wenn Sie einen Wert verwenden, der nicht in der Liste enthalten ist, wird die Größe auf die nächst höhere Größe geändert. Wenn Sie keine Einheit angeben, wird von Byte ausgegangen.

Wenn der dynamische Speicher Ihrer Plattform begrenzt ist, sollten Sie eine kleinere Seitengröße verwenden, um die Auswirkungen der Anforderungen an den Synchronisationsspeicher zu begrenzen.

Wenn Sie eine Seitengröße auswählen, sollten Sie die folgenden Richtlinien beachten:

- **Datenbankgröße** Große Seiten bringen im Allgemeinen bei größeren Datenbanken Vorteile. Größere Seiten enthalten mehr Informationen und führen daher zu einer effizienteren Speichernutzung, v.a. wenn Zeilen eingefügt werden, die etwas größer als eine halbe Seite sind: Je größer die Seite, desto weniger Seiten müssen ein- und ausgelagert werden.
- **Zeilenanzahl** Da eine Zeile (ausgenommen BLOBs) auf eine Seite passen muss, legt die Seitengröße fest, wie groß die größte gepackte Zeile sein kann und wie viele Zeilen auf einer Seite gespeichert werden können. In manchen Fällen kann das Lesen von einer Seite, um die Werte einer Zeile zu erhalten, als Nebeneffekt bewirken, dass der Inhalt der nächsten Zeilen in den Speicher geladen wird.
- **Abfragentypen** Im Allgemeinen sind kleine Seiten eher bei Abfragen nützlich, die eine relativ kleine Anzahl von Zeilen von wahlfreien Positionen abrufen. Im Gegensatz dazu sind größere Seiten günstiger für Abfragen, die sequenzielle Table-Scans ausführen.
- **Cachegröße** Größere Seiten erfordern möglicherweise auch einen größeren Cache. Wenn die dynamische Cachedimensionierung aktiv ist, fordert UltraLite gegebenenfalls zusätzlichen Cache an.
- **Indexeinträge** Die Seitengröße hat auch Einfluss auf Indizes. Je größer die Datenbankseite ist, umso mehr Indexeinträge kann sie speichern.
- **Gerätespeicher** Kleine Seiten sind besonders nützlich, wenn Sie Ihre Datenbanken auf kleinen Geräten mit beschränktem Speicher betreiben. So kann zum Beispiel ein 1-MB-Speicher 1000 Seiten speichern, die jeweils 1 kB groß sind, aber nur 250 Seiten mit einer Größe von 4 kB.

Es ist nicht möglich, die Seitengröße einer vorhandenen Datenbank zu ändern. Sie müssen stattdessen eine neue Datenbank erstellen.

In Sybase Central können Sie die Seitengröße in jedem Assistenten einstellen, der eine Datenbank erstellt. Auf der Seite **Einstellungen für die Speicherung der neuen Datenbank** klicken Sie auf den entsprechenden Bytewert.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- [Gepackte Zeilen und Tabellendefinitionen auf Seite 50](#)
- [„UltraLite-Indizes“ auf Seite 56](#)
- [„UltraLite-Erstellungsparameter case“ auf Seite 143](#)
- [„UltraLite-Verbindungsparameter CACHE_SIZE“ auf Seite 172](#)
- [„UltraLite-Verbindungsparameter CACHE_MIN_SIZE“ auf Seite 171](#)
- [„UltraLite-Verbindungsparameter CACHE_MAX_SIZE“ auf Seite 170](#)
- [„UltraLite-Verbindungsparameter RESERVE_SIZE“ auf Seite 187](#)
- [„UltraLite-Dienstprogramm zum Initialisieren einer Datenbank \(ulinit\)“ auf Seite 214](#)
- [„UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien \(uload\)“ auf Seite 222](#)
- [ULCreateDatabase-Methode \[UltraLite Embedded SQL\] \[*UltraLite - C- und C++-Programmierung*\]](#)
- [ULDatabaseManager.CreateDatabase-Methode \[UltraLite C++\] \[*UltraLite - C- und C++-Programmierung*\]](#)
- [ULDatabaseManager.CreateDatabase-Methode \[UltraLite.NET\] \[*UltraLite - .NET-Programmierung*\]](#)
- [Auf Erstellungsparameterwerte zugreifen auf Seite 26](#)

Beispiel

Um die Seitengröße der Datenbank auf 8 kB zu setzen, geben Sie **page_size=8k** oder **page_size=8192** an:

```
ulinit test.udb -a --page_size=8k
```

UltraLite-Erstellungsparameter precision

Legt die maximale Anzahl von Stellen im Ergebnis aller Berechnungen mit Dezimaltrennzeichen fest.

Für Android-Smartphones können Sie `Connection.setOption(OPTION_PRECISION, Wert)` als Alternative zum Einstellen dieses Erstellungsparameters verwenden. Siehe [Connection.setOption-Methode \[UltraLiteJ\] \[*UltraLite® – Java-Programmierung*\]](#) und [Connection.OPTION_PRECISION-Variable \[UltraLiteJ\] \[*UltraLite® – Java-Programmierung*\]](#).

Syntax

```
ulinit precision=value
```

Zulässige Werte

Ganzzahl zwischen 1 und 127 (inklusive)

Standardwert

30

Bemerkungen

Die Position des Dezimalzeichens wird von der Anzahl der Gesamtstellen und der Dezimalstellen der Zahl bestimmt: Die Gesamtstellenzahl ist die Anzahl der Ziffern links und rechts vom Dezimalzeichen, während die Dezimalstellen die Mindestzahl der Ziffern nach dem Dezimalzeichen ist, wenn ein arithmetisches Ergebnis auf die maximale Gesamtstellenzahl gekürzt wird.

Die richtige Position des Dezimalzeichens wird wie folgt ermittelt:

- **Typ der durchgeführten arithmetischen Prozeduren** Multiplikation, Division, Addition, Subtraktion und Aggregatfunktionen können jeweils Ergebnisse haben, die die maximale Gesamtstellenzahl übersteigen.

Wenn zum Beispiel DECIMAL(8,2) mit DECIMAL(9,2) multipliziert wird, könnte das Ergebnis DECIMAL(17,4) erfordern. Wenn die Anzahl der Gesamtstellen (PRECISION) 15 ist, werden nur 15 Stellen im Ergebnis gehalten. Wenn die Dezimalstellenzahl (SCALE) 4 ist, lautet das Ergebnis DECIMAL(15,4). Wenn die Dezimalstellenzahl (SCALE) 2 ist, lautet das Ergebnis DECIMAL(15,2). In beiden Fällen ist ein Überlauf-Fehler möglich.

- **Beziehung zwischen Dezimalstellenzahl und Gesamtstellenzahl** Die Dezimalstellenzahl legt die Anzahl der Ziffern nach dem Dezimalzeichen fest. Sie kann nicht negativ oder größer als die Gesamtstellenzahl sein.

Es ist nicht möglich, die Gesamtstellenzahl einer vorhandenen Datenbank zu ändern. Sie müssen stattdessen eine neue Datenbank erstellen.

Wenn Sie eine Oracle-Datenbank als konsolidierte Datenbank verwenden, müssen alle entfernten UltraLite-Datenbanken und die konsolidierte Oracle-Datenbank denselben Gesamtstellenwert verwenden.

In Sybase Central können Sie die Gesamtstellenzahl in jedem Assistenten einstellen, der eine Datenbank erstellt. Klicken Sie auf der Seite **Parameter für die Erstellung der neuen Datenbank** auf die Option **Gesamtstellenzahl** fest.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- ULCreateDatabase-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26

UltraLite-Erstellungsparameter scale

Legt die Mindestanzahl der Stellen nach dem Dezimalzeichen fest, wenn ein arithmetisches Ergebnis auf die maximale Gesamtstellenanzahl gekürzt wird

Für Android-Smartphones können Sie Connection.setOption(OPTION_SCALE, Wert) als Alternative zum Einstellen dieses Erstellungsparameters verwenden. Siehe [Connection.setOption-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*] und [Connection.OPTION_SCALE-Variable \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*].

Syntax

`ulinit --scale=value`

Zulässige Werte

Ganzzahl zwischen 0 und 127 (inklusive)

Standardwert

6

Bemerkungen

Die Position des Dezimalzeichens wird von der Anzahl der Gesamtstellen und der Dezimalstellen der Zahl bestimmt: Die Gesamtstellenzahl ist die Anzahl der Ziffern links und rechts vom Dezimalzeichen, während die Dezimalstellen die Mindestzahl der Ziffern nach dem Dezimalzeichen ist, wenn ein arithmetisches Ergebnis auf die maximale Gesamtstellenzahl gekürzt wird.

Die richtige Position des Dezimalzeichens wird wie folgt ermittelt:

- **Typ der durchgeführten arithmetischen Prozeduren** Multiplikation, Division, Addition, Subtraktion und Aggregatfunktionen können jeweils Ergebnisse haben, die die maximale Gesamtstellenzahl übersteigen.

Wenn zum Beispiel DECIMAL(8,2) mit DECIMAL(9,2) multipliziert wird, könnte das Ergebnis DECIMAL(17,4) erfordern. Wenn die Anzahl der Gesamtstellen (PRECISION) 15 ist, werden nur 15 Stellen im Ergebnis gehalten. Wenn die Dezimalstellenzahl (SCALE) 4 ist, lautet das Ergebnis DECIMAL(15,4). Wenn die Dezimalstellenzahl (SCALE) 2 ist, lautet das Ergebnis DECIMAL(15,2). In beiden Fällen ist ein Überlauf-Fehler möglich.

- **Beziehung zwischen Dezimalstellenzahl und Gesamtstellenzahl** Die Dezimalstellenzahl legt die Anzahl der Ziffern nach dem Dezimalzeichen fest. Sie kann nicht negativ oder größer als die Gesamtstellenzahl sein.

Es ist nicht möglich, die Zahl der Dezimalstellen einer vorhandenen Datenbank zu ändern. Sie müssen stattdessen eine neue Datenbank erstellen.

In Sybase Central können Sie die Dezimalstellenzahl in jedem Assistenten einstellen, der eine Datenbank erstellt. Klicken Sie auf der Seite **Parameter für die Erstellung der neuen Datenbank** auf die Option **Dezimalstellen** fest.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- „UltraLite-Erstellungsparameter precision“ auf Seite 157
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- ULCreateDatabase-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26

Beispiel

Wenn zum Beispiel DECIMAL(8,2) mit DECIMAL(9,2) multipliziert wird, könnte das Ergebnis DECIMAL(17,4) erfordern. Wenn die Anzahl der Gesamtstellen (PRECISION) 15 ist, werden nur 15 Stellen im Ergebnis gehalten. Wenn die Dezimalstellenzahl (SCALE) 4 ist, lautet das Ergebnis DECIMAL(15,4). Wenn die Dezimalstellenzahl (SCALE) 2 ist, lautet das Ergebnis DECIMAL(15,2). In beiden Fällen ist ein Überlauf möglich.

UltraLite-Erstellungsparameter time_format

Setzt das Format für Zeitwerte, die aus der Datenbank abgerufen werden

Für Android-Smartphones können Sie Connection.setOption(OPTION_TIME_FORMAT, Wert) als Alternative zum Einstellen dieses Erstellungsparameters verwenden. Siehe [Connection.setOption-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*] und [Connection.OPTION_TIME_FORMAT-Variable \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*].

Syntax

`ulinit --time_format=value`

Zulässige Werte

Zeichenfolge (aus den Symbolen in der Liste unten zusammengesetzt)

Standardwert

HH:NN:SS.SSS

Bemerkungen

UltraLite stellt Uhrzeiten aus Uhrzeitteilen zusammen, die Sie mit dem Erstellungsparameter time_format festlegen. Uhrzeitteile können Stunden, Minuten, Sekunden und Millisekunden umfassen.

Uhrzeitwerte können auch durch Zeichenfolgen dargestellt werden. Bevor der Wert abgerufen werden kann, muss er einer Zeichenfolgenvariablen zugeordnet werden.

ISO (HH:MM:SS) ist das Standardzeitformat. Die Uhrzeit "Mitternacht" wird in diesem internationalen Format in der folgenden Form angegeben: 00:00:00. Wenn Sie das ISO-Standardzeitformat nicht verwenden wollen, müssen Sie ein anderes Format und eine andere Reihenfolge für diese Uhrzeitteile festlegen.

Das Format ist eine Zeichenfolge mit folgenden Symbolen:

Symbol	Beschreibung
HH	Stunden zweistellig (24-Studentakt)
NN	Minuten zweistellig
MM	Minuten zweistellig, wenn ein Doppelpunkt vorangeht (z.B. HH:MM)
SS.SSS	Sekunden zweistellig plus Bruchteil

Es ist nicht möglich, das Zeitformat einer vorhandenen Datenbank zu ändern. Sie müssen stattdessen eine neue Datenbank erstellen.

Jedes Symbol wird durch die entsprechenden Daten für die Uhrzeit ersetzt, die formatiert wird. Alle Formatsymbole, die Zeichen anstelle von Ziffern darstellen, können in Großbuchstaben gesetzt werden, damit die ersetzten Zeichen in Großbuchstaben geschrieben werden. Bei Zahlen werden führende Nullen durch gemischte Schreibung in der Formatzeichenfolge unterdrückt.

Sie können mithilfe der Groß- und Kleinschreibung der Symbole das Auffüllen mit Nullen steuern:

- Geben Sie das Symbol in einheitlicher Schreibung (z.B. HH oder hh) ein, um das Auffüllen mit Nullen zu gestatten. Beispiel: HH:NN:SS kann 01:01:01 ergeben.
- Geben Sie das Symbol in Groß- und Kleinschreibung (z.B. Hh oder hH) ein, um das Auffüllen mit Nullen nicht zu gestatten. Beispiel: Hh:Nn:Ss kann 1:1:1 ergeben.

In Sybase Central können Sie das Zeitformat in jedem Assistenten einstellen, der eine Datenbank erstellt. Klicken Sie auf der Seite **Parameter für die Erstellung der neuen Datenbank** auf die Option **Zeitformat**.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- „UltraLite-Erstellungsparameter timestamp_format“ auf Seite 162
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- ULCreateDatabase-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26

Beispiel

Wenn eine Transaktion um 15:30 Uhr ausgeführt wurde und Sie die time_format-Standardsyntax HH:NN:SS.SSS verwenden, lautet das Ergebnis folgendermaßen:

15:30:55.0

UltraLite-Erstellungsparameter timestamp_format

Setzt das Format für Zeitstempel, die aus der Datenbank abgerufen werden

Für Android-Smartphones können Sie `Connection.setOption(OPTION_TIMESTAMP_FORMAT, Wert)` als Alternative zum Einstellen dieses Erstellungsparameters verwenden. Siehe [Connection.setOption-Methode \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#) und [Connection.OPTION_TIME_FORMAT-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#).

Syntax

`ulinit --timestamp_format=value`

Zulässige Werte

Zeichenfolge

Standardwert

YYYY-MM-DD HH:NN:SS.SSS

Bemerkungen

UltraLite erstellt einen Zeitstempel anhand eines Datums- und eines Uhrzeitteils, die Sie mit den Erstellungsparametern `date_format` und `time_format` festlegen. Zusammen bestehen Datum und Uhrzeit aus sieben Teilen (Jahr, Monat, Tag, Stunde, Minute, Sekunde und Millisekunde).

Zeitstempelwerte können auch durch Zeichenfolgen dargestellt werden. Um einen Zeitstempelwert abzurufen, muss er einer Zeichenfolgenvariablen zugewiesen werden.

Gewöhnlich stellen Zeitstempelspalten sicher, dass die Datenintegrität bei der Synchronisation mit einer konsolidierten Datenbank gewährleistet ist. Zeitstempel dienen zur Identifikation, wenn gleichzeitige Datenaktualisierungen zwischen mehreren entfernten Datenbanken durchgeführt wurden, indem sie den letzten Zeitpunkt der Synchronisation der einzelnen Benutzer protokollieren.

Tipp

Stellen Sie sicher, dass die konsolidierte Datenbank und die entfernte UltraLite-Datenbank dieselben Einstellungen für Zeitstempel und Zeitstempelerhöhungen verwenden. Sie können Ungleichheiten vermeiden, indem Sie die Erstellungsparameter so festlegen, dass der Wert mit demjenigen in der konsolidierten Datenbank übereinstimmt.

Das Format ist eine Zeichenfolge mit folgenden Symbolen:

Symbol	Beschreibung
YY	Jahr zweistellig.
YYYY	Jahr vierstellig.

Symbol	Beschreibung
MM	Monat zweistellig oder Minuten zweistellig, falls ein Doppelpunkt vorangeht (wie HH:MM)
MMM[m...]	Zeichenkurzform für Monate. Es werden so viele Zeichen angezeigt, wie "m" angegeben werden. Durch ein großes "M" werden Großbuchstaben ausgegeben.
D	Einzelzeichen für Wochentag (0 = Sonntag, 6 = Samstag).
DD	Monatstag zweistellig. Es ist keine vorangestellte Null erforderlich.
DDD[d...]	Zeichenkurzform für Wochentag. Durch ein großes "D" werden Großbuchstaben ausgegeben.
HH	Stunde zweistellig. Es ist keine vorangestellte Null erforderlich.
NN	Minuten zweistellig Es ist keine vorangestellte Null erforderlich.
SS.SSS	Sekunden und Sekundenbruchteile
AA	12-Studentakt verwenden. Uhrzeiten am Vormittag werden durch AM gekennzeichnet.
PP	12-Studentakt verwenden. Uhrzeiten am Nachmittag werden durch PM gekennzeichnet.
JJJ	Jahrestag von 1 bis 366.

Es ist nicht möglich, das Zeitstempelformat einer vorhandenen Datenbank zu ändern. Sie müssen stattdessen eine neue Datenbank erstellen.

Zulässige Werte werden aus den in der vorangehenden Tabelle aufgelisteten Symbolen erstellt. Jedes Symbol wird durch die entsprechenden Daten für das Datum ersetzt, das formatiert wird.

Bei den Zeichenkurzformen wird die Anzahl der angegebenen Buchstaben gezählt. Die Angabe A.M. oder P.M. (die lokalisiert sein könnte) wird gegebenenfalls auf die Anzahl der Byte gekürzt, die der angegebenen Anzahl von Zeichen entspricht.

Für Symbole, die Zeichendaten darstellen (wie z.B. MMM), können Sie die Groß- und Kleinschreibung in der Ausgabe wie folgt steuern:

- Wenn nur Großschreibung verwendet werden soll, geben Sie das Symbol in Großbuchstaben ein.
Beispiel: MMM ergibt JAN.
- Wenn nur Kleinschreibung verwendet werden soll, geben Sie das Symbol in Kleinbuchstaben ein.
Beispiel: mmm ergibt jan.

- Geben Sie das Symbol in Groß- und Kleinbuchstaben ein, damit UltraLite für die jeweils benutzte Sprache die richtige Schreibweise auswählt. In Deutsch ergibt z.B. die Schreibweise Mmm die Ausgabe Mai, während dieselbe Schreibweise in Französisch die Ausgabe mai ergibt.

Für Symbole, die numerische Daten darstellen, können Sie das Auffüllen mit Nullen durch die Schreibweise der Symbole steuern:

- Geben Sie das Symbol in einheitlicher Schreibung (z.B. MM oder mm) ein, um das Auffüllen mit Nullen zu gestatten. Beispiel: yyyy/mm/dd ergibt 2002/01/01.
- Geben Sie das Symbol in Groß- und Kleinschreibung (z.B. Mm) ein, wenn das Auffüllen mit Nullen nicht gestattet werden soll. Beispiel: yyyy/Mm/Dd ergibt 2002/1/1.

In Sybase Central können Sie das Zeitstempelformat in jedem Assistenten einstellen, der eine Datenbank erstellt. Klicken Sie auf der Seite **Parameter für die Erstellung der neuen Datenbank** auf die Option **Zeitstempelformat**.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- „UltraLite-Erstellungsparameter timestamp_increment“ auf Seite 164
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- ULCreateDatabase-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26
- „Implementieren zeitstempelbasierter Downloads“ [*MobiLink - Serveradministration*]
- „UltraLite-Parallelität“ auf Seite 484

Beispiel

Wenn eine Transaktion am Freitag, den 12. Mai 2006, um 3:30 PM ausgeführt wurde und Sie die timestamp_format-Standardsyntax YYYY-MM-DD HH:NN:SS.SSS verwenden, lautet das Ergebnis folgendermaßen:

2006-05-12 15:30:55.0

UltraLite-Erstellungsparameter timestamp_increment

Begrenzt die Auflösung von Zeitstempelwerten Wenn Zeitstempel in die Datenbank eingefügt werden, kürzt UltraLite sie, um sie an diese Erhöhung anzupassen.

Für Android-Smartphones können Sie Connection.setOption(OPTION_TIMESTAMP_INCREMENT, Wert) als Alternative zum Einstellen dieses Erstellungsparameters verwenden. Siehe [Connection.setOption-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*] und

[Connection.OPTION_TIMESTAMP_INCREMENT-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#).

Syntax

`ulinit timestamp_increment=value`

Zulässige Werte

1 bis 60000000 Mikrosekunden

Standardwert

1 Mikrosekunde

Bemerkungen

1000000 Mikrosekunden entsprechen 1 Sekunde.

Es ist nicht möglich, die Zeitstempelerhöhung einer vorhandenen Datenbank zu ändern. Sie müssen stattdessen eine neue Datenbank erstellen.

Diese Erhöhung ist nützlich, wenn eine DEFAULT TIMESTAMP-Spalte als Primärschlüssel oder als Zeilenbezeichner verwendet wird.

In Sybase Central können Sie die Zeitstempelerhöhung in jedem Assistenten einstellen, der eine Datenbank erstellt. Klicken Sie auf der Seite **Parameter für die Erstellung der neuen Datenbank** auf die Option **Zeitstempel-Inkrementierungsstufe**.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- „UltraLite-Erstellungsparameter timestamp_format“ auf Seite 162
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- ULCreateDatabase-Methode [UltraLite Embedded SQL] [[UltraLite - C- und C++-Programmierung](#)]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [[UltraLite - C- und C++-Programmierung](#)]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [[UltraLite - .NET-Programmierung](#)]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26
- „Implementieren zeitstempelbasierter Downloads“ [[MobiLink - Serveradministration](#)]
- „UltraLite-Parallelität“ auf Seite 484

Beispiel

Um einen Wert wie '2000/12/05 10:50:53:700' zu speichern, setzen Sie diesen Erstellungsparameter auf 100000. Diese Einstellung kürzt den Zeitstempel nach der ersten Dezimalstelle in der Sekundenkomponente.

UltraLite-Erstellungsparameter

timestamp_with_time_zone_format

Stellt das Format für von der Datenbank abgerufene TIMESTAMP WITH TIME ZONE-Werte ein.

Syntax

```
ulinit --timestamp_with_time_zone_format=value
```

Zulässige Werte

Zeichenfolge (aus den Symbolen in der Liste unten zusammengesetzt)

Standardwert

```
YYYY-MM-DD HH:NN:SS.SSS+HH:NN
```

Bemerkungen

Das Format ist eine Zeichenfolge mit folgenden Symbolen:

Symbol	Beschreibung
YY	Jahr zweistellig
YYYY	Jahr vierstellig
MM	Monat zweistellig oder Minuten zweistellig, falls ein Doppelpunkt vorangeht (wie HH:MM)
MMM[m...]	Zeichenkurzform für Monate: so viele Zeichen wie "m"
DD	Monatstag zweistellig
DDD[d...]	Zeichenkurzform für Wochentag
HH	Stunde zweistellig
NN	Minuten zweistellig
SS.SSSSSS	Sekunden und Sekundenbruchteile bis zu sechs Dezimalstellen. Nicht alle Plattformen unterstützen Zeitstempel mit einer Genauigkeit von bis zu sechs Stellen.
AA	A.M. oder P.M. (12-Stunden-Angabe), AA und PP bei 24-Stunden-Angabe weglassen
PP	P.M. falls nötig (12-Stunden-Angabe), AA und PP bei 24-Stunden-Angabe weglassen
HH	Stunden zweistellig (Zeitzoneverschiebung)

Symbol	Beschreibung
NN	Minuten zweistellig (Zeitzoneverschiebung)

Jedes Symbol wird durch die entsprechenden Daten für das Datum ersetzt, das formatiert wird.

Für Symbole, die Zeichendaten darstellen (wie z.B. MMM), können Sie die Groß- und Kleinschreibung in der Ausgabe wie folgt steuern:

- Wenn nur Großschreibung verwendet werden soll, geben Sie das Symbol in Großbuchstaben ein.
Beispiel: MMM ergibt JAN.
- Wenn nur Kleinschreibung verwendet werden soll, geben Sie das Symbol in Kleinbuchstaben ein.
Beispiel: mmm ergibt jan.
- Geben Sie das Symbol in Groß- und Kleinbuchstaben ein, damit UltraLite für die jeweils benutzte Sprache die richtige Schreibweise auswählt. In Deutsch ergibt z.B. die Schreibweise Mmm die Ausgabe Mai, während dieselbe Schreibweise in Französisch die Ausgabe mai ergibt.

Wenn die Zeichendaten Mehrbyte-Zeichen sind, spiegelt die Länge der einzelnen Symbole die Anzahl der Zeichen und nicht die Anzahl der Bytes wieder. Das Symbol "MMM" gibt z.B. eine Länge von drei Zeichen für den Monat an.

Für Symbole, die numerische Daten darstellen, können Sie das Auffüllen mit Nullen durch die Schreibweise der Symbole steuern:

- Geben Sie das Symbol in einheitlicher Schreibung (z.B. MM oder mm) ein, um das Auffüllen mit Nullen zu gestatten. Beispiel: yyyy/mm/dd ergibt 2002/01/01.
- Geben Sie das Symbol in Groß- und Kleinschreibung (z.B. Mm) ein, wenn das Auffüllen mit Nullen nicht gestattet werden soll. Beispiel: yyyy/Mm/Dd ergibt 2002/1/1.

Hinweis

Wenn Sie die Einstellung für die timestamp_with_time_zone_format-Option so ändern, dass das Datumsformat neu geordnet wird, achten Sie darauf, auch die Option date_order zu ändern, damit dieselbe Änderung darin nachvollzogen wird, und umgekehrt. Siehe „[date_order-Option](#)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Siehe auch

- „TIMESTAMP WITH TIME ZONE-Datentyp“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

UltraLite-Erstellungsparameter utf8_encoding

Kodiert Daten im UTF-8-Format, der 8-Bit-Mehrbyte-Kodierung für Unicode.

Syntax

```
uload -c --utf8_encoding=value
```

Werte

Boolescher Wert.

Standardwert

1 (Datenbanken werden mit UTF-8 kodiert)

Bemerkungen

UTF-8-Zeichen werden durch ein bis vier Byte repräsentiert. Bei anderen Mehrbyte-Kollationen werden ein oder zwei Byte verwendet. In allen bereitgestellten Mehrbyte-Kollationen werden Zeichen, die zwei oder mehr Byte umfassen, als alphabetisch behandelt. Sie können diese Zeichen in Bezeichnern verwenden, ohne dass Anführungszeichen erforderlich sind.

Zeichen in einer UltraLite-Datenbank werden entweder über die in der gewählten Kollation implizite Codepage oder mit UTF8 kodiert. UltraLite-Datenbanken, die die UTF8BIN-Kollation verwenden, werden automatisch mit UTF8 kodiert. Wenn das Betriebssystem, auf dem Sie Ihre UltraLite-Anwendung bereitstellen, UTF8 oder Unicode verwendet (wie die meisten Linux-Distributionen, Windows Mobile und iPhone) oder wenn Sie Zeichen aus mehreren Sprachen in Ihrer Datenbank speichern wollen, sollten Sie die Datenbank unter Verwendung einer UTF8-Kodierung erstellen. Wenn Sie versuchen, UTF-8-kodierte Zeichen in eine konsolidierte Tabelle zu synchronisieren, die Unicode nicht unterstützt, wird ein Benutzerfehler gemeldet.

In Sybase Central können Sie die UTF-8-Kodierung in jedem Assistenten einstellen, der eine Datenbank erstellt. Klicken Sie auf der Seite **Kollatierung und Zeichensatz der neuen Datenbank** auf die Option **Ja, UTF8 als Zeichensatz der Datenbank verwenden**.

Aus einer Clientanwendung legen Sie diesen Parameter als einen der Erstellungsparameter für die CreateDatabase-Methode in der Datenbankmanager-Klasse fest.

Siehe auch

- „Plattformanforderungen für die Zeichensatzkodierung in UltraLite“ auf Seite 27
- „UltraLite-Zeichensätze“ auf Seite 26
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- ULDatabaseManager.CreateDatabase-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.CreateDatabase-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- Auf Erstellungsparameterwerte zugreifen auf Seite 26

UltraLite-Verbindungsparameter

Dieser Abschnitt beschreibt die UltraLite-Verbindungsparameter, die zur Verfügung stehen, wenn Sie eine Verbindung zu einer UltraLite-Datenbank herstellen.

Mitglieder

UltraLite unterstützt die folgenden Verbindungsparameter:

Parametername	Beschreibung
CACHE_MAX_SIZE, CACHE_MIN_SIZE, CACHE_SIZE	Legt die Größe des Cachespeichers der Datenbank fest. Siehe „ UltraLite-Verbindungsparameter CACHE_SIZE “ auf Seite 172, „ UltraLite-Verbindungsparameter CACHE_MIN_SIZE “ auf Seite 171 und „ UltraLite-Verbindungsparameter CACHE_MAX_SIZE “ auf Seite 170.
COMMIT_FLUSH	Legt fest, wann festgeschriebene Transaktionen nach einem Festschreibeaufruf bereinigt, d.h. in den Speicher geschrieben werden. Siehe „ UltraLite-Verbindungsparameter COMMIT_FLUSH “ auf Seite 175.
CON	Gibt einen Namen für die aktuelle Verbindung an. Siehe „ UltraLite-Verbindungsparameter CON “ auf Seite 177.
CE_FILE, DBF, desktop, device und NT_FILE	<p>Bei der Erstellung legen diese Parameter den Speicherort der Datenbank fest.</p> <p>Bei nachfolgenden Verbindungen informieren sie UltraLite darüber, wo sich die Datei befindet.</p> <p>Sie können DBF verwenden, wenn Sie eine Anwendung für eine einzelne Plattform erstellen oder eine Verbindung mit einem UltraLite-Administrationstool herstellen. Verwenden Sie die anderen plattformspezifischen Versionen, wenn Sie einen UltraLite-Client programmieren, der eine Verbindung mit verschiedenen plattformspezifischen Datenbanken herstellt. Siehe:</p> <ul style="list-style-type: none"> • „UltraLite-Verbindungsparameter DBF“ auf Seite 177 • „UltraLite-Verbindungsparameter CE_FILE“ auf Seite 174 • „UltraLite-Verbindungsparameter desktop“ auf Seite 181 • „UltraLite-Verbindungsparameter device“ auf Seite 182 • „UltraLite-Verbindungsparameter NT_FILE“ auf Seite 185
DBKEY	<p>Zum Zeitpunkt der Erstellung legt dieser Parameter den Chiffrierschlüssel zur Verschlüsselung der Datenbank fest.</p> <p>Für nachfolgende Verbindungen legt er den Chiffrierschlüssel zur Verschlüsselung der Datenbank fest.</p> <p>Siehe „UltraLite-Verbindungsparameter DBKEY“ auf Seite 179.</p>
DBN	Identifiziert eine laufende Datenbank anhand eines Namens anstatt des Dateinamens. Siehe „ UltraLite-Verbindungsparameter DBN “ auf Seite 180.

Parametername	Beschreibung
MIRROR_FILE	Gibt den Namen einer Datenbankspiegeldatei an. Siehe „ UltraLite-Verbindungsparameter MIRROR_FILE “ auf Seite 184.
PWD	<p>Dieser Parameter legt zum Zeitpunkt der Erstellung das anfängliche Kennwort für einen Benutzer fest.</p> <p>Für nachfolgende Verbindungen übergibt er das Kennwort für die Benutzer-ID.</p> <p>Siehe „UltraLite-Verbindungsparameter PWD“ auf Seite 186.</p>
RESERVE_SIZE	Weist für die UltraLite-Datenbank vorab Speicherplatz im Dateisystem zu, ohne Daten einzufügen. Siehe „ UltraLite-Verbindungsparameter RESERVE_SIZE “ auf Seite 187.
START	Gibt den Speicherort der Programmdatei der UltraLite-Engine an. Siehe „ UltraLite-Verbindungsparameter START “ auf Seite 189.
TEMP_DIR	Gibt den Namen des Verzeichnisses (das bereits vorhanden sein muss) an, in dem UltraLite eine temporäre Datei ablegt, deren Name vom Namen der Datenbank abgeleitet wird. Siehe „ UltraLite-Verbindungsparameter TEMP_DIR “ auf Seite 189.
UID	<p>Dieser Parameter legt bei der Erstellung die erste Benutzer-ID fest.</p> <p>Bei nachfolgenden Verbindungen wird damit ein Benutzer der Datenbank festgelegt.</p> <p>Die Benutzer-ID muss eine von maximal vier in der UltraLite-Datenbank gespeicherten Benutzer-IDs sein. Siehe „UltraLite-Verbindungsparameter UID“ auf Seite 190.</p>

Siehe auch

- „[UltraLite-Verbindungszeichenfolgen und Parameter](#)“ auf Seite 35

UltraLite-Verbindungsparameter **CACHE_MAX_SIZE**

Legt die maximale Größe des Cachespeichers der Datenbank fest. UltraLite verwaltet die Cachegröße automatisch, sodass die Einstellung dieses Parameters nicht erforderlich sein sollte.

Syntax

CACHE_MAX_SIZE=*number*{ **k** | **m** }

Standardwert

Die vorgegebene maximale Cachegröße ist 20 MB für mobile Geräte und 50 MB für PCs.

Bemerkungen

Der `cache_max_size`-Verbindungsparameter gibt die maximale Speichermenge an, die dem Dateicache zugewiesen werden soll. Standardmäßig ist die Größe in Byte. Verwenden Sie `k` oder `m`, um die Einheit von KB bzw. MB anzugeben.

Wenn die maximale Cachegröße überschritten wird, wird stattdessen der obere Grenzwert für die Cachegröße verwendet. UltraLite lässt kein Anwachsen der Cachegröße über die tatsächliche Größe der Datenbank zu.

Wenn Sie einen Grenzwert für die Cachegröße festlegen, der größer ist als die Größe Ihrer Datenbank, kann der überschüssige Speicherplatz verwendet werden, um Zeilen im Cache zu speichern.

Alle führenden bzw. nachgestellten Leerzeichen in Verbindungsparameterwerten werden ignoriert. Der Wert dieses Verbindungsparameters darf keine führenden Apostrophe, Anführungszeichen oder Semikolons enthalten.

Siehe auch

- „UltraLite-Verbindungsparameter `CACHE_MIN_SIZE`“ auf Seite 171
- „UltraLite-Verbindungsparameter `CACHE_SIZE`“ auf Seite 172
- „UltraLite-Option `cache_allocation`“ auf Seite 195
- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Erstellungsparameter `page_size`“ auf Seite 155
- „UltraLite-Verbindungsparameter `RESERVE_SIZE`“ auf Seite 187
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [*UltraLite - C- und C++-Programmierung*]
- `ULDatabaseManager.OpenConnection`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [*UltraLite - .NET-Programmierung*]
- `ULConnection.Open`-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- „Cachegrößenanpassung für eine UltraLite-Datenbank“ auf Seite 469

Beispiel

Das folgende Fragment einer Verbindungszeichenfolge legt die maximale Cachegröße auf 100 MB fest.

```
"CACHE_MAX_SIZE=100m"
```

UltraLite-Verbindungsparameter `CACHE_MIN_SIZE`

Legt die Mindestgröße des Cachespeichers der Datenbank fest. UltraLite verwaltet die Cachegröße automatisch, sodass die Einstellung dieses Parameters nicht erforderlich sein sollte.

Syntax

```
CACHE_MIN_SIZE=number{ k | m }
```

Standardwert

Die Standard-Cachegröße für Geräte ist 256 KB. Die Standard-Cachegröße für PCs beträgt 512 KB.

Bemerkungen

Der `cache_min_size`-Verbindungsparameter gibt die Speichermenge an, die dem Dateicache zugewiesen werden soll. Standardmäßig ist die Größe in Byte. Verwenden Sie `k` oder `m`, um die Einheit von `KB` bzw. `MB` anzugeben.

Wenn die Mindestcachegröße höher angesetzt wird als die maximale Cachegröße, gibt UltraLite eine Fehlermeldung aus und die Verbindung schlägt fehl.

Alle führenden bzw. nachgestellten Leerzeichen in Verbindungsparameterwerten werden ignoriert. Der Wert dieses Verbindungsparameters darf keine führenden Apostrophe, Anführungszeichen oder Semikolons enthalten.

Siehe auch

- „UltraLite-Verbindungsparameter `CACHE_MAX_SIZE`“ auf Seite 170
- „UltraLite-Verbindungsparameter `CACHE_SIZE`“ auf Seite 172
- „UltraLite-Option `cache_allocation`“ auf Seite 195
- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Erstellungsparameter `page_size`“ auf Seite 155
- „UltraLite-Verbindungsparameter `RESERVE_SIZE`“ auf Seite 187
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [[UltraLite - C- und C++-Programmierung](#)]
- `ULDatabaseManager.OpenConnection`-Methode [UltraLite C++] [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [[UltraLite - .NET-Programmierung](#)]
- `ULConnection.Open`-Methode [UltraLite.NET] [[UltraLite - .NET-Programmierung](#)]

Beispiel

Das folgende Fragment einer Verbindungszeichenfolge legt die Mindestcachegröße auf 1 MB fest.

```
"CACHE_MIN_SIZE=1m"
```

UltraLite-Verbindungsparameter `CACHE_SIZE`

Legt die Anfangsgröße des Cachespeichers der Datenbank fest. UltraLite verwaltet die Cachegröße automatisch, sodass die Einstellung dieses Parameters nicht erforderlich sein sollte.

Für Android-Smartphones können Sie `Configuration.setPageSize` als Alternative zum Einstellen dieses Verbindungsparameter verwenden. Siehe [Configuration.setPageSize-Methode \[UltraLiteJ\]](#) [[UltraLite® – Java-Programmierung](#)].

Syntax

`CACHE_SIZE=number{ k | m }`

Standardwert

Die Standard-Cachegröße wird durch die Menge an verfügbarem Speicher auf Ihrem System und der Größe der Datenbank bestimmt.

Bemerkungen

Der `cache_size`-Verbindungsparameter gibt die ursprüngliche Speichermenge an, die dem Dateicache zugewiesen werden soll. Dieser Cache enthält kürzlich verwendete Seiten aus der Datenbankdatei, sodass darauf bei Bedarf schnell erneut zugegriffen werden kann. Außerdem dient er zum Erfassen der Änderungen an einer Seite, bevor sie zurück in den Speicher geschrieben werden. Eine Seite kann aus dem Cache sehr viel schneller ausgelesen werden als aus dem Speicher. Das Schreiben in den Speicher ist aufwändiger, sodass das Gruppieren mehrerer Änderungen in einem einzigen Schreibvorgang für die Performance wichtig ist. Auch für verschlüsselte Datenbanken hat der Cache Vorteile: Die Entschlüsselung erfolgt erst, wenn die Seite in den Cache geladen ist. Die Verschlüsselung erfolgt, bevor die Seite in den Speicher zurückgeschrieben wird. Wenn der Cache ausreichend groß ist, ist der Aufwand für die Verschlüsselung unerheblich.

Ein Beispiel für die Cachenutzung ist die Synchronisation. Wenn UltraLite einen Download empfängt, werden die Zeilen in die Datenbank eingefügt und die referenzielle Integrität wird geprüft. Nach dem Einfügen werden die Zeilen auch indiziert, d. h. zu den einzelnen Indizes in der Tabelle hinzugefügt. Daher enthält der Cache während der Synchronisation die Seiten, für die neue Zeilen gespeichert werden, sowie die Indexseiten für die aktuelle Tabelle. Die Synchronisations-Performance hängt davon ab, ob der Cache groß genug ist, um einen geeigneten "Arbeitssatz" von Seiten für die zu synchronisierende Tabelle aufzunehmen. Wenn der Cache zu klein ist, müssen Indexseiten möglicherweise wiederholt aus dem Speicher gelesen werden, was zu deutlichen Einbußen bei der Performance führt. Eine bessere Performance kann erzielt werden, wenn der Cache alle erforderlichen Indexseiten aufnehmen kann.

Standardmäßig ist die Größe in Byte. Verwenden Sie `k` oder `m`, um die Einheit von `kB` bzw. `MB` anzugeben.

Wenn Sie die maximal zulässige Cachegröße überschreiten, wird sie automatisch durch den oberen Grenzwert für die Cachegröße der Plattform ersetzt.

Alle führenden bzw. nachgestellten Leerzeichen in Parameterwerten werden ignoriert. Der Wert dieses Verbindungsparameters darf keine führenden Apostrophe, Anführungszeichen oder Semikolons enthalten.

Siehe auch

- „UltraLite-Verbindungsparameter `CACHE_MAX_SIZE`“ auf Seite 170
- „UltraLite-Verbindungsparameter `CACHE_MIN_SIZE`“ auf Seite 171
- „UltraLite-Option `cache_allocation`“ auf Seite 195
- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Erstellungsparameter `page_size`“ auf Seite 155
- „UltraLite-Verbindungsparameter `RESERVE_SIZE`“ auf Seite 187
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [*UltraLite - C- und C++-Programmierung*]
- `ULDatabaseManager.OpenConnection`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [*UltraLite - .NET-Programmierung*]
- `ULConnection.Open`-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Das folgende Fragment einer Verbindungszeichenfolge legt die Cachegröße auf 20 MB fest.

```
"CACHE_SIZE=20m"
```

UltraLite-Verbindungsparameter `CE_FILE`

Dieser Verbindungsparameter enthält bei der Erstellung einer neuen UltraLite-Datenbank den Namen für die neue Datenbankdatei.

Beim Verbindungsaufbau mit einer bestehenden Datenbank wird diese über den Parameter identifiziert.

Für Android-Smartphones können Sie `Configuration.setDatabaseName` als Alternative zum Einstellen dieses Verbindungsparameter verwenden. Siehe [Configuration.setDatabaseName-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*].

Syntax

```
CE_FILE=path\ce-db
```

Standardwert

DBF-Verbindungsparameter.

Verhalten

1. Wenn DBF angegeben ist, wird nach einer entsprechenden Datenbank (identischer Dateiname) gesucht. Wenn eine solche Datenbank gefunden wird, wird eine Verbindung hergestellt. Wenn keine solche Datenbank gefunden wird, wird der automatische Start fortgesetzt.
2. Eine Datenbank wird bei Bedarf automatisch gestartet, wenn DBF angegeben ist.

Bemerkungen

Verwenden Sie den Verbindungsparameter `CE_FILE` bei UltraLite-Clientanwendungen, die dieselbe Verbindungszeichenfolge zum Verbinden mit einem Microsoft Windows Mobile-Gerät und mit anderen Plattformen verwenden.

Der Verbindungsparameter `CE_FILE` hat Vorrang vor dem Verbindungsparameter `DBF`. Wenn Sie mit einem UltraLite-Administrationstool eine Verbindung herstellen oder das Verbindungsobjekt nur eine Verbindung mit einer Windows Mobile-Datenbank herstellt, verwenden Sie den Verbindungsparameter `DBF`.

Der Wert von `CE_FILE` muss die Anforderungen an Dateinamen für Windows Mobile erfüllen. Wenn Sie einen absoluten Pfad für die Datenbank angeben, müssen alle betreffenden Verzeichnisse vorhanden sein, bevor Sie den Pfad dieser Datei angeben. UltraLite erstellt die Verzeichnisse nicht automatisch.

Alle führenden bzw. nachgestellten Leerzeichen in Parameterwerten werden ignoriert. Der Wert dieses Verbindungsparameters darf keine führenden Apostrophe, Anführungszeichen oder Semikolons enthalten.

Siehe auch

- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Dateipfadformate in Verbindungsparametern“ auf Seite 37
- Vorrang von Verbindungsparametern für UltraLite-Administrationstools auf Seite 36
- „UltraLite-Verbindungsparameter DBF“ auf Seite 177
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [*UltraLite - C- und C++-Programmierung*]
- `ULDatabaseManager.OpenConnection`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [*UltraLite - .NET-Programmierung*]
- `ULConnection.Open`-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Das folgende Beispiel stellt eine neue Verbindung her und gibt verschiedene Datenbankdateien für die Windows-PC- und die Windows Mobile-Plattform an:

```
Set Connection = DatabaseMgr.OpenConnection("DBF=d:\Dbfile.udb;CE_FILE=\myapp\MyDB.udb")
```

UltraLite-Verbindungsparameter COMMIT_FLUSH

Legt fest, wann festgeschriebene Transaktionen nach einem Festschreibeaufruf bereinigt, d.h. in den Speicher geschrieben werden. Wenn die UltraLite-Anwendung keine Festschreibeaufrufe durchführt, kann keine Bereinigung durchgeführt werden.

Syntax

COMMIT_FLUSH={ **immediate** | **grouped** | **on_checkpoint** }

Standardwert

immediate

Bemerkungen

Dieser Verbindungsparameter legt fest, welche Transaktionen nach einem Hardwarefehler oder Crash wiederhergestellt werden. Sie können logische Autocommit-Vorgänge in einen einzelnen Wiederherstellungspunkt zusammenfassen.

Durch die Gruppierung dieser Vorgänge können Sie die UltraLite-Performance verbessern. Dies geht jedoch auf Kosten der Wiederherstellbarkeit der Daten. Es besteht eine geringe Wahrscheinlichkeit, dass eine Transaktion verloren geht, selbst wenn sie festgeschrieben wurde, wenn nach einem Festschreibevorgang und bevor die Transaktion in den Speicher geschrieben wird, ein Hardwarefehler oder Systemabsturz auftritt.

Die folgenden Parameter werden unterstützt:

- **immediate** Festgeschriebene Transaktionen werden sofort nach einem commit-Aufruf und vor dem Abschluss des Festschreibvorgangs in den Speicher geschrieben.
- **grouped** Festgeschriebene Transaktionen werden bei einem commit-Aufruf in den Speicher geschrieben, jedoch erst nachdem ein konfigurierter Schwellenwert erreicht wurde. Sie können mit der Datenbankoption `commit_flush_count` einen Schwellenwert für einen Transaktionszähler oder mit der Datenbankoption `commit_flush_timeout` einen zeitbasierten Schwellenwert konfigurieren.

Die Optionen `commit_flush_count` und `commit_flush_timeout` agieren beide als mögliche Trigger für die Bereinigung der Festschreibungen. Der zuerst erreichte Schwellenwert löst jeweils den Trigger aus. Wenn die Bereinigung eintritt, setzt UltraLite den Zähler und den Zeitgeber auf 0 zurück. Danach werden sowohl der Zähler als auch der Zeitgeber überwacht, bis einer dieser Schwellenwerte erneut erreicht wird.

- **on_checkpoint** Festgeschriebene Transaktionen werden bei einem Checkpoint-Vorgang in den Speicher geschrieben. Sie können einen Checkpoint mit einer der folgenden Methoden setzen:
 - CHECKPOINT-Anweisung. APIs, die über keine Checkpoint-Methode verfügen, müssen diese SQL-Anweisung verwenden.
 - Funktion `ULCheckpoint` für UltraLite Embedded SQL
 - Checkpoint-Methode für ein Verbindungsobjekt in einer C++-Komponente

Siehe auch

- „Bereinigen einzelner oder gruppierter Transaktionen“ auf Seite 487
- „UltraLite-Option `commit_flush_count` [temporär]“ auf Seite 196
- „UltraLite-Option `commit_flush_timeout` [temporär]“ auf Seite 197
- „CHECKPOINT-Anweisung [UltraLite]“ auf Seite 432
- `ULCheckpoint`-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- `ULConnection.Checkpoint`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]

UltraLite-Verbindungsparameter CON

Benennt eine Verbindung, um das Umschalten auf diese Verbindung in Anwendungen mit mehreren Verbindungen zu erleichtern

Syntax

CON=*name*

Standardwert

Kein Verbindungsname

Bemerkungen

Der CON-Verbindungsparameter ist für die gesamte Anwendung gültig.

Verwenden Sie diesen Verbindungsparameter nur, wenn Sie zwei oder mehr gleichzeitige Verbindungen einrichten und zwischen ihnen wechseln wollen.

Der Verbindungsname ist nicht mit dem Datenbanknamen identisch.

Alle führenden bzw. nachgestellten Leerzeichen in Parameterwerten werden ignoriert. Der Wert dieses Verbindungsparameters darf keine führenden Apostrophe, Anführungszeichen oder Semikolons enthalten.

Siehe auch

- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Verbindungsparameter DBN“ auf Seite 180
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [[UltraLite - C- und C++-Programmierung](#)]
- `ULDatabaseManager.OpenConnection`-Methode [UltraLite C++] [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [[UltraLite - .NET-Programmierung](#)]
- `ULConnection.Open`-Methode [UltraLite.NET] [[UltraLite - .NET-Programmierung](#)]

Beispiel

Das folgende Fragment einer Verbindungszeichenfolge legt den Verbindungsnamen auf `MyFirstCon` fest.

```
"CON=MyFirstCon"
```

UltraLite-Verbindungsparameter DBF

Dieser Verbindungsparameter enthält bei der Erstellung einer neuen UltraLite-Datenbank den Namen für die neue Datenbankdatei.

Beim Öffnen einer Verbindung mit einer existierenden Datenbank gibt er an, welche Datenbankdatei geladen werden soll, um eine Verbindung herzustellen.

Syntax

DBF=*ul-db*

Verhalten

1. Stellt beim Herstellen der Verbindung fest, ob bereits eine Datenbank ausgeführt wird. Wenn DBF angegeben ist, wird nach einer entsprechenden Datenbank gesucht. Wenn eine solche Datenbank gefunden wird, wird eine Verbindung hergestellt. Wenn keine solche Datenbank gefunden wird, wird der automatische Start fortgesetzt.
2. Wenn DBF angegeben ist, wird nach einer entsprechenden Datenbank (identischer Dateiname) gesucht. Wenn eine solche Datenbank gefunden wird, wird eine Verbindung hergestellt. Wenn keine solche Datenbank gefunden wird, wird der automatische Start fortgesetzt.
3. Wenn weder DBN noch DBF angegeben ist und eine einzelne Datenbank ausgeführt wird, wird eine Verbindung mit dieser Datenbank hergestellt.
4. Eine Datenbank wird bei Bedarf automatisch gestartet, wenn DBF angegeben ist. Wenn DBN ebenfalls angegeben ist, wird dies der Name der laufenden Datenbank, andernfalls wird ein Name anhand des Basisdateinamens generiert.

Bemerkungen

Da es sich um Aliase handelt, hat bei der gleichzeitigen Angabe von DBF der zuletzt angegebene Wert Vorrang.

Wenn Sie mit einer einzelnen Verbindungszeichenfolge eine Verbindung mit mehreren Datenbanken auf verschiedenen Geräten herstellen, können Sie die folgenden Parameter verwenden, um plattformspezifische Alternativen anzugeben:

- CE_FILE
- desktop
- device
- NT_FILE

Wenn dieser Parameter angegeben wird, haben die plattformspezifischen Verbindungsparameter Vorrang vor DBF.

Der Wert von DBF muss die Anforderungen an Dateinamen für die Plattform erfüllen.

Windows Mobile Wenn Sie ein Deployment auf einem Windows Mobile-Gerät durchführen, können UltraLite-Dienstprogramme und Assistenten eine UltraLite-Datenbank auf einem zugeordneten Mobile-Gerät verwalten. Um eine Datei auf einem Windows Mobile-Gerät zu identifizieren, müssen Sie den erforderlichen absoluten Pfad angeben und das **wce:**-Präfix verwenden.

Alle führenden bzw. nachgestellten Leerzeichen in Parameterwerten werden ignoriert. Der Wert darf keine führenden Apostrophe, Anführungszeichen oder Semikolons enthalten.

Siehe auch

- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Dateipfadformate in Verbindungsparametern“ auf Seite 37
- Vorrang von Verbindungsparametern für UltraLite-Administrationstools auf Seite 36
- „UltraLite-Verbindungsparameter DBN“ auf Seite 180
- „UltraLite-Verbindungsparameter CE_FILE“ auf Seite 174
- „UltraLite-Verbindungsparameter NT_FILE“ auf Seite 185
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.OpenConnection-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [*UltraLite - .NET-Programmierung*]
- ULConnection.Open-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiele

Um eine Verbindung zur Datenbank *MyULdb.udb* im PC-Verzeichnis *c:\mydb* herzustellen, verwenden Sie folgende Verbindungszeichenfolge:

```
"DBF=c:\mydb\MyULdb.udb"
```

Um eine Verbindung mit derselben Datenbank im Ordner *UltraLite* des zugeordneten Windows Mobile-Geräts herzustellen, verwenden Sie die folgende Verbindungszeichenfolge:

```
"DBF=wce:\UltraLite\MyULdb.udb"
```

UltraLite-Verbindungsparameter DBKEY

Beim Erstellen einer neuen UltraLite-Datenbank liefert dieser Parameter einen Chiffrierschlüssel für die Datenbank.

Beim Öffnen einer Verbindung mit einer vorhandenen Datenbank liefert er den Chiffrierschlüssel für die Datenbank.

Syntax

```
DBKEY=string
```

Standardwert

Es wird kein Schlüssel geliefert.

Bemerkungen

Wenn Sie nicht den korrekten Chiffrierschlüssel für die Datenbank angeben, schlägt die Verbindung fehl.

Wenn eine Datenbank mit einem Chiffrierschlüssel erstellt wird, wird die Datenbank mit dem AES 256-Bit- oder dem AES FIPS-zertifizierten Algorithmus stark verschlüsselt. Die Verwendung der starken Verschlüsselung bietet Sicherheit gegenüber fachmännischen und zielgerichteten Versuchen, Zugriff auf die Daten zu gewinnen. Sie hat jedoch deutliche Auswirkungen auf die Performance.

Alle führenden bzw. nachgestellten Leerzeichen in Parameterwerten werden ignoriert. Der Wert darf keine führenden Apostrophe, Anführungszeichen oder Semikolons enthalten.

Siehe auch

- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „Datenbanksicherheit“ auf Seite 29
- „UltraLite-Erstellungsparameter obfuscate“ auf Seite 154
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.OpenConnection-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [*UltraLite - .NET-Programmierung*]
- ULConnection.Open-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

UltraLite-Verbindungsparameter DBN

Unterscheidet Datenbanken anhand des Namens, wenn sich Anwendungen mit mehr als einer Datenbank verbinden

Syntax

DBN=*db-name*

Standardwert

Keine.

Verhalten

1. Stellt beim Herstellen der Verbindung fest, ob bereits eine Datenbank ausgeführt wird. Wenn DBF angegeben ist, wird nach einer entsprechenden Datenbank gesucht. Wenn eine solche Datenbank gefunden wird, wird eine Verbindung hergestellt. Wenn keine solche Datenbank gefunden wird, wird der automatische Start fortgesetzt.
2. Wenn DBF angegeben ist, wird nach einer entsprechenden Datenbank (identischer Dateiname) gesucht. Wenn eine solche Datenbank gefunden wird, wird eine Verbindung hergestellt. Wenn keine solche Datenbank gefunden wird, wird der automatische Start fortgesetzt.
3. Wenn weder DBN noch DBF angegeben ist und eine einzelne Datenbank ausgeführt wird, wird eine Verbindung mit dieser Datenbank hergestellt.
4. Eine Datenbank wird bei Bedarf automatisch gestartet, wenn DBF angegeben ist. Wenn DBN ebenfalls angegeben ist, wird dies der Name der laufenden Datenbank, andernfalls wird ein Name anhand des Basisdateinamens generiert.

Bemerkungen

UltraLite legt den Datenbanknamen fest, nachdem die Datenbank geöffnet wurde. Clientanwendungen können mit dieser Datenbank über ihren Namen anstatt über ihre Datei eine Verbindung herstellen.

Alle führenden bzw. nachgestellten Leerzeichen in Parameterwerten werden ignoriert. Der Wert dieses Verbindungsparameters darf keine führenden Apostrophe, Anführungszeichen oder Semikolons enthalten.

Siehe auch

- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Verbindungsparameter DBF“ auf Seite 177
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.OpenConnection-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [*UltraLite - .NET-Programmierung*]
- ULConnection.Open-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Verwenden Sie die folgenden Parameter, um eine Verbindung mit der laufenden UltraLite-Datenbank namens Kitchener herzustellen:

```
DBN=Kitchener;DBF=cities.udb
```

UltraLite-Verbindungsparameter desktop

Dieser Verbindungsparameter enthält bei der Erstellung einer neuen UltraLite-Datenbank den Namen für die neue Datenbankdatei.

Beim Verbindungsaufbau mit einer bestehenden Datenbank wird diese über den Parameter identifiziert.

Syntax

```
desktop:DBF=path\db | temp_dir=Temp
```

Bemerkungen

Benutzen Sie den Verbindungsparameter "desktop" für UltraLite-Clientanwendungen, die dieselbe Verbindungszeichenfolge zur Verbindung mit einem Microsoft Windows - oder Mac-Gerät verwenden.

Der Doppelpunkt (:) wird zusätzlich zum Unterstrich (_) als Trennzeichen verwendet. Optionen mit einem Präfix haben Vorrang vor den Optionen ohne Präfix.

Der **temp_dir**-Verbindungsparameter ist nun verfügbar. Dieser muss der Name eines Verzeichnisses sein, das bereits vorhanden ist. UL legt die temporäre Datei (mit einer vom Namen der Datenbank abgeleiteten Bezeichnung) in dem angegebenen Verzeichnis ab, und nicht neben der Datenbankdatei (Standardwert und früheres Verhalten). Die Angabe von einem Verzeichnis für temporäre Dateien mit schnelleren I/O-Merkmalen kann die Performance verbessern, beispielsweise von temporären Tabellen, die im Vergleich zur Cachelgröße groß sind. Transaktionen mit langen Laufzeiten können auch viel Speicherplatz in der temporären Datei beanspruchen.

Der Wert von "desktop" muss den Anforderungen für Dateinamen von Windows oder Mac entsprechen. Wenn Sie einen absoluten Pfad für die Datenbank angeben, müssen alle betreffenden Verzeichnisse

vorhanden sein, bevor Sie den Pfad dieser Datei angeben. UltraLite erstellt die Verzeichnisse nicht automatisch.

Alle führenden bzw. nachgestellten Leerzeichen in Parameterwerten werden ignoriert. Der Wert dieses Verbindungsparameters darf keine führenden Apostrophe, Anführungszeichen oder Semikolons enthalten.

Verhalten

1. Wenn DBF angegeben ist, wird nach einer entsprechenden Datenbank (identischer Dateiname) gesucht. Wenn eine solche Datenbank gefunden wird, wird eine Verbindung hergestellt. Wenn keine solche Datenbank gefunden wird, wird der automatische Start fortgesetzt.
2. Eine Datenbank wird bei Bedarf automatisch gestartet, wenn DBF angegeben ist.

Siehe auch

- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Dateipfadformate in Verbindungsparametern“ auf Seite 37
- Vorrang von Verbindungsparametern für UltraLite-Administrationstools auf Seite 36
- „UltraLite-Verbindungsparameter DBF“ auf Seite 177
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.OpenConnection-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [*UltraLite - .NET-Programmierung*]
- ULConnection.Open-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Das folgende Beispiel stellt eine neue Verbindung her und gibt verschiedene Datenbankdateien für die Windows-PC- und die Windows Mobile-Plattform an:

```
"desktop:DBF=C:\dir\db.udb; device:DBF=\SD Card\db.udb; device:temp_dir=\Temp; device:cache_size=4M"
```

UltraLite-Verbindungsparameter device

Dieser Verbindungsparameter enthält bei der Erstellung einer neuen UltraLite-Datenbank den Namen für die neue Datenbankdatei.

Beim Verbindungsaufbau mit einer bestehenden Datenbank wird diese über den Parameter identifiziert.

Syntax

device:DBF=*path***\db | temp_dir=***Temp*

Bemerkungen

Verwenden Sie den Verbindungsparameter "device" für UltraLite-Clientanwendungen, die dieselbe Verbindungszeichenfolge zur Verbindung mit einem Microsoft Windows CE- oder iPhone-Gerät verwenden.

Der Doppelpunkt (:) wird zusätzlich zum Unterstrich (_) als Trennzeichen verwendet. Optionen mit einem Präfix haben Vorrang vor den Optionen ohne Präfix.

Der **temp_dir**-Verbindungsparameter ist nun verfügbar. Dieser muss der Name eines Verzeichnisses sein, das bereits vorhanden ist. UL legt die temporäre Datei (mit einer vom Namen der Datenbank abgeleiteten Bezeichnung) in dem angegebenen Verzeichnis ab, und nicht neben der Datenbankdatei (Standardwert und früheres Verhalten). Die Angabe von einem Verzeichnis für temporäre Dateien mit schnelleren I/O-Merkmalen kann die Performance verbessern, beispielsweise von temporären Tabellen, die im Vergleich zur Cachegröße groß sind. Transaktionen mit langen Laufzeiten können auch viel Speicherplatz in der temporären Datei beanspruchen.

Der Wert von "device" muss die Anforderungen für Dateinamen von Windows CE oder iPhone erfüllen. Wenn Sie einen absoluten Pfad für die Datenbank angeben, müssen alle betreffenden Verzeichnisse vorhanden sein, bevor Sie den Pfad dieser Datei angeben. UltraLite erstellt die Verzeichnisse nicht automatisch.

Alle führenden bzw. nachgestellten Leerzeichen in Parameterwerten werden ignoriert. Der Wert dieses Verbindungsparameters darf keine führenden Apostrophe, Anführungszeichen oder Semikolons enthalten.

Siehe auch

- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Dateipfadformate in Verbindungsparametern“ auf Seite 37
- Vorrang von Verbindungsparametern für UltraLite-Administrationstools auf Seite 36
- „UltraLite-Verbindungsparameter DBF“ auf Seite 177
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.OpenConnection-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [*UltraLite - .NET-Programmierung*]
- ULConnection.Open-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Das folgende Beispiel stellt eine neue Verbindung her und gibt verschiedene Datenbankdateien für die Windows-PC- und die Windows Mobile-Plattform an:

```
"desktop:DBF=C:\dir\db.udb; device:DBF=\SD Card\db.udb; device:temp_dir=
\Temp; device:cache_size=4M"
```

UltraLite-Verbindungsparameter MIRROR_FILE

Gibt den Namen der Datenbankspiegeldatei an, in die alle Schreibvorgänge der Datenbank ausgegeben werden (zum gleichen Zeitpunkt, zu dem sie in die Hauptdatenbankdatei geschrieben werden).

Syntax

MIRROR_FILE=*path\mirrorfile-db*

Standardwert

Keine.

Bemerkungen

UltraLite ermöglicht die Spiegelung von Datenbankdateien, um die Fehlertoleranz in potenziell unzuverlässigen Speichersystemen zu erhöhen. Für diesen Zweck wird die Spiegeldatei verwendet. Alle Datenbankschreibvorgänge werden gleichzeitig in die Spiegeldatei und in die Hauptdatenbankdatei ausgegeben. (Der Overhead für den Schreibvorgang wird daher verdoppelt. Der Lese-Overhead ändert sich nicht). Wenn eine beschädigte Seite aus der Datenbankdatei gelesen wird, wird die Seite wiederhergestellt, indem sie aus der Spiegeldatei gelesen wird.

Das Spiegeln wird auf allen Plattformen mit einem dateibasierten Speicher unterstützt.

Wenn beim Start der Datenbank die Option **mirror_file**= angegeben wird, öffnet UltraLite die angegebene Datei und prüft zunächst, ob sie mit der Hauptdatenbank übereinstimmt. Falls die Spiegeldatei nicht vorhanden ist, wird sie zu diesem Zeitpunkt durch Kopieren der Hauptdatei erstellt. Falls die Spiegeldatei keine Datenbankdatei oder beschädigt ist, wird ein Fehler gemeldet und die Datenbank wird erst gestartet, wenn die Datei entfernt wurde oder eine andere Spiegeldatei angegeben wird. Wenn die Spiegeldatei nicht mit der Datenbank übereinstimmt, wird **SQL_E_MIRROR_FILE_MISMATCH** generiert und die Datenbank wird nicht gestartet. Wenn eine beschädigte Seite wiederhergestellt wird, wird die Warnung **SQL_E_CORRUPT_PAGE_READ_RETRY** generiert. (Ohne Spiegelung oder wenn die Spiegeldatei ebenfalls beschädigt ist, wird der Fehler **SQL_E_DEVICE_ERROR** generiert und die Datenbank wird angehalten.)

Um einen effektiven Schutz gegen Datenträgerausfälle zu gewährleisten, müssen Seitenprüfsummen bei der Verwendung einer Spiegeldatei aktiviert werden. (Sowohl mit als auch ohne Spiegelung kann UltraLite mithilfe von Prüfsummen die Beschädigung einer Seite bereits beim Laden der Seite erkennen und somit vermeiden, beschädigte Daten zu referenzieren.) Zur Aktivierung von Prüfsummen geben Sie bei der Datenbankerstellung die Option **checksum_level** an. Wenn eine Spiegeldatei verwendet wird und Prüfsummen nicht aktiviert sind, generiert UltraLite die Warnung **SQL_E_MIRROR_FILE_REQUIRES_CHECKSUMS**.

Die Spiegeldatei ist eine exakte Kopie der Datenbankdatei und kann daher direkt als Datenbank gestartet werden. Das Dienstprogramm ulvalid meldet beschädigte Seiten.

Siehe auch

- „UltraLite-Erstellungsparameter checksum_level“ auf Seite 144
- „UltraLite-Dienstprogramm zum Validieren von Datenbanken (ulvalid)“ auf Seite 237
- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Dateipfadformate in Verbindungsparametern“ auf Seite 37
- Vorrang von Verbindungsparametern für UltraLite-Administrationstools auf Seite 36
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.OpenConnection-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [*UltraLite - .NET-Programmierung*]
- ULConnection.Open-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Das folgende Beispiel erstellt eine neue Verbindung und eine Spiegeldatei:

```
Connection = DatabaseMgr.OpenConnection("DBF=c:\Dbfile.udb;  
UID=JDoe;PWD=ULdb;  
MIRROR_FILE=c:\test\MyMirrorDB.udb")
```

UltraLite-Verbindungsparameter NT_FILE

Dieser Verbindungsparameter enthält bei der Erstellung einer neuen UltraLite-Datenbank den Namen für die neue Datenbankdatei.

Beim Verbindungsaufbau mit einer bestehenden Datenbank wird diese über den Parameter identifiziert.

Für Android-Smartphones können Sie Configuration.setDatabaseName als Alternative zum Einstellen dieses Verbindungsparameter verwenden. Siehe [Configuration.setDatabaseName-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*].

Syntax

NT_FILE=path\nt-db

Standardwert

DBF-Verbindungsparameter.

Bemerkungen

Verwenden Sie den Verbindungsparameter NT_FILE bei UltraLite-Clientanwendungen, die dieselbe Verbindungszeichenfolge für eine PC-Datenbank und eine Datenbank auf anderen Plattformen verwenden.

Dieser Verbindungsparameter hat Vorrang vor dem Parameter DBF. Wenn Sie mit einem UltraLite-Administrationstool eine Verbindung aufnehmen oder das Verbindungsobjekt nur eine Verbindung mit einer PC-Datenbank herstellt, verwenden Sie den Verbindungsparameter DBF.

Der Wert von NT_FILE muss die Anforderungen an Dateinamen für Windows PC-Plattformen erfüllen.

Der Pfad kann in absoluter oder relativer Form angegeben werden. Wenn Sie ein Verzeichnis als Teil des Dateinamens angeben, müssen alle betreffenden Verzeichnisse vorhanden sein, bevor Sie den Pfad zu dieser Datei angeben. UltraLite erstellt die Verzeichnisse nicht automatisch.

Alle führenden bzw. nachgestellten Leerzeichen in Parameterwerten werden ignoriert. Der Wert dieses Verbindungsparameters darf keine führenden Apostrophe, Anführungszeichen oder Semikolons enthalten.

Siehe auch

- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Dateipfadformate in Verbindungsparametern“ auf Seite 37
- Vorrang von Verbindungsparametern für UltraLite-Administrationstools auf Seite 36
- „UltraLite-Verbindungsparameter DBF“ auf Seite 177
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseManager.OpenConnection-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [*UltraLite - .NET-Programmierung*]
- ULConnection.Open-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Das folgende Beispiel stellt eine neue Verbindung her und gibt verschiedene Datenbankdateien für die Windows-PC- und die Mobile-Plattform an:

```
Connection = DatabaseMgr.OpenConnection( "UID=JDoe;PWD=ULdb;  
NT_FILE=c:\test\MyTestDB.udb;CE_FILE=\database\MyCEDB.udb" )
```

UltraLite-Verbindungsparameter PWD

Wenn Sie eine neue UltraLite-Datenbank erstellen, legt dieser Verbindungsparameter das Kennwort für den Standardbenutzer fest.

Wenn Sie eine Verbindung zu einer vorhandenen Datenbank herstellen, definiert er das Kennwort für eine Benutzer-ID, die für die Authentifizierung verwendet wird.

Für Android-Smartphones können Sie Configuration.setPassword als Alternative zum Einstellen dieses Verbindungsparameters verwenden. Siehe [Configuration.setPassword-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*].

Syntax

PWD=password

Standardwert

Wenn Sie weder UID noch PWD angeben, öffnet UltraLite Verbindungen mit **UID=DBA** und **PWD=sql**.

Bemerkungen

Jeder Benutzer einer Datenbank hat ein Kennwort. UltraLite unterstützt bis zu vier Kombinationen aus Benutzer-IDs und Kennwörtern.

Es ist möglich, Kennwörter auf NULL oder eine leere Zeichenfolge zu setzen.

Ein Zufallswert mit 4 Byte Länge (Salt) wird generiert, wenn ein neuer Benutzer erstellt wird oder ein bestehender Benutzer sein Kennwort ändert. Der Salt-Wert wird an das Kennwort des Benutzers angehängt, wenn der Kennwort-Hash berechnet und in der Datenbank zusammen mit dem Hash gespeichert wird. Dies vermindert deutlich die Anfälligkeit gegenüber Wörterbuchangriffen und stellt sicher, dass für Benutzer mit demselben Kennwort verschiedene Kennwort-Hashes vorhanden sind.

Dieser Verbindungsparameter ist nicht verschlüsselt. UltraLite erstellt jedoch Hashes vor dem Speichern eines Kennworts, sodass Kennwörter nur über Sybase Central geändert werden können.

Siehe auch

- „Benutzer“ [*SQL Anywhere Server - Datenbankadministration*]
- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Benutzer“ auf Seite 61
- „UltraLite-Verbindungsparameter UID“ auf Seite 190
- UltraLite für C/C++: „Benutzerauthentifizierung“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [*UltraLite - C- und C++-Programmierung*]
- `ULDatabaseManager.OpenConnection`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „Benutzerauthentifizierung“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [*UltraLite - .NET-Programmierung*]
- `ULConnection.Open`-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiele

Der folgende Abschnitt einer Verbindungszeichenfolge gibt die Benutzer-ID DBA und das Kennwort sql an.

```
"UID=DBA;PWD=sql"
```

Der folgende Abschnitt einer Verbindungszeichenfolge gibt die Benutzer-ID DBA und ein leeres Kennwort an:

```
"UID=DBA;PWD= ' ' "
```

UltraLite-Verbindungsparameter RESERVE_SIZE

Weist für die UltraLite-Datenbank vorab Speicherplatz im Dateisystem zu, ohne Daten einzufügen. Das Reservieren von Speicherplatz im Dateisystem bedeutet, dass der Speicherplatz nicht von anderen Dateien belegt werden kann.

Syntax

RESERVE_SIZE= *number*{ **k** | **m** | **g** }

Standardwert

0 (keine Reservierungsgröße).

Bemerkungen

Der angegebene Wert kann zwischen 0 und der maximalen Datenbankgröße liegen. Verwenden Sie das Suffix k, m oder g, um die Einheit in kB, MB oder GB anzugeben. Wenn Sie keine Einheit angeben, wird standardmäßig von Byte ausgegangen.

Sie sollten die Datenbank mit Testdaten ausführen und die Datenbankgröße beobachten, um anschließend die für Ihr UltraLite-Deployment geeignete Reservierungsgröße zu wählen.

Wenn der Wert von RESERVE_SIZE kleiner als die Datenbankdateigröße ist, ignoriert UltraLite den Parameter.

Das Reservieren von Speicherplatz im Dateisystem kann die Performance aus folgenden Gründen etwas erhöhen:

- Weil es, verglichen mit inkrementellem Zuwachs, das Ausmaß der Dateifragmentierung vermindert.
- Weil es Speicherfehler aufgrund von fehlendem Speicherplatz vermeidet.

Da eine UltraLite-Datenbank aus Daten und Metadaten besteht, nimmt die Datenbankgröße nur zu, wenn dies erforderlich ist (wenn die Anwendung die Datenbank aktualisiert).

Siehe auch

- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Verbindungsparameter CACHE_SIZE“ auf Seite 172
- „UltraLite-Erstellungsparameter page_size“ auf Seite 155
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [[UltraLite - C- und C++-Programmierung](#)]
- ULDatabaseManager.OpenConnection-Methode [UltraLite C++] [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [[UltraLite - .NET-Programmierung](#)]
- ULConnection.Open-Methode [UltraLite.NET] [[UltraLite - .NET-Programmierung](#)]

Beispiel

Das folgende Fragment einer Verbindungszeichenfolge legt die reservierte Größe auf 128 kB fest, sodass das System beim Start Systemspeicher in diesem Umfang für die Datenbank reserviert.

```
"RESERVE_SIZE=128K"
```

UltraLite-Verbindungsparameter START

Startet die Programmdatei der UltraLite-Engine. Dieser Parameter wird von UltraLite für Android nicht unterstützt. Dieser Parameter ist nur erforderlich, wenn die Engine sich an keinem der erwarteten Speicherorte befindet.

Syntax

START=*path***\uleng16.exe**

Bemerkungen

Geben Sie den Verbindungsparameter StartLine (START) nur beim Verbinden mit einer Engine an, die aktuell nicht läuft.

Pfade mit Leerstellen erfordern Anführungszeichen. Anderenfalls gibt der Client SQLE_UNABLE_TO_CONNECT_OR_START zurück.

Siehe auch

- „Start der UltraLite-Engine“ auf Seite 128
- „UltraLite-Engine-Dienstprogramm (uleng16)“ auf Seite 210
- „UltraLite-Datenverwaltungskomponenten für Windows Mobile“ auf Seite 19
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [[UltraLite - C- und C++-Programmierung](#)]
- ULDatabaseManager.OpenConnection-Methode [UltraLite C++] [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [[UltraLite - .NET-Programmierung](#)]
- ULConnection.Open-Methode [UltraLite.NET] [[UltraLite - .NET-Programmierung](#)]

Beispiel

Der folgende Befehl startet die UltraLite-Engine, die sich im Verzeichnis *Programme* befindet.

```
Start="Program Files\uleng16.exe"
```

Eine andere Möglichkeit, diesen Pfad festzulegen, wäre, die gesamte Zeichenfolge in Apostrophe zu setzen:

```
Start='"\Program Files\uleng16.exe"'
```

UltraLite-Verbindungsparameter TEMP_DIR

Gibt den Namen des Verzeichnisses (das bereits vorhanden sein muss) an, in dem UltraLite eine temporäre Datei ablegt, deren Name vom Namen der Datenbank abgeleitet wird.

Syntax

TEMP_DIR=*path*

Bemerkungen

Zusätzlich zur Datenbankdatei erstellt und verwaltet UltraLite während des Datenbankvorgangs eine temporäre Datei. Sie müssen diese Datei weder verwenden noch verwalten.

Die temporäre UltraLite-Datei befindet sich standardmäßig im selben Ordner wie die UltraLite-Datenbank (sofern vorhanden). Die temporäre Datei hat denselben Dateinamen wie die Datenbank, aber bei dateibasierten Plattformen wird eine Tilde in die Dateierweiterung einbezogen. Wenn Sie z. B. die Beispieldatenbank *CustDB.udb* ausführen, wird die temporäre Datei namens *CustDB.~db* in demselben Verzeichnis gespeichert wie die Datenbankdatei.

Die Angabe von einem Verzeichnis für temporäre Dateien mit schnelleren I/O-Merkmalen kann die Performance verbessern, beispielsweise von temporären Tabellen, die im Vergleich zur Cachegröße groß sind. Transaktionen mit langen Laufzeiten können auch viel Speicherplatz in der temporären Datei beanspruchen.

Pfade mit Leerstellen erfordern Anführungszeichen. Anderenfalls gibt der Client `SQL_UNABLE_TO_CONNECT_OR_START` zurück.

Siehe auch

- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [[UltraLite - C- und C++-Programmierung](#)]
- `ULDatabaseManager.OpenConnection`-Methode [UltraLite C++] [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [[UltraLite - .NET-Programmierung](#)]
- `ULConnection.Open`-Methode [UltraLite.NET] [[UltraLite - .NET-Programmierung](#)]

Beispiel

Mit dem folgenden Fragment einer Verbindungszeichenfolge wird die temporäre Datei im Verzeichnis `\Temp` abgelegt:

```
temp_dir=\Temp;
```

UltraLite-Verbindungsparameter UID

Wenn Sie eine neue UltraLite-Datenbank erstellen, legt dieser Verbindungsparameter die Standard-Benutzer-ID für die Datenbank fest.

Wenn Sie eine Verbindung zu einer vorhandenen Datenbank herstellen, gibt er die Benutzer-ID an, mit der Sie sich mit der Datenbank verbinden. Der Wert muss ein authentifizierter Benutzer für die Datenbank sein.

Syntax

`UID=user`

Standardwert

Wenn Sie bei der Verbindungsaufnahme weder UID noch PWD angeben, öffnet UltraLite-Verbindungen mit **UID=DBA** und **PWD=sql**.

Bemerkungen

Jeder Benutzer einer Datenbank hat eine Benutzer-ID. UltraLite unterstützt bis zu vier Kombinationen aus Benutzer-IDs und Kennwörtern.

UltraLite-Benutzer-IDs unterscheiden sich von MobiLink-Benutzernamen und anderen SQL Anywhere-Benutzer-IDs. Es ist nicht möglich, eine Benutzer-ID nach ihrer Erstellung zu ändern. Sie müssen stattdessen die Benutzer-ID löschen und eine neue hinzufügen.

Es ist nicht möglich, die UID auf NULL oder eine leere Zeichenfolge zu setzen. Die maximale Länge einer Benutzer-ID beträgt 31 Zeichen. Benutzer-IDs reagieren nicht auf Groß-/Kleinschreibung.

Alle führenden bzw. nachgestellten Leerzeichen in Parameterwerten werden ignoriert. Der Wert dieses Verbindungsparameters darf keine führenden Apostrophe, Anführungszeichen oder Semikolons enthalten.

Siehe auch

- „UltraLite-Verbindungszeichenfolgen und Parameter“ auf Seite 35
- „UltraLite-Benutzer“ auf Seite 61
- UltraLite für C/C++: „Benutzerauthentifizierung“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite für C/C++: „UltraLite-Datenbankverbindungen“ [*UltraLite - C- und C++-Programmierung*]
- `ULDatabaseManager.OpenConnection`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „Benutzerauthentifizierung“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite für Embedded SQL: „UltraLite-Datenbankverbindung mithilfe von Embedded SQL verwenden“ [*UltraLite - C- und C++-Programmierung*]
- UltraLite.NET: „Verbindungseinrichtung für eine UltraLite-Datenbank“ [*UltraLite - .NET-Programmierung*]
- `ULConnection.Open`-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Das folgende Fragment einer Verbindungszeichenfolge liefert die Benutzer-ID DBA und das Kennwort sql für eine Datenbank:

```
"UID=DBA;PWD=sql"
```

UltraLite-Datenbankeigenschaften

UltraLite-Datenbankeigenschaftswerte werden definiert, wenn die Datenbank erstellt wird. Sie können geändert werden, indem Sie die UltraLite-Datenbank neu erstellen oder die entsprechenden Datenbankoptionen bearbeiten.

UltraLite unterstützt die folgenden Datenbankeigenschaften:

Eigenschaft	Beschreibung
cache_allocation	Gibt die aktuelle Cachegröße als Prozentsatz der Mindest- und Höchsteinstellungen an. Siehe „ UltraLite-Option cache_allocation “ auf Seite 195.
CaseSensitive	Gibt den Status der Berücksichtigungsfunktion für Groß- und Kleinschreibung zurück. Gibt "On" zurück, wenn die Datenbank die Groß- und Kleinschreibung berücksichtigt. Anderenfalls wird "Off" zurückgegeben. Siehe „ UltraLite-Erstellungsparameter case “ auf Seite 143.
CharSet	Gibt den CHAR-Zeichensatz der Datenbank zurück. Der von der Datenbank verwendete Zeichensatz hängt von der Kollationssequenz der Datenbank ab und davon, ob die Daten UTF-8-kodiert sind. Siehe auch: <ul style="list-style-type: none"> • „UltraLite-Erstellungsparameter utf8_encoding“ auf Seite 167 • „UltraLite-Erstellungsparameter collation“ auf Seite 145
ChecksumLevel	Gibt die Ebene der Prüfsummenvalidierung in der Datenbank zurück: 0 (keine Prüfsummen hinzufügen). 1 (Prüfsummen nur wichtigen Seiten hinzufügen) oder 2 (Prüfsummen allen Seiten hinzufügen). Siehe „ UltraLite-Erstellungsparameter checksum_level “ auf Seite 144.
Collation	Gibt den Namen der Kollationssequenz der Datenbank zurück. Siehe „ UltraLite-Erstellungsparameter collation “ auf Seite 145.
commit_flush_count	Gibt den Wert der Option commit_flush_count zurück, die einen Schwellenwert für den Festschreibungszähler setzt. Siehe „ UltraLite-Option commit_flush_count [temporär] “ auf Seite 196.
commit_flush_timeout	Gibt den Wert der Option commit_flush_timeout zurück, die einen Schwellenwert für das Zeitintervall setzt. Siehe „ UltraLite-Option commit_flush_timeout [temporär] “ auf Seite 197.
ConnCount	Gibt die Anzahl der Verbindungen zur Datenbank zurück. Der Wert ist dynamisch und kann je nach Anzahl der aktuell vorhandenen Verbindungen variieren. UltraLite unterstützt bis zu 14 gleichzeitige Datenbankverbindungen.
date_format	Gibt das Datumsformat zurück, das die Datenbank für Zeichenfolge-Konvertierungen verwendet. Siehe „ UltraLite-Erstellungsparameter date_format “ auf Seite 146.

Eigenschaft	Beschreibung
date_order	Gibt die Datumsreihenfolge zurück, die die Datenbank für Zeichenfolge-Konvertierungen verwendet. Siehe „ UltraLite-Erstellungsparameter date_order “ auf Seite 149.
Encryption	<p>Gibt den Typ der Datenbankverschlüsselung zurück: None, Simple, AES oder AES FIPS.</p> <p>Die von der Datenbank verwendete Verschlüsselung hängt davon ab, ob Sie die starke Verschlüsselung (AES oder AES_FIPS) und den DBKEY-Verbindungsparameter oder die Verschleierung (einfache Verschlüsselung) konfiguriert haben.</p> <p>Diese Eigenschaft kann sich nur ändern, wenn der Wert ursprünglich None ist (d.h. es wird weder FIPS noch Verschleierung verwendet) und Sie dann den Chiffrierschlüssel ändern, indem Sie im Connection-Objekt einen neuen Chiffrierschlüssel erstellen, indem Sie die für Ihre API korrekte Funktion oder Methode aufrufen. In diesem Fall wird der Wert auf AES geändert, da der Erstellungsparameter fips nicht festgelegt werden kann, nachdem die Datenbank erstellt wurde. Siehe:</p> <ul style="list-style-type: none"> • ULChangeEncryptionKey-Methode [UltraLite Embedded SQL] [UltraLite - C- und C++-Programmierung] • ULConnection.ChangeEncryptionKey-Methode [UltraLite.NET] [UltraLite - .NET-Programmierung] • „Datenbanksicherheit“ auf Seite 29 • „UltraLite-Erstellungsparameter fips“ auf Seite 150 • „UltraLite-Erstellungsparameter obfuscate“ auf Seite 154 • „UltraLite-Verbindungsparameter DBKEY“ auf Seite 179
File	<p>Gibt den Dateinamen der Datenbankstammdatei für die aktuelle Verbindung einschließlich des Pfads zurück. Dies ist der Wert, der im DBF-Verbindungsparameter angegeben ist. Siehe:</p> <ul style="list-style-type: none"> • „UltraLite-Verbindungsparameter DBF“ auf Seite 177
global_database_id	Gibt den Wert der Option global_database_id zurück, der für globalAutoincrement-Spalten verwendet wird. Siehe „ UltraLite-Option global_database_id “ auf Seite 198.
isolation_level	Gibt die aktuelle Isolationsstufe der Datenbank zurück. Der Wert kann read_committed oder read_uncommitted sein. Siehe „ Isolationsstufen “ auf Seite 43.

Eigenschaft	Beschreibung
MaxHashSize	Gibt die standardmäßige maximale Anzahl von Byte zurück, die für den Index-Hash verwendet werden. Diese Eigenschaft kann pro Index festgelegt werden. Siehe „ UltraLite-Erstellungsparameter max_hash_size “ auf Seite 151.
ml_remote_id	Gibt den Wert der Option ml_remote_id zurück, der die Datenbank bei der MobiLink-Synchronisation eindeutig identifiziert. Siehe „ UltraLite ml_remote_id-Option “ auf Seite 199.
Name	Gibt den Namen (oder Alias) der Datenbank für die aktuelle Verbindung zurück. Der zurückgegebene Name entspricht dem Wert des DBF-Verbindungsparameters. Wenn Sie den DBN-Verbindungsparameter nicht verwendet haben, ist der zurückgegebene Name die Datenbankdatei ohne Pfad und Erweiterung. Siehe auch: <ul style="list-style-type: none"> • „UltraLite-Verbindungsparameter DBN“ auf Seite 180 • „UltraLite-Verbindungsparameter DBF“ auf Seite 177
nearest_century	Gibt das nächste Jahrhundert zurück, das die Datenbank für Zeichenfolge-Konvertierungen verwendet. Siehe „ UltraLite-Erstellungsparameter nearest_century “ auf Seite 153.
PageSize	Gibt die Seitengröße der Datenbank (in Byte) zurück. Siehe „ UltraLite-Erstellungsparameter page_size “ auf Seite 155.
precision	Gibt die Gleitkomma-Gesamtstellenzahl zurück, die die Datenbank für Zeichenfolge-Konvertierungen verwendet. Siehe „ UltraLite-Erstellungsparameter precision “ auf Seite 157.
scale	Gibt die minimale Anzahl der Stellen nach dem Dezimalzeichen zurück, wenn ein arithmetisches Ergebnis während Zeichenfolge-Konvertierungen auf die maximale PRECISION von der Datenbank gekürzt wird. Siehe „ UltraLite-Erstellungsparameter scale “ auf Seite 158.
time_format	Gibt das Zeitformat zurück, das die Datenbank für Zeichenfolge-Konvertierungen verwendet. Siehe „ UltraLite-Erstellungsparameter time_format “ auf Seite 160.
timestamp_format	Gibt das Zeitstempelformat zurück, das die Datenbank für Zeichenfolge-Konvertierungen verwendet. Siehe „ UltraLite-Erstellungsparameter timestamp_format “ auf Seite 162.

Eigenschaft	Beschreibung
timestamp_increment	Gibt die minimale Differenzmenge zwischen zwei eindeutigen Zeitstempeln in Mikrosekunden zurück. Siehe „ UltraLite-Erstellungsparameter timestamp_increment “ auf Seite 164.
timestamp_with_time_zone	Gibt das Zeitstempelformat für TIMESTAMP WITH TIME ZONE-Werte zurück. Siehe „ UltraLite-Erstellungsparameter timestamp_with_time_zone_format “ auf Seite 166.

Siehe auch

- „Lesen der Datenbankeigenschaften“ auf Seite 39
- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „UltraLite-Erstellungsparameter“ auf Seite 141
- „UltraLite-Datenbankoptionen“ auf Seite 195

UltraLite-Datenbankoptionen

Dieser Abschnitt beschreibt die UltraLite-Datenbankoptionen, die verfügbar sind.

UltraLite-Datenbankoptionen werden definiert, wenn die Datenbank erstellt wird, und können geändert werden, während Sie mit der Datenbank verbunden sind.

Siehe auch

- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „Lesen der Datenbankeigenschaften“ auf Seite 39
- „UltraLite-Erstellungsparameter“ auf Seite 141

UltraLite-Option `cache_allocation`

Bestimmt die Cachegröße explizit neu. Der Wert ist ein Prozentsatz des Bereichs zwischen Minimum und Maximum. Ein Wert von 0 stellt die minimale Größe dar, ein Wert von 100 ist die Maximalgröße.

Zulässige Werte

Ganzzahl

Standardwert

Keine.

Bemerkungen

Die Eigenschaft `cache_allocation` gibt die aktuelle Cachegröße als Prozentsatz der Minimum- und Maximum-Cachegröße an.

Siehe auch

- „UltraLite-Datenbankeigenschaften“ auf Seite 191
- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- „UltraLite-Verbindungsparameter CACHE_SIZE“ auf Seite 172
- „UltraLite-Verbindungsparameter CACHE_MIN_SIZE“ auf Seite 171
- „UltraLite-Verbindungsparameter CACHE_MAX_SIZE“ auf Seite 170
- ULConnection.SetDatabaseOptionInt-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULSetDatabaseOptionULong-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseSchema.SetDatabaseOption-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

UltraLite-Option `commit_flush_count` [temporär]

Legt einen Schwellenwert für den Festschreibungszähler fest, nach dessen Überschreiten die Festschreibungen bereinigt werden

Zulässige Werte

Ganzzahl

Standardwert

10

Bemerkungen

Mit 0 deaktivieren Sie den Transaktionszähler. Bei deaktiviertem Transaktionszähler ist die Anzahl von Festschreibungen unbegrenzt, wenn eine Bereinigung ausgelöst wird.

Sowohl `commit_flush_count` als auch `commit_flush_timeout` sind temporäre Datenbankoptionen. Sie müssen diese Optionen bei jedem Start der Datenbank neu einstellen. Sie gelten nur für den Zeitraum, in dem die Datenbank aktiv ist. Sie sind nur erforderlich, wenn Sie `COMMIT_FLUSH=grouped` als Teil einer Verbindungszeichenfolge festlegen.

Wenn Sie diese Option festlegen und den Verbindungsparameter `COMMIT_FLUSH` auf "grouped" in Ihrer Verbindungszeichenfolge setzen, löst einer der beiden Schwellenwerte eine Bereinigung aus. Wenn die Bereinigung ausgeführt wird, setzt UltraLite den Zähler *und* den Zeitgeber zurück auf 0. Anschließend werden sowohl der Zähler als auch der Zeitgeber überwacht, bis nachfolgend einer der beiden Schwellenwerte erreicht wird.

Eine wichtige Überlegung beim Einstellen der Optionen zum Bereinigen von Festschreibungen ist, wie stark die Verzögerung der Bereinigung von festgeschriebenen Transaktionen die Wiederherstellbarkeit der Daten gefährdet. Es besteht eine geringe Möglichkeit, dass eine Transaktion verloren geht, nachdem sie festgeschrieben wurde. Wenn nach einer Festschreibung und bevor die Transaktion in den Speicher geschrieben wird, ein schwerwiegender Hardwarefehler auftritt, wird die Transaktion bei einer Wiederherstellung zurückgesetzt. Eine längere Verzögerung kann die UltraLite- Performance steigern. Sie müssen den Schwellenwert für den Zähler mit Umsicht auswählen.

Um die Option `commit_flush_count` von einer Clientanwendung aus zu setzen, legen Sie die Option unter Verwendung der Funktion `SetDatabaseOption` für die von Ihnen verwendete Programmierschnittstelle fest oder verwenden Sie die SQL-Anweisung `SET OPTION`.

Siehe auch

- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- „Bereinigen einzelner oder gruppierter Transaktionen“ auf Seite 487
- „UltraLite-Option `commit_flush_timeout` [temporär]“ auf Seite 197
- „UltraLite-Verbindungsparameter `COMMIT_FLUSH`“ auf Seite 175
- `ULConnection.SetDatabaseOptionInt`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- `ULSetDatabaseOptionULong`-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- `ULDatabaseSchema.SetDatabaseOption`-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

UltraLite-Option `commit_flush_timeout` [temporär]

Legt einen Schwellenwert für das Zeitintervall fest, nach dem eine gruppierte Festschreibung bereinigt wird

Zulässige Werte

Ganzzahl, in Millisekunden

Standardwert

10000 Millisekunden

Bemerkungen

Mit 0 deaktivieren Sie den Zeitschwellenwert.

Sowohl `commit_flush_count` als auch `commit_flush_timeout` sind temporäre Datenbankoptionen. Sie müssen diese Optionen bei jedem Start der Datenbank neu einstellen. Sie gelten nur für den Zeitraum, in dem die Datenbank aktiv ist. Sie sind nur erforderlich, wenn Sie `COMMIT_FLUSH=grouped` als Teil einer Verbindungszeichenfolge festlegen.

Wenn Sie diese Option zusätzlich zur Option `commit_flush_timeout` festlegen und den Verbindungsparameter `COMMIT_FLUSH` auf `grouped` gesetzt haben, lösen beide Schwellenwerte eine Bereinigung aus. Wenn die Bereinigung ausgeführt wird, setzt UltraLite den Zähler *und* den Zeitgeber zurück auf 0. Anschließend werden sowohl der Zähler als auch der Zeitgeber überwacht, bis nachfolgend einer der beiden Schwellenwerte erreicht wird.

Eine wichtige Überlegung beim Einstellen der Optionen zum Bereinigen von Festschreibungen ist, wie stark die Verzögerung der Bereinigung von festgeschriebenen Transaktionen die Wiederherstellbarkeit der Daten gefährdet. Es besteht eine geringe Möglichkeit, dass eine Transaktion verloren geht, nachdem sie festgeschrieben wurde. Wenn nach einer Festschreibung und bevor die Transaktion in den Speicher geschrieben wird, ein schwerwiegender Hardwarefehler auftritt, wird die Transaktion bei einer

Wiederherstellung zurückgesetzt. Eine längere Verzögerung kann die UltraLite- Performance steigern. Sie müssen den Schwellenwert für den Timeout mit Umsicht wählen.

Um die Option `commit_flush_timeout` von einer Clientanwendung aus zu setzen, legen Sie sie unter Verwendung der Funktion `SetDatabaseOption` für die von Ihnen verwendete Programmierschnittstelle fest oder verwenden Sie die SQL-Anweisung `SET OPTION`.

Siehe auch

- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- „UltraLite-Option `commit_flush_count` [temporär]“ auf Seite 196
- „UltraLite-Verbindungsparameter `COMMIT_FLUSH`“ auf Seite 175
- `ULConnection.SetDatabaseOptionInt`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- `ULSetDatabaseOptionULong`-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- `ULDatabaseSchema.SetDatabaseOption`-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

UltraLite-Option `global_database_id`

Legt die Datenbank-Identifizierungsnummer fest

Zulässige Werte

Eindeutige, nicht-negative Ganzzahl

Standardwert

Der Bereich der Standardwerte für eine bestimmte `global autoincrement`-Spalte ist $pn + 1$ bis $p(n + 1)$. Dabei gilt: p ist die Partitionsgröße der Spalte und n ist die globale Datenbank-Identifizierungsnummer.

Bemerkungen

Um die Eindeutigkeit des Primärschlüssels bei der Synchronisation mit einem MobiLink-Server aufrechtzuerhalten, legt die globale ID einen Startwert für `GLOBAL AUTOINCREMENT`-Spalten fest. Die globale ID muss festgelegt werden, bevor Standardwerte zugeordnet werden können. Wenn einer Tabelle eine Zeile hinzugefügt wird und noch kein Wert für sie festgelegt wurde, generiert UltraLite durch die Kombination des Werts von `global_database_id` und der Partitionsgröße für die Spalte einen Wert für die Spalte.

Wenn diese Option eingestellt ist, führt UltraLite ein Festschreiben durch.

Beim Deployment einer Anwendung müssen Sie jeder Datenbank für die Synchronisation mit dem MobiLink-Server eine andere Identifizierungsnummer zuweisen. Sie können die globale ID einer vorhandenen Datenbank jederzeit ändern.

Um die Option `global_database_id` von einer Clientanwendung aus zu setzen, legen Sie sie unter Verwendung der Funktion `SetDatabaseOption` für die von Ihnen verwendete Programmierschnittstelle fest oder verwenden Sie die SQL-Anweisung `SET OPTION`.

Siehe auch

- „GLOBAL AUTOINCREMENT“ [\[MobiLink - Serveradministration\]](#)
- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- „GLOBAL AUTOINCREMENT-Spalten in UltraLite deklarieren“ auf Seite 71
- ULConnection.SetDatabaseOptionInt-Methode [UltraLite C++] [\[UltraLite - C- und C++-Programmierung\]](#)
- ULSetDatabaseID-Methode [UltraLite Embedded SQL] [\[UltraLite - C- und C++-Programmierung\]](#)
- ULConnection.DatabaseID-Eigenschaft [UltraLite.NET] [\[UltraLite - .NET-Programmierung\]](#)
- Connection.setDatabaseId-Methode [UltraLiteJ] [\[UltraLite® – Java-Programmierung\]](#)
- Connection.OPTION_DATABASE_ID-Variable [BlackBerry] [UltraLiteJ] [\[UltraLite® – Java-Programmierung\]](#)

Beispiel

Um die UltraLite-Datenbankspalten mit autoincrement von 3001 auf 4000 zu erhöhen, setzen Sie die globale ID auf 3.

```
SET OPTION global_database_id=3
```

UltraLite-Option isolation_level

Isolationsstufen legen fest, wie weit Vorgänge aus einer Transaktion für Vorgänge anderer paralleler Transaktionen sichtbar sind. UltraLite verwendet die Standardisolationsstufe read_committed bei Verbindungen im Autocommit-Modus.

Siehe auch

- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- „Isolationsstufen“ auf Seite 43
- ULConnection.SetDatabaseOption-Methode [UltraLite C++] [\[UltraLite - C- und C++-Programmierung\]](#)
- ULConnection.BeginTransaction-Methode [UltraLite.NET] [\[UltraLite - .NET-Programmierung\]](#)
- „Datensynchronisation auf einem BlackBerry-Smartphone“ [\[UltraLite® – Java-Programmierung\]](#)

UltraLite ml_remote_id-Option

Die **entfernte ID** ist ein eindeutiger Bezeichner für eine UltraLite-Datenbank und wird von MobiLink verwendet, um die Datenbank für die Synchronisation zu identifizieren. Die entfernte ID kann jede beliebige Zeichenfolge sein, die für Sie aussagekräftig ist, vorausgesetzt, diese Zeichenfolge ist unter allen entfernten MobiLink-Clients eindeutig. Die ID kann auch auf NULL gesetzt sein (NULL ist der anfängliche Wert). Wenn während der Synchronisation die entfernte ID NULL ist, ordnet UltraLite sie einer generierten GUID zu.

Wenn Sie eine UltraLite-Datenbank per Synchronisation für die Verteilung auf mehrere Geräte mit Daten füllen, müssen Sie die entfernte ID vor der Verteilung auf NULL setzen, um sicherzustellen, dass jede Datenbank eine eindeutige entfernte ID hat. Nach der Verteilung kann eine neue eindeutige entfernte ID

explizit festgelegt werden. Sie kann aber auch auf NULL belassen werden, sodass UltraLite automatisch einen neuen eindeutigen Wert generiert.

Zulässige Werte

Ein beliebiger Wert, der die Datenbank für die MobiLink-Synchronisation eindeutig kennzeichnet

Standardwert

Null

Bemerkungen

MobiLink verwendet die entfernte ID zum Speichern der Synchronisationsinformationen für die entfernte Datenbank. Wenn die entfernte ID gegeben ist, müssen MobiLink-Benutzernamen nicht mehr eindeutig sein. Die entfernte ID ist besonders nützlich, wenn mehrere MobiLink-Benutzer dieselbe UltraLite-Datenbank synchronisieren. In diesem Fall sollten die Synchronisationsskripten die entfernte ID referenzieren und nicht nur den Benutzernamen.

Wenn diese Option eingestellt ist, führt UltraLite ein Festschreiben durch.

Um die Option `ml_remote_id` von einer Clientanwendung aus zu setzen, legen Sie sie unter Verwendung der Funktion `SetDatabaseOption` für die von Ihnen verwendete Programmierschnittstelle fest oder verwenden Sie die SQL-Anwendung `SET OPTION`.

Siehe auch

- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- „Entfernte IDs“ [*MobiLink - Clientadministration*]
- „Synchronisationsparameter User Name“ auf Seite 112
- `ULConnection.SetDatabaseOption`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- `ULSetDatabaseOptionString`-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- `ULDatabaseSchema.SetDatabaseOption`-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- `Connection.setOption`-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- `Connection.OPTION_ML_REMOTE_ID`-Variable [BlackBerry] [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

UltraLite-Dienstprogramme

UltraLite umfasst eine Reihe von Dienstprogrammen, mit denen grundlegende Aufgaben der Datenbankverwaltung von der Eingabeaufforderung aus ausgeführt werden können. Viele dieser Dienstprogramme haben eine ähnliche Funktionalität wie die SQL Anywhere-Dienstprogramme. Die Verwendung der Optionen kann jedoch variieren. Genauere Hinweise zu der UltraLite-Implementierung dieser Optionen finden Sie in der UltraLite-Dokumentation.

Hinweis

Die Optionen für die in diesem Abschnitt dokumentierten Dienstprogramme berücksichtigen die Groß- und Kleinschreibung, sofern nicht anders angegeben. Geben Sie daher die Optionen *genau so* ein, wie hier angegeben.

Unterstützte Beendigungscode

Die Dienstprogramme ulload, ulsync und ulunload geben Beendigungscode zurück, die angeben, ob der betreffende Vorgang eines Dienstprogramms erfolgreich abgeschlossen wurde. 0 gibt an, dass der Vorgang erfolgreich war. Jeder andere Wert gibt an, dass der Vorgang fehlgeschlagen ist.

Beendigungscode	Status	Beschreibung
0	EXIT_OKAY	Vorgang erfolgreich
1	EXIT_FAIL	Vorgang fehlgeschlagen
3	EXIT_FILE_ERROR	Datenbank konnte nicht gefunden werden
4	EXIT_OUT_OF_MEMORY	Dynamischer Speicher des Geräts ist erschöpft
6	EXIT_COMMUNICATIONS_FAIL	Kommunikationsfehler während des Datenaustauschs mit der UltraLite-Engine
9	EXIT_UNABLE_TO_CONNECT	Ungültige Benutzer-ID oder ungültiges Kennwort angegeben. Daher kann keine Verbindung mit der Datenbank hergestellt werden.
12	EXIT_BAD_ENCRYPT_KEY	Fehlender oder ungültiger Chiffrierschlüssel
13	EXIT_DB_VER_NEWER	Es wurde erkannt, dass die Datenbankversion nicht kompatibel ist. Die Datenbank muss auf eine neuere Version umgestellt werden.
255	EXIT_USAGE	Ungültige Befehlszeilenoptionen.

Interactive SQL-Dienstprogramm für UltraLite (dbisql)

Führt in einer Datenbank SQL-Anweisungen und Skriptdateien aus.

Syntax

```
dbisql -c "connection-string" [ options ] [ dbisql-statement | dbisql-script-file ]
```

```
dbisql -c "connection-string" -ul [ options ] [ dbisql-statement | dbisql-script-file ]
```

dbisql-statement: Eine SQL-Anweisung oder eine Reihe von durch ein Befehlstrennzeichen getrennten SQL-Anweisungen.

Option	Beschreibung
<i>@data</i>	<p>Liest Optionen aus der angegebenen Umgebungsvariable oder Konfigurationsdatei ein.</p> <p>Wenn die Umgebungsvariable und die Konfigurationsdatei denselben Namen haben, wird die Umgebungsvariable verwendet. Siehe „Konfigurationsdateien“ [SQL Anywhere Server - Datenbankadministration].</p> <p>Wenn Sie Kennwörter oder andere Informationen in der Konfigurationsdatei schützen möchten, können Sie das Dienstprogramm zum Verschleiern von Dateien auf die Konfigurationsdatei anwenden. Siehe „Dienstprogramm zum Verschleiern von Dateien (dbfhide)“ [SQL Anywhere Server - Datenbankadministration].</p>
-c "keyword=value; ..."	<p>Legt Verbindungsparameter fest. Wenn Interactive SQL keine Verbindung herstellen kann, erscheint ein Fenster, in das Sie die Verbindungsparameter eingeben können. Wenn Sie keine Benutzer-ID und kein Kennwort angeben, werden die Standardbenutzer-ID DBA und das Standardkennwort sql verwendet. Siehe „Verbindungsparameter“ [SQL Anywhere Server - Datenbankadministration].</p>
-d <i>delimiter</i>	<p>Gibt ein Befehlstrennzeichen an. Anführungszeichen um das Trennzeichen sind optional und nur dann erforderlich, wenn die Befehlsebene selbst das Trennzeichen auf spezielle Art interpretiert.</p> <p>Diese Option setzt die Einstellung der Option <code>command_delimiter</code> außer Kraft. Siehe „<code>command_delimiter</code>-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbankadministration].</p>

Option	Beschreibung
-d1	Zeigt alle vom Benutzer explizit ausgeführten Anweisungen im Befehlsfenster an (STDOUT). Dadurch erhalten Sie nützliche Informationen bei der Fehlersuche in SQL-Skripten, oder wenn Interactive SQL umfangreiche SQL-Skripten verarbeitet. (Das letzte Zeichen ist die Ziffer 1 und nicht ein "L" in Kleinschreibung.) Diese Option ist nur verfügbar, wenn Sie SQL als Befehlszeilenprogramm ausführen.
-datasource <i>DSN-name</i>	Legt eine ODBC-Datenquelle zum Verbinden fest.
-f <i>filename</i>	<p>Öffnet die Datei <i>Dateiname</i> im Fensterausschnitt "SQL-Anweisungen" (führt sie aber nicht aus).</p> <p>Wenn die Option -f angegeben ist, wird die Option -c ignoriert, d.h. dass keine Verbindung zur Datenbank hergestellt wird.</p> <p>Der Dateiname kann in Anführungszeichen gesetzt sein. Dies <i>muss</i> erfolgen, wenn der Dateiname eine Leerstelle enthält.</p> <p>Wenn die Datei nicht vorhanden oder eigentlich ein Verzeichnis und keine Datei ist, gibt DBISQL eine Fehlermeldung aus und wird beendet.</p> <p>Wenn der Dateiname keine volle Laufwerks- und Pfadangabe enthält, wird angenommen, dass die Datei im aktuellen Verzeichnis liegt.</p> <p>Diese Option wird nur unterstützt, wenn Interactive SQL als Fensteranwendung ausgeführt wird.</p>
-host <i>hostname</i>	Gibt den <i>hostnamen</i> oder die IP-Adresse des Computers an, auf dem der Datenbankserver läuft. Sie können den Namen "localhost" verwenden, um das aktuelle System zu bezeichnen.

Option	Beschreibung
-nogui	<p>Führt Interactive SQL als Konsolenanwendung aus, ohne das Fenster der Benutzeroberfläche zu verwenden. Dies ist für Batch-Vorgänge nützlich.</p> <p>Wenn Sie entweder <i>dbisql-statement</i> oder <i>dbisql-script-file</i> angeben, wird die Option -nogui angenommen.</p> <p>In diesem Modus setzt Interactive SQL den Code zum Beenden des Programms, um Erfolg oder Scheitern zu melden. Bei Windows-Betriebssystemen wird die Umgebungsvariable ERRORLEVEL auf den Code zum Beenden des Programms gesetzt. Siehe „Exit-Codes der Softwarekomponenten“ [SQL Anywhere Server - Programmierung].</p>
-onerror { continue exit }	<p>Steuert, was passiert, wenn während des Lesens von Anweisungen aus einer Skriptdatei ein Fehler auftritt. Dies ist von Vorteil, wenn Interactive SQL in Batch-Vorgängen verwendet wird. Diese Option setzt die Einstellung von on_error außer Kraft. Siehe „on_error-Option [Interactive SQL]“ [SQL Anywhere Server - Datenbank-administration].</p> <p>Definieren Sie einen der folgenden unterstützten Werte für das <i>Verhalten</i>:</p> <ul style="list-style-type: none"> • Continue Der Fehler wird ignoriert und Interactive SQL fährt mit der Ausführung der Anweisungen fort. • Exit Interactive SQL wird beendet.
-q	<p>Unterdrückt die Ausgabe von Meldungen. Führt das Dienstprogramm im stillen Modus aus. Dies ist nur dann nützlich, wenn Sie Interactive SQL mit einer Anweisung oder einer Skriptdatei starten. Das Angeben dieser Option unterdrückt keine Fehlermeldungen, sondern Folgendes:</p> <ul style="list-style-type: none"> • Warnungen und andere nicht-schwerwiegende Meldungen • Die Ausgabe von Ergebnismengen

Option	Beschreibung
-ul	<p>Gibt an, dass UltraLite-Datenbanken Standardvorgabe sind. Interactive SQL passt die Ihnen verfügbaren Optionen abhängig vom Typ der Datenbank an, mit der Sie verbunden sind.</p> <p>Standardmäßig nimmt Interactive SQL an, dass Sie mit SQL Anywhere-Datenbanken verbunden sind. Wenn Sie die Option -ul angeben, ändert sich der Standard zu UltraLite-Datenbanken. Unabhängig von dem als Standard festgelegten Datenbanktyp können Sie eine Verbindung zu einer SQL Anywhere- oder zu einer UltraLite-Datenbank herstellen, indem Sie den Datenbanktyp aus der Dropdown-Liste Datenbanktyp ändern im Fenster Verbinden auswählen.</p> <p>Weitere Hinweise zum Herstellen einer Verbindung mit UltraLite-Datenbanken von Interactive SQL aus finden Sie unter „Interactive SQL-Dienstprogramm für UltraLite (dbisql)“ auf Seite 201.</p>
-version	<p>Zeigt die Versionsnummer von Interactive SQL an. Sie können die Versionsnummer auch in Interactive SQL anzeigen, indem Sie im Menü Hilfe auf Info über Interactive SQL klicken.</p>
-x	<p>Durchsucht Anweisungen, führt sie aber nicht aus. Dies ist nützlich, um lange Skriptdateien auf Syntaxfehler zu überprüfen.</p> <p>Ausführliche Beschreibungen von SQL-Anweisungen und Anweisungen in Interactive SQL finden Sie unter „SQL-Sprachelemente“ [SQL Anywhere Server - SQL-Referenzhandbuch].</p>
<i>dbisql-statement</i> <i>dbisql-script-file</i>	<p>Führt die SQL-Anweisung bzw. die angegebene <i>dbisql-Skriptdatei</i> aus.</p> <p>Wenn Sie weder <i>dbisql-statement</i> noch <i>dbisql-script-file</i> angeben, wechselt Interactive SQL in den interaktiven Modus, in dem Sie eine Anweisung in ein Befehlsfenster eingeben können.</p>

Bemerkungen

Mit Interactive SQL können Sie die Datenbank durchsuchen sowie SQL-Anweisungen und Skriptdateien ausführen. Sie erhalten auch Rückmeldungen zu folgenden Elementen:

- Anzahl der betroffenen Zeilen
- Für jede Anweisung benötigte Zeit
- Ausführungsplan von Abfragen
- Etwaige Fehlermeldungen

Sie können sich nicht sowohl mit SQL Anywhere- als auch mit UltraLite-Datenbanken verbinden. Weitere Hinweise zum Herstellen einer Verbindung zu SQL Anywhere-Datenbanken finden Sie unter „Interactive SQL-Dienstprogramm (dbisql)“ [[SQL Anywhere Server - Datenbankadministration](#)].

Interactive SQL wird unter Windows, Solaris, Linux und Mac OS X unterstützt. Weitere Hinweise finden Sie unter <http://www.sybase.com/detail?id=1061806>.

Für Windows gibt es zwei Programmdateien:

1. Batch-Skripten sollten *dbisql* oder *dbisql.com* anstelle von *dbisql.exe* aufrufen. Die Programmdatei *dbisql.com* ist als Konsolenanwendung gelinkt.
2. Die Programmdatei *dbisql.exe* ist als Fensteranwendung gelinkt und blockiert die Befehls-Shell, aus der sie gestartet wurde, nicht. Wenn *dbisql.exe* aus einer Batchdatei ausgeführt wird, sehen Sie keine Ausgabedaten in der Standardausgabe oder in den Standard-Fehlerdateien.

Die Standardkodierung für Interactive SQL kann auch temporär mit der Option `default_isql_encoding` gesetzt werden. Siehe „`default_isql_encoding`-Option [Interactive SQL]“ [[SQL Anywhere Server - Datenbankadministration](#)].

Sie können die Kodierung angeben, die zum Lesen und Schreiben von Dateien verwendet werden soll, indem Sie die ENCODING-Klausel der INPUT-, OUTPUT- oder READ-Anweisung verwenden. Siehe:

- „INPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „OUTPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „READ-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Beendigungscode sind 0 (Erfolg) oder eine von 0 verschiedene Zahl (Fehlschlag). Beendigungscode ungleich Null werden nur verwendet, wenn Interactive SQL im Batchmodus verwendet wird (mit einer Befehlszeile, die eine SQL-Anweisung oder den Namen einer Skriptdatei verwendet). Siehe „Exit-Codes der Softwarekomponenten“ [[SQL Anywhere Server - Programmierung](#)].

Im Befehlseingabemodus setzt Interactive SQL den Code zum Beenden des Programms, um Erfolg oder Scheitern zu melden. Bei Windows-Betriebssystemen wird die Umgebungsvariable ERRORLEVEL auf den Code zum Beenden des Programms gesetzt.

Wenn Sie eine *reload.sql*-Datei mit Interactive SQL ausführen, müssen Sie den Chiffrierschlüssel als Parameter angeben. Wenn Sie den Schlüssel nicht in der READ-Anweisung übergeben, werden Sie von Interactive SQL aufgefordert, den Schlüssel bereitzustellen.

Sie können Interactive SQL auf folgende Arten starten:

- Von Sybase Central aus, indem Sie auf **Datei » Interactive SQL öffnen** klicken.
- Vom Menü **Start** aus, indem Sie auf **Start » Programme » SQL Anywhere 16 » Administrationstools » Interactive SQL** klicken.
- Indem Sie den dbisql-Befehl an einer Eingabeaufforderung ausführen.

Siehe auch

- „SQL-Anweisungen für Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Interactive SQL“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Konfigurationsdateien“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Dienstprogramm zum Verschleiern von Dateien (dbfhide)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „INPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „OUTPUT-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „READ-Anweisung [Interactive SQL]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „UltraLite-Verbindungsparameter“ auf Seite 168
- „Unterstützte BeendigungsCodes“ auf Seite 201

Beispiel

Mit dem folgenden Befehl wird die Skriptdatei *mycom.sql* in der Datenbank *CustDB.udb* für UltraLite ausgeführt. Da keine Benutzer-ID und kein Kennwort festgelegt sind, werden die Standard-Benutzer-ID **DBA** und das Kennwort **sql** verwendet. Die Option `-onerror` ist als "Exit" festgelegt. Wenn es also einen Fehler in der Skriptdatei gibt, wird der Prozess beendet.

```
dbisql -ul -c DBF=CustDB.udb -onerror exit mycom.sql
```

UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp)

Erledigt die Vorverarbeitung eines C-/C++-Programms, das Embedded SQL (ESQL) enthält, sodass der für dieses Programm erforderliche Code vor dem Aufruf des Compilers generiert werden kann. Die untenstehende Tabelle enthält aus Vollständigkeitsgründen die gesamte Menge der Befehlszeilenoptionen, die für UltraLite einzig relevanten Optionen sind indessen **-eu** und **-wu**.

Syntax

```
sqlpp -u [ options ] esql-filename [ output-filename ]
```

Option	Beschreibung
-d	Generiert Code, der die Datenspeichergröße verringert, aber die Codegröße erhöht. Datenstrukturen werden vor der Verwendung zur Ausführungszeit wieder benutzt und initialisiert.

Option	Beschreibung
-e Stufe	<p>Diese Option kennzeichnet jedes Static Embedded SQL-Element als Fehler, das nicht Teil eines angegebenen Standards ist. Der Wert <i>Stufe</i> gibt den zu verwendenden Standard an. Beispiel: <code>sqlpp -e c03 . . .</code> kennzeichnet alle Syntaxelemente, die nicht Teil des SQL/2003-Kernstandards sind.</p> <p>Die folgenden Werte sind für <i>Stufe</i> zulässig:</p> <ul style="list-style-type: none"> • c03 Kennzeichnet Syntax, die nicht SQL/2003-Kernsyntax ist • p03 Kennzeichnet Syntax, die nicht Full-SQL/2003-Syntax ist • c99 Kennzeichnet Syntax, die nicht SQL/1999-Kernsyntax ist • p99 Kennzeichnet Syntax, die nicht Full-SQL/1999-Syntax ist • e92 Kennzeichnet Syntax, die nicht Entry-Level der SQL/1992 Syntax ist • i92 Kennzeichnet Syntax, die nicht Intermediate-Level der SQL/1992 Syntax ist • f92 Kennzeichnet Syntax, die nicht Full-SQL/1992 Syntax ist • t Kennzeichnet Nicht-Standard-Hostvariablentypen • u Kennzeichnet Syntax, die nicht von UltraLite unterstützt wird <p>Für die Kompatibilität mit früheren Versionen von SQL Anywhere können Sie auch e, i und f angeben, die e92, i92 bzw. f92 entsprechen.</p>
-h width	Begrenzt die maximale Länge von getrennten Ausgabezeilen durch <code>sqlpp</code> in der <code>.c</code> -Datei auf <i>width</i> . Am Ende von getrennten Zeilen werden Backslashes hinzugefügt, sodass ein C-Compiler die getrennten Zeilen als eine durchgängige Zeile analysieren kann. Der Standardwert ist keine maximale Zeilenlänge (Ausgabezeilen werden standardmäßig nicht unterteilt).
-k	Teilt dem Präprozessor mit, dass das zu kompilierende Programm eine Benutzerdeklaration von <code>SQLCODE</code> enthält
-m mode	Cursor-Aktualisierbarkeitsmodus. Entweder HISTORICAL oder READONLY .
-n	<p>Generiert Zeilennummerinformationen in der C-Datei, wobei <code>#line</code>-Direktiven an den betreffenden Positionen im generierten Code verwendet werden.</p> <p>Verwenden Sie diese Option, um die Quelle in den entsprechenden Zeilenzahlen in der Datei <i>esql-filename</i> und nicht in der Datei <i>output-filename</i> zu untersuchen und Quellfehler dort zu melden.</p>
-o O/S spec	Für UltraLite nicht anwendbar.

Option	Beschreibung
-q	Führt das Dienstprogramm im stillen Modus aus. Informative Banner, Versionsnummern und Statusmeldungen werden unterdrückt. Fehlermeldungen werden jedoch weiter angezeigt.
-r-	Für UltraLite nicht anwendbar.
-s string-length	Setzt die maximale Größe für Zeichenfolgen fest, die der Präprozessor in die C-Datei ausgibt. Zeichenfolgen, die länger als dieser Wert sind, werden mithilfe einer Liste von Zeichen initialisiert ('a','b','c' etc.). Die meisten C-Compiler sind in der Größe der Zeichenfolgenlitterale begrenzt, die Sie handhaben können. Mit dieser Option wird die obere Grenze festgesetzt. Der Standardwert ist "500".
-u	Für UltraLite erforderlich. Generiert eine Ausgabe, die speziell für UltraLite-Datenbanken erforderlich ist.
-w level	<p>Nicht-konforme SQL-Syntax als Warnung kennzeichnen. Der Wert <i>level</i> gibt den zu verwendenden Standard an. Beispiel: <code>sqlpp -w c03 . . .</code> kennzeichnet alle SQL-Syntaxelemente, die nicht Teil der SQL/2003-Kernsyntax sind.</p> <p>Die folgenden Werte sind für <i>level</i> zulässig:</p> <ul style="list-style-type: none"> • c03 Kennzeichnet Syntax, die nicht SQL/2003-Kernsyntax ist • p03 Kennzeichnet Syntax, die nicht Full-SQL/2003-Syntax ist • c99 Kennzeichnet Syntax, die nicht SQL/1999-Kernsyntax ist • p99 Kennzeichnet Syntax, die nicht Full-SQL/1999-Syntax ist • e92 Kennzeichnet Syntax, die nicht Entry-Level der SQL/1992 Syntax ist • i92 Kennzeichnet Syntax, die nicht Intermediate-Level der SQL/1992 Syntax ist • f92 Kennzeichnet Syntax, die nicht Full-SQL/1992 Syntax ist • t Kennzeichnet Nicht-Standard-Hostvariablentypen • u Kennzeichnet Syntax, die nicht von UltraLite unterstützt wird <p>Für die Kompatibilität mit früheren Versionen von SQL Anywhere können Sie auch e, i und f angeben, die e92, i92 bzw. f92 entsprechen.</p>
-x	Ändert Mehrbyte-Zeichenfolgen in Escape-Sequenzen, sodass sie vom Compiler verarbeitet werden können.
-z collation-sequence	Geben Sie die Kollationssequenz an.

Bemerkungen

Dieser Präprozessor konvertiert die SQL-Anweisungen in der Eingabedatei in C/C++. Er schreibt das Ergebnis in die *output-filename*. Die normale Dateinamenerweiterung für Quelldateien in Embedded SQL ist *sql*. Der Standard-*output-filename* ist der Basisname von *esql-filename* mit der Erweiterung *c*. Wenn der *esql-filename* jedoch bereits die Erweiterung *.c* hat, ist die Standarderweiterung für die Ausgabe *.cc*.

Die Kollationssequenz hilft dem Präprozessor, die Zeichen zu verstehen, die im Quellcode des Programms verwendet werden, z.B. bei der Identifizierung von alphabetischen Zeichen, die für die Verwendung in Bezeichnern geeignet sind. In UltraLite umfassen Kollationen eine Codepage sowie eine Sortierreihenfolge. Wenn Sie **-z** nicht angeben, versucht der Präprozessor, anhand des Betriebssystems eine geeignete Kollation zu ermitteln.

Sie können eine Liste der unterstützten Kollationen (und der entsprechenden Codepage) anzeigen, indem Sie den Befehl **ulinit -Z** ausführen.

Tipp

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. Sie können die Kompatibilität von SQL sowohl für SQL Anywhere als auch für UltraLite-Anwendungen mithilfe der Optionen **-e** bzw. **-w** testen. Einen Überblick über den SQL Flagger finden Sie unter „[Testen der SQL-Konformität mit dem SQL Flagger](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Siehe auch

- „Embedded SQL“ [*SQL Anywhere Server - Programmierung*]
- „UltraLite-Zeichensätze“ auf Seite 26

Beispiel

Der folgende Befehl bearbeitet die Embedded SQL-Datei *srcfile.sql* im dialogfreien Betrieb für eine UltraLite-Anwendung vor.

```
sqlpp -u -q MyEsqFile.sql
```

UltraLite-Engine-Dienstprogramm (uleng16)

Verwaltet gleichzeitige UltraLite-Datenbankverbindungen von Anwendungen und gestattet es der UltraLite-Engine, als Daemon ausgeführt werden, indem Sie die **-ud**-Option verwenden.

Syntax

```
uleng16 [ -ud ] [ db-file-name ]
```

Option	Beschreibung
-ud <i>db-file-name</i>	Damit können Sie die Engine so ausführen, dass sie weiter läuft, nachdem die aktuelle Benutzersitzung beendet wurde. Wenn Sie den Daemon direkt mit der Option -ud starten, erstellt der uleng16-Befehl den Daemonprozess und kehrt sofort zurück (er wird beendet und der nächste Befehl kann ausgeführt werden), bevor der Daemon sich selbst initialisiert oder versucht, eine der im Befehl angegebenen Datenbanken zu öffnen.

Bemerkungen

Die UltraLite-Engine zeigt beim Start kein Meldungsfenster an.

Die UltraLite-Engine sollte von einer Anwendung in Szenarien verwendet werden, in denen mehrere Prozesse auf dieselbe Datenbank zur gleichen Zeit zugreifen können. Die Engine befindet sich im SQL Anywhere *bin32*- oder *bin64*-Verzeichnis, weil die UltraLite-Desktop-Administrationstools die Engine zur Verbindung mit Datenbanken verwenden.

Mit der Option -ud können Sie die UltraLite-Engine so ausführen, dass die Datenbank-Engine weiterläuft, wenn Sie sich abmelden. (Normalerweise gilt: Wenn Sie sich abmelden, werden alle in dieser Sitzung geöffneten Anwendungen heruntergefahren.)

Siehe auch

- „UltraLite-Deployment“ auf Seite 117
- „So erstellen und Sie UltraLite C++-Anwendungen und führen ein Deployment durch.“ [*UltraLite - C- und C++-Programmierung*]
- „UltraLite-Datenverwaltungskomponenten für Windows Mobile“ auf Seite 19
- „UltraLite-Dienstprogramm zum Stoppen der Engine (ulstop)“ auf Seite 211
- „UltraLite-Verbindungsparameter START“ auf Seite 189

UltraLite-Dienstprogramm zum Stoppen der Engine (ulstop)

Stoppt die UltraLite-Engine.

Syntax

ulstop

Bemerkungen

Verwenden Sie ulstop bei der Entwicklung, um die Engine manuell zu stoppen. In Live-Deployments wird ulstop gewöhnlich nicht benötigt.

Siehe auch

- „UltraLite-Datenverwaltungskomponenten für Windows Mobile“ auf Seite 19
- „UltraLite-Engine-Dienstprogramm (uleng16)“ auf Seite 210

UltraLite-Dienstprogramm zum Löschen von Datenbanken (ulerase)

Löscht eine UltraLite-Datenbank

Syntax

ulerase [*options*] [*db-file-name*]

Option	Beschreibung
<i>@data</i>	Liest Optionen aus der angegebenen Umgebungsvariablen oder Konfigurationsdatei ein. Wenn beide mit demselben Namen vorhanden sind, wird die Umgebungsvariable verwendet.
-k <i>key</i> ODER --ek= <i>key</i>	Gibt den Chiffrierschlüssel für eine verschlüsselte Datenbank an.
-p ODER --ep	Gibt an, dass Sie zur Eingabe des Chiffrierschlüssel aufgefordert werden wollen.
--log	Protokolliert Vorgänge in der festgelegten Datei.
-q ODER --quiet	Führt das Dienstprogramm im stillen Modus aus. Informative Banner, Versionsnummern und Statusmeldungen werden unterdrückt. Fehlermeldungen werden jedoch weiter angezeigt.
-u <i>uid,pwd</i> ODER --dba= <i>uid,pwd</i>	Gibt die Benutzer-ID und das Kennwort für den Zugriff auf die Datenbank an.
-? ODER --help	Zeigt Informationen über die Verwendung des Dienstprogramms an und beendet das Programm.
<i>db-file-name</i>	Löscht die angegebene Datenbank.

Bemerkungen

Die Datenbank muss zugänglich sein. Die Kombination aus Benutzer-ID und Kennwort muss eine Verbindung zulassen, sonst wird die Datenbank nicht gelöscht.

Verschlüsselte Datenbanken erfordern einen in der Verbindungszeichenfolge übergebenen Schlüssel oder die Verwendung von **-k** *key* oder **-p**.

Siehe auch

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

UltraLite-Informationsdienstprogramm (ulinfo)

Zeigt Informationen über eine UltraLite-Datenbank an.

Syntax

ulinfo -c *options*

Option	Beschreibung
@ <i>data</i>	Liest Optionen aus der angegebenen Umgebungsvariablen oder Konfigurationsdatei ein. Wenn beide mit demselben Namen vorhanden sind, wird die Umgebungsvariable verwendet.
-c " <i>connection-string</i> " ODER --connect= " <i>connection-string</i> "	Datenbank-Verbindungsparameter angeben (erforderlich).
-q ODER --quiet	Führt das Dienstprogramm im stillen Modus aus. Informative Banner, Versionsnummern und Statusmeldungen werden unterdrückt. Fehlermeldungen werden jedoch weiter angezeigt.
--log= <i>filename</i>	Protokolliert Vorgänge in der festgelegten Datei.
-? ODER --help	Zeigt Informationen über die Verwendung des Dienstprogramms an und beendet das Programm.

Bemerkungen

Warnungen, die beim Öffnen einer UltraLite-Datenbank generiert werden, werden immer angezeigt, außer Sie verwenden die Option **-q**.

Siehe auch

- „Konfigurationsdateien“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „UltraLite-Verbindungsparameter“ auf Seite 168
- „UltraLite-Option global_database_id“ auf Seite 198
- „UltraLite ml_remote_id-Option“ auf Seite 199

Beispiel

So werden grundlegende interne Datenbankvorgänge für eine Datei namens *cv_dbattr.udb* angezeigt, die bereits synchronisiert wurde:

```
ulinfo -c DBF=cv_dbattr.udb
```

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit)

Erstellt eine neue UltraLite-Datenbank.

Dieses Dienstprogramm funktioniert in einem der folgenden Modi:

- **Leermodus** Erstellt eine leere Datenbank mit den Merkmalen, die in den Befehlszeilenargumenten angegeben werden.
- **Extraktionsmodus** Erstellt eine neue Datenbank, basierend auf einer SQL Anywhere-Datenbank.

Ein Anfangsschema wird erstellt, das Tabellen und Indizes in der SQL Anywhere-Referenzdatenbank entspricht. Viele der Referenzdatenbank-Eigenschaften werden extrahiert und in der neuen UltraLite-Datenbank benutzt.

Syntax

```
ulinit options dbname
```

Option	Beschreibung
@data	Liest Optionen aus der angegebenen Umgebungsvariablen oder Konfigurationsdatei ein. Wenn beide mit demselben Namen vorhanden sind, wird die Umgebungsvariable verwendet. Siehe „Konfigurationsdateien“ [SQL Anywhere Server - Datenbankadministration].

Option	Beschreibung
-a "keyword=value;..." ODER --SAconnect ="keyword=value;..."	<p>Setzt das Dienstprogramm in den Extraktionsmodus und stellt eine Verbindung zu einer vorhandenen Datenbank unter Verwendung der angegebenen Verbindungsparameter her.</p> <p>Wenn diese Option nicht vorhanden ist, erstellt das Dienstprogramm eine neue Datenbank unter Verwendung der angegebenen Verbindungsparameter Leermodus).</p>
-c ODER --case	<p>Leermodus.</p> <p>Berücksichtigung von Groß- und Kleinschreibung bei allen Zeichenfolgenvergleichen.</p>
-d ODER --datacopy	<p>Extraktionsmodus.</p> <p>Kopiert für jede Tabelle in der neuen UltraLite-Datenbank die Daten aus der entsprechenden Tabelle in der SQL Anywhere-Datenbank. Die neue Datenbank ist anfangs leer, es sei denn, Sie verwenden diese Option.</p> <p>Standardmäßig werden diese Daten in den nachfolgenden Synchronisationen nicht hochgeladen. Um die Daten in die nächste Upload-Synchronisation einzubeziehen, verwenden Sie -i mit -d.</p>
--date_format =format	<p>Leermodus.</p> <p>Legt das Format für Datumsangaben fest, die aus der Datenbank abgerufen werden. Siehe „UltraLite-Erstellungsparameter date_format“ auf Seite 146.</p>
--date_order =date-format-interpretation	<p>Leermodus.</p> <p>Legt die Interpretation des Datumsformats fest. Siehe „UltraLite-Erstellungsparameter date_order“ auf Seite 149.</p>

Option	Beschreibung
-e value ODER --fips=value	Leermodus. On oder Off, 1 oder 0, usw. Diese Option steuert die AES FIPS-zertifizierte Verschlüsselung mithilfe eines Certicom-zertifizierten Verschlüsselungsalgorithmus. Siehe „ Datenbank-sicherheit “ auf Seite 29 und „ UltraLite-Erstellungsparameter fips “ auf Seite 150.
-f ODER --exactschema	Extraktionsmodus. Schlägt fehl, wenn das exakte Schema nicht in UltraLite unterstützt wird. Andernfalls werden Warnmeldungen angezeigt, wenn es Unterschiede beim Schema gibt.
-g id ODER --databaseid=id	Setzt die Ausgangsdatenbank-ID auf den angegebenen INTEGER-Wert. Dieser Ausgangswert wird mit einer Partitionsgröße für neue Zeilen verwendet, die global autoincrement-Spalten enthalten. Beim Deployment einer Anwendung müssen Sie jeder Datenbank für die Synchronisation mit dem MobiLink-Server einen anderen Bereich von Identifizierungsnummern zuweisen. Siehe „ UltraLite-Option global_database_id “ auf Seite 198.
-i ODER --insertforupload	Extraktionsmodus. Verwendung mit -d . Eingefügte Zeilen in nächste Upload-Synchronisation einbeziehen. Standardmäßig werden die von diesem Dienstprogramm eingefügten Zeilen während der Synchronisation nicht gesendet.
--identity-file=file	Gibt die Datei an, die die Client-TLS-Identität enthält. Siehe „ identity “ [MobiLink - Clientadministration].
--identity-password=password	Legt das Kennwort für die Client-TLS-Identität fest. Siehe „ identity_password “ [MobiLink - Clientadministration].

Option	Beschreibung
-k <i>key</i> ODER --key= <i>key</i>	Extraktionsmodus. Gibt den Chiffrierschlüssel für eine verschlüsselte Datenbank an.
-K ODER --prompt	Leermodus. Gibt an, dass Sie zur Eingabe des Chiffrierschlüssels aufgefordert werden wollen.
-l <i>filename</i> ODER --sql= <i>filename</i>	Extraktionsmodus. Protokolliert SQL-Anweisungen zur DDL-Datenbankschemaerstellung nach der Ausführung in der angegebenen Datei.
--log= <i>filename</i>	Leermodus. Protokolliert Vorgänge in der festgelegten Datei.
-m <i>filename</i> ODER --mirror_file= <i>filename</i>	Extraktionsmodus. Geben Sie die Datenbankspiegeldatei an. Siehe „UltraLite-Verbindungsparameter MIRROR_FILE“ auf Seite 184.
--max_hash_size= <i>size</i>	Leermodus. Legt die Standardgröße für den Index-Hash in Byte fest. Siehe „UltraLite-Erstellungsparameter max_hash_size“ auf Seite 151.

Option	Beschreibung
-n <i>pubname</i> ODER --publication= <i>pubname</i>	<p>Extraktionsmodus.</p> <p>Erforderlich. Fügt Tabellen zum UltraLite-Datenbankschema hinzu.</p> <p><i>pubname</i> definiert eine Publikation in der Referenzdatenbank. Die Tabellen in der Publikation werden zur UltraLite-Datenbank hinzugefügt.</p> <p>Geben Sie die Option mehrmals an, wenn der UltraLite-Datenbank mehrere Publikationen hinzugefügt werden sollen. Um alle Tabellen in der Referenzdatenbank zur UltraLite-Datenbank hinzuzufügen, geben Sie -n* an.</p>
--nearest_century= <i>yy</i>	<p>Leermodus.</p> <p>Steuert die Interpretation von zweistelligen Jahresangaben bei Konvertierungen von Zeichenfolgen in Datumsangaben. Siehe „UltraLite-Erstellungsparameter nearest_century“ auf Seite 153.</p>
-o <i>value</i> ODER --obfuscate= <i>value</i>	<p>Leermodus.</p> <p>On oder Off, 1 oder 0, usw. Steuert, ob Daten in der Datenbank verschleiert werden. Die Verschleierung ist eine vereinfachte Form der Verschlüsselung. Siehe „Datenbanksicherheit“ auf Seite 29 und „UltraLite-Erstellungsparameter obfuscate“ auf Seite 154.</p>
-p <i>size</i> ODER --page_size= <i>size</i>	<p>Leermodus.</p> <p>Geben Sie die Seitengröße der Datenbank an.</p>
--precision= <i>precision</i>	<p>Leermodus.</p> <p>Legt die maximale Anzahl von Stellen im Ergebnis aller Berechnungen mit Dezimaltrennzeichen fest. Siehe „UltraLite-Erstellungsparameter precision“ auf Seite 157.</p>

Option	Beschreibung
-q ODER --quiet	Führt das Dienstprogramm im stillen Modus aus. Informative Banner, Versionsnummern und Statusmeldungen werden unterdrückt. Fehlermeldungen und Warnungen werden jedoch weiter angezeigt.
-r size ODER --reserve_size	Nur Datenbankverbindung. Reservierungsgröße. Siehe „ UltraLite-Verbindungsparameter RESERVE_SIZE “ auf Seite 187.
-s pubname ODER --sync_publication	Extraktionsmodus. Erstellt eine Publikation in der UltraLite-Datenbank mit derselben Definition wie <i>pubname</i> in der Referenzdatenbank. Publikationen werden verwendet, um die Synchronisation zu konfigurieren. Wenn Sie mehrere Synchronisationspublikationen verwenden wollen, geben Sie die Option -s mehrfach an. Die Tabellen in dieser Publikation müssen in einer Publikation enthalten sein, die durch die Option -n aufgelistet ist. Wird die Option -s nicht angegeben, hat die entfernte UltraLite-Datenbank keine benannten Publikationen. Weitere Informationen dazu, wie Sie Publikationen für die MobiLink-Synchronisation erstellen, finden Sie unter „ Publizieren von Daten in UltraLite “ auf Seite 78.
-S checksum_level ODER --checksum_level=checksum_level	Leermodus. 0, 1 oder 2. Gibt die Ebene der Prüfsummenvalidierung auf Datenbankseiten an. Siehe „ UltraLite-Erstellungsparameter checksum_level “ auf Seite 144.

Option	Beschreibung
--scale=scale	<p>Leermodus.</p> <p>Legt die Mindestanzahl der Stellen nach dem Dezimalzeichen fest, wenn ein arithmetisches Ergebnis auf die maximale Gesamtstellenanzahl gekürzt wird. Siehe „UltraLite-Erstellungsparameter scale“ auf Seite 158.</p>
-t file ODER --rootcert=file	<p>Gibt die Datei an, die das vertrauenswürdige Stammzertifikat enthält. Dieses Zertifikat ist für die Serverauthentifizierung erforderlich.</p>
--time_format=format	<p>Leermodus.</p> <p>Setzt das Format für Zeitwerte, die aus der Datenbank abgerufen werden. Siehe „UltraLite-Erstellungsparameter time_format“ auf Seite 160.</p>
--timestamp_format=format	<p>Leermodus.</p> <p>Legt das Format für Zeitstempelwerte fest, die aus der Datenbank abgerufen werden. Siehe „UltraLite-Erstellungsparameter timestamp_format“ auf Seite 162.</p>
--timestamp_increment=increment	<p>Leermodus.</p> <p>Legt fest, wie der Zeitstempel in UltraLite gekürzt wird. Siehe „UltraLite-Erstellungsparameter timestamp_increment“ auf Seite 164.</p>
--timestamp_with_time_zone_format=format	<p>Leermodus.</p> <p>Diese Option stellt das Format für von der Datenbank abgerufene TIMESTAMP WITH TIME ZONE-Werte ein. Siehe „UltraLite-Erstellungsparameter timestamp_with_time_zone_format“ auf Seite 166.</p>
-u <uid>,<pwd> ODER --dba=<uid>,<pwd>	<p>Nur Datenbankverbindung.</p> <p>Geben Sie die Benutzer-ID und das Kennwort an.</p>

Option	Beschreibung
--utf8_encoding=value	Leermodus. On oder Off, 1 oder 0, usw. Kodiert Daten im UTF-8-Format, der 8-Bit-Mehrbyte-Kodierung für Unicode. Siehe „ UltraLite-Zeichensätze “ auf Seite 26 und „ UltraLite-Erstellungsparameter utf8_encoding “ auf Seite 167.
-w ODER --nowarnings	Extraktionsmodus. Zeigt Warnungen nicht an.
-x table ODER --exclude	Extraktionsmodus. Schließt in der Liste aufgeführte Tabellen aus.
-y ODER --overwrite	Überschreibt die vorhandene Datenbankdatei.
-z collation-sequence ODER --collation=collation-sequence	Leermodus. Geben Sie die Kollationssequenz an.
-Z ODER --listcollation	Leermodus. Listen Sie die verfügbaren Kollationssequenzen auf und beenden Sie die Anwendung.
-? ODER --help	Zeigt die Verwendung des Dienstprogramms an und beendet die Anwendung.

Hinweis

Eine Option kann in einem der beiden Modi verwendet werden, falls nicht eine in der Beschreibung erwähnt ist.

Bemerkungen

Bei Ausführung im Extraktionsmodus versucht ULINIT, eine UltraLite-Datenbank zu erstellen, die so weit wie möglich mit der SQL Anywhere-Datenbank übereinstimmt. Wenn beispielsweise eine Spalte in der SQL Anywhere-Datenbank über eine Klausel verfügt, die UltraLite nicht unterstützt, wird der Standardwert ignoriert und stattdessen wird der UltraLite-Standardwert verwendet. Eine Warnung wird generiert und die Erstellung wird fortgesetzt. Dies ist sinnvoll, wenn SQL Anywhere-Tabellen nicht geändert werden können, aber eine angemessene UltraLite-Alternative zur Verfügung steht. Um eine genaue Schemaentsprechung zu erzwingen, verwenden Sie die **-f**-Option. Das Dienstprogramm ulinit schlägt fehl, wenn das Schema keine geeignete UltraLite-Alternative unterstützt.

Siehe auch

- „Aus einer SQL Anywhere-Datenbank in eine UltraLite-Datenbank konvertieren“ auf Seite 33
- „Synchronisationsmodelle“ [*MobiLink - Erste Orientierung*]
- „UltraLite-Verbindungsparameter“ auf Seite 168

Beispiele

Erstellen Sie eine Datei mit dem Namen *customer.udb*, die die Tabellen aus *TestPublication* enthält:

```
ulinit -a "DSN=MySADB;UID=JimmyB;PWD=secret" -n TestPublication -k mykey  
customer.udb
```

In diesem Beispiel wird eine Verbindung zu einer SQL Anywhere-Datenbank hergestellt, die in der *MySADB*-Datenquelle angegeben ist. Es wird eine UltraLite-Datenbank mit allen Optionen dieser Datenbank und allen Tabellen in der Publikation *TestPublication* erstellt. Die neue UltraLite-Datenbank wird *customer.udb* genannt und mit dem Schlüssel *mykey* verschlüsselt.

Erstellen Sie eine Datei namens *customer.udb*, die zwei unterschiedliche Publikationen enthält. *Pub1* könnte eine kleine Teilmenge von Daten für eine vorrangige Synchronisation enthalten, während *Pub2* den Großteil der Daten enthalten könnte:

```
ulinit -a "DSN=MySADB;UID=JimmyB;PWD=secret" --exactschema -n Pub1 -n Pub2 -  
s Pub1 -s Pub2 customer.udb
```

In diesem Beispiel wird eine Verbindung zu einer SQL Anywhere-Datenbank hergestellt, die in der *MySADB*-Datenquelle angegeben ist. Es wird eine UltraLite-Datenbank mit allen Optionen dieser Datenbank und allen Tabellen in den Publikationen *Pub1* und *Pub2* erstellt. Die neue UltraLite-Datenbank wird ebenfalls mit den Publikationen *Pub1* und *Pub2* erstellt. Da die Option **--exactschema** eingestellt ist, schlägt ulinit fehl, wenn das genaue Schema nicht extrahiert werden kann.

Erstellen Sie eine neue, leere Datenbank, die eine andere *customer.udb*-Datei überschreibt, wenn vorhanden. Die neue Datenbank hat kein Schema und alle Datenbankoptionen sind auf Standardwerte gesetzt.

```
ulinit -y customer.udb
```

UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)

Lädt Daten von einer XML-Datei in eine neue oder vorhandene Datenbank.

Syntax

ulload **-c** "connection-string" [options] xml-file

Option	Beschreibung
@data	Liest Optionen aus der angegebenen Umgebungsvariablen oder Konfigurationsdatei ein. Wenn beide mit demselben Namen vorhanden sind, wird die Umgebungsvariable verwendet. Siehe „ Konfigurationsdateien “ [SQL Anywhere Server - Datenbankadministration].
-a ODER --append	Fügt Daten und Schemadefinitionen in eine vorhandene Datenbank ein.
-c "connection-string" ODER --connect= "connection-string"	Gibt die Parameter für die Datenbankverbindung an.
-d ODER --dataonly	Lädt nur Daten. Alle Schema-Metadaten in der XML-Dateieingabe werden ignoriert. Die Parameter -d oder --dataonly können nur verwendet werden, wenn -a angegeben ist (weil nur Daten geladen werden, muss die UDB zum Laden der Daten vorhanden sein und ein Schema aufweisen, das die zu ladenden Daten unterstützt).
-e value ODER --fips= value	Geben Sie on oder off, 1 oder 0 usw. an. Diese Option steuert die AES FIPS-zertifizierte Verschlüsselung mithilfe eines Certicom-zertifizierten Verschlüsselungsalgorithmus. Siehe „ Datenbanksicherheit “ auf Seite 29 und „ UltraLite-Erstellungsparameter fips “ auf Seite 150.

Option	Beschreibung
-E behavior ODER --onerror=behavior	Steuert, was passiert, wenn ein Fehler während des Lesens von Daten aus der XML-Datei auftritt. Geben Sie einen der folgenden unterstützten Werte für das <i>behavior</i> an: <ul style="list-style-type: none"> • continue ulload ignoriert den Fehler und fährt fort, XML zu laden. • prompt ulload fordert Sie zur Bestätigung auf. • quit ulload stoppt das Laden des XML-Codes und wird mit einem Fehler beendet. Dieses Verhalten ist das Standardverhalten, falls kein Verhalten festgelegt wird. • exit ulload wird beendet.
-f directory ODER --filedir=directory	Legt das Verzeichnis fest, das Dateien mit zusätzlich zu ladenden Daten enthält. Siehe „ UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload) “ auf Seite 233.
-g ID ODER --databaseid=ID	Setzt die Ausgangsdatenbank-ID auf den angegebenen INTEGER-Wert. Dieser Ausgangswert wird mit einer Partitionsgröße für neue Zeilen verwendet, die global autoincrement-Spalten enthalten. Beim Deployment einer Anwendung müssen Sie jeder Datenbank für die Synchronisation mit dem MobiLink-Server einen anderen Bereich von Identifizierungsnummern zuweisen. Siehe „ UltraLite-Option global_database_id “ auf Seite 198.
-i ODER --insertforsync	Eingefügte Zeilen in nächste Upload-Synchronisation einbeziehen. Standardmäßig werden die von diesem Dienstprogramm eingefügten Zeilen während der Synchronisation nicht gesendet.
--identity-file = file	Gibt die Datei an, die die Client-TLS-Identität enthält. Siehe „ identity “ [MobiLink - Clientadministration].
--identity-password = password	Legt das Kennwort für die Client-TLS-Identität fest. Siehe „ identity_password “ [MobiLink - Clientadministration].
-l filename ODER --log=filename	Protokolliert Vorgänge in der festgelegten Datei.

Option	Beschreibung
-n ODER --schemaonly	Lädt nur Schema-Metadaten. Alle Daten in der XML-Eingabedatei werden ignoriert.
-o value ODER --obfuscate=value	On oder Off, 1 oder 0, usw. Steuert, ob Daten in der Datenbank verschleiert werden. Die Verschleierung ist eine vereinfachte Form der Verschlüsselung. Siehe „Datenbanksicherheit“ auf Seite 29 und „UltraLite-Erstellungsparameter obfuscate“ auf Seite 154.
-p page-size ODER --page_size=page-size	Definiert die Seitengröße in der Datenbank. Siehe „UltraLite-Erstellungsparameter page_size“ auf Seite 155.
-q ODER --quiet	Führt das Dienstprogramm im stillen Modus aus. Informative Banner, Versionsnummern und Statusmeldungen werden unterdrückt. Fehlermeldungen werden jedoch weiter angezeigt.
-s file ODER --sql=file	Protokolliert die SQL-Anweisungen, die verwendet werden, um die Datenbank in die angegebene <i>Datei</i> zu laden.
-t file ODER --rootcert=file	Gibt die Datei an, die das vertrauenswürdige Stammzertifikat enthält. Dieses Zertifikat ist für die Serverauthentifizierung erforderlich.
--utf8_encoding=value	On oder Off, 1 oder 0, usw. Kodiert Daten im UTF-8-Format, der 8-Bit-Mehrbyte-Kodierung für Unicode. Siehe „UltraLite-Zeichensätze“ auf Seite 26 und „UltraLite-Erstellungsparameter utf8_encoding“ auf Seite 167.
-v ODER --verbose	Gibt Meldungen ausführlich aus.

Option	Beschreibung
-y ODER --overwrite	Überschreibt die Datenbankdatei ohne Bestätigung. Dies gilt nur, wenn Sie ulload verwenden, um eine neue Datenbank zu erstellen.
-? ODER --help	Zeigt die Verwendung des Dienstprogramms an und beendet das Programm.

Bemerkungen

Das Dienstprogramm ulload akzeptiert eine XML-Eingabedatei, die von ulunload, ulunloadold (mit SQL Anywhere 10 bereitgestellt) oder ulxml (in UltraLite Versionen 8 und 9) generiert wurde. Wenn es zusammen mit ulunload verwendet wird, ermöglicht es Ihnen dieses Dienstprogramm, eine Datenbank neu aufzusetzen. Eine alternative Methode, um eine Datenbank neu aufzusetzen, ist die Verwendung von ulunload, um SQL-Anweisungen zu generieren, und dann DBISQL zu verwenden, um sie in eine neue Datenbank zu lesen.

Die XML-Datei kann Metadaten für das Schema bzw. Metadaten für die Datenbankdaten enthalten. **-**ignoriert die Schema-Metadaten und fügt nur Daten in die *.udb*-Datei hinzu. **-n**ignoriert die Daten und die Metadaten und fügt nur das Schema in die *.udb*-Datei hinzu.

Die Angabe einer Option oder eines Zertifikats in der Befehlszeile hat Vorrang vor allen Einstellungen in der von ulload verarbeiteten *xml-file*.

Das Dienstprogramm ulload stellt etwaige Synchronisationsprofile in der Datenbank beim Lesens der XML wieder her.

Dieses Dienstprogramm gibt Fehlercodes zurück. Jeder andere Wert als 0 weist darauf hin, dass der Vorgang fehlgeschlagen ist.

Siehe auch

- „UltraLite-Verbindungsparameter“ auf Seite 168
- „UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload)“ auf Seite 233
- „Unterstützte Beendigungscodes“ auf Seite 201
- „UltraLite-Option `global_database_id`“ auf Seite 198

Beispiel

So erstellen Sie die neue UltraLite-Datenbankdatei *sample.udb* und laden sie mit Daten in *sample.xml*:

```
ulload -c DBF=sample.udb sample.xml
```

So laden Sie die Daten aus *sample.xml* in die vorhandene Datenbank *sample.udb* und fordern den Benutzer im Falle eines Fehlers zu einer Aktion auf:

```
ulload -d -c DBF=sample.udb --onerror=prompt sample.xml
```

So erstellen Sie das Schema und die Daten, die in *test_data.xml* in der Datenbank *sample.udb* gespeichert werden. Da der Parameter *-a* angegeben ist, muss *sample.udb* vorhanden sein, bevor dieser Befehl ausgeführt wird. Zudem schlägt der ULLOAD-Befehl fehl, wenn ein Schema oder Daten mit dem Inhalt von *sample.udb* in Konflikt steht.

```
ulload -c DBF=sample.udb -a test_data.xml
```

UltraLite-Synchronisationsdienstprogramm (ulsync)

Synchronisiert eine UltraLite-Datenbank mit einem MobiLink-Server. Dieses Tool kann zum Testen der Synchronisation während der Anwendungsentwicklung verwendet werden.

Syntax

```
ulsync -c [ options ] [ synchronization parameters]
```

Option	Beschreibung
@data	Liest Optionen aus der angegebenen Umgebungsvariablen oder Konfigurationsdatei ein. Wenn beide mit demselben Namen vorhanden sind, wird die Umgebungsvariable verwendet. Siehe „Konfigurationsdateien“ [SQL Anywhere Server - Datenbankadministration].
-c "connection-string" ODER --connect="connection-string"	Erforderlich. Stellt eine Verbindung mit der Datenbank her, wie im Parameter DBF oder file_name der <i>connection-string</i> angegeben. Wenn Sie keine Benutzer-ID und kein Kennwort angeben, werden die Standardbenutzer-ID DBA und das Standardkennwort sql verwendet.

Option	Beschreibung
-p <i>profile-name</i> ODER --profile=profile	<p>Führt eine Synchronisation unter Verwendung des angegebenen Synchronisationsprofils durch. Ist äquivalent mit:</p> <pre>SYNCHRONIZE <i>profileName</i> MERGE <i>syncOptions</i></pre> <p>Hier werden die Synchronisationsoptionen von den nachgestellten ulsync-Optionen übernommen. Beispiel:</p> <pre>ulsync -p <i>profileName</i> "MobiLinkId=ml;ScriptVersion=Version 001...<i>syncOptions</i>"</pre> <p>Siehe „Synchronisationsprofiloptionen“ auf Seite 230.</p>
-q ODER --quiet	<p>Führt das Dienstprogramm im stillen Modus aus. Informative Banner, Versionsnummern und Statusmeldungen werden unterdrückt. Fehlermeldungen werden jedoch weiter angezeigt.</p>
-r ODER --result	<p>Zeigt die letzten Synchronisationsergebnisse an und beendet die Anwendung.</p>
-v ODER --verbose	<p>Zeigt Meldungen über den Synchronisationsfortschritt an. Dies legt auch fest, ob der Verarbeitungsfortschritt bei einer Synchronisation angezeigt wird, unabhängig davon, ob die C++ API oder die SQL-Anweisung für die Profilsynchronisierung verwendet wird. Siehe „CREATE SYNCHRONIZATION PROFILE-Anweisung [Ultra-Lite]“ auf Seite 436.</p>
--log <i>filename</i>	<p>Protokolliert Vorgänge in der festgelegten Datei.</p>
-? ODER --help	<p>Zeigt Informationen über die Verwendung des Dienstprogramms an und beendet das Programm.</p>

Bemerkungen

Wenn eine Zertifikatsdatei mit der Option **trusted_certificate** oder **e2ee_public_key** definiert ist, sucht die UltraLite-Laufzeitbibliothek nach diesen Dateien nur in dem Haupt-Ressourcen-Bundle, das Bestandteil jedes Anwendungs-Deployment-Packages für iPhone ist. Sie fügen diesem Bundle Elemente hinzu, indem Sie sie im */Resources*-Ordner Ihres Xcode-Projekts speichern. Dies ist nicht anwendbar auf Zertifikate, die in der UltraLite-Datenbank gespeichert werden, und hat keinen Einfluss auf Mac OS-Clients (nur iPhone). Siehe „trusted_certificates“ [*MobiLink - Clientadministration*].

Die folgenden Optionen, die für die Versionen 10 und früher gültig waren, werden nicht mehr unterstützt: *-a authenticate-parameters*, *-e sync-parms*, *-k stream-type*, *-n* (keine Synchronisation) und *-x protocol-options*. *-e <keyword=value>* ist nun Teil der Synchronisationsparameter-Zeichenfolge, *-k* und *-x* sind nun Teil der Synchronisationsparameter-Zeichenfolge **Stream= stream{stream parms}**.

ulsync kann als gleichwertig mit einer der folgenden SQL-Anweisungen angesehen werden, abhängig vom Verwendungszweck:

```
ulsync -p profile "parms"
```

Diese Anweisung ist gleichwertig mit folgender Anweisung:

```
SYNCHRONIZE PROFILE profile MERGE  
parms
```

und

```
ulsync "parms"
```

Diese Anweisung ist gleichwertig mit folgender Anweisung:

```
SYNCHRONIZE USING <parms>
```

Bei einer sicheren Synchronisation muss die UltraLite-Anwendung Zugriff auf das öffentliche Zertifikat haben. Sie können ein Zertifikat folgendermaßen referenzieren:

- Durch Aufnahme der Zertifikatsdaten in die UltraLite-Datenbank zum Erstellungszeitpunkt mit der Option **-t file** und ulinit oder ulload.
- Durch Referenzierung einer externen Zertifikatsdatei während der Synchronisation mit der Datenstromoption **trusted_certificate=file**.

Dieses Dienstprogramm gibt Fehlercodes zurück. Jeder andere Wert als 0 weist darauf hin, dass der Vorgang fehlgeschlagen ist.

Siehe auch

- „Synchronisationsprofiloptionen“ auf Seite 230
- „Ende-zu-Ende-Verschlüsselung“ [*SQL Anywhere Server - Datenbankadministration*]
- „trusted_certificates“ [*MobiLink - Clientadministration*]
- „UltraLite-Verbindungsparameter“ auf Seite 168
- „UltraLite-Clients“ auf Seite 69
- „Unterstützte Beendigungscodes“ auf Seite 201
- „MobiLink-Dienstprogramm für die Dateiübertragung (mlfiletransfer)“ [*MobiLink - Clientadministration*]

Beispiele

Der folgende Befehl synchronisiert eine Datenbankdatei namens *myuldb.udb* für den MobiLink-Benutzer namens **remoteA**.

```
ulsync -c DBF=myuldb.udb "MobiLinkUid=remoteA;Stream=http;ScriptVersion=2"
```

Der folgende Befehl synchronisiert eine Datenbankdatei namens *myuldb.udb* über HTTPS unter Verwendung des Zertifikats *C:\Users\Public\Documents\SQL Anywhere 16\Samples\Certificates\rsaroot.crt*. Die Option **trusted_certificate=file** muss verwendet werden, weil die Datei mit dem vertrauenswürdigen Zertifikat nicht der Datenbank hinzugefügt wurde, als die Datenbank erstellt wurde. Der MobiLink-Benutzername lautet **remoteB**.

```
ulsync -c DBF=myuldb.udb "Stream=https{trusted_certificate=C:\Users\Public\Documents\SQL Anywhere 16\Samples\Certificates\rsaroot.crt};MobiLinkUid=remoteB;ScriptVersion=2;UploadOnly=ON"
```

Der folgende Befehl zeigt die letzten Synchronisationsergebnisse für die Datenbankdatei *synced.udb* an.

```
ulsync -r -c dbf=synced.udb
```

Die vorherigen Synchronisationsergebnisse werden wie folgt aufgelistet:

```
SQL Anywhere UltraLite-Dienstprogramm zur Datenbanksynchronisation Version
XX.X
  Ergebnisse der letzten Synchronisation:
  Erfolgreich
    Download-Zeitstempel: 2006-07-25 16:39:36.708000
    Upload OK
    Keine ignorierten Zeilen
    Teil-Download beibehalten
    Authentifizierungswert: 1000 (0x3e8)
```

Das folgende Beispiel zeigt die Verwendung der Befehlszeile, um die CustDB-Datenbank mit dem Benutzernamen 50 über TCP/IP auf dem Port 2439 zu synchronisieren. Es werden ausführliche Meldungen zum Verarbeitungsfortschritt verwendet.

```
ulsync -c "dbf=C:\Users\Public\Documents\SQL Anywhere 16\Samples\UltraLite\custdb.udb"
"MobiLinkUid=50;ScriptVersion=custdb 12.0;Stream=tcipip{port=2439}"
```

Der folgende Befehl veranschaulicht die Verwendung von TLS-Verschlüsselung mit E2EE:

```
ulsync -c "uid=dba;pwd=sql;dbf=myuldb.db"
"MobiLinkUid=rem1;MobiLinkPwd=password;ScriptVersion=v1;Stream=tls{host=myServer;port=2439;trusted_certificate=c:\clientcert.pem;e2ee_public_key=c:\e2eeublic.pem}"
```

Synchronisationsprofiloptionen

Mit dem Dienstprogramm *ulsync* geben Sie Synchronisationsprofiloptionen in der Befehlszeile an, nachdem Sie alle anderen Befehlszeilenoptionen definiert haben. Die Schlüsselwörter berücksichtigen nicht die Groß- und Kleinschreibung.

Synchronisationsprofiloption	Gültige Werte	Beschreibung
AllowDownloadDupRows	Boolescher Wert	Diese Option verhindert die Meldung von Fehlern, wenn mehrere Zeilen heruntergeladen werden, die denselben Primärschlüssel haben. Damit kann zugelassen werden, dass inkonsistente Daten synchronisiert werden, ohne ein Fehlschlagen der Synchronisation zu bewirken. Der Standardwert ist "no". Siehe „ Synchronisationsparameter Additional Parameters “ auf Seite 91
AuthParms	Zeichenfolge (durch Kommas getrennt)	Gibt die Liste der Authentifizierungsparameter an, die an den MobiLink-Server gesendet werden. Sie können Authentifizierungsparameter verwenden, um angepasste Authentifizierungen in MobiLink-Skripten durchzuführen. Siehe „ Synchronisationsparameter Authentication Parameters “ auf Seite 93.
CheckpointStore	Boolescher Wert	Fügt während der Synchronisation in die Datenbank zusätzliche Checkpoints ein, um das Anwachsen der Datenbank während des Synchronisationsprozesses zu verhindern. Siehe „ Synchronisationsparameter Additional Parameters “ auf Seite 91.
ContinueDownload	Boolescher Wert	Startet einen fehlgeschlagenen Download erneut. Wenn ein Download fortgesetzt wird, werden nur die Änderungen empfangen, die für den Download mit der fehlgeschlagenen Synchronisation ausgewählt waren. Standardmäßig setzt UltraLite Downloads nicht fort. Siehe „ Wiederaufnahme fehlgeschlagener Downloads “ [MobiLink - Serveradministration].
DisableConcurrency	Boolescher Wert	Unterbindet während der Synchronisation den Datenbankzugriff von anderen Threads. Siehe „ Synchronisationsparameter Additional Parameters “ auf Seite 91.
DownloadOnly	Boolescher Wert	Führt eine reine Download-Synchronisation durch. Siehe „ Synchronisationsparameter Download Only “ auf Seite 96.
KeepPartialDownload	Boolescher Wert	Steuert, ob UltraLite einen teilweisen Download bewahrt, wenn ein Verbindungsfehler auftritt. Standardmäßig setzt UltraLite teilweise heruntergeladene Änderungen nicht zurück. Siehe „ Synchronisationsparameter Keep Partial Download “ auf Seite 97.

Synchronisationsprofiloption	Gültige Werte	Beschreibung
MobiLinkPwd	String	Gibt das bestehende MobiLink-Kennwort an, das dem Benutzernamen zugeordnet ist. Siehe „ Erweiterte Option MobiLinkPwd (mp) “ [<i>MobiLink - Clientadministration</i>].
MobiLinkUid	Zeichenfolge	Gibt den MobiLink-Benutzernamen an. Siehe „ dbmlsync-Option -u (nicht mehr empfohlen) “ [<i>MobiLink - Clientadministration</i>] und „ dbmlsync-Option -mn “ [<i>MobiLink - Clientadministration</i>].
NewMobiLinkPwd	Zeichenfolge	Übergibt ein neues Kennwort für den MobiLink-Benutzer. Benutzen Sie diese Option, wenn Sie ein bestehendes Kennwort ändern wollen. Siehe „ dbmlsync-Option -mn “ [<i>MobiLink - Clientadministration</i>].
Ping	Boolescher Wert	Bestätigt nur die Kommunikation mit dem Server. Es findet keine Synchronisation statt. Siehe „ Synchronisationsparameter Ping “ auf Seite 102.
Publikationen	Zeichenfolge (durch Kommas getrennt)	Gibt die zu synchronisierenden Publikationen an. Die Publikationen legen die Tabellen in der entfernten Datenbank fest, die an der Synchronisation teilnehmen. Wenn dieser Parameter leer ist (Standardwert), werden alle Tabellen synchronisiert. Wenn der Parameter ein Sternchen (*) ist, werden alle Publikationen synchronisiert. Siehe „ Publizieren von Daten in UltraLite “ auf Seite 78.
ScriptVersion	String	Gibt die MobiLink-Skriptversion an. Die Skriptversion legt fest, welche Skripten von MobiLink in der konsolidierten Datenbank während der Synchronisation ausgeführt werden. Wenn Sie keine Skriptversion angeben, wird 'default' verwendet. Siehe „ Erweiterte Option ScriptVersion (sv) “ [<i>MobiLink - Clientadministration</i>].
SendDownloadACK	Boolescher Wert	Legt fest, dass eine Downloadbestätigung vom Client an den Server gesendet werden soll. Standardmäßig sieht der MobiLink-Server keine Downloadbestätigung vor. Siehe „ Synchronisationsparameter Send Download Acknowledgement “ auf Seite 104.
Stream	Zeichenfolge (mit Unterliste)	Gibt das MobiLink-Netzwerk-Synchronisationsprotokoll an. Siehe „ Synchronisationsparameter Stream Type “ auf Seite 107.

Synchronisationsprofiloption	Gültige Werte	Beschreibung
TableOrder	Zeichenfolge (durch Kommas getrennt)	Legt die Reihenfolge der Tabellen im Upload fest. Standardmäßig wählt UltraLite eine auf Fremdschlüsselbeziehungen basierende Reihenfolge aus. Siehe „ Synchronisationsparameter Additional Parameters “ auf Seite 91.
UploadOnly	String	Legt fest, dass die Synchronisation nur einen Upload umfasst und dass kein Download erfolgen wird. Siehe „ Synchronisationsparameter Upload Only “ auf Seite 110.

Die booleschen Werte können als YES/NO, 1/0, TRUE/FALSE, ON/OFF angegeben werden. In allen booleschen Fällen ist der Standardwert "No". Bei allen anderen Werten wird als Standardwert einfach nichts angegeben.

Siehe auch

- „[ALTER SYNCHRONIZATION PROFILE-Anweisung \[UltraLite\]](#)“ auf Seite 425
- „[DROP SYNCHRONIZATION PROFILE-Anweisung \[UltraLite\]](#)“ auf Seite 448
- „[SYNCHRONIZE-Anweisung \[UltraLite\]](#)“ auf Seite 462
- „[UltraLite-Erstellungsparameter](#)“ auf Seite 141

UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload)

Entlädt folgende Elemente, abhängig von den verwendeten Optionen:

- Eine vollständige UltraLite-Datenbank in XML oder SQL
- Die gesamten oder einen Teil der UltraLite-Daten in XML oder SQL

Syntax

ulunload -c "*connection-string*" [*options*] *output-file*

Option	Beschreibung
@data	Liest Optionen aus der angegebenen Umgebungsvariablen oder Konfigurationsdatei ein. Wenn beide mit demselben Namen vorhanden sind, wird die Umgebungsvariable verwendet.

Option	Beschreibung
-b <i>max-size</i> ODER --maxblob= <i>max-size</i>	Legt die Maximalgröße der Spaltendaten fest, die in der XML-Datei gespeichert werden können. Der Standardwert beträgt 10 kB. Um alle Daten in der XML-Datei zu speichern (keine maximale Größe), verwenden Sie -b -1 .
-c <i>"connection-string"</i> ODER --connect= <i>"connection-string"</i>	Erforderlich. Stellt eine Verbindung mit der Datenbank her, wie im Parameter DBF oder file_name der <i>connection-string</i> angegeben. Wenn Sie keine Benutzer-ID und kein Kennwort angeben, werden die Standardbenutzer-ID DBA und das Standardkennwort sql verwendet.
-d ODER --dataonly	Entlädt nur die Daten von der Datenbank in die Ausgabedatei. Entlädt keine Schemainformationen.
-e <i>table,...</i> ODER --exclude= <i>table,...</i>	Schließt die benannte <i>table</i> beim Entladen der Datenbank aus. Sie können mehrere Tabellen in einer durch Kommas getrennten Liste angeben. Beispiel: -e mydbtable1,mydbtable5
-f <i>directory</i> ODER --filedir= <i>directory</i>	Legt das Verzeichnis fest, in dem Daten gespeichert werden sollen, die größer als die in -b angegebene Maximalgröße sind. Das Standardverzeichnis ist das Verzeichnis der Ausgabedatei.
-l <i>filename</i> ODER --log= <i>filename</i>	Protokolliert Vorgänge in der festgelegten Datei.
-n ODER --schemaonly	Entlädt nur Schemadaten. Alle Daten in der Datenbank werden ignoriert.

Option	Beschreibung
-q ODER --quiet	Führt das Dienstprogramm im stillen Modus aus. Informative Banner, Versionsnummern und Statusmeldungen werden unterdrückt. Fehlermeldungen werden jedoch weiter angezeigt.
-s ODER --sql	Entlädt als SQL Anywhere-kompatible SQL-Anweisungen. Die SQL-Dateiausgabe kann von UltraLite oder SQL Anywhere unter Verwendung von DBISQL gelesen werden.
-t table,... ODER --include=table,...	Entlädt nur Daten in der genannten <i>table</i> . Sie können mehrere Tabellen in einer durch Kommas getrennten Liste angeben. Beispiel: -t mydbtable2,mydbtable6
-v ODER --verbose	Gibt Meldungen ausführlich aus.
-x owner ODER --owner=owner	Gibt Tabellen aus, sodass sie einer bestimmten Benutzer-ID gehören. Sie können diese Option zusammen mit der Option -s verwenden.
-y ODER --overwrite	Überschreibt die <i>output-file</i> ohne Bestätigung.
-? ODER --help	Zeigt Informationen über die Verwendung des Dienstprogramms an und beendet das Programm.
<i>output-file</i>	Erforderlich. Gibt den Namen der Datei an, in die die Datenbank entladen werden soll. Wenn Sie die Option -s verwenden, wird die Datenbank in Form von SQL-Anweisungen entladen. Andernfalls wird die Datenbank als XML entladen.

Bemerkungen

Standardmäßig gibt ulunload XML-Code aus, der das Schema und die Daten in der Datenbank beschreibt. Sie können die Ausgabe für Archivierungszwecke verwenden oder um die UltraLite-Datenbank über alle Versionen portierbar zu halten.

Das Speichern einer Datenbank mit einem Synchronisationsprofil führt zu einer XML, die mit früheren Versionen der UltraLite-Dienstprogramme nicht kompatibel ist. Eine Behelfslösung wäre, die XML zu bearbeiten und den Textabschnitt zu entfernen, der gekennzeichnet ist mit

```
<syncprofiles>...</syncprofiles>
```

Beim Entladen einer Datenbank werden folgende Elemente nicht beibehalten:

- Synchronisationsstatus, gespeicherte Synchronisationszähler und Zeilenlöschungen. Synchronisieren Sie die Datenbank, bevor Sie sie entladen.
- UltraLite-Benutzereinträge

Führen Sie ulinfo aus, nachdem Sie die Datenbank mit dem Dienstprogramm ulload neu geladen haben, um zu überprüfen, welche Datenbankoptionen oder -eigenschaften beibehalten wurden.

Wenn Spaltendaten die maximale Größe überschreiten, die Sie mit der Option -b angegeben haben, wird der Überlauf in einer *.bin-Datei gespeichert, und zwar an folgenden Positionen:

- Im Verzeichnis der XML-Datei
- Im durch -f angegebenen Verzeichnis.

Die Datei befolgt diese Namenskonvention:

```
tablename-columnname-rownumber.bin
```

Mit der Option -x können Sie UltraLite-Tabellen Eigentümer zuweisen. Sie müssen einer Tabelle nur dann einen Eigentümer zuweisen, wenn Sie die resultierenden SQL-Anweisungen zur Erstellung oder Änderung einer SQL Anywhere-Datenbank verwenden wollen. Beim Lesen durch UltraLite werden Eigentümernamen stillschweigend ignoriert.

Dieses Dienstprogramm gibt Fehlercodes zurück. Jeder andere Wert als 0 weist darauf hin, dass der Vorgang fehlgeschlagen ist.

Wenn Sie dieses Dienstprogramm verwenden, um eine Datenbank auf dem Windows Mobile-Gerät direkt zu entladen, kann UltraLite die Datenbank erst sichern, nachdem das Entladen durchgeführt wurde. Sie müssen diese Aktion manuell ausführen, bevor Sie diese Assistenten verwenden.

Siehe auch

- „Konfigurationsdateien“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „UltraLite-Verbindungsparameter“ auf Seite 168
- „Unterstützte Beendigungscodes“ auf Seite 201
- „UltraLite-Dienstprogramm zum Laden von Daten aus XML-Dateien (ulload)“ auf Seite 222
- „UltraLite-Informationsdienstprogramm (ulinfo)“ auf Seite 213

Beispiel

So entladen Sie die Datenbank *sample.udb* in die Datei *sample.xml*:

```
ulunload -c DBF=sample.udb sample.xml
```

So entladen Sie die Daten von der Datenbank *sample.udb* in eine SQL-Datei namens *sample1.sql*. Die SQL-Datei, falls eine vorhanden ist, wird überschrieben.

```
ulunload -c DBF=sample.udb -d -y -s sample.sql
```

UltraLite-Dienstprogramm zum Validieren von Datenbanken (ulvalid)

Führt eine vollständige (normale) Validierung einer UltraLite-Datenbank durch.

Syntax

```
ulvalid -c "connection-string" [ options ]
```

Option	Beschreibung
@data	Liest Optionen aus der angegebenen Umgebungsvariablen oder Konfigurationsdatei ein. Wenn beide mit demselben Namen vorhanden sind, wird die Umgebungsvariable verwendet.
-c "connection-string" ODER --connect="connection-string"	Erforderlich. Verbindet mit der Datenbank, wie in <i>connection-string</i> festgelegt. Wenn Sie keine Benutzer-ID und kein Kennwort angeben, werden die Standardbenutzer-ID DBA und das Standardkennwort sql verwendet.
-e ODER --express	Express-Validierung. Führt nur Tabellenvalidierung aus. Diese Option bietet eine schnellere Validierung als die normale Validierung.
-q ODER --quiet	Führt das Dienstprogramm im stillen Modus aus. Informative Banner, Versionsnummern und Statusmeldungen werden unterdrückt. Fehlermeldungen werden jedoch weiter angezeigt.
-v ODER --verbose	Gibt Meldungen ausführlich aus.

Option	Beschreibung
<code>--log=filename</code>	Protokolliert Vorgänge in der festgelegten Datei.
<code>-?</code> ODER <code>--help</code>	Zeigt Informationen über die Verwendung des Dienstprogramms an und beendet das Programm.

Bemerkungen

Das Validieren einer Datenbank verifiziert die Richtigkeit der Tabellenmetadaten und gewährleistet, dass die Datei nicht beschädigt ist.

Die Validierung enthält:

- **Datenbankseiten** Validieren Sie alle Datenbankseiten unter Verwendung von Prüfsummen, falls aktiviert. Kritische Seiten haben immer Prüfsummen und auch Seiten ohne Prüfsummen unterliegen einer grundsätzlichen Gültigkeitsprüfung.
- **Tables** Tabellen werden validiert, indem überprüft wird, dass die Tabellenzeilenanzahl mit der Anzahl im jeweiligen Index übereinstimmt.
- **Indizes** Indizes werden validiert, indem überprüft wird, ob Einträge gültige Zeilen referenzieren. `ulvalid -e` führt eine Express-Prüfung durch, die nur eine Tabellenvalidierung umfasst.

Siehe auch

- „UltraLite-Erstellungsparameter `checksum_level`“ auf Seite 144
- „Konfigurationsdateien“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „Eine UltraLite-Datenbank validieren“ auf Seite 46

Beispiel

Ein Beispiel einer Express-Validierung einer Datenbank namens *sample.udb*, ausgeführt im dialogfreien Modus.

```
ulvalid -c DBF=sample.udb -e -q
```

UltraLite-Systemtabellen

Das Schema einer UltraLite-Datenbank wird in einem systemeigenen Format gespeichert. Frühere Versionen von UltraLite-Datenbanken wurden mehreren Systemtabellen gespeichert. Diese Systemtabellen können weiterhin aus Gründen der Abwärtskompatibilität abgefragt werden (sie sind im Wesentlichen Systemansichten), aber sie enthalten nur Informationen über Benutzerschemata (wie Tabellen, Spalten, Indizes) und nicht über Systemschemata. Zum Beispiel können Sie in einer Systemtabelle nicht die Eigenschaften der Systemtabelle selbst abfragen. Sie können in der Systemtabelle nur die Eigenschaften von benutzerdefinierten Tabellen abfragen.

Jede UltraLite-Programmier-API unterstützt Objekte und Methoden, die zum Abfragen der Datenbank zu ihrem Schema verwendet werden können. Es wird empfohlen, diese Objekte und APIs zum Untersuchen von Schemata zu verwenden, statt Systemansichten abzufragen.

Alle für diese Systemansichten durchgeführten Abfragen sind gleichwertig mit vollständigen Table-Scans. Index-Scans werden für diese Systemansichten nicht unterstützt.

sysarticle-Systemtabelle

Jede Zeile in der Systemtabelle sysarticle beschreibt eine Tabelle, die einer Publikation gehört.

Spaltenname	Spaltentyp	Beschreibung
publication_id	UNSIGNED INT	Ein Bezeichner für die Publikation, zu der dieser Artikel gehört
table_id	UNSIGNED INT	Der Bezeichner der Tabelle, die zu der Publikation gehört.
where_expr	VARCHAR(256)	Ein optionales Prädikat zum Filtern von Zeilen

Integritätsregeln

PRIMARY KEY (publication_id, table_id)

FOREIGN KEY (publication_id) REFERENCES syspublication(publication_id)

FOREIGN KEY (table_id) REFERENCES systable (object_id)

Siehe auch

- „syspublication-Systemtabelle“ auf Seite 242

syscolumn-Systemtabelle

Jede Zeile in der Systemtabelle syscolumn beschreibt eine Spalte.

Spaltenname	Spaltentyp	Beschreibung
column_name	VARCHAR(128)	Der Name der Spalte.
default	VARCHAR(128)	Der Standardwert für diese Spalte, z.B. autoincrement
domain	UNSIGNED INT	Die Spaltendomäne, die ein enumerierter Wert ist, der die Domäne der Spalte angibt.
domain_info	UNSIGNED INT	Wird bei einer Domäne variabler Größe verwendet
nulls	VARCHAR(1)	Legt fest, ob die Spalte NULL als Standardwert zulässt

Spaltenname	Spaltentyp	Beschreibung
object_id	UNSIGNED INT	Ein eindeutiger Bezeichner für die Spalte
table_id	UNSIGNED INT	Der Bezeichner der Tabelle, zu der die Spalte gehört

Integritätsregeln

PRIMARY KEY(table_id, object_id)

FOREIGN KEY (table_id) REFERENCES systable (object_id)

sysindex-Systemtabelle

Jede Zeile in der Systemtabelle sysindex beschreibt einen Index in der Datenbank.

Spaltenname	Spaltentyp	Beschreibung
check_on_commit	BIT	Gibt an, wann die referenzielle Integrität geprüft wird, um sicherzustellen, dass für jeden Fremdschlüssel eine übereinstimmende primäre Zeile vorhanden ist. Nur erforderlich, wenn der Typ foreign ist.
index_name	VARCHAR(128)	Der Name des Indexes.
ixcol_count	UNSIGNED INT	Die Anzahl der Spalten im Index
nullable	BIT	Nur erforderlich, wenn der Typ foreign ist. Gibt an, ob NULL erlaubt ist.
object_id	UNSIGNED INT	Ein eindeutiger Bezeichner für einen Index
primary_index_id	UNSIGNED INT	Nur erforderlich, wenn der Typ foreign ist. Listet den Bezeichner des primären Indexes auf.
primary_table_id	UNSIGNED INT	Nur erforderlich, wenn der Typ foreign ist. Listet den Bezeichner der Primärtabelle auf.
root_handle	UNSIGNED INT	Wird nur intern verwendet.
table_id	UNSIGNED INT	Ein eindeutiger Bezeichner für die Tabelle, auf die sich der Index bezieht

Spaltenname	Spaltentyp	Beschreibung
type	VARCHAR(10)	Der Indextyp. Kann einer der folgenden Werte sein: <ul style="list-style-type: none"> • primary • foreign • key • unique • index
hash_size	UNSIGNED SHORTINT	Speichert die für den Hash-Index verwendete Hash-Größe

Integritätsregeln

PRIMARY KEY(table_id, object_id)

FOREIGN KEY(table_id) REFERENCES systable(object_id)

Siehe auch

- „sysixcol-Systemtabelle“ auf Seite 241

sysixcol-Systemtabelle

Jede Zeile in der Systemtabelle sysixcol beschreibt eine Spalte eines in sysindex aufgelisteten Indexes.

Spaltenname	Spaltentyp	Beschreibung
column_id	UNSIGNED INT	Ein eindeutiger Bezeichner für die indizierte Spalte
index_id	UNSIGNED INT	Ein eindeutiger Bezeichner für den Index, zu dem diese Indexspalte gehört
order	VARCHAR(1)	Zeigt an, ob die Spalte im Index in aufsteigender (A) oder absteigender (D) Reihenfolge gehalten wird.
sequence	UNSIGNED INT	Die Sortierung der Spalte im Index
table_id	UNSIGNED INT	Ein eindeutiger Bezeichner für die Tabelle, auf die sich der Index bezieht

Integritätsregeln

PRIMARY KEY(table_id, index_id, sequence)

FOREIGN KEY(table_id, index_id) REFERENCES sysindex(table_id, object_id)

FOREIGN KEY(table_id, index_id) REFERENCES syscolumn(table_id, object_id)

Siehe auch

- [„sysindex-Systemtabelle“ auf Seite 240](#)

syspublication-Systemtabelle

Jede Zeile in der Systemtabelle syspublication beschreibt eine Publikation.

Spaltenname	Spaltentyp	Beschreibung
download_timestamp	TIMESTAMP	Die Uhrzeit des letzten Downloads
last_sync	UNSIGNED BIGINT	Wird verwendet, um den Upload-Fortschritt zu protokollieren
publication_id	UNSIGNED INT	Ein eindeutiger Bezeichner für die Publikation
publication_name	VARCHAR(128)	Der Name der Publikation

Integritätsregeln

PRIMARY KEY (publication_id)

Siehe auch

- [„sysarticle-Systemtabelle“ auf Seite 239](#)

syssyncresult-Systemtabelle

Jede Zeile in der syssyncresult-Systemtabelle enthält Informationen über die letzte Synchronisation.

Spaltenname	Spaltentyp	Beschreibung
sql_code	INTEGER	Der SQL-Code der letzten Synchronisation.
error_string	CHAR(200)	Die Fehlermeldung der letzten Synchronisation.
stream_error_code	SMALLINT	Der spezifische Datenstromfehler. Siehe ss_error_code.
system_error_code	INTEGER	Ein systemspezifischer Fehlercode. Weitere Hinweise zu Fehlercodes finden Sie in der Dokumentation zu Ihrer Plattform.

Spaltenname	Spaltentyp	Beschreibung
stream_error_string	CHAR(80)	Eine Zeichenfolge mit zusätzlichen Informationen, sofern verfügbar, für den stream_error_code-Wert.
upload_ok	BIT	Wird bei erfolgreichem Upload auf TRUE gesetzt, andernfalls FALSE
ignored_rows	BIT	TRUE, wenn die übertragenen Zeilen ignoriert wurden, andernfalls FALSE
auth_status	UNSIGNED SMALLINT	Der Status der Synchronisationsauthentifizierung.
auth_value	INTEGER	Der vom MobiLink-Server zur Ermittlung des auth_status-Ergebnisses verwendete Wert.
auth_info	CHAR(1024)	Die vom MobiLink-Benutzerauthentifizierungsskript zurückgegebene Authentifizierungsnachricht.
partial_download_retained	BIT	Der Wert, der anzeigt, ob ein Teil-Download gespeichert wurde.
timestamp	TIMESTAMP	Datum und Zeit der letzten Synchronisation.
sent_bytes	UNSIGNED INT	Die Anzahl der derzeit für den Upload gesendeten Byte.
sent_inserts	UNSIGNED INT	Die Anzahl der aktuell eingefügten Zeilen für den Upload.
sent_updates	UNSIGNED INT	Die Anzahl der derzeit für den Upload gesendeten Zeilen.
sent_deletes	UNSIGNED INT	Die Anzahl der aktuell gesendeten gelöschten Zeilen für den Upload.
received_bytes	UNSIGNED INT	Die Anzahl der derzeit für den Download gesendeten Byte.
received_inserts	UNSIGNED INT	Die Anzahl der derzeit für den Download eingefügten Zeilen.
received_updates	UNSIGNED INT	Die Anzahl der derzeit für den Download gesendeten Zeilen.

Spaltenname	Spaltentyp	Beschreibung
received_ignored_updates	UNSIGNED INT	Die Anzahl von Duplikaten, die im Download empfangen wurden.
received_deletes	UNSIGNED INT	Die Anzahl der aktuell gesendeten gelöschten Zeilen für den Download.
received_ignored_deletes	UNSIGNED INT	Die Anzahl der gelöschten Zeilen, die im Download von Zeilen empfangen wurden und die bereits gelöscht sind.
received_truncate_deletes	UNSIGNED INT	Die Anzahl der Zeilen, die im Download durch einen Kürzungsvorgang gelöscht wurden.

Siehe auch

- „Synchronisationsparameter Keep Partial Download“ auf Seite 97

systable-Systemtabelle

Jede Zeile in der Systemtabelle systable beschreibt eine Tabelle in der Datenbank.

Spaltenname	Spaltentyp	Beschreibung
column_count	UNSIGNED INT	Die Anzahl der Spalten in der Tabelle
index_count	UNSIGNED INT	Die Anzahl der Indizes in der Tabelle
ixcol_count	UNSIGNED INT	Die Gesamtzahl der Spalten in allen Indizes der Tabelle
table_name	VARCHAR(128)	Der Name der Tabelle.
object_id	UNSIGNED INT	Ein eindeutiger Bezeichner für die Tabelle
sync_type	VARCHAR(32)	Wird für die MobiLink-Synchronisation verwendet. Kann no_sync für keine Synchronisation, all_sync für die Synchronisation jeder Zeile oder normal_sync für die Synchronisation der geänderten Zeilen sein.
table_type	VARCHAR(32)	user , um benutzerdefinierte Tabellen anzugeben.

Integritätsregeln

PRIMARY KEY (object_id)

UltraLite Java Edition-Datenbankeigenschaften

UltraLite-Java Edition-Datenbankeigenschaftswerte werden definiert, wenn die Datenbank erstellt wird. Sie können geändert werden, indem Sie die UltraLite Java Edition-Datenbank neu erstellen oder die entsprechenden Datenbankoptionen bearbeiten.

UltraLite Java Edition unterstützt die folgenden Datenbankeigenschaften:

Eigenschaft	Beschreibung
blob_file_base_dir	<p>Gibt das Basisverzeichnis zurück, das die Datenbank verwendet, um nach BLOB-Dateien zu suchen.</p> <p>Siehe auch:</p> <ul style="list-style-type: none"> • „blob_file_base_dir-Option der UltraLite Java Edition“ auf Seite 247 • „Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload)“ auf Seite 253
date_format	Gibt das Datumsformat zurück, das die Datenbank für Zeichenfolge-Konvertierungen verwendet. Siehe „UltraLite-Erstellungsparameter date_format“ auf Seite 146.
date_order	Gibt die Datumsreihenfolge zurück, die die Datenbank für Zeichenfolge-Konvertierungen verwendet. Siehe „UltraLite-Erstellungsparameter date_order“ auf Seite 149.
global_database_id	Gibt den Wert der Option global_database_id zurück, der für globalAutoincrement-Spalten verwendet wird. Siehe „UltraLite-Option global_database_id“ auf Seite 198.
ml_remote_id	Gibt den Wert der Option ml_remote_id zurück, der die Datenbank bei der MobiLink-Synchronisation eindeutig identifiziert. Siehe „UltraLite ml_remote_id-Option“ auf Seite 199.
database_name	<p>Gibt den Namen (oder Alias) der Datenbank für die aktuelle Verbindung zurück. Der zurückgegebene Name entspricht dem Wert des DBF-Verbindungsparameters. Wenn Sie den DBN-Verbindungsparameter nicht verwendet haben, ist der zurückgegebene Name die Datenbankdatei ohne Pfad und Erweiterung.</p> <p>Siehe auch:</p> <ul style="list-style-type: none"> • „UltraLite-Verbindungsparameter DBN“ auf Seite 180 • „UltraLite-Verbindungsparameter DBF“ auf Seite 177

Eigenschaft	Beschreibung
nearest_century	Gibt das nächste Jahrhundert zurück, das die Datenbank für Zeichenfolge-Konvertierungen verwendet. Siehe „ UltraLite-Erstellungsparameter nearest_century “ auf Seite 153.
page_size	Gibt die Seitengröße der Datenbank (in Byte) zurück. Siehe „ UltraLite-Erstellungsparameter page_size “ auf Seite 155.
precision	Gibt die Gleitkomma-Gesamtstellenzahl zurück, die die Datenbank für Zeichenfolge-Konvertierungen verwendet. Siehe „ UltraLite-Erstellungsparameter precision “ auf Seite 157.
scale	Gibt die minimale Anzahl der Stellen nach dem Dezimalzeichen zurück, wenn ein arithmetisches Ergebnis während Zeichenfolge-Konvertierungen auf die maximale Dezimalstellenzahl von der Datenbank gekürzt wird. Siehe „ UltraLite-Erstellungsparameter scale “ auf Seite 158.
time_format	Gibt das Zeitformat zurück, das die Datenbank für Zeichenfolge-Konvertierungen verwendet. Siehe „ UltraLite-Erstellungsparameter time_format “ auf Seite 160.
timestamp_format	Gibt das Zeitstempelformat zurück, das die Datenbank für Zeichenfolge-Konvertierungen verwendet. Siehe „ UltraLite-Erstellungsparameter timestamp_format “ auf Seite 162.
timestamp_increment	Gibt die minimale Differenzmenge zwischen zwei eindeutigen Zeitstempeln in Mikrosekunden zurück. Siehe „ UltraLite-Erstellungsparameter timestamp_increment “ auf Seite 164.
timestamp_with_time_zone	Gibt das Zeitstempelformat für TIMESTAMP WITH TIME ZONE-Werte zurück. Siehe „ UltraLite-Erstellungsparameter timestamp_with_time_zone_format “ auf Seite 166.

Siehe auch

- „[Lesen der Datenbankeigenschaften](#)“ auf Seite 39
- „[Zugriff auf Datenbankoptionen](#)“ auf Seite 40
- „[UltraLite-Erstellungsparameter](#)“ auf Seite 141
- „[UltraLite Java Edition-Datenbankoptionen](#)“ auf Seite 246

UltraLite Java Edition-Datenbankoptionen

UltraLite Java Edition-Datenbankoptionen werden definiert, wenn die Datenbank erstellt wird. Sie können geändert werden, während Sie mit der Datenbank verbunden sind.

Dieser Abschnitt beschreibt die UltraLite-Datenbankoptionen, die verfügbar sind. Zusätzlich zu diesen Optionen werden die folgenden UltraLite-Datenbankoptionen von UltraLite Java Edition-Datenbanken unterstützt:

- `global_database_id`
- `ml_remote_id`

Siehe auch

- „UltraLite-Option `global_database_id`“ auf Seite 198
- „UltraLite `ml_remote_id`-Option“ auf Seite 199
- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „Lesen der Datenbankeigenschaften“ auf Seite 39
- „UltraLite-Erstellungsparameter“ auf Seite 141

blob_file_base_dir-Option der UltraLite Java Edition

Legt das Basisverzeichnis fest, das die Datenbank verwendet, um nach BLOB-Dateien zu suchen.

Zulässige Werte

Zeichenfolge

Standardwert

Der Standardwert für BlackBerry-Smartphones ist `"file:///SDCard"`. Der Standardwert für andere Plattformen ist `""`.

Bemerkungen

Die Tabellenspalten, die Sie zum Speichern des Dateinamens verwendet haben, wird als Zeichenfolge `<base_dir></reference_to_external_file>` gespeichert, wobei `<base_dir>` das Basisverzeichnis ist, in dem sich die Blob-Datei befindet, und `<reference_to_external_file>` die exakte Zeichenfolge, die in der Inhaltsspalte in der XML-Datei enthalten ist.

Das Basisverzeichnis kann ursprünglich mit der Option `-f` des Datenbankladedienstprogramms von UltraLite Java Edition eingerichtet werden. Relative Dateinamen werden mit dem Optionswert `blob_file_base_dir` aufgelöst. Wenn der Dateiname nicht mit dem Präfix **file://** beginnt, stellt UltraLite dem Dateinamen den Optionswert voran, bevor versucht wird, die Datei zu öffnen.

Siehe auch

- „Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload)“ auf Seite 253
- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „Lesen der Datenbankeigenschaften“ auf Seite 39
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- `Connection.getDatabaseProperty`-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- `Connection.setOption`-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- `Connection.OPTION_BLOB_FILE_BASE_DIR`-Variable [BlackBerry] [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

date_format-Option der UltraLite Java Edition

Legt das Datumsformat fest, das die Datenbank für Zeichenfolge-Konvertierungen verwendet.

Zulässige Werte

Zeichenfolge

Standardwert

"YYYY-MM-DD"

Bemerkungen

Weitere Hinweise finden Sie unter [„UltraLite-Erstellungsparameter date_format“](#) auf Seite 146.

Siehe auch

- [„Zugriff auf Datenbankoptionen“](#) auf Seite 40
- [„Lesen der Datenbankeigenschaften“](#) auf Seite 39
- [„SET OPTION-Anweisung \[UltraLite\]“](#) auf Seite 459
- [Connection.getDatabaseProperty-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*]
- [Connection.setOption-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*]
- [Connection.OPTION_DATE_FORMAT-Variable \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*]

date_order-Option der UltraLite Java Edition

Legt die Datumsreihenfolge fest, die die Datenbank für Zeichenfolge-Konvertierungen verwendet.

Zulässige Werte

Zeichenfolge

Standardwert

"YMD"

Bemerkungen

Weitere Hinweise finden Sie unter [„UltraLite-Erstellungsparameter date_order“](#) auf Seite 149.

Siehe auch

- [„Zugriff auf Datenbankoptionen“](#) auf Seite 40
- [„Lesen der Datenbankeigenschaften“](#) auf Seite 39
- [„SET OPTION-Anweisung \[UltraLite\]“](#) auf Seite 459
- [Connection.getDatabaseProperty-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*]
- [Connection.setOption-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*]
- [Connection.OPTION_DATE_ORDER-Variable \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*]

nearest_century-Option der UltraLite Java Edition

Legt das nächste Jahrhundert fest, das die Datenbank für Zeichenfolge-Konvertierungen verwendet.

Zulässige Werte

Zeichenfolge

Standardwert

"50"

Bemerkungen

Weitere Hinweise finden Sie unter „UltraLite-Erstellungsparameter nearest_century“ auf Seite 153.

Siehe auch

- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „Lesen der Datenbankeigenschaften“ auf Seite 39
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- Connection.getDatabaseProperty-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- Connection.setOption-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- Connection.OPTION_NEAREST_CENTURY-Variable [] [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

precision-Option der UltraLite Java Edition

Legt die Gleitkomma-Gesamtstellenzahl fest, die die Datenbank für Zeichenfolge-Konvertierungen verwendet.

Zulässige Werte

Zeichenfolge

Standardwert

"30"

Bemerkungen

Weitere Hinweise finden Sie unter „UltraLite-Erstellungsparameter precision“ auf Seite 157.

Siehe auch

- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „Lesen der Datenbankeigenschaften“ auf Seite 39
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- Connection.getDatabaseProperty-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- Connection.setOption-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- Connection.OPTION_PRECISION-Variable [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

scale-Option der UltraLite Java Edition

Legt die minimale Anzahl der Stellen nach dem Dezimalzeichen fest, wenn ein arithmetisches Ergebnis während Zeichenfolge-Konvertierungen auf die maximale Gesamtstellenanzahl von der Datenbank gekürzt wird.

Zulässige Werte

Zeichenfolge

Standardwert

"6"

Bemerkungen

Weitere Hinweise finden Sie unter „[UltraLite-Erstellungsparameter scale](#)“ auf Seite 158.

Siehe auch

- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „Lesen der Datenbankeigenschaften“ auf Seite 39
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- [Connection.getDatabaseProperty-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*]
- [Connection.setOption-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*]
- [Connection.OPTION_SCALE-Variable \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*]

time_format-Option der UltraLite Java Edition

Legt das Zeitformat fest, das die Datenbank für Zeichenfolge-Konvertierungen verwendet.

Zulässige Werte

Zeichenfolge

Standardwert

"HH:NN:SS.SSS"

Bemerkungen

Weitere Hinweise finden Sie unter „[UltraLite-Erstellungsparameter time_format](#)“ auf Seite 160.

Siehe auch

- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „Lesen der Datenbankeigenschaften“ auf Seite 39
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- [Connection.getDatabaseProperty-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*]
- [Connection.setOption-Methode \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*]
- [Connection.OPTION_TIME_FORMAT-Variable \[UltraLiteJ\]](#) [*UltraLite® – Java-Programmierung*]

timestamp_format-Option der UltraLite Java Edition

Legt das Zeitstempelformat fest, das die Datenbank für Zeichenfolge-Konvertierungen verwendet.

Zulässige Werte

Zeichenfolge

Standardwert

"YYYY-MM-DD HH:NN:SS.SSS"

Bemerkungen

Weitere Hinweise finden Sie unter „UltraLite-Erstellungsparameter timestamp_format“ auf Seite 162.

Siehe auch

- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „Lesen der Datenbankeigenschaften“ auf Seite 39
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- Connection.getDatabaseProperty-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- Connection.setOption-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- Connection.OPTION_TIMESTAMP_FORMAT-Variable [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

timestamp_increment-Option der UltraLite Java Edition

Legt die minimale Differenzmenge zwischen zwei eindeutigen Zeitstempeln in Mikrosekunden fest.

Zulässige Werte

Zeichenfolge

Standardwert

"1"

Bemerkungen

Weitere Hinweise finden Sie unter „UltraLite-Erstellungsparameter timestamp_increment“ auf Seite 164.

Siehe auch

- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „Lesen der Datenbankeigenschaften“ auf Seite 39
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- Connection.getDatabaseProperty-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- Connection.setOption-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]
- Connection.OPTION_TIMESTAMP_INCREMENT-Variable [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

timestamp_with_time_zone_format-Option der UltraLite Java Edition

Legt das Zeitstempelformat für TIMESTAMP WITH TIME ZONE-Werte fest.

Zulässige Werte

Zeichenfolge

Standardwert

"YYYY-MM-DD HH:NN:SS.SSS+HH:NN"

Bemerkungen

Weitere Hinweise finden Sie unter „UltraLite-Erstellungsparameter timestamp_with_time_zone_format“ auf Seite 166.

Siehe auch

- „Zugriff auf Datenbankoptionen“ auf Seite 40
- „Lesen der Datenbankeigenschaften“ auf Seite 39
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- [Connection.getDatabaseProperty-Methode \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.setOption-Methode \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.OPTION_TIMESTAMP_WITH_TIME_ZONE_FORMAT-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)

Dienstprogramme der UltraLite Java Edition

UltraLite-Dienstprogramme werden für Wartungsarbeiten und Verwaltungsaufgaben in UltraLite Java Edition-Datenbanken verwendet.

UltraLite Java Edition-Dienstprogramm für Datenbankinformationen (uljinfo)

Zeigt Informationen über eine vorhandene UltraLite Java Edition-Datenbank an.

Syntax

uljinfo -c filename -p password [options]

Option	Beschreibung
-c filename	Erforderlich. Gibt den Dateinamen der zu prüfenden UltraLite Java Edition-Datenbank an.

Option	Beschreibung
-ek <i>key</i>	Gibt den Chiffrierschlüssel an, der für den Zugriff auf die verschlüsselte UltraLite Java Edition-Datenbank erforderlich ist.
-p <i>password</i>	Gibt das Kennwort für die Verbindung mit der UltraLite Java Edition-Datenbank an. Der Standardwert ist sql .
-v	Zeigt Meldungen ausführlich an.
-?	Zeigt Syntaxinformationen zur Befehlszeile an.

Beispiel

Nachfolgend finden Sie ein Beispiel für die Ausgabe mit dem uljinfo-Dienstprogramm:

```
C:\ULj\bin>uljinfo.cmd -c ..\Samples\Demol.ulj -p sql
SQL Anywhere UltraLiteJ-Dienstprogramm für Datenbankinformationen
Datenbankname: ..\Samples\Demol.ulj
Datei: '..\Samples\Demol.ulj'
Datenbank-ID: 0
Seitengröße: 1024
0 Zeilen für den nächsten Upload
Datumsformat: YYYY-MM-DD
Datumsreihenfolge: YMD
Nächstes Jahrhundert: 50
Anzahl Gesamtstellen: 30
Dezimalstellen: 6
Zeitformat: HH:NN:SS.SSS
Zeitstempelformat: YYYY-MM-DD HH:NN:SS.SSS
Zeitstempel-Inkrementierungsstufe: 1
Anzahl von Tabellen: 1
Anzahl von Spalten: 2
Anzahl von Publikationen: 0
Anzahl von Tabellen, die immer eingelesen werden: 0
Anzahl von Tabellen, die nie synchronisiert werden: 0
Anzahl von Primärschlüsseln: 1
Anzahl von Fremdschlüsseln: 0
Anzahl von Indizes: 0
Letzter Download am Thu Jul 05 11:31:05 EDT 2007
Upload OK: TRUE
```

Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload)

Stellt die Funktionalität bereit, um eine UltraLite Java Edition-Datenbank aus einer XML-Quelldatei zu laden. Die XML-Datei wird häufig vom uljunload-Dienstprogramm erstellt und ist anpassbar.

Hinweis

Datenbankkonfigurationsparameter können nicht in der Befehlszeile gesetzt werden. Das uljload-Dienstprogramm verwendet die Standardkonfiguration, wenn eine neue Datenbank erstellt wird. Um eine Datenbank mit einzelnen Konfigurationsparametern zu erstellen, z.B. mit einer anderen Seitengröße, erstellen Sie die Datenbank zuerst in Java und verwenden dann **uljload -a**, um den XML-Inhalt in die erstellte Datenbank zu laden.

Syntax

uljload -c filename -p password [options] inputfile

Option	Beschreibung
-a	Fügt einer vorhandenen Datenbank Informationen aus der XML-Datei hinzu. Wenn diese Option nicht festgelegt ist, wird eine neue Datenbank erstellt.
-c filename	Erforderlich. Gibt den Namen der Datenbankdatei an.
-d	Lädt nur Daten. Schemainformationen werden ignoriert.
-ek key	Gibt den Chiffrierschlüssel an, der für den Zugriff auf die verschlüsselte UltraLite Java Edition-Datenbank erforderlich ist.
-f directory	Legt das Verzeichnis zum Abfragen von Daten für Spalten fest, die die maximale Blob-Größe (mit der Option -b bei uljunload festgelegt) überschreiten.
-i	Fügt Zeilen für die Upload-Synchronisation ein.
-n	Lädt nur Schemainformationen, Zeilendaten werden ignoriert.
-p password	Optional. Legt das Kennwort fest. Das Standardkennwort ist sql .
-q	Führt das Dienstprogramm im stillen Modus aus. Es werden keine Nachrichten angezeigt.
-v	Zeigt Meldungen ausführlich an.
-y	Überschreibt die Ausgabedatei, wenn vorhanden (und wenn die Option -a nicht festgelegt ist).
-z pagesize	Legt die Seitengröße (in Byte) fest, um die Datenbank zu erstellen.
-?	Zeigt Syntaxinformationen zur Befehlszeile an.
<i>inputfile</i>	Legt die Eingabedatei mit XML-Anweisungen fest.

Beispiel für das Laden von blobfile-Typen

In diesem Beispiel wird angenommen, dass Sie die folgende SQL-Anweisung auf Ihre UltraLite Java Edition-Datenbank angewendet und die Datei mit dem UltraLite Java Edition-Dienstprogramm zum Entladen einer Datenbank entladen haben.

```
CREATE TABLE blobfile_example
  file_name CHAR(size) DEFAULT AUTOFILENAME( prefix, extension ),
  file_contents LONG BINARY STORE AS FILE( file_name ) CASCADE DELETE
```

Wenn das uljload-Dienstprogramm auf eine vom uljunload exportierte Datei stößt, behandelt es die file_name-Spalte als CHAR-Spalte und die file_contents-Spalte als LONG BINARY-Spalte.

Wenn uljload auf dieselbe Datei im XML-Format stößt, rekonstruiert es den blobfile-Typ. Die Spalte file_name speichert die Zeichenfolge <base_dir>/<reference_to_external_file>, wobei <base_dir> die exakte Zeichenfolge in der Option -f und <reference_to_external_file> die exakte Zeichenfolge ist, die in der Spalte file_contents in der XML-Datei enthalten ist. Die neu aufgebaute Datenbank enthält danach die gültigen Referenzen auf die externen Dateien.

Der load-Befehl schlägt fehl und in der Fehlermeldung werden Sie aufgefordert, die Größe der file_name-Spalte zu erhöhen, wenn der generierte Dateiname zu lang ist.

Siehe auch

- „Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload)“ auf Seite 256

Beispiel

Die Option -? zeigt Befehlszeileninformationen an.

```
uljload -?
```

Wenn die Befehlszeile des uljload-Dienstprogramms die Option -? enthält, werden die folgenden Informationen über die Syntax angezeigt:

```
Lade-Dienstprogramm für SQL Anywhere UltraLiteJ-Datenbanken
Verwendung: uljload [Optionen] <XML-Datei>
            Erstellt und lädt Daten aus <XML file> in eine neue UltraLiteJ-
Datenbank.
```

Optionen:

```
-a      Zu einer vorhandenen Datenbank hinzufügen.
-c <Datei> Datenbankdatei.
-d      Nur Daten -- Schema ignorieren.
-ek <Schlüssel> Chiffrierschlüssel.
-f <Verzeichnis>
        Verzeichnis zum Laden von .File-Pfaden.
-i      Zeilen für Upload-Synchronisation einfügen.
-n      Nur Schema -- Daten ignorieren.
-p      Kennwort für die Verbindung zur Datenbank.
-q      Still: Keine Meldungen ausgeben.
-v      Nachrichten ausführlich anzeigen.
-y      Bereits vorhandene Datei überschreiben.
-z <Seitengröße>
        Seitengröße, mit der die Datenbank erstellt werden soll.
```

Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload)

Stellt die Funktionalität zum Entladen einer UltraLite Java Edition-Datenbank (die Daten, das Schema oder beides) in eine XML-Datei bereit.

Syntax

uljunload -c filename -p password [options] outputfile

Optionen	Beschreibung
-b <i>max-blob-size</i>	Gibt die maximale Größe (in Byte) der Blob/Char-Datenausgabe in XML an.
-c <i>filename</i>	Erforderlich. Gibt den Namen der Datenbankdatei an, die heruntergeladen werden soll.
-d	Entlädt nur Daten und gibt keine Schemainformationen aus.
-e <i>table, ...</i>	Schließt Daten für in der Liste genannte Tabellen aus.
-ek <i>key</i>	Gibt den Chiffrierschlüssel an, der für den Zugriff auf die verschlüsselte UltraLite Java Edition-Datenbank erforderlich ist.
-f <i>directory</i>	Gibt das Verzeichnis zum Speichern von Daten für Spalten, die die mit der Option -b festgelegte maximale BLOB-Größe überschreiten.
-n	Entlädt nur Schemainformationen und gibt keine Daten aus.
-p <i>password</i>	Gibt das Kennwort für die Verbindung mit der Datenbank an. Der Standardwert ist sql .
-q	Führt das Dienstprogramm im dialogfreien Modus aus. Meldungen werden nicht angezeigt.
-t <i>table, ...</i>	Gibt Daten nur für Tabellen aus, die in der Liste genannt sind.
-v	Zeigt Meldungen ausführlich an.
-y	Überschreibt die Ausgabedatei, wenn sie bereits vorhanden ist.
-?	Zeigt Optionsverwendung und Hilfeinformationen.
<i>outputfile</i>	Gibt den Dateinamen aus. (Diese Datei enthält XML-Anweisungen, die die Datenbankinhalte beschreiben.)

Beispiel für das Entladen von blobfile-Typen

Wenden Sie die folgende SQL-Beispielanweisung auf Ihre Datenbank an, um einen blobfile-Typ zu entladen:

```
CREATE TABLE blobfile_example
  file_name CHAR(size) DEFAULT AUTOFILENAME( prefix, extension ),
  file_contents LONG BINARY STORE AS FILE( file_name ) CASCADE DELETE
```

Wenn Sie dieses Beispiel anwenden und dieses Dienstprogramm verwenden, exportiert uljunload die file_name-Spalte als normale CHAR-Spalte, jedoch mit einem zusätzlichen Attribut, default_autofilename, welches das Präfix und die Erweiterungszeichenfolgen in der Form 'Präfix', 'Erweiterung' speichert. Die file_contents-Spalte wird als LONG BINARY exportiert, jedoch mit einem zusätzlichen Attribut, filename_col, das den Namen der referenzierten file_name-Spalte speichert. In der Zeile, welche die blobfile-Spalten enthält, bleibt der Inhalt der file_name-Spalte unverändert. Die file_contents-Spalte verhält sich wie eine extern gespeicherte BLOB-Spalte und hat die folgende Form:

```
file_contents.File="tablename-columnname-rownumber.bin"
```

Der Inhalt der file_contents-Spalte wird in Form von *-bin*-Dateien an dem durch die Option -f festgelegten Ort gespeichert.

Siehe auch

- „Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload)“ auf Seite 256

Beispiel für XML-Datei-Inhalte

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<ul:ulschema xmlns:ul="urn:ultralite">
  <collation name="1252LATIN1" case_sensitive="no"/>
  <options>
    <option name="dateformat" value="YYYY-MM-DD"/>
    <option name="dateorder" value="YMD"/>
    <option name="nearestcentury" value="50"/>
    <option name="precision" value="30"/>
    <option name="scale" value="6"/>
    <option name="timeformat" value="HH:NN:SS.SSS"/>
    <option name="timestampformat" value="YYYY-MM-DD HH:NN:SS.SSS"/>
    <option name="timestampincrement" value="1"/>
  </options>
  <tables>
    <table name="ULCustomer" sync="changes">
      <columns>
        <column name="cust_id" type="integer" null="no"/>
        <column name="cust_name" type="char(30)" null="yes"/>
      </columns>
      <primarykey>
        <primarycolumn name="cust_id" direction="asc"/>
      </primarykey>
      <indexes/>
    </table>
  </tables>
  <uldata>
    <table name="ULCustomer">
      <row cust_id="2000" cust_name="Apple St. Builders"/>
      <row cust_id="2001" cust_name="Art's Renovations"/>
      <row cust_id="2002" cust_name="Awnings R Us"/>
      <row cust_id="2003" cust_name="Al's Interior Design"/>
      <row cust_id="2004" cust_name="Alpha Hardware"/>
    </table>
  </uldata>
</ul:ulschema>
```

```
<row cust_id="2005" cust_name="Ace Properties"/>
<row cust_id="2006" cust_name="Al Contracting"/>
<row cust_id="2007" cust_name="Archibald Inc."/>
<row cust_id="2008" cust_name="Acme Construction"/>
<row cust_id="2009" cust_name="ABCXYZ Inc."/>
<row cust_id="2010" cust_name="Buy It Co."/>
<row cust_id="2011" cust_name="Bill's Cages"/>
<row cust_id="2012" cust_name="Build-It Co."/>
<row cust_id="2013" cust_name="Bass Interiors"/>
<row cust_id="2014" cust_name="Burger Franchise"/>
<row cust_id="2015" cust_name="Big City Builders"/>
<row cust_id="2016" cust_name="Bob's Renovations"/>
<row cust_id="2017" cust_name="Basements R Us"/>
<row cust_id="2018" cust_name="BB Interior Design"/>
<row cust_id="2019" cust_name="Bond Hardware"/>
<row cust_id="2020" cust_name="Cat Properties"/>
<row cust_id="2021" cust_name="C & C Contracting"/>
<row cust_id="2022" cust_name="Classy Inc."/>
<row cust_id="2023" cust_name="Cooper Construction"/>
<row cust_id="2024" cust_name="City Schools"/>
<row cust_id="2025" cust_name="Can Do It Co."/>
<row cust_id="2026" cust_name="City Corrections"/>
<row cust_id="2027" cust_name="City Sports Arenas"/>
<row cust_id="2028" cust_name="Cantaloupe Interiors"/>
<row cust_id="2029" cust_name="Chicken Franchise"/>
</table>
</uldata>
</ul:ulschema>
```

UltraLite Java Edition-Dienstprogramm für die Datenbankübertragung

Das uljdbtserv-Dienstprogramm stellt die Funktionalität bereit, um eine UltraLite-Datenbank von einem BlackBerry-Smartphone auf ein externes Gerät, z.B. auf einen Desktop, ein Notebook oder einen Server, zu übertragen. Außerdem können Sie eine Datenbank löschen, Datenbankinformationen anzeigen sowie das Datenbankübertragungslog anzeigen oder per E-Mail senden. Das Dienstprogramm besteht aus zwei Anwendungen, die gleichzeitig ausgeführt werden müssen: der Desktopanwendung für die UltraLite Java Edition-Datenbankübertragung (uljdbt) und der Clientanwendung für das BlackBerry Smartphone *ULjDatabaseTransfer.cod*.

Die Desktopanwendung empfängt UltraLite Java Edition-Datenbanken unter Verwendung einer USB- oder HTTP-Verbindungsmethode. Wenn Sie die Serveranwendung starten, wartet sie auf ein BlackBerry-Smartphone, das die Datenbank über die durch die Clientanwendung angegebene Verbindung überträgt. Die Verbindung wird entweder manuell über die Anwendungsschnittstelle geschlossen, wenn bei der Anwendung eine Zeitüberschreitung eintritt, oder wenn die Übertragung abgeschlossen ist.

Die BlackBerry-Smartphone-Clientanwendung sendet UltraLite Java Edition-Datenbanken über ein USB-Kabel bzw. einen festgelegten TCP-Port an die Desktopanwendung.

Clientanwendung starten

Mit dem uljdbtserv-Dienstprogramm können Sie eine UltraLite Java Edition-Datenbank übertragen, löschen, anzeigen und per E-Mail versenden.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Die Clientanwendung ist eine signierte Datei, die sich im Verzeichnis *UltraLite\UltraLiteJ\BlackBerry4.2* der SQL Anywhere-Installation befindet.

Aufgabe

1. Laden Sie *ULjDatabaseTransfer.cod* aus dem Verzeichnis *UltraLite\UltraLiteJ\BlackBerry4.2* der SQL Anywhere-Installation.

Das Symbol der Clientanwendung erscheint in der Liste der Anwendungen.

2. Starten Sie die Anwendung und drücken Sie das Trackwheel.
3. Füllen Sie im Fenster **Datenbankverbindung** folgende Felder aus:
 - **Datenbankname** Der Name der Datenbank, die auf das externe Gerät übertragen werden soll. Wenn der Name mit *file://* (Groß-/Kleinschreibung beachten) beginnt, versucht die Clientanwendung, die Datenbank im Dateisystem zu finden, sonst findet sie die Datenbank im Objektspeicher.
 - **Datenbank-Kennwort** Das Datenbankkennwort, das zur Genehmigung der Datenübertragung verwendet wird. Wenn Sie das Feld **Datenbank-Kennwort** leer lassen, wird das Standardkennwort verwendet.
 - **Chiffrierschlüssel** Der Datenbank-Chiffriercode zum Verschlüsseln der Datenbank. Diese Option ist nur erforderlich, wenn die Datenbank verschlüsselt ist.
4. Klicken Sie auf **Weiter**.

Ergebnisse

Der Bildschirm **Aktion** wird angezeigt. Auf diesem Bildschirm können Sie auf alle Funktionen der Clientanwendung zugreifen.

UltraLite Java Edition-Datenbank übertragen

Übertragen einer Datenbank unter Verwendung der BlackBerry-Smartphone-Clientanwendung.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. Führen Sie das *uljdbt*-Dienstprogramm, das sich in *%SQLANY16%\Bin32\uljdbtserv.cmd* befindet, auf dem BlackBerry-Smartphone-Client aus.

2. Klicken Sie im Fenster **Aktion** Ihres Blackberrys auf die gewünschte Verbindungsmethode (USB oder HTTP).
3. Für eine USB-Übertragung klicken Sie auf **USB-Datenbankübertragung**. Für eine HTTP-Übertragung gehen Sie zum nächsten Schritt.
4. Folgen Sie den Anweisungen zum Starten der Desktopanwendung für die Datenbankübertragung.

Hinweis

Um eine erfolgreiche Datenbankübertragung zu gewährleisten, stellen Sie sicher, dass das Gerät oder der Simulator mit dem BlackBerry Device Manager verbunden ist. Für einen Simulator stellen Sie sicher, dass mit der Option **USB Cable Connected** eine USB-Verbindung simuliert wird.

- a. Klicken Sie in der Clientanwendung auf **Weiter**.
 - b. Stellen Sie in der Desktopanwendung sicher, dass **USB** gewählt ist, und klicken Sie auf **Start**.
Das BlackBerry-Smartphone beginnt, die Datenbank auf das externe Gerät zu übertragen. In der Desktopanwendung wird der Verarbeitungsfortschritt angezeigt.
 - c. Klicken Sie sowohl in der Client- als auch in der Desktopanwendung auf **OK**, um die Anwendungen zu schließen.
5. Für eine HTTP-Übertragung klicken Sie auf **HTTP-Datenbankübertragung**.
- a. Klicken Sie im Fenster **HTTP-Übertragung** auf **Weiter**.
 - b. Geben Sie folgende Werte ein:
 - **Host** Die IP-Adresse Ihres Desktop-PCs.
 - **Port** Den Port, der unter **Verbindungseigenschaften** in der Desktopanwendung festgelegt wurde.
 - **URL-Suffix** Den Hostnamen des Servers, der die Übertragung empfängt, einschließlich des **http://**-Suffixes (dies ist erforderlich).
 - c. Klicken Sie auf **Weiter**.

Hinweis

Um über BES auf ein Non-Ident-Gerät zu übertragen, lassen Sie **Suffix** leer. Bei Ident-Geräten verwenden Sie das Suffix **;deviceside=false**.

Zur Übertragung über Direct TCP verwenden Sie das Suffix **;deviceside=true**. Nicht alle Netzbetreiber unterstützen diese Funktion.

Das WAP-Gateway des Netzbetreibers kann benutzt werden, falls Sie die entsprechenden APN-Daten kennen. Sie müssen diese Informationen auch an das Suffix anhängen. Auch bei Verwendung eines BES befindet sich möglicherweise eine Firewall zwischen dem BES und dem Gerät, auf dem das UltraLite Java Edition-Dienstprogramm für die Datenbankübertragung ausgeführt wird. In diesem Fall müssen Sie einen SSL-Tunnel verwenden. Geben Sie im Fenster **HTTP Transfer Parms** den Port und den Namen oder die IP-Adresse des SSL-Servers an, der auf der BES-Seite der Firewall läuft. Sie müssen der Übertragungsanwendung außerdem den Port mitteilen, dem der SSL-Client zugeordnet ist.

Wenn Sie eine Datenbank von einem BlackBerry-Simulator aus übertragen, muss ein BlackBerry MDS-Simulator laufen oder Sie müssen dem UltraLite Java Edition-Dienstprogramm für die Datenbankübertragung, das den BlackBerry-Simulator ausführt, das URL-Suffix **;deviceside=true** angeben.

- d. Stellen Sie in der Desktopanwendung sicher, dass **HTTP** gewählt ist, und klicken Sie auf **Start**.
- e. Klicken Sie in der Clientanwendung auf **Weiter**.
Das BlackBerry-Smartphone beginnt, die Datenbank auf das externe Gerät zu übertragen. In der Desktopanwendung wird der Verarbeitungsfortschritt angezeigt.
- f. Klicken Sie sowohl in der Client- als auch in der Desktopanwendung auf **OK**, um die Anwendungen zu schließen.

Ergebnisse

Die Datenbank wird übertragen.

Eine UltraLite Java Edition-Datenbank empfangen

Verwenden Sie das UltraLite Java Edition-Dienstprogramm für die Datenbankübertragung, um eine Datenbank zu empfangen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. Führen Sie *uljdbtserv.cmd* vom *Bin32*-Verzeichnis Ihrer SQL Anywhere-Installation aus.
2. Auf der Registerkarte **Verbinden** klicken Sie auf die gewünschte **Verbindungsmethode**.

3. Unter **Verbindungseigenschaften** geben Sie folgende Werte an:

- **Port** Dieses Feld gilt nur für HTTP-Verbindungen. Geben Sie die TCP-Portnummer ein, zu der das BlackBerry Smartphone eine Verbindung herstellen soll. Normalerweise entspricht diese Portnummer der Portnummer, die dem UltraLite Java Edition-Dienstprogramm für die Datenbankübertragung mitgeteilt wurde, das auf dem BlackBerry-Smartphone läuft. Wenn Sie jedoch SSL verwenden, kann es eine andere Nummer sein.
- **BlackBerry-Kennwort** Dieses Feld gilt nur für USB-Verbindungen. Wenn das angeschlossene BlackBerry-Smartphone gesperrt ist, geben Sie das Kennwort für den Zugriff ein. Lassen Sie dieses Feld leer, wenn es kein Kennwort gibt.
- **Timeout** Die Anzahl von inaktiven Minuten, bevor bei der Serveranwendung eine Zeitüberschreitung auftritt und sie die Verbindung schließt.
- **Ausgabe** Geben Sie einen Dateinamen und einen Speicherort für die übertragene Datenbank ein.

4. Klicken Sie auf **Start**, um eine Verbindung zum BlackBerry Smartphone herzustellen.

Die Serveranwendung wartet, bis entweder eine Zeitüberschreitung eintritt oder eine Verbindung hergestellt ist. Wenn Sie eine vorhandene Datei angegeben haben, werden Sie gefragt, ob sie überschrieben werden soll.

Die Registerkarte **Logs** enthält Details zum Serverstatus und zum Fortschritt der Übertragung einschließlich eventueller Fehlermeldungen.

Ergebnisse

Die Datenbank wird empfangen.

Eine UltraLite Java Edition-Datenbank löschen

Löschen einer Datenbank.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. Klicken Sie im Fenster **Aktion** Ihres Blackberrys auf die Option **Datenbank löschen**.
2. Klicken Sie im Bestätigungsdialogfeld auf **Löschen**, um die Datenbank zu löschen.
3. Klicken Sie im Dialogfeld **Datenbank gelöscht** auf **OK**, um den Client zu schließen.

Ergebnisse

Die Datenbank wird gelöscht.

UltraLite Java Edition-Datenbankinformationen anzeigen

Anzeigen der Datenbankinformationen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. Klicken Sie im Fenster **Aktion** Ihres Blackberrys auf die Option **Datenbankinfos anzeigen**. Blättern Sie nach unten, um die Datenbankinformationen anzuzeigen.
2. Klicken Sie auf **Zurück**, um zum Bildschirm **Aktion** zurückzukehren.

Ergebnisse

Sie können die Datenbankinformationen lesen.

Logdateien anzeigen

Zeigen Sie die Logdatei an.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. Rufen Sie im Fenster **Datenbankverbindung** Ihres Blackberrys das Menü auf.
2. Klicken Sie auf **Log**. Der Anmeldebildschirm wird angezeigt.
3. Um die Logdatei per E-Mail zu senden, geben Sie die E-Mail-Adresse ein und klicken Sie auf **E-Mail senden**. Wenn Sie zum vorherigen Fenster zurückkehren möchten, drücken Sie die Eingabetaste.

Ergebnisse

Sie können die Logdatei lesen.

Systemtabellen der UltraLite Java Edition

Dieser Abschnitt enthält eine Einführung in die Systemtabellen, die UltraLite Java Edition-Datenbanken zur Verfügung stehen.

sysarticles-Systemtabelle

Jede Zeile in der Systemtabelle sysarticle beschreibt eine Tabelle, die einer Publikation gehört.

Spaltenname	Spaltentyp	Beschreibung
publication_id	UNSIGNED INTEGER	Ein Bezeichner für die Publikation, zu der dieser Artikel gehört
table_id	UNSIGNED INTEGER	Der Bezeichner der Tabelle, die zu der Publikation gehört.
predicate	VARCHAR(256)	Der predicate-Ausdruck, der von der dazugehörigen WHERE-Klausel einer CREATE PUBLICATION- oder ALTER PUBLICATION-Anweisung angegeben wird.

Integritätsregeln

PRIMARY KEY (publication_id, table_id)

syscolumn-Systemtabelle

Jede Zeile in der Systemtabelle syscolumn beschreibt eine Spalte.

Spaltenname	Spaltentyp	Beschreibung
table_id	UNSIGNED INTEGER	Der Bezeichner der Tabelle, zu der die Spalte gehört
column_id	UNSIGNED INTEGER	Ein eindeutiger Bezeichner für die Spalte
column_name	VARCHAR(128)	Der Name der Spalte. Siehe Domain-Schnittstelle [UltraLiteJ] [<i>UltraLite® – Java-Programmierung</i>].
column_flags	TINY	Eine Bit-für-Bit-Kombination aus den folgenden, die Attribute beschreibenden Parametern: <ul style="list-style-type: none">● 0x01 Spalte ist der Primärschlüssel.● 0x02 Spalte ist nullwertfähig
column_domain	TINY	Die Spaltendomäne, ein enumerierter Wert, der die Domäne der Spalte in den 6 Bits niedriger Ordnung angibt. Die restlichen Bits werden intern verwendet.

Spaltenname	Spaltentyp	Beschreibung
column_length	UNSIGNED SHORT	Die Spaltenlänge. Bei Spalten vom Typ VARCHAR und BINARY, die in der Domain-Schnittstelle definiert sind, ist dies die maximale Länge in Byte. Bei Spalten vom Typ NUMERIC wird die Gesamtstellenzahl im ersten Byte gespeichert, und die Dezimalstellen werden im zweiten Byte gespeichert.
column_default_value	VARCHAR(128)	Der Standardwert für diese Spalte, der durch einen der COLUMN_DEFAULT-Werte in der ColumnSchema-Schnittstelle angegeben wird. Beispiel: COLUMN_DEFAULT_AUTOINC gibt einen selbstinkrementierenden Standardwert an. Wenn für eine varchar-Spalte die Anweisung DEFAULT AUTOFILENAME angegeben wurde, speichert sie die Parameter für Präfix und Erweiterung in der Form Präfix Erweiterung.
filename_colid	UNSIGNED INTEGER	Speichert die Spalten-ID der Spalte, die den Dateinamen für eine als STORE AS FILE deklarierte LONG BINARY-Spalte enthält. Sonst hat diese Spalte den Wert NULL.

Integritätsregeln

PRIMARY KEY (table_id, column_id)

sysforeignkey-Systemtabelle

Jede Zeile in der Systemtabelle sysforeignkey beschreibt einen Fremdschlüssel, der einer Tabelle gehört.

Spaltenname	Spaltentyp	Beschreibung
table_id	UNSIGNED INTEGER	Der Bezeichner der Tabelle, zu der der Fremdschlüssel gehört
foreign_table_id	UNSIGNED SHORT	Der Bezeichner der Tabelle, den diese Fremdschlüssel-Spalte referenziert
foreign_key_id	UNSIGNED INTEGER	Der Bezeichner des Fremdschlüssels
name	VARCHAR(128)	Der Name des Fremdschlüssels.

Spaltenname	Spaltentyp	Beschreibung
index_name	VARCHAR(128)	Der Name des Indexes, den der Fremdschlüssel referenziert

Integritätsregeln

PRIMARY KEY (table_id, foreign_key_id)

sysfkcol-Systemtabelle

Jede Zeile in der Systemtabelle sysfkcol beschreibt eine Fremdschlüssel-Spalte.

Spaltenname	Spaltentyp	Beschreibung
table_id	UNSIGNED INTEGER	Ein eindeutiger Bezeichner für die Tabelle, auf die sich der Fremdschlüssel bezieht
foreign_key_id	UNSIGNED INTEGER	Ein eindeutiger Bezeichner für den Fremdschlüssel, zu dem diese Spalte gehört
item_no	UNSIGNED SHORT	Die Sortierung der Spalte im Fremdschlüssel
column_id	UNSIGNED SHORT	Ein eindeutiger Bezeichner der Tabellenspalte, die die Fremdspalte referenziert
foreign_column_id	UNSIGNED SHORT	Ein eindeutig Bezeichner der Tabellenspalte, die referenziert wird

Integritätsregeln

PRIMARY KEY (table_id, foreign_key_id, item_no)

sysindex-Systemtabelle

Jede Zeile in der Systemtabelle sysindex beschreibt einen Index in der Datenbank.

Spaltenname	Spaltentyp	Beschreibung
table_id	UNSIGNED INTEGER	Ein eindeutiger Bezeichner für die Tabelle, auf die sich der Index bezieht
index_id	UNSIGNED INTEGER	Ein eindeutiger Bezeichner für einen Index
index_name	VARCHAR(128)	Der Name des Indexes.

Spaltenname	Spaltentyp	Beschreibung
index_flags	TINY	Eine Bit-für-Bit-Kombination aus den folgenden Parametern, die den Typ des Indexes und seine Beständigkeit angeben: <ul style="list-style-type: none"> • 0x01 Eindeutiger Schlüssel. • 0x02 Eindeutiger Index. • 0x04 Der Index ist beständig. • 0x08 Primärschlüssel.
index_data	UNSIGNED INTEGER	Wird nur intern verwendet.
hash_size	UNSIGNED SHORT	Die Hash-Größe für den Hash-Index

Integritätsregeln

PRIMARY KEY (table_id, index_id)

sysindexcolumn-Systemtabelle

Jede Zeile in der Systemtabelle sysindexcolumn beschreibt eine Spalte eines in sysindex aufgelisteten Indexes.

Spaltenname	Spaltentyp	Beschreibung
table_id	UNSIGNED INTEGER	Ein eindeutiger Bezeichner für die Tabelle, auf die sich der Index bezieht
index_id	UNSIGNED INTEGER	Ein eindeutiger Bezeichner für den Index, zu dem diese Indexspalte gehört
order	UNSIGNED INTEGER	Die Sortierung der Spalte im Index
column_id	UNSIGNED SHORT	Ein eindeutiger Bezeichner für die indizierte Spalte
index_column_flag	TINY	Ein Hinweis, wo in der Spalte sich der Index befindet, entweder in aufsteigender (1) oder in absteigender (2) Folge

Integritätsregeln

PRIMARY KEY (table_id, index_id, order)

syspublications-Systemtabelle

Jede Zeile in der Systemtabelle syspublication beschreibt eine Publikation.

Spaltenname	Spaltentyp	Beschreibung
publication_id	UNSIGNED INTEGER	Ein eindeutiger Bezeichner für die Publikation
publication_name	VARCHAR(128)	Der Name der Publikation
download_timestamp	TIMESTAMP	Die Uhrzeit des letzten Downloads
last_sync_sent	UNSIGNED INTEGER	Eine Ganzzahl, die einen an MobiLink gesendeten Upload protokolliert
last_sync_confirmed	UNSIGNED INTEGER	Eine Ganzzahl, die einen Upload protokolliert, der als von MobiLink empfangen bestätigt wurde

Integritätsregeln

PRIMARY KEY (publication_id)

sysstable-Systemtabelle

Jede Zeile in der Systemtabelle sysstable beschreibt eine Tabelle in der Datenbank.

Spaltenname	Spaltentyp	Beschreibung
table_id	UNSIGNED INTEGER	Ein eindeutiger Bezeichner für die Tabelle
table_name	VARCHAR(128)	Der Name der Tabelle.
table_flags	UNSIGNED SHORT	Eine Bit-für-Bit-Kombination aus einem der folgenden Parameter: <ul style="list-style-type: none">• TableSchema.TABLE_IS_SYSTEM• TableSchema.TABLE_IS_NO_SYNC• TableSchema.TABLE_IS_DOWNLOAD_ONLY
table_data	UNSIGNED INTEGER	Wird nur intern verwendet.

Spaltenname	Spaltentyp	Beschreibung
table_partition_size	UNSIGNED INTEGER	Die Partitionsgröße der Tabelle, wenn diese angegeben ist oder die Tabelle einen Standardwert enthält; ansonsten NULL. Dieser Wert ist 0, wenn die Partitionsgröße außerhalb des zulässigen Bereichs liegt.

Integritätsregeln

PRIMARY INDEX (table_id)

sysuldata-Systemtabelle

Jede Zeile in der Systemtabelle sysuldata enthält Wertepaare von Optionen und Eigenschaften.

Synchronisationsprofile werden in der sysuldata-Systemtabelle gespeichert, wobei **name** auf den Namen des Profils gesetzt wird, **type** auf **ulsync** und **long_setting** auf die UTF-8-Kodierung der Profilzeichenfolge. Genau eine der Spalteneinstellungen und **long_setting** enthalten einen zugeordneten Wert, abhängig vom Verwendungszweck.

Spaltenname	Spaltentyp	Beschreibung
long_setting	LONGBINARY	Ein BLOB für lange Werte
name	VARCHAR(128)	Der Name der Eigenschaft.
setting	VARCHAR(128)	Der Wert der Eigenschaft
type	VARCHAR(128)	Entweder opt für Optionen oder prop für Eigenschaften

Integritätsregeln

PRIMARY KEY (name, type)

Siehe auch

- [Connection.OPTION_BLOB_FILE_BASE_DIR-Variable \[BlackBerry\] \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.OPTION_DATABASE_ID-Variable \[BlackBerry\] \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.OPTION_DATE_FORMAT-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.OPTION_DATE_ORDER-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.OPTION_ML_REMOTE_ID-Variable \[BlackBerry\] \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.OPTION_NEAREST_CENTURY-Variable \[\] \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.OPTION_PRECISION-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.OPTION_SCALE-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.OPTION_TIME_FORMAT-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.OPTION_TIMESTAMP_FORMAT-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.OPTION_TIMESTAMP_INCREMENT-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.OPTION_TIMESTAMP_WITH_TIME_ZONE_FORMAT-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.PROPERTY_DATABASE_NAME-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)
- [Connection.PROPERTY_PAGE_SIZE-Variable \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#)

UltraLite-SQL-Referenz

Dieser Abschnitt enthält Referenzinformationen für UltraLite-SQL. UltraLite-SQL ist eine eindeutige Teilmenge des von SQL Anywhere-Datenbanken unterstützten SQL.

UltraLite-SQL-Sprachelemente

Schlüsselwörter in UltraLite

Jede SQL-Anweisung enthält ein oder mehrere Schlüsselwörter. SQL-Schlüsselwörter berücksichtigen die Groß- und Kleinschreibung nicht. In dieser Dokumentation werden Schlüsselwörter in Großbuchstaben geschrieben. Manche Schlüsselwörter können nur als Bezeichner verwendet werden, wenn sie in Anführungszeichen gesetzt werden. Diese Schlüsselwörter werden als **reservierte Wörter** bezeichnet.

Hinweis

UltraLite unterstützt nur eine Teilmenge der SQL Anywhere-Schlüsselwörter. Dennoch sollten Sie, um Probleme bei zukünftigen Releases zu vermeiden, annehmen, dass alle reservierten Wörter in SQL Anywhere auch für UltraLite gelten.

Siehe auch

- „Reservierte Wörter“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Bezeichner in UltraLite

Bezeichner sind die Namen von Objekten in der Datenbank – z.B. Benutzer-IDs, Tabellen und Spalten. Bezeichner haben eine maximale Länge von 128 Byte.

Bezeichner müssen zwischen Anführungszeichen gestellt werden, wenn eine der folgenden Bedingungen zutrifft:

- Der Bezeichner enthält Leerstellen.
- Das erste Zeichen des Bezeichners ist kein alphabetisches Zeichen. Die Kollationssequenz der Datenbank bestimmt, welche Zeichen als alphabetische Zeichen gelten und welche als Ziffern.
- Der Bezeichner enthält ein reserviertes Wort.
- Der Bezeichner enthält nicht-alphabetische Zeichen und Ziffern.

Sie können einen einzelnen Backslash in einem Bezeichner nur verwenden, wenn er als Escapezeichen verwendet wird.

Siehe auch

- „Reservierte Wörter“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Zeichenfolgen in UltraLite

Mit den Zeichenfolgen werden die Zeichendaten in der Datenbank gespeichert. UltraLite unterstützt dieselben Regeln für Zeichenfolgen wie SQL Anywhere. Die Ergebnisse eines Zeichenfolgenvergleichs und die Sortierfolge der Zeichenfolgen hängt davon ab, ob die Datenbank die Groß- und Kleinschreibung berücksichtigt, sowie vom Zeichensatz und von der Kollationssequenz. Diese Eigenschaften werden gesetzt, wenn die Datenbank erzeugt wird.

Siehe auch

- „Zeichenfolgen“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „UltraLite-Zeichensätze“ auf Seite 26

Kommentare in UltraLite

Mit Kommentaren wird erklärender Text zu SQL-Anweisungen oder Anweisungsblöcken hinzugefügt. Die UltraLite-Laufzeitanwendung führt Kommentare nicht aus.

Folgende Kommentarindikatoren sind in UltraLite verfügbar:

- **-- (Doppel-Bindestrich)** Der Datenbankserver ignoriert alle restlichen Zeichen in der Zeile. Dieser Indikator ist der Kommentarindikator von SQL/2003.
- **// (Doppel-Schrägstrich)** Der Doppel-Schrägstrich hat dieselbe Bedeutung wie der Doppel-Bindestrich.
- **/* ... */ (Schrägstrich-Stern)** Alle Zeichen zwischen den beiden Kommentarmarkierungen werden ignoriert. Die beiden Kommentarmarkierungen können in derselben oder in verschiedenen Zeilen sein. In diesem Stil markierte Kommentare können verschachtelt sein. Dieser Kommentarstil wird auch C-Stil-Kommentar genannt.

Hinweis

Das Prozentzeichen (%) wird in UltraLite nicht unterstützt.

Beispiele

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

```
CREATE TABLE borrowed_book (  
    loaner_name CHAR(100) PRIMARY KEY,  
    date_borrowed DATE NOT NULL DEFAULT CURRENT DATE,  
    date_returned DATE,  
    book CHAR(20)  
    FOREIGN KEY book REFERENCES library_books (isbn),  
)  
--This statement creates a table for a library database to hold  
information on borrowed books.  
--The default value for date_borrowed indicates that the book is borrowed  
on the day the entry is made.  
--The date_returned column is NULL until the book is returned.
```

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

```

CREATE TABLE borrowed_book (
    loaner_name CHAR(100)      PRIMARY KEY,
    date_borrowed DATE NOT NULL DEFAULT CURRENT DATE,
    date_returned      DATE,
    book                CHAR(20)
    FOREIGN KEY book REFERENCES library_books (isbn),
)
/* This statement creates a table for a library database to hold
information on borrowed books.
The default value for date_borrowed indicates that the book is borrowed on
the day the entry is made.
The date_returned column is NULL until the book is returned. */

```

Zahlen in UltraLite

Mit Zahlen werden die numerischen Daten in der Datenbank gespeichert. Eine Zahl kann Folgendes sein bzw. enthalten:

- Eine beliebige Sequenz von Ziffern
- Sie kann Dezimalstellen enthalten
- Sie kann ein optionales Minuszeichen (-) oder Pluszeichen (+) enthalten
- Sie kann von einem e und einem numerischen Exponentialwert gefolgt sein

Alle im Folgenden aufgelisteten Zahlen werden z.B. von UltraLite unterstützt:

42

-4.038

.001

3.4e10

1e-10

NULL in UltraLite

Wie in SQL Anywhere ist NULL ein spezieller Wert, der sich von allen gültigen Werten der verschiedenen Datentypen unterscheidet. NULL ist jedoch in jedem Datentyp ein zulässiger Wert. NULL wird verwendet, um eine unbekannte (kein Wert) oder nicht zutreffende Information darzustellen.

Siehe auch

- „NULL-Spezialwert“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Spezialwerte in UltraLite

Sie können Spezialwerte in Ausdrücken und als Standardwerte für Spalten verwenden, wenn Sie Tabellen erstellen.

CURRENT DATE-Spezialwert

Gibt das aktuelle Jahr sowie den aktuellen Monat und Tag zurück

Datentyp

DATE

Bemerkungen

Das zurückgegebene Datum basiert auf dem Wert der Systemuhr zu dem Zeitpunkt, als die SQL-Anweisung von der UltraLite-Laufzeitanwendung ausgeführt wurde. Wenn Sie CURRENT DATE mit einem der folgenden Datentypen verwenden, basieren alle Werte auf verschiedenen Uhrzeitwerten:

- CURRENT DATE mehrere Uhrzeiten innerhalb einer Anweisung
- CURRENT DATE mit CURRENT TIME oder CURRENT TIMESTAMP innerhalb einer einzelnen Anweisung
- CURRENT DATE mit der Funktion NOW oder GETDATE innerhalb einer einzelnen Anweisung

Siehe auch

- [„Ausdrücke in UltraLite“ auf Seite 277](#)
- [„GETDATE-Funktion \[Datum und Uhrzeit\]“ auf Seite 356](#)
- [„NOW-Funktion \[Datum und Uhrzeit\]“ auf Seite 379](#)

CURRENT TIME-Spezialwert

Die aktuelle Stunde, Minute, Sekunde und Sekundenbruchteile

Datentyp

TIME

Bemerkungen

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

Das zurückgegebene Datum basiert auf dem Wert der Systemuhr zu dem Zeitpunkt, als die SQL-Anweisung von der UltraLite-Laufzeitanwendung ausgeführt wurde. Wenn Sie CURRENT TIME mit einem der folgenden Datentypen verwenden, basieren alle Werte auf verschiedenen Uhrzeitwerten:

- CURRENT TIME mehrere Uhrzeiten innerhalb einer Anweisung
- CURRENT TIME mit CURRENT DATE oder CURRENT TIMESTAMP innerhalb einer einzelnen Anweisung
- CURRENT TIME mit der Funktion NOW oder GETDATE innerhalb einer einzelnen Anweisung

Siehe auch

- „Ausdrücke in UltraLite“ auf Seite 277
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 356
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 379

CURRENT TIMESTAMP-Spezialwert

Kombiniert CURRENT DATE und CURRENT TIME zu einem TIMESTAMP-Wert, der Jahr, Monat, Tag, Stunde, Minute, Sekunde und Sekundenbruchteile enthält

Datentyp

TIMESTAMP

Bemerkungen

Der Sekundenbruchteil wird mit 3 Dezimalstellen gespeichert. Die Genauigkeit ist durch die Genauigkeit der Systemuhr begrenzt.

Spalten, die als DEFAULT CURRENT TIMESTAMP deklariert sind, enthalten nicht unbedingt eindeutige Werte.

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

CURRENT_TIMESTAMP ist mit CURRENT TIMESTAMP äquivalent.

Das zurückgegebene Datum basiert auf dem Wert der Systemuhr zu dem Zeitpunkt, als die SQL-Anweisung von der UltraLite-Laufzeitanwendung ausgeführt wurde. Wenn Sie CURRENT TIMESTAMP mit einem der folgenden Datentypen verwenden, basieren alle Werte auf verschiedenen Uhrzeitwerten:

- CURRENT TIMESTAMP mehrere Uhrzeiten innerhalb einer Anweisung
- CURRENT TIMESTAMP mit CURRENT DATE oder CURRENT TIME innerhalb einer einzelnen Anweisung
- CURRENT TIMESTAMP mit der Funktion NOW oder GETDATE innerhalb einer einzelnen Anweisung

Siehe auch

- „CURRENT TIME-Spezialwert“ auf Seite 274
- „Ausdrücke in UltraLite“ auf Seite 277
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 379
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 356
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 379

CURRENT UTC TIMESTAMP-Spezialwert

Gibt einen TIMESTAMP WITH TIME ZONE-Wert zurück, der die aktuelle UTC-Zeit bestehend aus Jahr, Monat, Tag wiedergibt.

Datentyp

DATE

Bemerkungen

Das zurückgegebene Datum basiert auf dem Wert der Systemuhr zu dem Zeitpunkt, als die SQL-Anweisung von der UltraLite-Laufzeitanwendung ausgeführt wurde.

- CURRENT DATE mehrere Uhrzeiten innerhalb einer Anweisung
- CURRENT DATE mit CURRENT TIME oder CURRENT TIMESTAMP innerhalb einer einzelnen Anweisung
- CURRENT DATE mit der Funktion NOW oder GETDATE innerhalb einer einzelnen Anweisung

Siehe auch

- [„Ausdrücke in UltraLite“ auf Seite 277](#)
- [„GETDATE-Funktion \[Datum und Uhrzeit\]“ auf Seite 356](#)
- [„CURRENT TIMESTAMP-Spezialwert“ auf Seite 275](#)
- [„NOW-Funktion \[Datum und Uhrzeit\]“ auf Seite 379](#)

SQLCODE-Spezialwert

Der aktuelle SQLCODE-Wert zu dem Zeitpunkt, zu dem der Spezialwert ausgewertet wird. Nicht unterstützt von UltraLite Java Edition-Datenbanken.

Datentyp

String

Bemerkungen

Der SQLCODE-Wert wird nach jeder Anweisung gesetzt. Sie können SQLCODE prüfen, um zu ermitteln, ob die Anweisung erfolgreich war.

Siehe auch

- [„Ausdrücke in UltraLite“ auf Seite 277](#)
- [Fehlermeldungen](#)

Beispiel

Verwenden Sie eine SELECT-Anweisung, um für jeden Versuch, eine neue Zeile aus der Ergebnismenge abzurufen, einen Fehlercode zu produzieren. Zum Beispiel: SELECT a, b, SQLCODE FROM MyTable.

Datums- und Zeitangaben in UltraLite

Zahlreiche Datums- und Zeitfunktionen verwenden Datumsangaben, die aus Datums- und Zeitteilen bestehen. UltraLite und SQL Anywhere unterstützen dieselben Datumsteile.

Siehe auch

- [Angaben von Datumsteilen auf Seite 317](#)

Ausdrücke in UltraLite

Ausdrücke werden durch die Kombination von Daten und oft in Form von Spaltenreferenzen, mit Operatoren oder Funktionen gebildet.

Syntax

```
expression:
  case-expression
  | constant
  | [correlation-name.]column-name
  | - expression
  | expression operator expression
  | ( expression )
  | function-name ( expression, ... )
  | if-expression
  | special value
  | input-parameter
```

Parameter

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

alternative form of case-expression:

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

constant:

```
integer | number | string | host-variable
```

special-value:

```
CURRENT { DATE | TIME | TIMESTAMP }
| NULL
| SQLCODE
| SQLSTATE
```

if-expression:

```
IF condition
```

```
THEN expression
[ ELSE expression ]
ENDIF
```

```
input-parameter:
{ ? | :name [ : indicator-name ] }
```

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

Siehe auch

- „Konstanten in Ausdrücken“ auf Seite 278
- „Spezialwerte in UltraLite“ auf Seite 273
- „Spaltennamen in Ausdrücken“ auf Seite 278
- „UltraLite SQL-Funktionen“ auf Seite 316
- „Unterabfragen in Ausdrücken“ auf Seite 281
- „Suchbedingungen in UltraLite“ auf Seite 283
- „UltraLite, SQLDatentypen“ auf Seite 296
- „CASE-Ausdrücke“ auf Seite 280
- „Eingabeparameter“ auf Seite 282

Konstanten in Ausdrücken

In UltraLite sind Konstanten Zahlen oder Zeichenfolgenlitterale.

Syntax

```
' constant '
```

Verwendung

Zeichenfolgenkonstanten werden in Apostrophe ('einfache Anführungszeichen') eingeschlossen.

Ein Apostroph innerhalb einer Zeichenfolge wird durch zwei aufeinanderfolgende Apostrophe dargestellt.

Siehe auch

- [Escapesequenzen \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)

Beispiel

Um einen englischen Possessivausdruck zu verwenden, geben Sie die Zeichenfolgenlitterale folgendermaßen ein:

```
'John's database'
```

Spaltennamen in Ausdrücken

Ein Bezeichner in einem Ausdruck

Syntax

correlation-name.column-name

Bemerkungen

Einem Spaltennamen wird ein optionaler Korrelationsname vorangestellt, bei dem es sich gewöhnlich um den Namen einer Tabelle handelt.

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

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

Siehe auch

- „FROM-Klausel [UltraLite]“ auf Seite 450

IF-Ausdrücke

Legt eine Suchbedingung fest, um eine spezifische Teilmenge von Daten zurückzugeben

Syntax 1

```
IF search-condition
THEN expression1
[ ELSE expression2 ]
ENDIF
```

Bemerkungen

Aus Kompatibilitätsgründen kann dieser Ausdruck entweder mit ENDIF oder mit END IF aufhören.

Dieser Ausdruck gibt Folgendes zurück:

- Wenn *Suchbedingung* den Wert TRUE hat, gibt der IF-Ausdruck *Ausdruck1* zurück.
- Wenn *Suchbedingung* den Wert FALSE hat und eine ELSE-Klausel angegeben wurde, gibt der IF-Ausdruck *Ausdruck2* zurück.
- Wenn *Suchbedingung* den Wert FALSE hat und kein *Ausdruck2* vorhanden ist, gibt der IF-Ausdruck NULL zurück.
- Wenn *Suchbedingung* den Wert UNKNOWN hat, gibt der IF-Ausdruck NULL zurück.

Siehe auch

- „NULL-Spezialwert“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Suchbedingungen“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

CASE-Ausdrücke

Stellt bedingte SQL-Ausdrücke bereit

Syntax 1

```
CASE expression1
WHEN expression2 THEN expression3, ...
[ ELSE expression4 ]
END

SELECT ID,
      ( CASE name
        WHEN 'Tee Shirt' THEN 'Shirt'
        WHEN 'Sweatshirt' THEN 'Shirt'
        WHEN 'Baseball Cap' THEN 'Hat'
        ELSE 'Unknown'
      END ) as Type
FROM Product
```

Syntax 2

```
CASE
WHEN search-condition
THEN expression1, ...
[ ELSE expression2 ]
END
```

Bemerkungen

Aus Kompatibilitätsgründen können Sie diesen Ausdruck entweder mit ENDCASE oder mit END CASE abschließen.

Sie können CASE-Ausdrücke überall dort verwenden, wo reguläre Ausdrücke zulässig sind.

Syntax 1 Falls der Ausdruck, der dem Schlüsselwort CASE folgt, gleich dem Ausdruck ist, der dem ersten Schlüsselwort WHEN folgt, wird der Ausdruck zurückgegeben, der dem Schlüsselwort WHEN folgt. Andernfalls wird der Ausdruck zurückgegeben, der dem Schlüsselwort ELSE folgt, sofern angegeben.

Der folgende Code verwendet zum Beispiel eine CASE-Anweisung als zweite Klausel in einer SELECT-Anweisung. Er wählt eine Zeile aus der Tabelle Product aus, in der der Spaltenname den Wert Sweatshirt hat.

Syntax 2 Wenn die Suchbedingung, die dem ersten Schlüsselwort WHEN folgt, den Wert TRUE hat, wird der Ausdruck, der dem zugeordneten Schlüsselwort THEN folgt, zurückgegeben. Andernfalls wird der Ausdruck zurückgegeben, der der ELSE-Klausel folgt, sofern angegeben.

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

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

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

Beispiel

Die folgende Anweisung verwendet einen CASE-Ausdruck als dritte Klausel einer SELECT-Anweisung, um eine Zeichenfolge einer Suchbedingung zuzuordnen. Wenn der Wert der Namensspalte **Tee Shirt** ist, gibt diese Abfrage **Sale** zurück. Und wenn der Wert der Namensspalte nicht **Tee Shirt** und die Menge größer als fünfzig ist, wird **Big Sale** zurückgegeben. Ansonsten gibt die Abfrage stets **Regular price** zurück.

```
SELECT ID, name,
       ( CASE
         WHEN name='Tee Shirt' THEN 'Sale'
         WHEN quantity >= 50 THEN 'Big Sale'
         ELSE 'Regular price'
       END ) as Type
FROM Product
```

Aggregatausdrücke

Führt eine Aggregatberechnung durch, die von der UltraLite-Laufzeitbibliothek nicht bereitgestellt wird

Syntax

SUM(*expression*)

Bemerkungen

Ein Aggregatausdruck berechnet aus einer Reihe von Zeilen einen einzelnen Wert.

Ein Aggregatausdruck ist immer dann gegeben, wenn entweder eine Aggregatfunktion verwendet wird oder wenn ein oder mehrere Operanden ein Aggregatausdruck sind.

Hat eine SELECT-Anweisung keine GROUP BY-Klausel, können alle Ausdrücke in der Auswahlliste entweder nur Aggregatausdrücke oder keine Aggregatausdrücke enthalten. Hat eine SELECT-Anweisung aber eine GROUP BY-Klausel, muss jeder Nicht-Aggregatausdruck in der Auswahlliste in der Liste GROUP BY stehen.

Beispiel

Beispiel: In der folgenden Abfrage werden alle Gehälter für die Mitarbeiter in der Tabelle employee zusammengerechnet. In dieser Abfrage ist SUM(salary) ein Aggregatausdruck.

```
SELECT SUM( salary )
FROM employee
```

Unterabfragen in Ausdrücken

Eine SELECT-Anweisung, die in einer anderen SELECT-Anweisung verschachtelt ist

Syntax

Eine Unterabfrage ist wie eine normale Abfrage aufgebaut.

Bemerkungen

In UltraLite können Sie nur in den folgenden Situationen Referenzen zu Unterabfragen verwenden:

- Als Tabellenausdruck in der FROM-Klausel. Diese Form von Tabellenausdruck (auch bezeichnet als **abgeleitete Tabelle**) muss einen abgeleiteten Tabellennamen und Spaltennamen haben, in denen Werte in der SELECT-Liste abgerufen werden.
- Um Werte für die Suchbedingungen EXISTS, ANY, ALL und IN bereitzustellen.

Sie können Unterabfragen zu Namen schreiben, die vor (links von) der Unterabfrage angegeben werden. Sie werden dann "linke äußere Referenzen" genannt. Es ist jedoch nicht möglich, Referenzen zu Elementen innerhalb von Unterabfragen (auch als innere Referenzen bezeichnet) zu verwenden.

Siehe auch

- „SELECT-Anweisung [UltraLite]“ auf Seite 457
- „Verwendung von Unterabfragen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Suchbedingungen in UltraLite“ auf Seite 283

Beispiel

Die folgende Unterabfrage wird verwendet, um alle Produkt-IDs für Artikel aufzulisten, die nur noch in geringem Umfang auf Lager sind (weniger als 20 Stück).

```
FROM SalesOrderItems
( SELECT ID
  FROM Products
  WHERE Quantity < 20 )
```

Eingabeparameter

Agiert als Platzhalter, um Endbenutzern Werte für eine vorbereitete Anweisung bereitzustellen. Diese vom Benutzer angegebenen Werte werden dann zur Ausführung der Anweisung verwendet.

Syntax

```
{ ? | :name [ : indicator-name ] }
```

Bemerkungen

Verwenden Sie das Platzhalterzeichen ? oder die benannte Form in Ausdrücken. Sie können Eingabeparameter überall dort verwenden, wo Sie einen Spaltennamen oder eine Konstante verwenden können.

Das genaue Verfahren zur Bereitstellung der Werte für die Anweisung hängen von der API ab, die Sie bei der Erstellung des UltraLite-Clients verwenden.

Benannte Form verwenden Die benannte Form eines Eingabeparameters hat eine spezielle Bedeutung. Im Allgemeinen wird *name* verwendet, um mehrere Orte anzugeben, an denen ein tatsächlicher Wert bereitgestellt wird.

Bei Embedded SQL-Anwendungen liefert *indicator-name* die Variable, in die der Nullindikator positioniert wird. Wenn Sie die benannte Form mit den anderen Komponenten verwenden, wird *indicator-name* ignoriert.

Datentypen ableiten Der Datentyp des Eingabeparameters wird abgeleitet, wenn die Anweisung mit einem der folgenden Muster vorbereitet wird:

- **CAST (? AS Typ)**

In diesem Fall ist *type* eine Spezifikation eines Datenbanktyps, wie etwa CHAR(32).

- Genau ein Operand eines Binäroperators ist ein Eingabeparameter. Es wird abgeleitet, dass der Typ der Operandtyp ist.

Wenn der Typ nicht abgeleitet werden kann, generiert UltraLite einen Fehler. Beispiel:

- **-?:** Der Operand ist unär.
- **? + ?:** Beide sind Eingabeparameter.

Siehe auch

- „Hostvariablen“ [[UltraLite - C- und C++-Programmierung](#)]
- „Vorbereitete Anweisungen“ [[SQL Anywhere Server - Programmierung](#)]
- UltraLite C/C++: „Datenmodifikation mit INSERT, UPDATE und DELETE“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite.NET: „Datenänderung mit INSERT, UPDATE und DELETE“ [[UltraLite - .NET-Programmierung](#)]

Beispiel

Die folgende Embedded SQL-Anweisung hat zwei Eingabeparameter:

```
INSERT INTO MyTable VALUES ( :v1, :v2, :v1)
```

Die erste Instanz von v1 stellt seinen Wert sowohl den Positionen v2 und v1 in der Anweisung bereit.

Suchbedingungen in UltraLite

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

Syntax

```
search-condition:
expression comparison-operator expression
| expression IS [ NOT ] NULL
| expression [ NOT ] BETWEEN expression AND expression
| expression [ NOT ] IN ( expression, ... )
| expression [ NOT ] IN ( subquery )
| expression [ NOT ] { ANY | ALL } ( subquery )
| expression [ NOT ] EXISTS ( subquery )
| expression [ NOT ] LIKE ( pattern )
| NOT search-condition
| search-condition AND search-condition
| search-condition OR search-condition
| ( search-condition IS [ NOT ] { TRUE | FALSE | UNKNOWN } )
```

comparison-operator :

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

Parameter

Die folgenden Typen von Suchbedingungen werden von UltraLite unterstützt:

- ALL-Bedingung
- ANY-Bedingung
- BETWEEN-Bedingung
- EXISTS-Bedingung
- IN-Bedingung
- LIKE-Bedingung

Bemerkungen

In UltraLite können Suchbedingungen in folgenden Elementen enthalten sein:

- WHERE-Klausel
- HAVING-Klausel
- ON-Klausel
- SQL-Abfragen

Suchbedingungen können verwendet werden, um eine Teilmenge der Zeilen einer Tabelle in einer FROM-Klausel in einer SELECT-Anweisung auszuwählen, oder in Ausdrücken, wie etwa IF oder CASE, um spezifische Werte auszuwählen. In UltraLite wird jede Bedingung als einer von drei Zuständen ausgewertet: TRUE, FALSE oder UNKNOWN. Kombiniert werden diese Zustände als **dreiwertige Logik** bezeichnet. Das Ergebnis eines Vergleichs ist UNKNOWN, falls einer der beiden verglichenen Werte NULL ist. Suchbedingungen sind nur erfüllt, wenn das Ergebnis der Bedingungen den Wert TRUE hat.

Siehe auch

- „Suchbedingungen in UltraLite“ auf Seite 283
- „Abfragen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Musterübereinstimmung von Zeichenfolgen in der WHERE-Klausel“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Vergleichsoperatoren“ auf Seite 285
- „Drei-Werte-Logik“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Unterabfragen in Ausdrücken“ auf Seite 281

Vergleichsoperatoren

Jeder Operator, der es gestattet, zwei oder mehr Ausdrücke in einer Suchbedingung zu vergleichen

Syntax

expression operator expression

Parameter

Operator	Interpretation
=	Gleich
>	Größer als
<	Kleiner als
>=	Größer als oder gleich
<=	Kleiner als oder gleich
!=	Ungleich
<>	Ungleich
!>	Nicht größer als
!<	Nicht kleiner als

Bemerkungen

Daten vergleichen Beim Vergleichen von Daten bedeutet < früher und > später.

Vergleich von LONG VARCHAR- und LONG BINARY-Werten UltraLite unterstützt keine Vergleiche mit LONG VARCHAR- oder LONG BINARY-Werten.

Berücksichtigung von Groß- und Kleinschreibung In UltraLite werden Vergleiche unter derselben Berücksichtigung von Groß- und Kleinschreibung ausgeführt, die der Datenbank entspricht, mit der Sie arbeiten. Standardmäßig werden die UltraLite-Datenbanken so erstellt, dass die Groß- und Kleinschreibung nicht berücksichtigt wird.

NOT-Operator Der NOT-Operator negiert einen Ausdruck.

Siehe auch

- „Logische Operatoren“ auf Seite 286
- „Suchbedingungen in UltraLite“ auf Seite 283

Beispiel

Beide nachstehenden Abfragen finden alle Tee Shirts und Baseball Caps, 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
```

Logische Operatoren

Führt eine der folgenden Aktionen aus:

- Bedingungen vergleichen (AND, OR und NOT)
- Die Wahrheit oder NULL des Ausdrucks überprüfen (IS)

Syntax 1

condition1 logical-operator condition2

Syntax 2

NOT *condition*

Syntax 3

expression **IS** [**NOT**] { *truth-value* | **NULL** }

Bemerkungen

Suchbedingungen können verwendet werden, um eine Teilmenge der Zeilen einer Tabelle in einer FROM-Klausel in einer SELECT-Anweisung auszuwählen, oder in Ausdrücken, wie etwa IF oder CASE, um spezifische Werte auszuwählen. In UltraLite wird jede Bedingung als einer von drei Zuständen ausgewertet: TRUE, FALSE oder UNKNOWN. Kombiniert werden diese Zustände als **dreiwertige Logik** bezeichnet. Das Ergebnis eines Vergleichs ist UNKNOWN, falls einer der beiden verglichenen Werte NULL ist. Suchbedingungen sind nur erfüllt, wenn das Ergebnis der Bedingungen den Wert TRUE hat.

AND Die kombinierte Bedingung ist TRUE, falls beide Bedingungen TRUE sind; FALSE, falls eine Bedingung FALSE ist; und sonst UNKNOWN.

condition1 **AND** *condition2*

OR Die kombinierte Bedingung ist TRUE, wenn eine Bedingung TRUE ist; FALSE, wenn beide Bedingungen FALSE sind; und sonst UNKNOWN.

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

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

Siehe auch

- „Drei-Werte-Logik“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „Vergleichsoperatoren“ auf Seite 285
- „Suchbedingungen in UltraLite“ auf Seite 283

Beispiel

Die Bedingung IS NULL ist erfüllt, wenn die Spalte NULL enthält. Wenn Sie den Operator IS NOT NULL verwenden, ist die Bedingung erfüllt, wenn die Spalte einen anderen Wert als NULL enthält. Dieses Beispiel zeigt eine IS NULL-Bedingung: WHERE paid_date IS NULL.

ALL-Suchbedingung

Verwenden Sie die ALL-Bedingung mit einem Vergleichsoperator, um einen einzelnen Wert mit den von der Unterabfrage produzierten Datenwerten zu vergleichen.

Syntax

expression compare [**NOT**] **ALL** (*subquery*)

Parameter

compare:

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

Bemerkungen

UltraLite verwendet den angegebenen Vergleichsoperator, um den Testwert mit den einzelnen Datenwerten in der Ergebnismenge zu vergleichen. Wenn alle Vergleiche mit einem TRUE-Ergebnis enden, gibt der ALL-Test den Wert TRUE zurück.

Siehe auch

- „Unterabfragen und der ALL-Test“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Vergleichsoperatoren“ auf Seite 285

Beispiel

Es werden die Bestellnummern und Kundennummern derjenigen Bestellungen gesucht, 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)
```

ANY-Suchbedingung

Verwenden Sie die ANY-Bedingung mit einem Vergleichsoperator, um einen einzelnen Wert mit den von der Unterabfrage produzierten Spaltenwerten zu vergleichen.

Syntax 1

expression compare [**NOT**] **ANY** (*subquery*)

Syntax 2

expression = **ANY** (*subquery*)

Parameter

compare:

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

Bemerkungen

UltraLite verwendet den angegebenen Vergleichsoperator, um den Testwert mit den einzelnen Datenwerten in der Spalte zu vergleichen. Wenn irgendeiner (any) der Vergleiche das Ergebnis TRUE ausgibt, wird vom ANY-Test der Wert TRUE übergeben.

Syntax 1 Sie ist TRUE, wenn *expression* gleich einem Wert im Ergebnis der Unterabfrage ist, und FALSE, wenn der Ausdruck nicht NULL und nicht gleich einem der von der Unterabfrage zurückgegebenen Werte ist. Die ANY-Bedingung ist UNKNOWN, wenn *expression* NULL ist, außer wenn das Ergebnis der Unterabfrage keine Zeilen enthält, da in diesem Fall die Bedingung immer FALSE wäre.

Siehe auch

- „Unterabfragen und der ANY-Test“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]
- „Vergleichsoperatoren“ auf Seite 285

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)
```

BETWEEN-Suchbedingung

Gibt einen Inklusivbereich an, in dem nach dem niedrigeren Wert und dem höheren Wert sowie den davon begrenzten Werten gesucht wird.

Syntax

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

Bemerkungen

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

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

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

Beispiel

Es sollen alle Preise, die unter 10 oder über 15 Dollar liegen, aufgelistet werden.

```
SELECT Name, UnitPrice
FROM Products
WHERE UnitPrice NOT BETWEEN 10 AND 15
```

EXISTS-Suchbedingung

Prüft, ob eine Unterabfrage Zeilen in Abfrageergebnissen ergibt

Syntax

```
[ NOT ] EXISTS ( subquery )
```

Bemerkungen

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

Sie können die Logik der EXISTS-Bedingung mit der Form NOT EXISTS umkehren. In diesem Fall gibt der Test TRUE zurück, wenn die Unterabfrage keine Zeilen ergibt, und FALSE, wenn Zeilen gefunden werden.

Beispiel

Es werden die Kunden aufgelistet, die nach dem 13. Juli 2001 Bestellungen aufgegeben haben.

```
SELECT GivenName, Surname
FROM Customers
WHERE EXISTS (
  SELECT *
  FROM SalesOrders
  WHERE (OrderDate > '2001-07-13') AND
        (Customers.ID = SalesOrders.CustomerID))
```

IN-Suchbedingung

Überprüft die Mitgliedschaft, indem ein Wert in der Hauptabfrage mit einem anderen Wert in der Unterabfrage durchsucht wird

Syntax

expression [**NOT**] **IN**
{ (*subquery*) | (*value-expr*, ...) }

Parameter

value-expr ist ein Ausdruck, der nur einen Wert annimmt, bei dem es sich um eine Zeichenfolge, eine Zahl, ein Datum oder einen anderen SQL-Datentyp handeln kann.

Bemerkungen

Eine IN-Bedingung ohne das Schlüsselwort NOT wird nach den folgenden Regeln bewertet:

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

Sie können die Logik der IN-Bedingung mit der Form NOT IN umkehren.

Die Suchbedingung *expression* **IN** (*values*) ist identisch mit der Suchbedingung *expression* = **ANY** (*values*). Die Suchbedingung *expression* **NOTIN** (*values*) ist identisch mit der Suchbedingung *expression* <> **ALL** (*values*).

Beispiel

Es wird der Firmenname und der Staat für Kunden gesucht, die in den folgenden kanadischen Provinzen leben: Ontario, Manitoba und Québec.

```
SELECT CompanyName , Province
FROM Customers
WHERE State IN( 'ON', 'MB', 'PQ' )
```

LIKE-Suchbedingung

Syntax

Dies ist die Syntax für die LIKE-Suchbedingung:

expression [**NOT**] **LIKE** *pattern*

Parameter

- **expression** Die zu durchsuchende Zeichenfolge.
- **pattern** Das Muster, nach dem in *expression* gesucht werden soll.

Bemerkungen

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

expression mit *pattern* übereinstimmt (wenn nicht NOT angegeben wurde). Falls entweder *expression* oder *sample* der NULL-Wert ist, wird diese Suchbedingung mit UNKNOWN ausgewertet.

Das Schlüsselwort NOT kehrt die Bedeutung der Suchbedingung um, lässt UNKNOWN aber unverändert.

expression und *sample* werden als CHAR Zeichenfolgen interpretiert. *sample* kann eine beliebige Anzahl von Platzhalterzeichen aus der folgenden Tabelle enthalten:

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

Beispiel

Die folgende Suchbedingung ergibt TRUE für eine Zeile, in der der Spaltenname (column-name) mit dem Buchstaben a beginnt und den Buchstaben b als zweitletztes Zeichen hat:

```
SELECT * FROM table-name WHERE column-name LIKE 'a%b_'
```

Unterschiedliche Methoden für die Verwendung der LIKE-Suchbedingung

Suche nach	Beispiel	Zusätzliche Informationen
Eine Zeichenmenge	LIKE 'sm[iy]th'	Die Suche einer Menge von Zeichen wird festgelegt, indem die Zeichen innerhalb von eckigen Klammern aufgelistet werden. In diesem Beispiel findet die Suchbedingung <i>smith</i> und <i>smyth</i> .
Ein Zeichenbereich	LIKE '[a-r]ough'	<p>Die Suche nach einem Bereich von Zeichen wird festgelegt, indem die Bereichsgrenzen innerhalb von eckigen Klammern durch einen Bindestrich getrennt angegeben werden. In diesem Beispiel findet die Suchbedingung <i>bough</i> und <i>rough</i>, aber nicht <i>tough</i>.</p> <p>Der Bereich der Zeichen [a-z] wird als "größer als oder gleich mit a und kleiner als oder gleich mit z" interpretiert, wobei die Vorgänge "größer als" und "kleiner als" innerhalb der Kollation der Datenbank ausgeführt werden.</p> <p>Die untere Bereichsgrenze muss vor der oberen Bereichsgrenze stehen. Beispiel: [z-a] findet nichts, weil kein Zeichen zum Bereich [z-a] gehört.</p>

Suche nach	Beispiel	Zusätzliche Informationen
Bereiche und Mengen kombiniert	<code>... LIKE '[a-rt]ough'</code>	<p>Sie können Bereiche und Mengen innerhalb eckiger Klammern kombinieren. In diesem Beispiel findet <code>... LIKE '[a-rt]ough'</code> <code>bough</code>, <code>rough</code> und <code>tough</code>.</p> <p>Das Muster "[a-rt]" wird als "Genau ein Zeichen, das entweder im Bereich von a bis inklusive r liegt oder t ist" interpretiert.</p>
Ein Zeichen außerhalb des Bereichs	<code>... LIKE '[^a-r]ough'</code>	<p>Mit dem Einschaltungszeichen (^) wird ein Bereich von Zeichen festgelegt, der von einer Suche ausgeschlossen ist. In diesem Beispiel findet <code>LIKE '[^a-r]ough'</code> die Zeichenfolge <code>tough</code>, aber nicht <code>rough</code> oder <code>bough</code>.</p> <p>Das Einschaltungszeichen negiert den Rest des Inhalts der eckigen Klammern. Der Klammerausdruck <code>[^a-rt]</code> wird als "Genau ein Zeichen, das nicht im Bereich von a bis inklusive r liegt oder nicht t ist" interpretiert.</p>
Suchmuster mit nachgestellten Leerzeichen	<code>'90 '</code> , <code>'90[]'</code> und <code>'90_'</code>	<p>Wenn Ihr Suchmuster nachgestellte Leerzeichen enthält, bringt der Datenbankserver das Suchmuster nur mit Werten in Übereinstimmung, die Leerzeichen enthalten – es wird nicht mit Leerzeichen aufgefüllt. Die Suchmuster <code>"90 "</code>, <code>"90[]"</code> und <code>"90_"</code> stimmen z.B. mit dem Ausdruck <code>"90 "</code> überein, nicht aber mit <code>"90"</code>, auch wenn der geprüfte Wert in einer CHAR- oder VARCHAR-Spalte steht, die drei oder mehr Zeichen breit ist.</p>

Spezialfälle von Bereichen und Mengen

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

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

Berücksichtigung von Groß- und Kleinschreibung und Vorgehensweise beim Vergleichen

Wenn die Kollation der Datenbank Groß-/Kleinschreibung berücksichtigt, tut die Suchbedingung dies ebenfalls. Wenn Sie eine Suche ohne Berücksichtigung der Groß- und Kleinschreibung mit einer

Kollation durchführen möchten, in der die Groß- und Kleinschreibung berücksichtigt wird, müssen Sie Groß- und Kleinbuchstaben eingeben. Die folgende Suchbedingung ergibt zum Beispiel für die Zeichenfolgen Bough, rough und TOUGH den Wert TRUE.

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

Operatoren in UltraLite

Operatoren dienen zur Berechnung von Vergleichen, die ihrerseits als Operanden in einem Ausdruck einer höheren Ebene verwendet werden können.

UltraLite SQL unterstützt die folgenden Typen von Operatoren:

- Vergleichsoperator zur Auswertung und Rückgabe eines Ergebnisses unter Verwendung von einem (unär) oder zwei (binär) Vergleichsoperanden. Vergleiche ergeben die drei bekannten logischen Werte TRUE, FALSE und UNKNOWN.
- Arithmetische Operatoren zur Auswertung und Rückgabe einer Ergebnismenge für alle Gleitkomma-, Dezimal- und Ganzzahlwerte.
- Zeichenfolgenoperatoren zum Verketteten von zwei Zeichenfolgenwerten. Beispiel: "meine" + "Zeichenfolge" gibt die Zeichenfolge "meine Zeichenfolge" zurück.
- Bit-Operatoren zur Auswertung und Aktivierung bzw. Deaktivierung spezifischer Bits in der internen Repräsentation einer Ganzzahl.
- Logische Operatoren, die Suchbedingungen auswerten. Logische Auswertungen ergeben die drei bekannten logischen Werte TRUE, FALSE und UNKNOWN.

Es wird der normale Vorrang von mathematischen Operationen angewendet.

Siehe auch

- „Vorrang der Operatoren“ auf Seite 295
- „Vergleichsoperatoren“ auf Seite 285
- „Arithmetische Operatoren“ auf Seite 293
- „Zeichenfolgenoperatoren“ auf Seite 294
- „Bit-Operatoren“ auf Seite 294
- „Logische Operatoren“ auf Seite 286

Arithmetische Operatoren

Mit arithmetischen Operatoren können Sie Berechnungen durchführen.

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

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

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

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

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

Ausdruck % Ausdruck Modulo ergibt den ganzzahligen Rest nach einer Division, die zwei ganze Zahlen betrifft. Beispiel: 21 % 11 = 10, da 21 geteilt durch 11 den Ganzzahlwert 1 mit einem Rest von 10 ergibt. Falls ein Ausdruck NULL ist, ist das Ergebnis NULL.

Siehe auch

- [Arithmetische Vorgänge \[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#)

Zeichenfolgenoperatoren

Mit Zeichenfolgenoperatoren können Sie Zeichenfolgen verketteten, ausgenommen LONGVARCHAR- und LONGBINARY-Datentypen.

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

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

Die folgende Abfrage ergibt zum Beispiel den ganzzahligen Wert **579**:

```
SELECT 123 + 456
```

Die folgende Abfrage ergibt dagegen die Zeichenfolge **123456**:

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

Mit den Funktionen CAST und CONVERT können Sie Datentypen explizit konvertieren.

Bit-Operatoren

Bit-Operatoren führen Bitmanipulationen zwischen zwei Ausdrücken aus. Die folgenden Operatoren können bei Ganzzahl-Datentypen in UltraLite verwendet werden.

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

Die Bit-Operatoren &, | und ~ sind mit den logischen Operatoren AND, OR und NOT nicht austauschbar. Die Bit-Operatoren arbeiten mit Ganzzahlwerten, die in Bit dargestellt werden.

Beispiel

Die folgende Anweisung wählt Zeilen aus, in denen die angegebenen Bits gesetzt sind.

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

Vorrang der Operatoren

Der Vorrang bei Operatoren in Ausdrücken wird in der untenstehenden Liste dargestellt. Ausdrücke in Klammern werden zuerst ausgewertet, dann Multiplikation und Division vor Addition und Subtraktion. Zeichenfolgenverkettung erfolgt nach Addition und Subtraktion. Die Operatoren am Anfang der Liste werden vor denen weiter unten berücksichtigt.

Tipp

Legen Sie die Reihenfolge der Operationen in UltraLite explizit fest, anstatt sich auf den Vorrang der Operatoren zu verlassen. Benutzen Sie daher bei der Verwendung mehrerer Operatoren in einem Ausdruck Klammern, um die Reihenfolge der Operationen eindeutig zu ordnen.

1. Namen, Funktionen, Konstante, IF-Ausdrücke, CASE-Ausdrücke
2. ()
3. unäre Operatoren (Operatoren, die einen einzelnen Operanden benötigen): +, -
4. ~
5. &, |, ^
6. *, /, %
7. +, -
8. ||
9. Vergleiche: >, <, <>, !=, <=, >=, [NOT] BETWEEN, [NOT] IN, [NOT] LIKE
10. Vergleiche: IS [NOT] TRUE, FALSE, UNKNOWN
11. NOT
12. AND
13. ODER

Variable in UltraLite

Es ist nicht möglich, SQL-Variable (einschließlich globale Variable) in UltraLite-Anwendungen zu verwenden.

UltraLite, SQLDatentypen

UltraLite unterstützt eine Untermenge der Datentypen, die in SQL Anywhere verfügbar sind.

Zeichendatentypen

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

UltraLite unterstützt die Datentypen CHAR, VARCHAR und LONG VARCHAR, die in einem Einbyte- oder Mehrbyte-Zeichensatz gespeichert werden und häufig so gewählt werden, dass sie der Primärsprache bzw. den gespeicherten Sprachen der Datenbank möglichst weitgehend entsprechen.

Speicherung

Feste Zeichentypen, z.B. VARCHAR, werden in die Zeile eingebettet, während lange Zeichentypen, z.B. LONG VARCHAR, separat gespeichert werden.

Berücksichtigen Sie die Seitengröße, wenn Sie eine Tabelle mit vielen Spalten von großen festen Typen erstellen. Eine vollständige Zeile muss auf eine Seite passen und bestimmte feste Zeichenspaltentypen werden zusammen mit einer Zeile gespeichert. Eine Datenbank mit einer Seitengröße von 1000 kann beispielsweise keine Zeichenwerte aufnehmen, die größer sind als 1000, weil diese nicht auf die Seite passen.

Siehe auch

- [„CREATE TABLE-Anweisung \[UltraLite\]“ auf Seite 438](#)

CHAR-Datentyp

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

Syntax

CHAR [(*max-length*)]

Bemerkungen

CHAR ist eine als VARCHAR implementierte Domäne.

Siehe auch

- [„VARCHAR-Datentyp“ auf Seite 297](#)

LONG VARCHAR-Datentyp

Der LONG VARCHAR-Datentyp speichert Zeichendaten von beliebiger Länge.

Syntax

LONG VARCHAR

Bemerkungen

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

Indizes können nicht für einen LONG VARCHAR-Typ erstellt werden.

Ein LONG VARCHAR-Typ kann nur in den Funktionen LENGTH und CAST verwendet werden.

Siehe auch

- „CHAR-Datentyp“ auf Seite 296
- „VARCHAR-Datentyp“ auf Seite 297
- „LENGTH-Funktion [Zeichenfolge]“ auf Seite 365
- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 330

VARCHAR-Datentyp

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

Syntax

VARCHAR [(*max-length*)]

Parameter

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

UltraLite-Datenbanken unterstützen nur Bytelänge-Semantik. Nicht-englische Zeichen können bis zu 3 Byte Speicherplatz erfordern.

UltraLite Java Edition-Datenbanken unterstützen nur Zeichenlängensemantik. *max-length* kann bis zu 32767 Zeichen betragen.

Bemerkungen

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

UltraLite komprimiert Daten so weit wie möglich. Wenn ein VARCHAR-Wert nicht die durch *max-length* angegebene Anzahl an Byte erfordert, wird nur die Anzahl von Byte verwendet, die zum Speichern des Werts erforderlich ist.

Vorsicht

Obwohl es möglich ist, eine Tabelle mit einer VARCHAR-Spalte zu erstellen, bei der die *max-length* die Seitengröße überschreitet, tritt ein Fehler auf, wenn Sie einen Wert einfügen, dessen Länge die Seitengröße überschreitet.

Beim Auswerten von Ausdrücken beträgt die Maximallänge eines temporären Zeichenwerts 2048 Byte.

Siehe auch

- „CHAR-Datentyp“ auf Seite 296
- „LONG VARCHAR-Datentyp“ auf Seite 297

Nummerische Datentypen

Nummerische Datentypen speichern numerische Daten.

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

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

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

Kompatibilität

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

Der FLOAT (*p*)-Datentyp ist ein Synonym für REAL oder DOUBLE, abhängig vom Wert *p*.

Hinweise zum Ändern der Standardwerte durch das Einstellen von Datenbankoptionen finden Sie unter „UltraLite-Erstellungsparameter precision“ auf Seite 157 und „UltraLite-Erstellungsparameter scale“ auf Seite 158.

BIGINT-Datentyp

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

Syntax

[UNSIGNED] BIGINT

Bemerkungen

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

Ein BIGINT-Wert erfordert 8 Byte Speicherplatz.

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

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

Standardmäßig ist der Datentyp mit Vorzeichen (SIGNED).

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

Siehe auch

- „BIT-Datentyp“ auf Seite 299
- „INTEGER-Datentyp“ auf Seite 302
- „SMALLINT-Datentyp“ auf Seite 304
- „TINYINT-Datentyp“ auf Seite 305
- „UltraLite - Numerische Funktionen“ auf Seite 320
- „UltraLite-Aggregatfunktionen“ auf Seite 317

BIT-Datentyp

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

Syntax

BIT

Bemerkungen

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

Standardmäßig ist beim BIT-Datentyp NULL nicht zulässig.

Der BIT-Wert erfordert 1 Bit Speicherplatz.

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

Siehe auch

- „BIGINT-Datentyp“ auf Seite 298
- „INTEGER-Datentyp“ auf Seite 302
- „SMALLINT-Datentyp“ auf Seite 304
- „TINYINT-Datentyp“ auf Seite 305
- „UltraLite - Numerische Funktionen“ auf Seite 320
- „UltraLite-Aggregatfunktionen“ auf Seite 317

DECIMAL-Datentyp

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

Syntax

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

Parameter

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

Sie können die Standardwerte ändern, indem Sie den entsprechenden Erstellungsparameter festlegen.

Bemerkungen

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

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

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

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

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

Hinweis

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

Siehe auch

- „FLOAT-Datentyp“ auf Seite 302
- „REAL-Datentyp“ auf Seite 304
- „DOUBLE-Datentyp“ auf Seite 301
- „NUMERIC-Datentyp“ auf Seite 303
- „UltraLite - Numerische Funktionen“ auf Seite 320
- „UltraLite-Aggregatfunktionen“ auf Seite 317
- „UltraLite-Erstellungsparameter precision“ auf Seite 157
- „UltraLite-Erstellungsparameter scale“ auf Seite 158

DOUBLE-Datentyp

Der DOUBLE-Datentyp speichert Gleitkommazahlen mit doppelter Genauigkeit.

Syntax

DOUBLE

Bemerkungen

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

DOUBLE-Werte erfordern 8 Byte Speicherplatz.

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

Siehe auch

- „FLOAT-Datentyp“ auf Seite 302
- „REAL-Datentyp“ auf Seite 304
- „DECIMAL-Datentyp“ auf Seite 300
- „NUMERIC-Datentyp“ auf Seite 303
- „UltraLite - Numerische Funktionen“ auf Seite 320
- „UltraLite-Aggregatfunktionen“ auf Seite 317
- „Konvertierungen von numerischen Datentypen“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

FLOAT-Datentyp

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

Syntax

FLOAT [(*precision*)]

Parameter

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

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

Bemerkungen

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

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

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

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

Siehe auch

- „DOUBLE-Datentyp“ auf Seite 301
- „REAL-Datentyp“ auf Seite 304
- „DECIMAL-Datentyp“ auf Seite 300
- „NUMERIC-Datentyp“ auf Seite 303
- „UltraLite - Numerische Funktionen“ auf Seite 320
- „UltraLite-Aggregatfunktionen“ auf Seite 317

INTEGER-Datentyp

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

Syntax

[UNSIGNED] INTEGER

Bemerkungen

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

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

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

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

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

Siehe auch

- „BIGINT-Datentyp“ auf Seite 298
- „BIT-Datentyp“ auf Seite 299
- „SMALLINT-Datentyp“ auf Seite 304
- „TINYINT-Datentyp“ auf Seite 305
- „UltraLite - Numerische Funktionen“ auf Seite 320
- „UltraLite-Aggregatfunktionen“ auf Seite 317

NUMERIC-Datentyp

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

Syntax

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

Bemerkung

NUMERIC ist eine als DECIMAL implementierte Domäne.

Siehe auch

- „DECIMAL-Datentyp“ auf Seite 300

REAL-Datentyp

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

Syntax

REAL

Bemerkungen

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

REAL-Werte erfordern 4 Byte Speicherplatz.

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

Siehe auch

- „DOUBLE-Datentyp“ auf Seite 301
- „FLOAT-Datentyp“ auf Seite 302
- „DECIMAL-Datentyp“ auf Seite 300
- „NUMERIC-Datentyp“ auf Seite 303
- „UltraLite - Numerische Funktionen“ auf Seite 320
- „UltraLite-Aggregatfunktionen“ auf Seite 317

SMALLINT-Datentyp

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

Syntax

[UNSIGNED] SMALLINT

Bemerkungen

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

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

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

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

Zeichenfolge unzulässige Zeichen enthält oder die Zeichenfolge nicht als Ganzzahlwert dekodiert werden kann.

Siehe auch

- „BIGINT-Datentyp“ auf Seite 298
- „BIT-Datentyp“ auf Seite 299
- „INTEGER-Datentyp“ auf Seite 302
- „TINYINT-Datentyp“ auf Seite 305
- „UltraLite - Numerische Funktionen“ auf Seite 320
- „UltraLite-Aggregatfunktionen“ auf Seite 317

TINYINT-Datentyp

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

Syntax

TINYINT

Bemerkungen

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

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

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

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

Siehe auch

- „BIGINT-Datentyp“ auf Seite 298
- „BIT-Datentyp“ auf Seite 299
- „INTEGER-Datentyp“ auf Seite 302
- „SMALLINT-Datentyp“ auf Seite 304
- „UltraLite - Numerische Funktionen“ auf Seite 320
- „UltraLite-Aggregatfunktionen“ auf Seite 317

Datentypen für Datum und Uhrzeit

Die folgende Liste gibt einen kurzen Überblick, wie Datumsangaben verarbeitet werden:

- Für alle zulässigen arithmetischen und logischen Operationen mit Datumsangaben werden immer richtige Werte zurückgegeben, unabhängig davon, ob die berechneten Werte über Jahrhundertgrenzen hinwegreichen.
- Die interne Speicherung von Datumsangaben bezieht immer explizit den Jahrhundertteil eines Jahreswerts ein.
- Datumswerte können immer im vollen Jahrhundertformat ausgegeben werden.

DATE-Datentyp

Der DATE-Datentyp speichert Kalenderdatumsangaben, z.B. Jahr, Monat und Tag.

Syntax

DATE

Bemerkungen

Das Format, in dem DATE-Werte von Anwendungen als Zeichenfolgen abgerufen werden, wird durch den `date_format`-Erstellungsparameter gesteuert. Ein DATE-Wert, der den 19. Juli 2010 darstellt, kann beispielsweise als "2010/07/19" oder als "Jul 19, 2010" an eine Anwendung zurückgegeben werden, je nach `date_format`-Erstellungsparameter.

Ein DATE-Wert benötigt 4 Byte Speicherplatz.

Siehe auch

- „CURRENT TIME-Spezialwert“ auf Seite 274
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 275
- „UltraLite-Datums- und Zeitfunktionen“ auf Seite 317
- Datumsformate [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 343
- „UltraLite-Erstellungsparameter date_format“ auf Seite 146
- „date_format-Option der UltraLite Java Edition“ auf Seite 248
- „UltraLite-Erstellungsparameter date_order“ auf Seite 149
- „date_order-Option der UltraLite Java Edition“ auf Seite 248
- „DATETIME-Datentyp“ auf Seite 307
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 348
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 362
- „UltraLite-Erstellungsparameter nearest_century“ auf Seite 153
- „nearest_century-Option der UltraLite Java Edition“ auf Seite 249
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 379
- „TIME-Datentyp“ auf Seite 307
- „TIMESTAMP-Datentyp“ auf Seite 308
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 275
- „TIMESTAMP WITH TIME ZONE-Datentyp“ auf Seite 309

DATETIME-Datentyp

DATETIME speichert Datums- und Uhrzeitangaben.

Syntax

DATETIME

Bemerkungen

DATETIME ist eine als TIMESTAMP implementierte Domäne.

Siehe auch

- „TIMESTAMP-Datentyp“ auf Seite 308

TIME-Datentyp

Der TIME-Datentyp speichert die Uhrzeit mit Stunden, Minuten, Sekunden und Sekundenbruchteil.

Syntax

TIME

Bemerkungen

Das Format, in dem TIME-Werte von Anwendungen als Zeichenfolgen abgerufen werden, wird durch den time_format-Erstellungsparameter gesteuert. Der TIME-Wert 23:59:59.999999 kann beispielsweise

als 23:59:59, 23:59:59.999 oder 23:59:59.999999 an eine Anwendung zurückgegeben werden, je nach `time_format`-Erstellungsparameter.

Ein TIME-Wert benötigt 8 Byte Speicherplatz.

Siehe auch

- [Zeitformate \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- [„CURRENT TIME-Spezialwert“ auf Seite 274](#)
- [„CURRENT TIMESTAMP-Spezialwert“ auf Seite 275](#)
- [„CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 276](#)
- [„UltraLite-Datums- und Zeitfunktionen“ auf Seite 317](#)
- [„DATE-Datentyp“ auf Seite 306](#)
- [„DATETIME-Datentyp“ auf Seite 307](#)
- [„DATE-Funktion \[Datum und Uhrzeit\]“ auf Seite 343](#)
- [„DATETIME-Funktion \[Datum und Uhrzeit\]“ auf Seite 348](#)
- [„Ausdrücke in UltraLite“ auf Seite 277](#)
- [„GETDATE-Funktion \[Datum und Uhrzeit\]“ auf Seite 356](#)
- [„ISDATE-Funktion \[Datentypkonvertierung\]“ auf Seite 362](#)
- [„NOW-Funktion \[Datum und Uhrzeit\]“ auf Seite 379](#)
- [„TIMESTAMP-Datentyp“ auf Seite 308](#)
- [„TIMESTAMP WITH TIME ZONE-Datentyp“ auf Seite 309](#)
- [„UltraLite-Erstellungsparameter `timestamp_format`“ auf Seite 162](#)
- [„`timestamp_format`-Option der UltraLite Java Edition“ auf Seite 251](#)
- [„UltraLite-Erstellungsparameter `timestamp_with_time_zone_format`“ auf Seite 166](#)
- [„`timestamp_with_time_zone_format`-Option der UltraLite Java Edition“ auf Seite 252](#)

TIMESTAMP-Datentyp

Der TIMESTAMP-Datentyp speichert einen Zeitpunkt mit Jahr, Monat, Tag, Stunde, Minute, Sekunde und Sekundenbruchteil, auf sechs Dezimalstellen genau.

Syntax

TIMESTAMP

Bemerkungen

Das Format, in dem TIMESTAMP-Werte von Anwendungen als Zeichenfolgen abgerufen werden, wird durch den `timestamp_format`-Erstellungsparameter gesteuert. Der TIMESTAMP-Wert 2010/04/01T23:59:59.999999 kann beispielsweise als "2010/04/01 23:59:59" oder als "April 1, 2010 23:59:59.999999" an eine Anwendung zurückgegeben werden, je nach `timestamp_format`-Erstellungsparameter.

Der TIMESTAMP-Wert benötigt 8 Byte Speicherplatz.

Obwohl der Bereich der möglichen Datumsangaben beim TIMESTAMP-Datentyp derselbe ist wie beim DATE-Typ (abgedeckte Jahre 0001 bis 9999), liegt der nützliche Bereich der TIMESTAMP-Datumstypen zwischen 1600-02-28 23:59:59 und 7911-01-01 00:00:00. Vor und nach diesem Bereich wird der Stunden- und Minutenteil des TIMESTAMP-Werts nicht gespeichert.

Beim Konvertieren eines `TIMESTAMP`-Werts in `TIMESTAMP WITH TIME ZONE` wird im Endergebnis der lokale Zeitzonen-Offset des Systems verwendet.

Siehe auch

- „Möglichkeiten zum Senden von Datums- und Zeitangaben an die Datenbank“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „`CURRENT TIME`-Spezialwert“ auf Seite 274
- „`CURRENT TIMESTAMP`-Spezialwert“ auf Seite 275
- „`CURRENT UTC TIMESTAMP`-Spezialwert“ auf Seite 276
- „UltraLite-Datums- und Zeitfunktionen“ auf Seite 317
- „`DATE`-Datentyp“ auf Seite 306
- „`DATETIME`-Datentyp“ auf Seite 307
- „`DATE`-Funktion [Datum und Uhrzeit]“ auf Seite 343
- „`DATETIME`-Funktion [Datum und Uhrzeit]“ auf Seite 348
- „UltraLite-Erstellungsparameter `date_order`“ auf Seite 149
- „`date_order`-Option der UltraLite Java Edition“ auf Seite 248
- „Ausdrücke in UltraLite“ auf Seite 277
- „`GETDATE`-Funktion [Datum und Uhrzeit]“ auf Seite 356
- „`ISDATE`-Funktion [Datentypkonvertierung]“ auf Seite 362
- „UltraLite-Erstellungsparameter `nearest_century`“ auf Seite 153
- „`nearest_century`-Option der UltraLite Java Edition“ auf Seite 249
- „`NOW`-Funktion [Datum und Uhrzeit]“ auf Seite 379
- „`TIME`-Datentyp“ auf Seite 307
- „`TIMESTAMP`-Datentyp“ auf Seite 308
- „`TIMESTAMP WITH TIME ZONE`-Datentyp“ auf Seite 309
- „UltraLite-Erstellungsparameter `timestamp_format`“ auf Seite 162
- „`timestamp_format`-Option der UltraLite Java Edition“ auf Seite 251
- „UltraLite-Erstellungsparameter `timestamp_with_time_zone_format`“ auf Seite 166
- „`timestamp_with_time_zone_format`-Option der UltraLite Java Edition“ auf Seite 252

TIMESTAMP WITH TIME ZONE-Datentyp

Der `TIMESTAMP WITH TIME ZONE`-Datentyp speichert einen Zeitpunkt mit Zeitzonen-Offset.

Syntax

`TIMESTAMP WITH TIME ZONE`

Bemerkungen

Der `TIMESTAMP WITH TIME ZONE`-Wert enthält Jahr, Monat, Tag, Stunde, Minute, Sekunde, Sekundenbruchteil und die Anzahl der Minuten vor oder nach der Coordinated Universal Time (UTC). Der Sekundenbruchteil wird mit sechs Dezimalstellen gespeichert.

Das Format, in dem `TIMESTAMP WITH TIME ZONE`-Werte von Anwendungen als Zeichenfolgen abgerufen werden, wird durch den `timestamp_with_time_zone_format`-Erstellungsparameter gesteuert. Der `TIMESTAMP WITH TIME ZONE`-Wert 2010/04/01T23:59:59.999999-6:00 kann beispielsweise als "2010/04/01 23:59:59 -06:00" oder als "April 1, 2010 23:59:59.999999 -06:00" an eine Anwendung zurückgegeben werden, je nach `timestamp_with_time_zone_format`-Erstellungsparameter.

Ein **TIMESTAMP WITH TIME ZONE**-Wert benötigt 10 Byte Speicherplatz.

Obwohl der Bereich der möglichen Datumsangaben beim **TIMESTAMP WITH TIME ZONE**-Datentyp derselbe ist wie beim **DATE**-Typ (abgedeckte Jahre 0001 bis 9999), liegt der nützliche Bereich der **TIMESTAMP WITH TIME ZONE**-Datumstypen zwischen 1600-02-28 23:59:59 und 7911-01-01 00:00:00. Vor und nach diesem Bereich wird der Stunden- und Minutenteil des **TIMESTAMP WITH TIME ZONE**-Werts nicht gespeichert.

Zwei **TIMESTAMP WITH TIME ZONE**-Werte gelten als identisch, wenn sie in UTC denselben Zeitpunkt darstellen, und zwar unabhängig vom angewendeten Zeitzonen-Offset. Zum Beispiel liefert die folgende Anweisung **Yes**, weil die Ergebnisse als identisch gelten:

```
SELECT IF CAST('2009-07-15 08:00:00 -08:00' AS TIMESTAMP WITH TIME ZONE) =  
        CAST('2009-07-15 11:00:00 -05:00' AS TIMESTAMP WITH TIME ZONE)  
        THEN 'Yes'  
        ELSE 'No'  
END IF;
```

Wenn Sie den Zeitzonen-Offset bei einem **TIMESTAMP WITH TIME ZONE**-Wert weglassen, wird standardmäßig der aktuelle UTC-Offset für den Client verwendet, unabhängig davon, ob der Zeitstempel Datum und Uhrzeit in Standard- oder Sommerzeit darstellt. Beispiel: Wenn sich der Client in der Eastern Standard-Zeitzone befindet und die folgende Anweisung ausführt, während Sommerzeit aktiv ist, wird ein Zeitstempel für die Atlantic Standard-Zeitzone (UTC -4 Stunden) zurückgegeben.

```
SELECT CAST('2009/01/30 12:34:55' AS TIMESTAMP WITH TIME ZONE)
```

- **Vergleich von **TIMESTAMP WITH TIME ZONE** mit anderen Datentypen** Der Vergleich von **TIMESTAMP WITH TIME ZONE**-Werten mit Zeitstempeln ohne Zeitzonen wird nicht empfohlen, weil sich der standardmäßige Zeitzonen-Offset des Clients nach dem geografischen Standort des Clients und nach der Jahreszeit richtet.
- **Konvertierung in oder aus **TIMESTAMP WITH TIME ZONE**** Beim Konvertieren eines **TIMESTAMP**-Werts in **TIMESTAMP WITH TIME ZONE** wird die Zeitzone des Clients für den Zeitzonen-Offset im Ergebnis herangezogen. Mit anderen Worten, der Wert gilt für die Verbindung als "lokal". Beim Konvertieren eines **TIMESTAMP WITH TIME ZONE**-Werts in **TIMESTAMP** wird der Offset verworfen. Konvertierungen in oder aus Typen, bei denen es sich nicht um Zeichenfolgen, Datumsangaben oder Datums- und Uhrzeitangaben handelt, werden nicht unterstützt.

Siehe auch

- „Vergleiche von Datumsangaben und Uhrzeiten“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Möglichkeiten zum Senden von Datums- und Zeitangaben an die Datenbank“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CURRENT TIME-Spezialwert“ auf Seite 274
- „CURRENT TIMESTAMP-Spezialwert“ auf Seite 275
- „CURRENT UTC TIMESTAMP-Spezialwert“ auf Seite 276
- „UltraLite-Datums- und Zeitfunktionen“ auf Seite 317
- „DATE-Datentyp“ auf Seite 306
- „DATETIME-Datentyp“ auf Seite 307
- „DATE-Funktion [Datum und Uhrzeit]“ auf Seite 343
- „DATETIME-Funktion [Datum und Uhrzeit]“ auf Seite 348
- „UltraLite-Erstellungsparameter date_order“ auf Seite 149
- „date_order-Option der UltraLite Java Edition“ auf Seite 248
- „Ausdrücke in UltraLite“ auf Seite 277
- „GETDATE-Funktion [Datum und Uhrzeit]“ auf Seite 356
- „ISDATE-Funktion [Datentypkonvertierung]“ auf Seite 362
- „UltraLite-Erstellungsparameter nearest_century“ auf Seite 153
- „nearest_century-Option der UltraLite Java Edition“ auf Seite 249
- „NOW-Funktion [Datum und Uhrzeit]“ auf Seite 379
- „TIME-Datentyp“ auf Seite 307
- „TIMESTAMP-Datentyp“ auf Seite 308
- „TIMESTAMP WITH TIME ZONE-Datentyp“ auf Seite 309
- „UltraLite-Erstellungsparameter timestamp_format“ auf Seite 162
- „timestamp_format-Option der UltraLite Java Edition“ auf Seite 251
- „UltraLite-Erstellungsparameter timestamp_with_time_zone_format“ auf Seite 166
- „timestamp_with_time_zone_format-Option der UltraLite Java Edition“ auf Seite 252

Binärdatentypen

Binäre Datentypen speichern Binärdaten, darunter auch Bilder und andere Informationstypen, die von der Datenbank nicht interpretiert werden.

BINARY-Datentyp

Der BINARY-Datentyp speichert Binärdaten mit einer festgelegten Maximallänge (in Byte).

Syntax

BINARY [(*max-length*)]

Bemerkungen

BINARY ist eine als VARBINARY implementierte Domäne.

Siehe auch

- „VARBINARY-Datentyp“ auf Seite 313

LONG BINARY-Datentyp

Der LONG BINARY-Datentyp speichert Binärdaten von beliebiger Länge.

Syntax

LONG BINARY

Bemerkungen

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

Indizes können nicht für einen LONG BINARY-Typ erstellt werden.

Ein LONG BINARY-Typ kann nur in den Funktionen LENGTH und CAST verwendet werden.

Siehe auch

- „BINARY-Datentyp“ auf Seite 311
- „VARBINARY-Datentyp“ auf Seite 313

UNIQUEIDENTIFIER-Datentyp

Der UNIQUEIDENTIFIER-Datentyp speichert UUID-Werte (auch als GUID-Werte bezeichnet).

Syntax

UNIQUEIDENTIFIER

Bemerkungen

Der UNIQUEIDENTIFIER-Datentyp wird üblicherweise für eine Primärschlüsselspalte oder andere eindeutige Spalten verwendet, um UUID-(Universally Unique Identifier-)Werte aufzunehmen, die Zeilen eindeutig identifizieren. Die NEWID-Funktion generiert UUID-Werte so, dass ein auf einem Computer erzeugter Wert nicht mit einer auf einem anderen Computer erzeugten UUID übereinstimmt. Mit NEWID generierte UNIQUEIDENTIFIER-Werte können daher als Schlüssel in einer Synchronisationsumgebung verwendet werden.

Beispiel:

```
CREATE TABLE T1 (  
    pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),  
    c1 INT );
```

UUID-Werte werden auch als GUID-Werte (Globally Unique Identifier) bezeichnet. UUID-Werte enthalten Bindestriche, um mit anderen RDBMS kompatibel zu sein. Sie können diese Einstellung mithilfe der Funktionen UUIDTOSTR und STRTOUUID ändern.

UNIQUEIDENTIFIER-Werte werden bei Bedarf automatisch zwischen Zeichenfolgenwerten und binären Werten konvertiert.

UNIQUEIDENTIFIER-Werte werden als BINARY(16) gespeichert, aber für Clientanwendungen als BINARY(36) beschrieben. Das gewährleistet, dass der Client, wenn er einen Wert als Zeichenfolge abrufen, genügend Speicher für das Ergebnis zugeordnet hat.

Siehe auch

- „NEWID-Funktion [Verschiedene]“ auf Seite 379
- „UUIDTOSTR-Funktion [Zeichenfolge]“ auf Seite 416
- „STRTOUUID-Funktion [Zeichenfolge]“ auf Seite 407
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

VARBINARY-Datentyp

Der VARBINARY-Datentyp speichert Binärdaten mit einer festgelegten Maximallänge (in Byte).

Syntax

VARBINARY [(*max-length*)]

Parameter

- **max-length** Die maximale Länge des Werts in Byte. Wenn die Länge nicht angegeben wird, ist sie gleich 1.

Der Wert muss im Bereich 1 bis 32767 liegen.

Bemerkungen

Bei Vergleichsvorgängen werden VARBINARY-Werte Byte für Byte miteinander verglichen. Dies ist ein Unterschied zum CHAR-Datentyp, bei dem Werte anhand der Kollationssequenz der Datenbank verglichen werden.

Wenn eine binäre Zeichenfolge als Präfix der anderen fungiert, wird die kürzere Zeichenfolge mit der anderen verglichen, als wäre die kürzere Zeichenfolge mit Nullen aufgefüllt.

Beim Auswerten von Ausdrücken beträgt die Maximallänge eines temporären Zeichenwerts 2048 Byte.

VARBINARY-Werte werden während einer Zeichensatzkonvertierung nicht umgewandelt.

Siehe auch

- „BINARY-Datentyp“ auf Seite 311
- „LONG BINARY-Datentyp“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321
- „Bit-Operatoren“ auf Seite 294

Räumliche Datentypen

Räumliche Daten sind Daten, die die Position, Form und Ausrichtung von Objekten in einem definierten Raum beschreiben. UltraLite stellt Speicher- und Datenverwaltungsfunktionen für räumliche Daten in Form von Punkten bereit, mit denen Sie Informationen wie geografische Standorte und Routinginformationen speichern können. Punkte werden unter Verwendung des **räumlichen Typs** ST_Geometry definiert. Hierbei werden Funktionen und Konstruktoren verwendet, um auf die räumlichen Daten zuzugreifen und sie zu bearbeiten. UltraLite stellt auch eine Reihe von räumlichen SQL-Funktionen zur Kompatibilität mit anderen Produkten bereit.

Ein Punkt kennzeichnet eine einzelne Position in einem Raum. Eine Punktgeometrie hat keine Länge und keinen Bereich. Ein Punkt hat immer eine X- und Y-Koordinate.

In GIS-Daten werden Punkte gewöhnlich verwendet, um Standorte wie Adressen oder geografische Merkmale wie einen Berg darzustellen.

Siehe auch

- „Empfohlene Lektüre zu räumlichen Themen“ [[SQL Anywhere Server - Unterstützung für räumliche Daten](#)]

ST_GEOMETRY-Datentyp

Der ST_Geometry-Typ wird verwendet, um räumliche Daten in Form von Punkten zu speichern.

Funktionen

UltraLite unterstützt die folgenden Funktionen von ST_GEOMETRY:

- „ST_AsBinary-Funktion [Räumlich]“
- „ST_AsExtText-Funktion [Räumlich]“
- „ST_AsText-Funktion [Räumlich]“
- „ST_Distance-Funktion [Räumlich]“
- „ST_Equals-Funktion [Räumlich]“
- „ST_IntersectsRect-Funktion [Räumlich]“
- „ST_Point-Funktion [Räumlich]“
- „ST_PointFromExtText-Funktion [Räumlich]“
- „ST_PointFromText-Funktion [Spatial]“
- „ST_PointFromWKB-Funktion [Spatial]“
- „ST_SRID-Funktion [Räumlich]“
- „ST_X-Funktion [Räumlich]“
- „ST_Y-Funktion [Räumlich]“

Bemerkungen

Der ST_GEOMETRY-Typ ist der maximale übergeordnete Datentyp der Geometriedatentyp-Hierarchie. Der ST_GEOMETRY-Typ unterstützt Methoden, die auf jeden räumlichen Wert angewendet werden können. Der ST_Geometry-Datentyp kann nicht instanziiert werden. Stattdessen ist ein untergeordneter Typ zu instanziiieren. Bei der Arbeit mit Originalformaten (WKT oder WKB) können Sie Methoden wie ST_PointFromText/ST_PointFromWKB verwenden, um den geeigneten konkreten Typ zu instanziiieren, der den Wert im Originalformat darstellt.

Die ST_SRID dieser Methode kann verwendet werden, um das räumliche Bezugssystem abzurufen, das dem Wert zugeordnet ist.

Spalten vom ST_GEOMETRY-Typ oder seiner Untertypen können nicht in einen Primärschlüssel, einen eindeutigen Index oder eine Eindeutigkeits-Integritätsregel einbezogen werden.

Spalten- und Objektdefinitionen

UltraLite stellt einen festen Satz von drei verschiedenen Bezugssystemen bereit, die einer Spalte zugeordnet werden können, während diese erstellt wird. Einzelne Geometrieobjekte können einem beliebigen SRID-Wert zugeordnet werden (mit Ausnahme des nicht definierten Bezugssystems) und können nur in einer Spalte gespeichert werden, die einem übereinstimmenden SRID-Wert oder dem nicht definierten Bezugssystem zugeordnet ist.

Die vordefinierten Bezugssysteme sind:

- **Bezugssystem nicht definiert oder "null"** Hierbei handelt es sich um das Standardbezugssystem, wenn kein SRID-Wert angegeben wird. Es ermöglicht das Vorkommen enthaltener Geometriewerte in jedem gültigen Bezugssystem. Durch dieses Bezugssystem sind "Catch-All"-Spalten zulässig, die keine Bezugssystemkonsistenz zwischen den Geometrieobjekten erzwingen.
- **Standardmäßiges planares Bezugssystem** Diese Spalte wird durch Festlegung eines SRID-Werts von 0 während ihrer Erstellung definiert und kann nur Geometriewerte enthalten, die diesem Bezugssystem zugeordnet sind. Die Werte werden so behandelt, als befänden sie sich in einem planaren 2D-Raum.
- **Geodätisches WGS 84-Bezugssystem** Diese Spalte wird durch Festlegung eines SRID-Werts von 4326 während der Spaltenerstellung definiert und kann nur Geometriewerte enthalten, die diesem Bezugssystem zugeordnet sind. Die Werte werden so behandelt, als befänden sie sich auf der Erdoberfläche und Vorgänge werden dementsprechend durchgeführt.

Hinweis

Ein Punkt in SRID 4326 kann in einer Spalte mit dem Bezugssystem WGS 84 oder mit dem nicht definierten Bezugssystem gespeichert werden, nicht aber im standardmäßigen planaren System.

Es werden keine Transformationen zwischen Bezugssystemen unterstützt.

Beispiel

Im folgenden Beispiel wird eine Tabelle mit einer Spalte erstellt, die dem standardmäßigen planaren Bezugssystem zugeordnet ist, sowie einer Spalte mit einem nicht definierten Bezugssystem:

```
CREATE TABLE T1 (
  V1 INTEGER PRIMARY KEY,
  V2 ST_GEOMETRY(SRID=0),
  V3 ST_GEOMETRY
)
```

Die folgende SQL-Anweisung zeigt, wie Sie Daten in die Tabelle T1 aus dem vorherigen Beispiel einfügen:

```
INSERT INTO T1(V1, V2, V3)
VALUES (1, ST_POINTFROMTEXT('POINT(10 20)', 0), ST_POINT(5, 6, 2163))
```

Benutzerdefinierte Datentypen und ihre Entsprechungen

Im Unterschied zu SQL Anywhere-Datenbanken unterstützt UltraLite keine benutzerdefinierten Datentypen. Die folgende Tabelle listet die Entsprechungen von UltraLite-Datentypen zu integrierten SQL Anywhere-Aliasen auf:

SQL Anywhere-Datentyp	UltraLite-Entsprechung
MONEY	DECIMAL (19,4)
SMALLMONEY	DECIMAL(10,4)
TEXT	LONG VARCHAR
XML	LONG VARCHAR

Siehe auch

- „LONG VARCHAR-Datentyp“ auf Seite 297
- „DECIMAL-Datentyp“ auf Seite 300

UltraLite SQL-Funktionen

Funktionen werden zur Rückgabe von Informationen aus der Datenbank verwendet. Sie sind überall dort zulässig, wo Ausdrücke erlaubt sind.

Wenn in der Dokumentation nichts anderes angegeben ist, wird NULL für eine Funktion zurückgegeben, wenn eines der Argumente NULL ist.

Funktionen verwenden dieselben Syntaxkonventionen wie SQL-Anweisungen

Siehe auch

- „Syntaxkonventionen“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Funktionstypen

In diesem Abschnitt werden die verfügbaren Funktionen nach ihrem Typ gruppiert.

UltraLite unterstützt eine Teilmenge derselben Funktionen wie SQL Anywhere, manchmal mit einigen Unterschieden.

Hinweis

Wenn nicht anders festgelegt, gibt jede Funktion, die Null als Parameter erhält, NULL zurück.

Siehe auch

- „ST_GEOMETRY-Datentyp“ auf Seite 314

UltraLite-Aggregatfunktionen

Aggregatfunktionen fassen Daten über eine Gruppe von Zeilen in der Datenbank zusammen. Die Gruppen werden durch die GROUP BY-Klausel der SELECT-Anweisung gebildet. Aggregatfunktionen sind nur in der Auswahlliste und in den Klauseln HAVING und ORDER BY einer SELECT-Anweisung erlaubt.

Liste der Funktionen

Folgende Aggregatfunktionen stehen zur Verfügung:

- „AVG-Funktion [Aggregat]“
- „COUNT-Funktion [Aggregat]“
- „COUNT_UPLOAD_ROWS-Funktion [Aggregat]“
- „LIST-Funktion [Aggregat]“
- „MAX-Funktion [Aggregat]“
- „MIN-Funktion [Aggregat]“
- „SUM-Funktion [Aggregat]“

UltraLite-Funktionen zur Datentypkonvertierung

Datentypkonvertierungsfunktionen werden benutzt, um Argumente von einem Datentyp in einen anderen zu konvertieren oder um zu testen, ob eine Konvertierung möglich ist.

Liste der Funktionen

Die folgenden Datentypkonvertierungsfunktionen sind verfügbar:

- „CAST-Funktion [Datentypkonvertierung]“
- „CONVERT-Funktion [Datentypkonvertierung]“
- „HEXTOINT-Funktion [Datentypkonvertierung]“
- „INTTOHEX-Funktion [Datentypkonvertierung]“
- „ISDATE-Funktion [Datentypkonvertierung]“

UltraLite-Datums- und Zeitfunktionen

Datums- und Uhrzeitfunktionen führen Vorgänge mit den Datentypen DATE, TIME, TIMESTAMP, und TIMESTAMP WITH TIME ZONE aus.

SQL Anywhere umfasst Kompatibilitätsunterstützung für Datums und Zeittypen von Transact-SQL, einschließlich DATETIME und SMALLDATETIME. Diese Transact-SQL-Datentypen werden als Domänen über den nativen SQL Anywhere-Datentyp TIMESTAMP implementiert.

Angeben von Datumsteilen

Viele Datumsfunktionen benutzen Datumsangaben, die aus **Datumsteilen** zusammengesetzt sind. Die folgende Tabelle zeigt zulässige Werte der Datumsteile.

Bei Datums- und Zeitfunktionen können Sie ein Minuszeichen angeben, um von einem Datum oder einer Uhrzeit zu subtrahieren Sie können z.B. Folgendes ausführen, um einen Zeitstempel von vor 31 Tagen zu erzeugen:

```
SELECT DATEADD(day, -31, NOW());
```

Datumsteil	Abkürzung	Werte
Year	YY	1-9999
Quarter	QQ	1-4
Month	MM	1-12
Week	WK	1-54. Wochen beginnen am Sonntag. Ein Jahr mit 54 Wochen tritt in Schaltjahren auf, die an einem Samstag beginnen.
Day	DD	1-31
Dayofyear	DY	1-366
Weekday	DW	1-7 (Sonntag = 1, ..., Samstag = 7)
Hour	HH	0-23
Minute	MI	0-59
Second	SS	0-59
Millisecond	MS	0-999
Microsecond	MCS oder US	0-999999
Calyearofweek	CYR	1-9999. Das Jahr, in dem die Woche beginnt. Die Woche mit den ersten Tagen des neuen Jahres beginnt möglicherweise schon im Vorjahr, je nachdem, an welchem Wochentag das Jahr beginnt. Jahre, die von Montag bis Donnerstag beginnen, haben keine Tage, die Teil des vorhergehenden Jahres sind, aber Jahre, die von Freitag bis Sonntag beginnen, beginnen ihre erste Woche am ersten Montag des Jahres.
Calweekofyear	CWK	1-53. Die Nummer der Kalenderwoche im Jahr, die das angegebene Datum enthält. Weitere Hinweise zum ISO-Wochensystem und zum Datums- und Zeitstandard ISO 8601 finden Sie unter http://en.wikipedia.org/wiki/ISO_week_date .
Caldayofweek	CDW	1-7. (Montag = 1, ..., Sonntag = 7)
TZOffset	TZ	-840 bis 840

Liste der Funktionen

Die folgenden Datums- und Zeitfunktionen stehen zur Verfügung:

- „DATE-Funktion [Datum und Uhrzeit]“
- „DATEADD-Funktion [Datum und Uhrzeit]“
- „DATEDIFF-Funktion [Datum und Uhrzeit]“
- „DATEFORMAT-Funktion [Datum und Uhrzeit]“
- „DATENAME-Funktion [Datum und Uhrzeit]“
- „DATEPART-Funktion [Datum und Uhrzeit]“
- „DATETIME-Funktion [Datum und Uhrzeit]“
- „DAY-Funktion [Datum und Uhrzeit]“
- „DAYNAME-Funktion [Datum und Uhrzeit]“
- „DAYS-Funktion [Datum und Uhrzeit]“
- „DOW-Funktion [Datum und Uhrzeit]“
- „GETDATE-Funktion [Datum und Uhrzeit]“
- „HOUR-Funktion [Datum und Uhrzeit]“
- „HOURS-Funktion [Datum und Uhrzeit]“
- „MINUTE-Funktion [Datum und Uhrzeit]“
- „MINUTES-Funktion [Datum und Uhrzeit]“
- „MONTH-Funktion [Datum und Uhrzeit]“
- „MONTHNAME-Funktion [Datum und Uhrzeit]“
- „MONTHS-Funktion [Datum und Uhrzeit]“
- „NOW-Funktion [Datum und Uhrzeit]“
- „QUARTER-Funktion [Datum und Uhrzeit]“
- „SECOND-Funktion [Datum und Uhrzeit]“
- „SECONDS-Funktion [Datum und Uhrzeit]“
- „SWITCHOFFSET-Funktion [Datum und Zeit]“
- „TODAY-Funktion [Datum und Uhrzeit]“
- „TODATETIMEOFFSET-Funktion [Datum und Zeit]“
- „WEEKS-Funktion [Datum und Uhrzeit]“
- „YEAR-Funktion [Datum und Uhrzeit]“
- „YEARS-Funktion [Datum und Uhrzeit]“
- „YMD-Funktion [Datum und Uhrzeit]“

Siehe auch

- „Datentypen für Datum und Uhrzeit“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „UltraLite, SQLDatentypen“ auf Seite 296

UltraLite - Verschiedene Funktionen

Verschiedene Funktionen führen Vorgänge auf arithmetischen, Zeichenfolge- oder Datum/Zeit-Ausdrücken, einschließlich der Rückgabewerte anderer Funktionen, aus.

Liste der Funktionen

Folgende verschiedene Funktionen stehen zur Verfügung:

- „ARGN-Funktion [Verschiedene]“
- „COALESCE-Funktion [Verschiedene]“
- „EXPLANATION-Funktion [Verschiedene]“
- „GREATER-Funktion [Verschiedene]“
- „IFNULL-Funktion [Verschiedene]“
- „ISNULL-Funktion [Verschiedene]“
- „LESSER-Funktion [Verschiedene]“
- „NEWID-Funktion [Verschiedene]“
- „NULLIF-Funktion [Verschiedene]“

UltraLite - Numerische Funktionen

Numerische Funktionen führen mathematische Vorgänge auf numerischen Datentypen aus oder geben numerische Informationen zurück.

Liste der Funktionen

Die folgenden numerischen Funktionen stehen zur Verfügung:

- „ABS-Funktion [Numerisch]“
- „ACOS-Funktion [Numerisch]“
- „ASIN-Funktion [Numerisch]“
- „ATAN-Funktion [Numerisch]“
- „ATAN2-Funktion [Numerisch]“
- „CEILING-Funktion [Numerisch]“
- „COS-Funktion [Numerisch]“
- „COT-Funktion [Numerisch]“
- „DEGREES-Funktion [Numerisch]“
- „EXP-Funktion [Numerisch]“
- „FLOOR-Funktion [Numerisch]“
- „LOG-Funktion [Numerisch]“
- „LOG10-Funktion [Numerisch]“
- „MOD-Funktion [Numerisch]“
- „PI-Funktion [Numerisch]“
- „POWER-Funktion [Numerisch]“
- „RADIANS-Funktion [Numerisch]“
- „REMAINDER-Funktion [Numerisch]“
- „ROUND-Funktion [Numerisch]“
- „SIGN-Funktion [Numerisch]“
- „SIN-Funktion [Numerisch]“
- „SQRT-Funktion [Numerisch]“
- „TAN-Funktion [Numerisch]“
- „TRUNCNUM-Funktion [Numerisch]“

Räumliche UltraLite-Funktionen

Räumliche Daten sind Daten, die die Position, Form und Ausrichtung von Objekten in einem definierten Raum beschreiben. UltraLite stellt Speicher- und Datenverwaltungsfunktionen für räumliche Daten in Form von Punkten bereit, mit denen Sie Informationen wie zum Beispiel geografische Standorte und Routinginformationen speichern können. UltraLite stellt eine Reihe von räumlichen SQL-Funktionen bereit, um die Kompatibilität mit anderen Produkten zu gewährleisten. Diese Funktionen und Konstruktoren können Sie verwenden, um auf die räumlichen Daten zuzugreifen und sie zu bearbeiten.

Liste der Funktionen

Die folgenden räumlichen Funktionen stehen zur Verfügung:

- „ST_AsBinary-Funktion [Räumlich]“
- „ST_AsExtText-Funktion [Räumlich]“
- „ST_AsText-Funktion [Räumlich]“
- „ST_Distance-Funktion [Räumlich]“
- „ST_Equals-Funktion [Räumlich]“
- „ST_IntersectsRect-Funktion [Räumlich]“
- „ST_Point-Funktion [Räumlich]“
- „ST_PointFromExtText-Funktion [Räumlich]“
- „ST_PointFromText-Funktion [Spatial]“
- „ST_PointFromWKB-Funktion [Spatial]“
- „ST_SRID-Funktion [Räumlich]“
- „ST_X-Funktion [Räumlich]“
- „ST_Y-Funktion [Räumlich]“

UltraLite-Zeichenfolgenfunktionen

Zeichenfolgenfunktionen führen Konvertierungs-, Extraktions- oder Manipulationsvorgänge auf Zeichenfolgen aus oder geben Informationen über Zeichenfolgen zurück.

Prüfen Sie beim Arbeiten in einem Mehrzeichen-Zeichensatz sorgfältig, ob die verwendete Funktion Informationen über Zeichen oder Byte zurückgibt.

Liste der Funktionen

Folgende Zeichenfolgenfunktionen stehen zur Verfügung: Systemfunktionen

- „ASCII-Funktion [Zeichenfolge]“
- „BYTE_LENGTH-Funktion [Zeichenfolge]“
- „BYTE_SUBSTR-Funktion [Zeichenfolge]“
- „CHAR-Funktion [Zeichenfolge]“
- „CHARINDEX-Funktion [Zeichenfolge]“
- „CHAR_LENGTH-Funktion [Zeichenfolge]“
- „DIFFERENCE-Funktion [Zeichenfolge]“
- „INSERTSTR-Funktion [Zeichenfolge]“
- „LCASE-Funktion [Zeichenfolge]“
- „LEFT-Funktion [Zeichenfolge]“
- „LENGTH-Funktion [Zeichenfolge]“
- „LOCATE-Funktion [Zeichenfolge]“
- „LOWER-Funktion [Zeichenfolge]“
- „LTRIM-Funktion [Zeichenfolge]“
- „PATINDEX-Funktion [Zeichenfolge]“
- „REPEAT-Funktion [Zeichenfolge]“
- „REPLACE-Funktion [Zeichenfolge]“
- „REPLICATE-Funktion [Zeichenfolge]“
- „RIGHT-Funktion [Zeichenfolge]“
- „RTRIM-Funktion [Zeichenfolge]“
- „SIMILAR-Funktion [Zeichenfolge]“
- „SOUNDEX-Funktion [Zeichenfolge]“
- „SPACE-Funktion [Zeichenfolge]“
- „STR-Funktion [Zeichenfolge]“
- „STRING-Funktion [Zeichenfolge]“
- „STRTOUUID-Funktion [Zeichenfolge]“
- „STUFF-Funktion [Zeichenfolge]“
- „SUBSTRING-Funktion [Zeichenfolge]“
- „TRIM-Funktion [Zeichenfolge]“
- „UCASE-Funktion [Zeichenfolge]“
- „UPPER-Funktion [Zeichenfolge]“
- „UIDTOSTR-Funktion [Zeichenfolge]“

UltraLite-Systemfunktionen

Systemfunktionen geben Systeminformationen zurück.

Liste der Funktionen

Die folgenden Systemfunktionen sind in UltraLite verfügbar:

- „DB_PROPERTY-Funktion [System]“
- „ML_GET_SERVER_NOTIFICATION-Funktion [System]“
- „SYNC_PROFILE_OPTION_VALUE-Funktion [System]“

Funktionen

Jede Funktion ist mit ihrem Funktionstyp (numerisch, Zeichen, etc.) aufgelistet.

Verknüpfungen zu Funktionen eines gegebenen Typs finden Sie unter „[Funktionstypen](#)“ auf Seite 316.

ABS-Funktion [Numerisch]

Gibt den absoluten Wert eines numerischen Ausdrucks zurück.

Syntax

ABS(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Die Zahl, deren absoluter Wert zurückgegeben werden soll

Rückgabe

Ein absoluter Wert des numerischen Ausdrucks.

Datentyp numerischer Ausdruck	Rückgabe
INT	INT
FLOAT	FLOAT
DOUBLE	DOUBLE
NUMERIC	NUMERIC

Beispiel

Die folgende Anweisung gibt den Wert 66 zurück:

```
SELECT ABS( -66 );
```

ACOS-Funktion [Numerisch]

Gibt den Arkuskosinus eines numerischen Ausdrucks im Bogenmaß zurück. Nicht unterstützt von UltraLite Java Edition-Datenbanken.

Syntax

ACOS(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Der Kosinus des Winkels.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch.

Siehe auch

- „ASIN-Funktion [Nummerisch]“ auf Seite 325
- „ATAN-Funktion [Nummerisch]“ auf Seite 326
- „ATAN2-Funktion [Nummerisch]“ auf Seite 327
- „COS-Funktion [Nummerisch]“ auf Seite 339

Beispiel

Die folgende Anweisung gibt den Arkuskosinus-Wert für 0,52 zurück:

```
SELECT ACOS( 0.52 );
```

ARGN-Funktion [Verschiedene]

Gibt ein ausgewähltes Argument aus einer Argumentliste zurück.

Syntax

ARGN(*integer-expression*, *expression* [, ...])

Parameter

- **Ganzzahlausdruck** Die Position eines Arguments in der Liste der Ausdrücke.
- **expression** Ein Ausdruck eines beliebigen Datentyps, der an die Funktion übergeben wird. Alle übergebenen Ausdrücke müssen denselben Datentyp haben.

Rückgabe

Wenn der Wert von *Ganzzahlausdruck* *n* ist, wird das *n*-te Argument (beginnend bei 1) aus der verbleibenden Argumentliste zurückgegeben.

Bemerkungen

Für die Ausdrücke gibt es zwar keine Datentypbeschränkung, allerdings müssen alle den gleichen Datentyp haben. Der Ganzzahlausdruck muss zwischen eins und der Elementanzahl der Liste der Ausdrücke sein, weil sonst NULL zurückgegeben wird. Mehrfache Ausdrücke werden durch ein Komma getrennt.

Beispiel

Die folgende Anweisung gibt den Wert 6 zurück:

```
SELECT ARGN( 6, 1,2,3,4,5,6 );
```

ASCII-Funktion [Zeichenfolge]

Gibt den Ganzzahl-ASCII-Wert des ersten Bytes in einem Zeichenfolgenausdruck zurück.

Syntax

ASCII(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge.

Rückgabe

SMALLINT

Bemerkungen

Wenn die Zeichenfolge leer ist, gibt ASCII 0 zurück. Literal-Zeichenfolgen müssen von Anführungszeichen umschlossen sein. Wenn der Zeichensatz der Datenbank ein Mehrbyte-Zeichensatz ist und das erste Zeichen der Parameterzeichenfolge aus mehr als einem Byte besteht, ist das Ergebnis NULL.

Siehe auch

- „CHAR-Funktion [Zeichenfolge]“ auf Seite 333
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert 90 zurück:

```
SELECT ASCII( 'Z' );
```

ASIN-Funktion [Nummerisch]

Gibt den Arkussinus einer Zahl im Bogenmaß zurück.

Syntax

ASIN(*numeric-expression*)

Parameter

- **Nummerischer_Ausdruck** Der Sinus des Winkels.

Rückgabe

DOUBLE

Bemerkungen

Die SIN- und ASIN-Funktionen sind inverse Vorgänge.

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch.

Siehe auch

- „ACOS-Funktion [Nummerisch]“ auf Seite 323
- „ATAN-Funktion [Nummerisch]“ auf Seite 326
- „ATAN2-Funktion [Nummerisch]“ auf Seite 327
- „SIN-Funktion [Nummerisch]“ auf Seite 396

Beispiel

Die folgende Anweisung gibt den Arkussinus-Wert für 0,52 zurück:

```
SELECT ASIN( 0.52 );
```

ATAN-Funktion [Nummerisch]

Gibt den Arkustangens einer Zahl im Bogenmaß zurück. Nicht unterstützt von UltraLite Java Edition-Datenbanken.

Syntax

ATAN(*numeric-expression*)

Parameter

- **Nummerischer_Ausdruck** Der Tangens des Winkels.

Bemerkungen

Die ATAN- und TAN-Funktionen sind inverse Vorgänge.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch.

Siehe auch

- „ACOS-Funktion [Nummerisch]“ auf Seite 323
- „ASIN-Funktion [Nummerisch]“ auf Seite 325
- „ATAN2-Funktion [Nummerisch]“ auf Seite 327
- „TAN-Funktion [Nummerisch]“ auf Seite 412

Beispiel

Die folgende Anweisung gibt den Arkustangens-Wert für 0,52 zurück:

```
SELECT ATAN( 0.52 );
```

ATAN2-Funktion [Numerisch]

Gibt den Arkustangens des Bruchs aus zwei Zahlen im Bogenmaß zurück. Nicht unterstützt von UltraLite Java Edition-Datenbanken.

Syntax

ATAN2 (*numeric-expression-1*, *numeric-expression-2*)

Parameter

- **Numerischer_Ausdruck_1** Der Zähler des Bruchs, dessen Arkustangens berechnet wird.
- **Numerischer_Ausdruck_2** Der Nenner des Bruchs, dessen Arkustangens berechnet wird.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch.

Siehe auch

- „ACOS-Funktion [Numerisch]“ auf Seite 323
- „ASIN-Funktion [Numerisch]“ auf Seite 325
- „ATAN-Funktion [Numerisch]“ auf Seite 326
- „TAN-Funktion [Numerisch]“ auf Seite 412

Beispiel

Die folgende Anweisung gibt den Arkustangens-Wert für den Bruch 0,52 durch 0,60 zurück:

```
SELECT ATAN2( 0.52, 0.60 );
```

AVG-Funktion [Aggregat]

Berechnet bei einer Zeilenmenge den Durchschnitt eines numerischen Ausdrucks oder einer Menge eindeutiger Werte.

Syntax 1

AVG([**DISTINCT**] *numeric-expression*)

Parameter

- [**ALL**] **Numerischer_Ausdruck** Der Ausdruck, dessen Durchschnitt über die Zeilen in jeder Gruppe berechnet wird.
- **DISTINCT-Klausel** Berechnet den Durchschnitt der eindeutigen numerischen Werte in jeder Gruppe.

Rückgabe

Gibt NULL für eine Gruppe zurück, die keine Zeilen enthält.

Gibt DOUBLE zurück, wenn das Argument DOUBLE ist, sonst NUMERIC.

Bemerkungen

Dieser Durchschnitt schließt keine Zeilen mit ein, in denen *Nummerischer_Ausdruck* NULL ist.

Diese Funktion kann einen Überlauffehler erzeugen, was dazu führt, dass ein Fehler zurückgegeben wird. Sie können die CAST-Funktion auf *Nummerischer_Ausdruck* anwenden, um den Überlauffehler zu vermeiden.

Siehe auch

- „SUM-Funktion [Aggregat]“ auf Seite 409
- „COUNT-Funktion [Aggregat]“ auf Seite 340

Siehe auch

- „Fehlerbehandlung: Aggregatfunktionen und äußere Referenzen“ [*SQL Anywhere 16 - Änderungen und Upgrades*]

Beispiel

Die folgende Anweisung gibt den Wert 49988.623200 zurück, wenn eine Verbindung mit SQL Anywhere 16 Demo besteht:

```
SELECT AVG( Salary ) FROM Employees;
```

Die folgende Anweisung gibt den durchschnittlichen Produktpreis aus der Tabelle "Products" zurück, wenn eine Verbindung mit SQL Anywhere 16 Demo besteht:

```
SELECT AVG( DISTINCT UnitPrice ) FROM Products;
```

BYTE_LENGTH-Funktion [Zeichenfolge]

Gibt die Anzahl der Byte in einer Zeichenfolge zurück.

Syntax

BYTE_LENGTH(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, deren Länge berechnet werden soll.

Rückgabe

INT

Bemerkungen

Nachgestellte Leerzeichen in *Zeichenfolgenausdruck* sind in der zurückgegebenen Länge enthalten.

Der Rückgabewert einer NULL-Zeichenfolge ist NULL.

Wenn die Zeichenfolge zu einem Mehrbyte-Zeichensatz gehört, kann sich der BYTE_LENGTH-Wert von der Anzahl der Zeichen, die CHAR_LENGTH zurückgibt, unterscheiden.

Siehe auch

- „CHAR_LENGTH-Funktion [Zeichenfolge]“ auf Seite 334
- „DATALENGTH-Funktion [System]“ auf Seite 341
- „LENGTH-Funktion [Zeichenfolge]“ auf Seite 365
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert 12 zurück:

```
SELECT BYTE_LENGTH( 'Test Message' );
```

BYTE_SUBSTR-Funktion [Zeichenfolge]

Gibt eine Teilzeichenfolge einer Zeichenfolge zurück. Die Teilzeichenfolge wird nach Bytes berechnet, nicht nach Zeichen.

Syntax

BYTE_SUBSTR(*string-expression*, *start* [, *length*])

Parameter

- **Zeichenfolgenderausdruck** Die Zeichenfolge, der die Teilzeichenfolge entnommen wird.
- **start** Ein Ganzzahl-Ausdruck, der den Start der Teilkette angibt. Eine positive Ganzzahl startet am Anfang der Zeichenfolge, mit dem ersten Zeichen in der Position 1. Eine negative Ganzzahl gibt eine Teilzeichenfolge an, die am Ende der Zeichenfolge beginnt, wobei sich das letzte Zeichen in der Position -1 befindet.
- **length** Ein Ganzzahl-Ausdruck, der die Länge der Teilzeichenfolge angibt. Eine positive *length* gibt die Anzahl von Byte an, die genommen werden sollen, wobei mit der Startposition *begonnen* wird. Eine negative *length* gibt maximal *length* Bytes links von der Startposition bis zur und einschließlich der Startposition zurück.

Rückgabe

BINARY, oder VARCHAR. Der zurückgegebene Wert hängt vom Typ von *Zeichenfolgenderausdruck* ab. Außerdem bestimmen die von Ihnen angegebenen Argumente, ob der zurückgegebene Wert LONG ist. Beispiel: LONG wird nicht zurückgegeben, wenn Sie eine Konstante < 32 kB für die Länge angeben.

Bemerkungen

Wenn *length* angegeben ist, ist die Teilzeichenfolge auf diese Anzahl von Byte begrenzt. Sowohl *start* als auch *length* können entweder positiv oder negativ sein. Mithilfe von entsprechenden Kombinationen von negativen und positiven Zahlen können Sie eine Teilzeichenfolge entweder vom Anfang oder vom Ende der Zeichenfolge bekommen.

Wenn *start* Null und die Länge nicht-negativ ist, wird ein *start*-Wert von "1" verwendet. Wenn *start* Null und *length* negativ ist, wird ein Start-Wert von "-1" verwendet.

Siehe auch

- „SUBSTRING-Funktion [Zeichenfolge]“ auf Seite 408
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert `Test` zurück:

```
SELECT BYTE_SUBSTR( 'Test Message', 1, 4 );
```

CAST-Funktion [Datentypkonvertierung]

Gibt den in einen angegebenen Datentyp konvertierten Wert eines Ausdrucks zurück.

Syntax

CAST(*expression AS datatype*)

Parameter

- ***expression*** Der zu konvertierende Ausdruck.
- ***datatype*** Der Zieldatentyp.

Rückgabe

Abhängig vom angeforderten Datentyp.

Bemerkungen

Wenn Sie bei Zeichenfolgentypen keine Länge angeben, wird eine geeignete Länge gewählt. Wenn bei einer DECIMAL-Konvertierung weder Gesamtstellenzahl noch Dezimalstellen angegeben wurden, wird ein Standardwert ausgewählt. Es wird empfohlen, dass Sie Gesamtstellenzahl und Dezimalstellen in Ihrer CAST-Funktion explizit angeben.

Ob eine Konvertierung möglich ist, hängt von dem verwendeten Wert ab. Die Werte im ursprünglichen Datentyp müssen mit dem neuen Datentyp kompatibel sein, damit kein Konvertierungsfehler generiert wird.

Dem folgenden Diagramm können Sie entnehmen, ob eine Konvertierung unterstützt wird:

FROM:\nTO:	BIT	TINY INT	UNSIGNED SMALL INT	SMALL INT	UNSIGNED INTEGER	INTEGER	UNSIGNED BIGINT	BIGINT	FLOAT	REAL	DOUBLE	NUMERIC OR DECIMAL	DATE	TIME	DATETIME OR\nTIMESTAMP	TIMESTAMP WITH TIME\nZONE	UNIQUEIDENTIFIER	BINARY OR\nVARBINARY	LONG BINARY	CHAR OR VARCHAR	LONG VARCHAR	ST_GEOMETRY
BIT		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗
TINY INT	⚠		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
UNSIGNED SMALL INT	⚠	⚠		✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
SMALL INT	⚠	⚠	⚠		✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
UNSIGNED INTEGER	⚠	⚠	⚠	⚠		✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
INTEGER	⚠	⚠	⚠	⚠	⚠		✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
UNSIGNED BIGINT	⚠	⚠	⚠	⚠	⚠	⚠		✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
BIGINT	⚠	⚠	⚠	⚠	⚠	⚠	⚠		✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
FLOAT	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠		⚠	✓	✓	✗	✗	✗	✗	✗	⚠	✗	✓	✗	✗
REAL	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	✓		✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
DOUBLE	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	✓	⚠		✓	✗	✗	✗	✗	✗	⚠	✗	✓	✗	✗
NUMERIC OR DECIMAL	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	✓	✓	✓		✗	✗	✗	✗	✗	⚠	✗	✓	✗	✗
DATE	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗		✓	✓	✓	✗	✗	✗	✓	✗	✗
TIME	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓		✓	✓	✗	✗	✓	✗	✗	✗
DATETIME OR\nTIMESTAMP	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓		✓	✗	✗	✗	✓	✗	✗
TIMESTAMP WITH TIME\nZONE	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓		✗	✗	✗	✓	✗	✗
UNIQUEIDENTIFIER	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗		⚠	✗	✓	✗	✗
BINARY OR\nVARBINARY	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	⚠	✗	✗	✗	✗	⚠		✓	✓	✗	⚠
LONG BINARY	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓		✗	✗	✗
CHAR OR VARCHAR	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	✓	✗		✓	⚠
LONG VARCHAR	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓		✗
ST_GEOMETRY	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	⚠	✗	✓	✗	

Symbol	Kompatibilität
✓	Wird immer konvertiert
✗	Wird nie konvertiert
⚠	Wertabhängig

Hinweis

Damit eine Konvertierung zwischen VARBINARY und UNIQUEIDENTIFIER möglich ist, muss der VARBINARY-Wert 16 Byte lang sein.

Damit eine Konvertierung zwischen NUMERIC und VARBINARY möglich ist, muss die NUMERIC-Quelle einen Wert haben, der auch in BIGINT umgewandelt werden kann.

Damit eine Konvertierung von VARCHAR in ST_GEOMETRY möglich ist, muss die VARCHAR-Quelle eine gültige Geometrie darstellen, entweder im WKT-Format oder im EWKT-Format.

Damit eine Konvertierung von VARBINARY in ST_GEOMETRY möglich ist, muss die VARBINARY-Quelle eine gültige Geometrie im WKB-Format darstellen.

Beim Casting einer Quelle im WKB- oder WKT-Format in ein ST_GEOMETRY-Objekt wird dem ST_GEOMETRY-Wert die SRID 0 zugeordnet. Beim Casting eines ST_GEOMETRY-Objekts werden VARCHAR-Werte in EWKT formatiert und VARBINARY-Werte in WKB.

Die Funktionen HEXTOINT und INTTOHEX können zum Konvertieren in und aus hexadezimalen Werten verwendet werden. Weitere Hinweise zur Verwendung dieser Funktionen finden Sie unter [Konvertierung in und aus hexadezimalen Werten \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#).

Siehe auch

- „[CONVERT-Funktion \[Datentypkonvertierung\]](#)“ auf Seite 336

Beispiel

Die folgende Funktion stellt sicher, dass eine Zeichenfolge als Datum verwendet wird:

```
SELECT CAST( '2000-10-31' AS DATE );
```

Der Wert des Ausdrucks 1 + 2 wird berechnet und das Ergebnis als Zeichenfolge, bestehend aus einem Zeichen, ausgegeben.

```
SELECT CAST( 1 + 2 AS CHAR );
```

Sie können die CAST-Funktion zum Kürzen von Zeichenfolgen verwenden:

```
SELECT CAST ( 'Surname' AS CHAR(5) );
```

Das Casting zwischen VARCHAR und ST_GEOMETRY erfolgt normalerweise implizit. Die folgende Anweisung fügt beispielsweise Werte zu ST_GEOMETRY-Spalten hinzu und verwendet dafür die ST_POINT-Funktion und einen VARCHAR-Parameter. Jeder Wert wird implizit in einen mit den Spalten der Tabelle konsistenten ST_GEOMETRY-Datentyp umgewandelt, aber Ergebnisse werden weiterhin als VARCHAR angezeigt.

```
INSERT INTO T1 VALUES (2, ST_POINT(1,2,0), 'SRID=2163;Point(1 2)');
```

CEILING-Funktion [Numerisch]

Gibt die erste Ganzzahl zurück, die größer oder gleich einem gegebenen Wert ist. Bei positiven Zahlen wird dies "Aufrunden" genannt.

Syntax

{ **CEILING** | **CEIL** } (*numeric-expression*)

Parameter

- **Nummerischer_Ausdruck** Die Zahl, deren Obergrenze berechnet werden soll.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch.

Siehe auch

- [„FLOOR-Funktion \[Numerisch\]“ auf Seite 355](#)

Beispiel

Die folgende Anweisung gibt den Wert 60 zurück:

```
SELECT CEILING( 59.84567 );
```

CHAR-Funktion [Zeichenfolge]

Gibt das zu einem ASCII-Code gehörige Zeichen zurück.

Syntax

CHAR(*integer-expression*)

Parameter

- **Ganzzahlausdruck** Die Zahl, die in ein ASCII-Zeichen konvertiert werden soll. Die Zahl muss im Bereich 0 bis einschließlich 255 liegen.

Rückgabe

VARCHAR

Bemerkungen

Das zurückgegebene Zeichen entspricht dem angegebenen numerischen Ausdruck im aktuellen Zeichensatz nach binärer Sortierreihenfolge.

CHAR gibt NULL für Ganzzahl-Ausdrücke mit Werten größer als 255 und kleiner als Null zurück.

Siehe auch

- [„UltraLite-Zeichenfolgenfunktionen“ auf Seite 321](#)

Beispiel

Die folgende Anweisung gibt den Wert Y zurück:

```
SELECT CHAR( 89 );
```

CHAR_LENGTH-Funktion [Zeichenfolge]

Gibt die Anzahl der Zeichen in einer Zeichenfolge zurück.

Syntax

```
CHAR_LENGTH ( string-expression )
```

Parameter

- **Zeichenfolgenderausdruck** Die Zeichenfolge, deren Länge berechnet werden soll.

Rückgabe

INT

Bemerkungen

Nachgestellte Leerzeichen sind in der zurückgegebenen Länge enthalten.

Der Rückgabewert einer NULL-Zeichenfolge ist NULL.

Wenn die Zeichenfolge ein Mehrbyte-Zeichensatz ist, kann sich der von der CHAR_LENGTH-Funktion zurückgegebene Wert von der Anzahl von Bytes, die die BYTE_LENGTH-Funktion zurückgibt, unterscheiden.

Hinweis

Sie können die CHAR_LENGTH-Funktion und die LENGTH-Funktion gleichermaßen für CHAR, VARCHAR und LONG VARCHAR-Datentypen verwenden. Sie müssen allerdings die LENGTH-Funktion bei BINARY- und Bit-Array-Datentypen verwenden.

Siehe auch

- „BYTE_LENGTH-Funktion [Zeichenfolge]“ auf Seite 328
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert 8 zurück:

```
SELECT CHAR_LENGTH( 'Chemical' );
```

CHARINDEX-Funktion [Zeichenfolge]

Gibt die Position einer Zeichenfolge in einer anderen zurück.

Syntax

CHARINDEX(*string-expression-1*, *string-expression-2*)

Parameter

- **Zeichenfolgenausdruck_1** Die Zeichenfolge, nach der Sie suchen.
- **Zeichenfolgenausdruck_2** Die zu durchsuchende Zeichenfolge.

Rückgabe

INT

Bemerkungen

Das erste Zeichen von *Zeichenfolgenausdruck_1* ist als 1 definiert. Wenn die durchsuchte Zeichenfolge mehr als eine Instanz der anderen Zeichenfolge enthält, gibt die CHARINDEX-Funktion die Position der ersten Instanz zurück.

Wenn die durchsuchte Zeichenfolge die andere Zeichenfolge nicht enthält, gibt die CHARINDEX-Funktion "0" zurück.

Falls eines der Argumente NULL ist, ist auch das Ergebnis NULL.

Siehe auch

- [„SUBSTRING-Funktion \[Zeichenfolge\]“ auf Seite 408](#)
- [„REPLACE-Funktion \[Zeichenfolge\]“ auf Seite 388](#)
- [„LOCATE-Funktion \[Zeichenfolge\]“ auf Seite 367](#)
- [„UltraLite-Zeichenfolgenfunktionen“ auf Seite 321](#)

Beispiel

Die folgende Anweisung gibt die Nach- und Vornamen aus den Spalten Surname und GivenName der Tabelle "Employees" zurück, aber nur, wenn der Nachname mit dem Buchstaben K anfängt:

```
SELECT Surname, GivenName
FROM GROUPO.Employees
WHERE CHARINDEX( 'K', Surname ) = 1;
```

Zurückgegebene Ergebnisse:

Surname	GivenName
Klobucher	James
Kuo	Felicia
Kelly	Moir

COALESCE-Funktion [Verschiedene]

Gibt den ersten Nicht-NULL-Ausdruck in einer Liste zurück. Diese Funktion ist mit der ISNULL-Funktion identisch.

Syntax

COALESCE(*expression*, *expression* [, ...])

Parameter

- ***expression*** Ein beliebiger Ausdruck.

Mindestens zwei Ausdrücke müssen an die Funktion übergeben werden und alle Ausdrücke müssen vergleichbar sein.

Rückgabe

Der Rückgabotyp dieser Funktion hängt von den angegebenen Ausdrücken ab. Das bedeutet, dass der Datenbankserver beim Auswerten der Funktion zuerst nach einem Datentyp sucht, mit dem alle Ausdrücke verglichen werden können. Wenn er gefunden wird, vergleicht der Datenbankserver die Ausdrücke und gibt das Ergebnis in dem für den Vergleich verwendeten Datentyp zurück. Wenn der Datenbankserver keinen gemeinsamen Vergleichsdattentyp findet, wird ein Fehler zurückgegeben.

Bemerkungen

Das Ergebnis ist nur NULL, wenn alle Argumente NULL sind.

Der Parameter kann ein beliebiger Skalartyp sein, aber nicht notwendigerweise derselbe Typ.

Eine detaillierte Beschreibung, wie der Datenbankserver diese Funktion verarbeitet, finden Sie unter [„ISNULL-Funktion \[Verschiedene\]“ auf Seite 363](#).

Siehe auch

- [„ISNULL-Funktion \[Verschiedene\]“ auf Seite 363](#)

Beispiel

Die folgende Anweisung gibt den Wert 34 zurück:

```
SELECT COALESCE( NULL, 34, 13, 0 );
```

CONVERT-Funktion [Datentypkonvertierung]

Gibt einen in einen angegebenen Datentyp konvertierten Ausdruck zurück.

Diese Funktion ähnelt der CAST-Funktion, ermöglicht Ihnen jedoch das Angeben eines Formatstils, um Konvertierungen von Datums- und Zeitdatentypen zu unterstützen. Weitere Hinweise zu anderen Konvertierungen finden Sie unter [„CAST-Funktion \[Datentypkonvertierung\]“ auf Seite 330](#).

Syntax

CONVERT(*datatype*, *expression* [, *format-style*])

Parameter

- ***datatype*** Der Datentyp, in den der Ausdruck konvertiert wird.
- ***expression*** Der zu konvertierende Ausdruck.
- ***format-style*** Der Stil-Code, der für den ausgegebenen Wert gilt. Verwenden Sie diesen Parameter, wenn Sie Zeichenfolgen in Datumsangaben oder Zeitdatumsangaben bzw. umgekehrt konvertieren. Die untenstehende Tabelle enthält die unterstützten Stil-Codes, gefolgt von einer Darstellung der Ausgabeformate, die vom jeweiligen Stil-Code erzeugt werden. Die Stil-Codes sind auf zwei Spalten aufgeteilt, abhängig davon, ob das Jahrhundert im Ausgabeformat enthalten ist (z.B. 06 bzw. 2006).

Stilcode 0 wird verwendet, wenn kein Argument angegeben wurde.

Stil-Codes ohne Jahrhundert (jj)	Stil-Codes mit Jahrhundert (jjjj)	Ausgabeformat
-	0 oder 100	Mmm tt jjjj hh:nnAA
1	101	mm/tt/jj[jj]
2	102	[jj]jj.mm.tt
3	103	tt/mm/jj[jj]
4	104	tt.mm.jj[jj]
5	105	tt-mm-jj[jj]
6	106	tt Mmm jj[jj]
7	107	Mmm tt, jj[jj]
8	108	hh:nn:ss
-	9 oder 109	Mmm tt jjjj hh:nn:ss:sssAA
10	110	mm-tt-jj[jj]
11	111	[jj]jj/mm/tt
12	112	[jj]jjmmtt
-	13 oder 113	tt Mmm jjjj hh:nn:ss:sss (24-Stundenuhr, Europa-Standard + Millisekunden, vierstelliges Jahr)
-	14 oder 114	hh:nn:ss:sss (24-Stundenuhr)
-	20 oder 120	jjjj-mm-tt hh:nn:ss (24-Stundenuhr, ODBC entsprechend, vierstelliges Jahr)

Stil-Codes ohne Jahrhundert (jj)	Stil-Codes mit Jahrhundert (jjjj)	Ausgabeformat
-	21 oder 121	jjjj-mm-tt hh:nn:ss.sss (24-Stundenuhr, ODBC entsprechend mit Millisekunden, vierstelliges Jahr)

Rückgabe

Abhängig vom angegebenen Datentyp.

Bemerkungen

Die CONVERT-Funktion kann verwendet werden, um eine Zeichenfolge in einen DATE-, TIME- oder TIMESTAMP-Datentyp zu konvertieren, wenn es keine Mehrdeutigkeit bei der syntaktischen Analyse der Zeichenfolge gibt. Wenn *format-style* angegeben ist, verwendet der Datenbankserver diesen möglicherweise als Hinweis zur syntaktischen Analyse der Zeichenfolge. Der Datenbankserver gibt einen Fehler zurück, wenn er die Zeichenfolge syntaktisch nicht eindeutig analysieren kann.

Informationen über die Stile, die von den Ausgabesymbolen erzeugt werden (z.B. Mmm), finden Sie unter [„UltraLite-Erstellungsparameter date_format“ auf Seite 146](#).

Beispiel

Die folgenden Anweisungen veranschaulichen die Verwendung des Formatstils:

```
SELECT CONVERT( CHAR( 20 ), OrderDate, 104 ) FROM GROUPO.SalesOrders;
```

OrderDate
16.03.2000
20.03.2000
23.03.2000
25.03.2000
...

```
SELECT CONVERT( CHAR( 20 ), OrderDate, 7 ) FROM GROUPO.SalesOrders;
```

OrderDate
Mar 16, 00
Mar 20, 00
Mar 23, 00
Mar 25, 00

OrderDate
...

Die folgende Anweisung veranschaulicht die Konvertierung in eine Ganzzahl und gibt den Wert 5 zurück:

```
SELECT CONVERT( integer, 5.2 );
```

COS-Funktion [Numerisch]

Gibt den Kosinus des Winkels im Bogenmaß zurück, der im Argument angegeben ist.

Syntax

COS(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Der Winkel im Bogenmaß.

Rückgabe

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltgenauem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück. Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Siehe auch

- „ACOS-Funktion [Numerisch]“ auf Seite 323
- „COT-Funktion [Numerisch]“ auf Seite 339
- „SIN-Funktion [Numerisch]“ auf Seite 396
- „TAN-Funktion [Numerisch]“ auf Seite 412

Beispiel

Die folgende Anweisung gibt den Kosinuswert eines Winkels von 0,52 Bogenmaß zurück:

```
SELECT COS( 0.52 );
```

COT-Funktion [Numerisch]

Gibt den Kotangens des Winkels im Bogenmaß zurück, der im Argument angegeben ist.

Syntax

COT(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Der Winkel im Bogenmaß.

Rückgabe

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück. Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Siehe auch

- „COS-Funktion [Numerisch]“ auf Seite 339
- „SIN-Funktion [Numerisch]“ auf Seite 396
- „TAN-Funktion [Numerisch]“ auf Seite 412

Beispiel

Die folgende Anweisung gibt den Kotangenswert für 0,52 zurück:

```
SELECT COT( 0.52 );
```

COUNT-Funktion [Aggregat]

Zählt die Anzahl der Zeilen in einer Gruppe, abhängig von den angegebenen Parametern.

Syntax 1

COUNT([* | [**DISTINCT**] *expression*])

Parameter

- ***** Gibt die Anzahl der Zeilen in jeder Gruppe zurück. COUNT(*) und COUNT() sind semantisch gleichwertig.
- **expression** Gibt die Anzahl der Zeilen in jeder Gruppe zurück, wobei *expression* nicht den Wert NULL hat.
- **DISTINCT expression** Gibt die Anzahl der unterschiedlichen Werte von *expression* für alle Zeilen in jeder Gruppe zurück, in denen *expression* nicht den Wert NULL hat.

Rückgabe

Die COUNT Funktion gibt einen Wert vom Datentyp INT zurück.

COUNT liefert nie den Wert NULL. Wenn eine Gruppe keine Zeilen oder keine Nicht-NULL-Werte von *expression* enthält, gibt COUNT den Wert 0 zurück.

Bemerkungen

Die COUNT Funktion gibt einen Maximalwert von 2147483647 zurück.

Siehe auch

- „AVG-Funktion [Aggregat]“ auf Seite 327
- „SUM-Funktion [Aggregat]“ auf Seite 409
- „Fehlerbehandlung: Aggregatfunktionen und äußere Referenzen“ [*SQL Anywhere 16 - Änderungen und Upgrades*]

Beispiel

Die folgende Anweisung gibt jede eindeutige Stadt und die Anzahl der Mitarbeiter in der jeweiligen Stadt zurück:

```
SELECT City, COUNT( * ) FROM GROUPO.Employees GROUP BY City;
```

COUNT_UPLOAD_ROWS-Funktion [Aggregat]

Gibt einen Zählwert für die Anzahl der Zeilen zurück, die bei der nächsten Synchronisation hochgeladen werden.

Syntax

```
COUNT_UPLOAD_ROWS( pubs,threshold)
```

Parameter

- **pubs** Eine kommagetrennte Liste von Publikationen, in denen nach Zeilen gesucht werden soll.
- **threshold** Die maximale Anzahl der zu zählenden Zeilen. (Der Wert 0 entspricht der Obergrenze.)

Rückgabe

INT

Beispiel

Folgendes liefert die Gesamtzahl der Zeilen für den Upload in *mypub1* und *mypub2*:

```
SELECT COUNT_UPLOAD_ROWS( 'mypub1,mypub2', 0 );
```

DATALENGTH-Funktion [System]

Gibt in Byte die Länge der Basisspeicherung für das Ergebnis eines Ausdrucks zurück.

Syntax

```
DATALENGTH( expression )
```

Parameter

- **expression** In der Regel ein Spaltenname. Wenn *expression* eine Zeichenfolgenkonstante ist, muss er in Anführungszeichen eingeschlossen werden.

Rückgabe

UNSIGNED INT

Bemerkungen

Die Rückgabewerte der DATALENGTH-Funktion sind Folgende:

Datentyp	DATALENGTH
BIT	1
TINYINT	1
SMALLINT	2
INTEGER	4
BIGINT	8
REAL	4
DOUBLE	8
TIME	8
DATE	4
TIMESTAMP	8
DATETIME	8
TIMESTAMP WITH TIME ZONE	29
UNIQUEIDENTIFIER	16
CHAR	Länge der Daten
VARCHAR	Länge der Daten
BINARY	Länge der Daten
VARBINARY	Länge der Daten
NCHAR	Länge der Daten
NVARCHAR	Länge der Daten
TEXT	Länge der Daten
NTEXT	Länge der Daten
IMAGE	Länge der Daten
XML	Länge der Daten

Siehe auch

- „SQL-Datentypen“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Beispiel

Die folgende Anweisung gibt die Länge der längsten Zeichenfolge in der CompanyName-Spalte zurück:

```
SELECT MAX( DATALENGTH( CompanyName ) )
FROM GROUPO.Customers;
```

Die folgende Anweisung gibt die Länge der Zeichenfolge "8sdofinsv8s7a7s7gehe4h" zurück:

```
SELECT DATALENGTH( '8sdofinsv8s7a7s7gehe4h' );
```

DATE-Funktion [Datum und Uhrzeit]

Konvertiert den Ausdruck in ein Datum und entfernt Stunden, Minuten oder Sekunden.

Hinweise zur Steuerung der Interpretation von Datumsformaten finden Sie unter „[UltraLite-Erstellungsparameter date_order](#)“ auf Seite 149.

Syntax

DATE(*expression*)

Rückgabe

DATE

Parameter

- **expression** Der Wert, der ins Datumsformat konvertiert werden soll. Dies ist üblicherweise eine Zeichenfolge.

Beispiel

Die folgende Anweisung gibt den Wert 1999-01-02 als Datum zurück:

```
SELECT DATE( '1999-01-02 21:20:53' );
```

Die folgende Anweisung gibt das Erstellungsdatum aller Objekte zurück, die in der SYSOBJECT-Systemansicht aufgelistet sind:

```
SELECT DATE( creation_time ) FROM SYSOBJECT;
```

DATEADD-Funktion [Datum und Uhrzeit]

Gibt einen TIMESTAMP- oder TIMESTAMP WITH TIME ZONE-Wert zurück, der durch Addieren eines Datumsteils zu ihrem Argument erzeugt wird.

Syntax

DATEADD(*date-part*, *integer-expression*, *timestamp-expression*)

date-part :
year
quarter
month
week
day
dayofyear
hour
minute
second
millisecond
microsecond

Parameter

- **Datumsteil** Der Datumsteil, den der *Zeichenfolgenausdruck* darstellt.

Eine komplette Enumeration zulässiger Datumsteile finden Sie unter [Angeben von Datumsteilen auf Seite 317](#).
- **Ganzzahlausdruck** Die Anzahl der *Datumsteil*-Werte, die zu *Zeitstempelausdruck* addiert werden sollen. *Zeichenfolgenausdruck* kann jeder numerische Typ sein, aber der Wert wird auf INTEGER gekürzt.
- **Zeitstempelausdruck** Der TIMESTAMP- oder TIMESTAMP WITH TIME ZONE-Wert, der geändert werden muss.

Rückgabe

TIMESTAMP WITH TIME ZONE, wenn *Zeitstempelausdruck* vom Typ TIMESTAMP WITH TIME ZONE ist, andernfalls TIMESTAMP.

Beispiel

Die folgende Anweisung gibt den TIMESTAMP-Wert 1995-11-02 00:00:00.000 zurück:

```
SELECT DATEADD( month, 102, '1987/05/02' );
```

Die folgende Anweisung gibt den TIMESTAMP-Wert 1987-05-02 04:00:00.000 zurück:

```
SELECT DATEADD( hour, 4, '1987/05/02' );
```

Die folgende Anweisung gibt den TIMESTAMP WITH TIME ZONE-Wert 1987-05-06 11:33:00.000+04:00 zurück:

```
SELECT DATEADD( day, 4, CAST( '1987/05/02 11:33:00.000000+04:00' as  
TIMESTAMP WITH TIME ZONE ) );
```

DATEDIFF-Funktion [Datum und Uhrzeit]

Gibt das Intervall zwischen zwei Datumsangaben zurück.

Syntax

```
DATEDIFF( date-part, date-expression-1, date-expression-2 )
```

date-part :

- year**
- quarter**
- month**
- week**
- day**
- dayofyear**
- hour**
- minute**
- second**
- millisecond**
- microsecond**

Parameter

- **Datumsteil** Bestimmt den Datumsteil, in dem das Intervall gemessen werden soll.

Wählen Sie eines der oben aufgelisteten Datumsobjekte. Eine komplette Liste der Datumsteile finden Sie unter [Angaben von Datumsteilen auf Seite 317](#).
- **Datumsausdruck_1** Das Startdatum für das Intervall. Dieser Wert wird von *Datumsausdruck_2* abgezogen, um die Anzahl von *Datumsteilen* zwischen den beiden Argumenten zu erhalten.
- **Datumsausdruck_2** Das Enddatum für das Intervall. *Datumsausdruck_1* wird von diesem Wert abgezogen, um die Anzahl von *Datumsteilen* zwischen den beiden Argumenten zu erhalten.

Rückgabe

INT mit Jahr, Quartal, Monat, Woche, Tag und Jahrestag. BIGINT mit Stunde, Minute, Sekunde, Millisekunde und Mikrosekunde.

Bemerkungen

Diese Funktion berechnet die Anzahl von Datumsteilen zwischen zwei angegebenen Datumsangaben. Das Ergebnis ist ein Ganzzahlwert mit Vorzeichen gleich (*Datumsausdruck_2* - *Datumsausdruck_1*), in Datumsteilen.

Ergebnisse der DATEDIFF-Funktion werden gekürzt und nicht gerundet, wenn das Ergebnis nicht ein gerades Mehrfaches des Datumsteils ist.

Wenn Sie **day** als den Datumsteil verwenden, gibt die DATEDIFF-Funktion die Anzahl der Mitternächte zwischen den angegebenen Zeitpunkten zurück, einschließlich des zweiten Datums, aber nicht des ersten.

Wenn Sie **month** als den Datumsteil angeben, gibt die DATEDIFF-Funktion die Anzahl der Monatsersten zwischen den Datumsangaben zurück, einschließlich des zweiten Datums, aber nicht des ersten.

Wenn Sie **week** als den Datumsteil verwenden, gibt die DATEDIFF-Funktion die Anzahl der Sonntage zwischen den Datumsangaben zurück, einschließlich des zweiten Datums, aber nicht des ersten.

Beispiel

Die folgende Anweisung gibt den Wert 1 zurück:

```
SELECT DATEDIFF( hour, '4:00AM', '5:50AM' );
```

Die folgende Anweisung gibt den Wert 102 zurück:

```
SELECT DATEDIFF( month, '1987/05/02', '1995/11/15' );
```

Die folgende Anweisung gibt den Wert 0 zurück:

```
SELECT DATEDIFF( day, '00:00', '23:59' );
```

Die folgende Anweisung gibt den Wert 4 zurück:

```
SELECT DATEDIFF( day,  
    '1999/07/19 00:00',  
    '1999/07/23 23:59' );
```

Die folgende Anweisung gibt den Wert 0 zurück:

```
SELECT DATEDIFF( month, '1999/07/19', '1999/07/23' );
```

Die folgende Anweisung gibt den Wert 1 zurück:

```
SELECT DATEDIFF( month, '1999/07/19', '1999/08/23' );
```

DATEFORMAT-Funktion [Datum und Uhrzeit]

Gibt eine Zeichenfolge zurück, die einen Datumsausdruck im angegebenen Format darstellt.

Syntax

DATEFORMAT(*datetime-expression*, *string-expression*)

Parameter

- **DATETIME-Ausdruck** Die zu konvertierende Zeitangabe
- **Zeichenfolgenausdruck** Das Format des konvertierten Datums

Hinweise zu den Datumsformatbeschreibungen finden Sie unter „[UltraLite-Erstellungsparameter date_format](#)“ auf Seite 146.

Rückgabe

VARCHAR

Bemerkungen

Jedes erlaubte Datumsformat kann für den Zeichenfolgenausdruck verwendet werden.

Beispiel

Die folgende Anweisung gibt den Wert "Jan 01, 1989" zurück:

```
SELECT DATEFORMAT( '1989-01-01', 'Mmm dd, yyyy' );
```

DATENAME-Funktion [Datum und Uhrzeit]

Gibt den Namen des angegebenen Teils eines TIMESTAMP- oder TIMESTAMP WITH TIME ZONE-Wertes als Zeichenfolge zurück (z.B. Monat Juni).

Syntax

DATENAME(*date-part*, *timestamp-expression*)

Parameter

- **Datumsteil** Der Datumsteil, der benannt werden soll.

Eine komplette Enumeration zulässiger Datumsteile finden Sie unter [Angeben von Datumsteilen auf Seite 317](#).

- **Zeitstempelausdruck** Der TIMESTAMP- oder TIMESTAMP WITH TIME ZONE-Wert, für den der Datumsteil zurückgegeben werden soll. Damit die Funktion sinnvolle Ergebnisse liefert, sollte *Zeitstempelausdruck* den angeforderten *Datumsteil* enthalten.

Rückgabe

VARCHAR

Bemerkungen

Die DATENAME-Funktion gibt eine Zeichenfolge zurück, sogar wenn das Ergebnis numerisch ist, wie "23" für den Tag.

Siehe auch

- „[DATEPART-Funktion \[Datum und Uhrzeit\]](#)“ auf Seite 347

Beispiel

Die folgende Anweisung gibt den Wert "Mai" zurück:

```
SELECT DATENAME( month, '1987/05/02' );
```

DATEPART-Funktion [Datum und Uhrzeit]

Syntax

DATEPART(*date-part*, *timestamp-expression*)

Parameter

- **Datumsteil** Der Datumsteil, der zurückgegeben werden soll.

Eine komplette Enumeration zulässiger Datumsteile finden Sie unter [Angeben von Datumsteilen auf Seite 317](#).

- **Zeitstempelausdruck** Der TIMESTAMP- oder TIMESTAMP WITH TIME ZONE-Wert, für den der Teil zurückgegeben werden soll.

Rückgabe

INT

Bemerkungen

Damit die Funktion sinnvolle Ergebnisse liefert, sollte *Zeitstempelausdruck* den angeforderten *Datumsteil*-Anteil enthalten.

Beispiel

Die folgende Anweisung gibt den Wert 5 zurück:

```
SELECT DATEPART( month , '1987/05/02' );
```

Im folgenden Beispiel wird eine Tabelle namens TableStatistics erstellt und in diese wird die Gesamtanzahl der Aufträge pro Jahr eingefügt, wie sie in der Tabelle "SalesOrders" gespeichert sind:

```
CREATE TABLE TableStatistics (
    ID INTEGER NOT NULL DEFAULT AUTOINCREMENT,
    Year INT,
    NumberOrders INT );
INSERT INTO TableStatistics ( Year, NumberOrders )
SELECT DATEPART( Year, OrderDate ), COUNT(*)
FROM GROUPO.SalesOrders
GROUP BY DATEPART( Year, OrderDate );
```

DATETIME-Funktion [Datum und Uhrzeit]

Konvertiert einen Ausdruck in einen TIMESTAMP-Wert.

Syntax

DATETIME(*expression*)

Parameter

- **expression** Der zu konvertierende Ausdruck. Er ist üblicherweise eine Zeichenfolge.

Rückgabe

TIMESTAMP

Bemerkungen

Der Versuch, numerische Werte zu konvertieren, erzeugt einen Fehler.

Siehe auch

- [„CAST-Funktion \[Datentypkonvertierung\]“ auf Seite 330](#)

Beispiel

Die folgende Anweisung gibt einen Zeitstempel mit dem Wert "1998-09-09 12:12:12.000" zurück:

```
SELECT DATETIME( '1998-09-09 12:12:12.000' );
```

DAY-Funktion [Datum und Uhrzeit]

Gibt den Tag des Monats für das Argument als Ganzzahl zwischen 1 und 31 zurück.

Syntax

DAY(*date-expression*)

Parameter

- **Datumsausdruck** Das Datum als DATE-Datentyp.

Rückgabe

SMALLINT

Bemerkungen

Die DAY-Funktion gibt eine Ganzzahl zwischen 1 und 31 zurück, entsprechend dem Tag des Monats im Argument.

Beispiel

Die folgende Anweisung gibt den Wert 12 zurück:

```
SELECT DAY( '2001-09-12' );
```

DAYNAME-Funktion [Datum und Uhrzeit]

Gibt den Namen des Wochentags von einem Datum zurück.

Syntax

DAYNAME(*date-expression*)

Parameter

- **Datumsausdruck** Das Datum.

Rückgabe

VARCHAR

Bemerkungen

Die Tagesnamen im Deutschen werden folgendermaßen zurückgegeben: Sonntag, Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag.

Beispiel

Die folgende Anweisung gibt den Wert "Samstag" zurück:

```
SELECT DAYNAME ( '1987/05/02' );
```

DAYS-Funktion [Datum und Uhrzeit]

Bearbeitet einen TIMESTAMP-Wert oder gibt die Anzahl der Tage zwischen zwei TIMESTAMP-Werten zurück. Spezifische Details finden Sie im folgenden Abschnitt "Bemerkungen".

Syntax 1

DAYS(*timestamp-expression*)

Syntax 2

DAYS(*timestamp-expression*, *timestamp-expression*)

Syntax 3

DAYS(*timestamp-expression*, *integer-expression*)

Parameter

- **Zeitstempelausdruck** Ein TIMESTAMP-Wert.
- **Ganzzahlausdruck** Die Anzahl der Tage, die zu *Zeitstempelausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl an Tagen von *Zeitstempelausdruck* abgezogen. Wenn Sie einen Ganzzahlausdruck angeben, muss *Zeitstempelausdruck* explizit als TIME-, DATE- oder TIMESTAMP-Wert festgelegt sein. Wenn *Zeitstempelausdruck* ein TIME-Wert ist, wird das aktuelle Datum angenommen.

Hinweise zum Casting von Datentypen finden Sie unter „[CAST-Funktion \[Datentypkonvertierung\]](#)“ auf Seite 330.

Rückgabe

INTEGER bei Syntax 1 und Syntax 2.

TIMESTAMP bei Syntax 3.

Bemerkungen

Das Ergebnis der DAYS-Funktion hängt von deren Argumenten ab. Die DAYS-Funktion ignoriert Stunden, Minuten, und Sekunden in ihren Argumenten.

- **Syntax 1** Wenn Sie einen einzelnen *Zeitstempelausdruck* an die DAYS-Funktion übergeben, wird die Anzahl der Tage zwischen 0000-02-29 und *Zeitstempelausdruck* als INTEGER-Wert zurückgegeben.

Hinweis

"0000-02-29" gibt kein tatsächliches Datum wieder. Es ist das von der DAYS-Funktion verwendete Standarddatum.

- **Syntax 2** Wenn Sie zwei TIMESTAMP-Werte an die DAYS-Funktion übergeben, liefert die Funktion die als Ganzzahl ausgedrückte Anzahl der Tage zwischen den beiden.

- **Syntax 3** Wenn Sie einen TIMESTAMP-Wert und eine Ganzzahl an die DAYS-Funktion übergeben, liefert die Funktion das TIMESTAMP-Ergebnis nach dem Addieren der durch die Ganzzahl ausgedrückten Anzahl an Tagen zum *Zeitstempelausdruck*-Argument.

Verwenden Sie statt Syntax 2 die DATEDIFF-Funktion. Verwenden Sie statt Syntax 3 die DATEADD-Funktion.

Siehe auch

- „DATEDIFF-Funktion [Datum und Uhrzeit]“ auf Seite 344
- „DATEADD-Funktion [Datum und Uhrzeit]“ auf Seite 343

Beispiel

Die folgende Anweisung gibt die Ganzzahl 729889 zurück:

```
SELECT DAYS( '1998-07-13 06:07:12' );
```

Die folgenden Anweisungen geben den Ganzzahlwert -366 zurück, womit angezeigt wird, dass der zweite DATE-Wert 366 Tage vor dem ersten liegt. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEDIFF):

```
SELECT DAYS( '1998-07-13 06:07:12',
             '1997-07-12 10:07:12' );
```

```
SELECT DATEDIFF( day,
                 '1998-07-13 06:07:12',
                 '1997-07-12 10:07:12' );
```

Die folgenden Anweisungen geben den TIMESTAMP-Wert 1999-07-14 00:00:00.000 zurück. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEADD):

```
SELECT DAYS( CAST('1998-07-13' AS DATE ), 366 );
```

```
SELECT DATEADD( day, 366, '1998-07-13' );
```

DB_PROPERTY-Funktion [System]

Gibt den Wert der gegebenen Eigenschaft zurück. Nicht unterstützt von UltraLite Java Edition-Datenbanken.

Syntax

```
DB_PROPERTY( property-name )
```

Parameter

- **Eigenschaftsname** Der Name der Datenbankeigenschaft.

Rückgabe

VARCHAR, LONG VARCHAR

Bemerkungen

Gibt eine Zeichenfolge zurück.

Um eine Option in UltraLite festzulegen, verwenden Sie die SET OPTION-Anweisung oder die API-spezifische Methode zum Festlegen der Datenbankoptionen.

Privilegien

Keine Privilegien sind erforderlich, um diese Funktion für die aktuelle Datenbank auszuführen. Um diese Funktion für andere Datenbanken ausführen zu können, benötigen Sie das SERVER OPERATOR-Systemprivileg oder das MONITOR-Systemprivileg.

NULL wird zurückgegeben, wenn Sie einen ungültigen Parameterwert angeben oder eines der erforderlichen Systemprivilegien nicht haben.

Siehe auch

- „UltraLite-Datenbankeigenschaften“ auf Seite 191
- „SET OPTION-Anweisung [UltraLite]“ auf Seite 459
- ULConnection.SetDatabaseOption-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- ULDatabaseSchema.SetDatabaseOption-Methode [UltraLite.NET] [*UltraLite - .NET-Programmierung*]

Beispiel

Die folgende Anweisung gibt die Seitengröße der aktuellen Datenbank in Byte zurück:

```
SELECT DB_PROPERTY( 'page_size');
```

DEGREES-Funktion [Numerisch]

Konvertiert eine Zahl von Bogenmaß in Grad.

Syntax

DEGREES(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Ein Winkel im Bogenmaß

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihr Argument in DOUBLE, führt die Berechnung als doppelte genaue Gleitkommazahl durch und gibt die Gradzahl des durch *Numerischer_Ausdruck* angegebenen Winkels zurück. Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Beispiel

Die folgende Anweisung gibt den Wert 29,79380534680281 zurück:

```
SELECT DEGREES( 0.52 );
```

DIFFERENCE-Funktion [Zeichenfolge]

Gibt die Differenz der SOUNDEX-Werte zwischen den beiden Zeichenfolgenausdrücken zurück.

Syntax

DIFFERENCE (*string-expression-1*, *string-expression-2*)

Parameter

- **Zeichenfolgenausdruck_1** Das erste SOUNDEX-Argument.
- **Zeichenfolgenausdruck_2** Das zweite SOUNDEX-Argument.

Rückgabe

SMALLINT

Bemerkungen

Die DIFFERENCE-Funktion vergleicht die SOUNDEX-Werte von zwei Zeichenfolgen und berechnet die Ähnlichkeit zwischen ihnen, wobei ein Wert von "0" bis "4" zurückgegeben wird ("4" entspricht der höchsten Übereinstimmung).

Diese Funktion gibt immer einen Wert zurück. Das Ergebnis ist nur NULL, wenn eines der Argumente NULL ist.

Siehe auch

- „SOUNDEX-Funktion [Zeichenfolge]“ auf Seite 397
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung liefert die Ähnlichkeit zwischen den Wörtern "test" und "chest":

```
SELECT DIFFERENCE( 'test', 'chest' );
```

DOW-Funktion [Datum und Uhrzeit]

Gibt eine Zahl von 1 bis 7 zurück, die den Wochentag eines angegebenen Datums repräsentiert, wobei Sonntag=1, Montag=2 usw. ist.

Syntax

DOW(*date-expression*)

Parameter

- **Datumsausdruck** Der Wert (vom Typ DATE), der ausgewertet werden soll.

Rückgabe

SMALLINT

Bemerkungen

Die DOW-Funktion wird durch den in der Datenbankoption `first_day_of_week` angegebenen Wert nicht beeinflusst. Beispiel: Selbst wenn `first_day_of_week` auf Montag gesetzt ist, gibt die DOW-Funktion eine 2 für Montag zurück.

Beispiel

Die folgende Anweisung gibt den Wert 5 zurück:

```
SELECT DOW( '1998-07-09' );
```

Die folgende Anweisung gibt den Wert 1 zurück:

```
SELECT DOW( CAST( '2010/05/30 11:33:00.000000+04:00' as TIMESTAMP WITH TIME ZONE ) );
```

Die folgende Anweisung fragt die Tabelle "Employees" ab und gibt den StartDate-Wert des Mitarbeiters zurück, ausgedrückt als die Nummer des Wochentags:

```
SELECT DOW( StartDate ) FROM GROUPO.Employees;
```

EXP-Funktion [Numerisch]

Gibt das Ergebnis der Berechnung der Basis des natürlichen Logarithmus (e) zurück, potenziert mit dem angegebenen Argument. Nicht unterstützt von UltraLite Java Edition-Datenbanken.

Syntax

EXP(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Der Exponent.

Rückgabe

DOUBLE

Bemerkungen

Das Ergebnis der EXP-Funktion ist die Basis des natürlichen Logarithmus (e), potenziert mit dem durch *Numerischer_Ausdruck* angegebenen Wert.

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltgenauem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück. Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Beispiel

Die Anweisung gibt den Wert 3269017,3724721107 zurück:

```
SELECT EXP( 15 );
```

EXPLANATION-Funktion [Verschiedene]

Gibt die Optimierungsstrategie einer SQL-Anweisung als normale Textzeichenfolge zurück.

Syntax

```
EXPLANATION(  
  string-expression  
  [ , cursor-type ]  
  [ , update-status ]  
)
```

```
EXPLANATION( string-expression )
```

Parameter

- **Zeichenfolgenausdruck** Die SQL-Anweisung, die gewöhnlich eine SELECT-Anweisung ist, aber auch eine UPDATE-, MERGE- oder DELETE-Anweisung sein kann.

Rückgabe

LONG VARCHAR

Bemerkungen

Der Zugriffsplan der Anweisung wird als Zeichenfolge zurückgegeben. Hinweise zum Interpretieren des Ergebnisses finden Sie unter „[Fortgeschrittene Aufgaben: Abfrageausführungspläne](#)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*].

Die GRAPHICAL_PLAN-Funktion bietet deutlich mehr Informationen über Zugriffspläne, einschließlich der Systemeigenschaften die möglicherweise beeinflusst haben, wie die Anweisung optimiert wurde.

Anhand dieser Informationen können Sie entscheiden, ob Sie Indizes hinzufügen oder wie Sie Ihre Datenbank zur Steigerung der Performance strukturieren sollen.

Siehe auch

- „Ausführungspläne in UltraLite“ auf Seite 477

Beispiel

Die folgende Anweisung übergibt eine SELECT-Anweisung als Zeichenfolgenparameter und gibt den Ausführungsplan für die Abfrage zurück:

```
SELECT EXPLANATION( 'SELECT * FROM Departments WHERE DepartmentID > 100' );
```

FLOOR-Funktion [Numerisch]

Gibt die größte Ganzzahl zurück, die nicht größer ist als die angegebene Zahl.

Syntax

```
FLOOR( numeric-expression )
```

Parameter

- **Nummerischer_Ausdruck** Der Wert, der gekürzt werden soll, in der Regel eine fester numerischer Typ, bei dem die Anzahl der Dezimalstellen nicht Null ist, oder ein angenäherter numerischer Datentyp (DOUBLE, REAL oder FLOAT).

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelte genaue Gleitkommazahl durch.

Siehe auch

- [„CEILING-Funktion \[Numerisch\]“ auf Seite 332](#)

Beispiel

Die folgende Anweisung gibt einen FLOOR-Wert von "123" zurück:

```
SELECT FLOOR (123);
```

Die folgende Anweisung gibt einen FLOOR-Wert von 123 zurück:

```
SELECT FLOOR (123.45);
```

Die folgende Anweisung gibt einen FLOOR-Wert von 124 zurück:

```
SELECT FLOOR (-123.45);
```

GETDATE-Funktion [Datum und Uhrzeit]

Gibt die aktuellen Werte für Jahr, Monat, Tag, Stunde, Minute, Sekunde und Sekundenbruchteil zurück.

Syntax

GETDATE()

Rückgabe

TIMESTAMP

Bemerkungen

Die Genauigkeit ist durch die Genauigkeit der Systemuhr begrenzt.

Die von der GETDATE-Funktion zurückgegebenen Werte ähneln den von der NOW-Funktion zurückgegebenen Angaben und dem Spezialwert von CURRENT TIMESTAMP.

Siehe auch

- [„NOW-Funktion \[Datum und Uhrzeit\]“ auf Seite 379](#)

Beispiel

Die folgende Anweisung gibt das Systemdatum und die Systemzeit zurück:

```
SELECT GETDATE( );
```

GREATER-Funktion [Verschiedene]

Gibt den größeren von zwei Parameterwerten zurück.

Syntax

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

Parameter

- ***expression-1*** Der erste Parameterwert, der verglichen werden soll.
- ***expression-2*** Der zweite Parameterwert, der verglichen werden soll.

Rückgabe

Der Rückgabetyt dieser Funktion hängt von den angegebenen Ausdrücken ab. Das bedeutet, dass der Datenbankserver beim Auswerten der Funktion zuerst nach einem Datentyp sucht, mit dem alle Ausdrücke verglichen werden können. Wenn er gefunden wird, vergleicht der Datenbankserver die Ausdrücke und gibt das Ergebnis in dem für den Vergleich verwendeten Datentyp zurück. Wenn der Datenbankserver keinen gemeinsamen Vergleichsdattentyp findet, wird ein Fehler zurückgegeben.

Bemerkungen

Wenn die Parameter gleich sind, wird der erste zurückgegeben.

Siehe auch

- „[LESSER-Funktion \[Verschiedene\]](#)“ auf Seite 366

Beispiel

Die folgende Anweisung gibt den Wert 10 zurück:

```
SELECT GREATER( 10, 5 ) FROM dummy;
```

HEXTOINT-Funktion [Datentypkonvertierung]

Gibt das dezimale Ganzzahl-Äquivalent einer hexadezimalen Zeichenfolge zurück

Die Funktionen CAST, CONVERT, HEXTOINT und INTTOHEX können benutzt werden, um in und aus hexadezimalen Werten zu konvertieren. Weitere Hinweise zur Verwendung dieser Funktionen finden Sie unter [Konvertierung in und aus hexadezimalen Werten \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#).

Syntax

```
HEXTOINT( hexadecimal-string )
```

Parameter

- **hexadecimal-string** Die Zeichenfolge, die in eine Ganzzahl konvertiert werden soll.

Rückgabe

Die HEXTOINT-Funktion gibt das plattformunabhängige SQL INTEGER-Äquivalent der hexadezimalen Zeichenfolge als INT-Wert zurück. Der hexadezimale Wert stellt eine negative Ganzzahl dar, wenn die 8. Ziffer von rechts aus den Ziffern 8-9 besteht und wenn die Buchstaben A-F in Groß- oder Kleinschreibung und die vorhergehenden führenden Ziffern alle aus dem Buchstaben F in Groß- oder Kleinschreibung bestehen. Das Folgende ist keine zulässige Verwendung von HEXTOINT, weil das Argument einen positiven Ganzzahlwert darstellt, der nicht als 32-Bit-Ganzzahl mit Vorzeichen dargestellt werden kann:

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

Bemerkungen

Die HEXTOINT-Funktion akzeptiert Zeichenfolgenlitterale oder Variable, die ausschließlich aus Ziffern und den groß- oder kleingeschriebenen Buchstaben A-F mit oder ohne 0x-Präfix bestehen. Die folgenden Verwendungen von HEXTOINT sind zulässig:

```
SELECT HEXTOINT( '0xFFFFFFFF' );
SELECT HEXTOINT( '0x00000100' );
SELECT HEXTOINT( '100' );
SELECT HEXTOINT( '0xfffffffff80000001' );
```

Die HEXTOINT-Funktion entfernt ggf. das 0x-Präfix. Wenn die Daten 8 Stellen überschreiten, müssen sie einen Wert darstellen, der als 32-Bit-Ganzzahlwert mit Vorzeichen dargestellt werden kann.

Siehe auch

- [„INTTOHEX-Funktion \[Datentypkonvertierung\]“ auf Seite 362](#)

Beispiel

Die folgende Anweisung gibt den Wert 420 zurück:

```
SELECT HEXTOINT( '1A4' );
```

HOUR-Funktion [Datum und Uhrzeit]

Gibt die Stundenkomponente eines TIMESTAMP-Werts zurück.

Syntax

HOUR(*timestamp-expression*)

Parameter

- **Zeitstempelausdruck** Ein TIMESTAMP-Wert.

Rückgabe

SMALLINT

Bemerkungen

Der zurückgegebene Wert ist der Stundenteil des TIMESTAMP-Ausdrucks, ein SMALLINT-Wert zwischen 0 und 23.

Beispiel

Die folgende Anweisung liefert den Wert 21:

```
SELECT HOUR( '1998-07-09 21:12:13' );
```

HOURS-Funktion [Datum und Uhrzeit]

Bearbeitet einen TIMESTAMP-Wert oder gibt die Anzahl von Stunden zwischen zwei TIMESTAMP-Werten zurück. Spezifische Details finden Sie unter dem Verwendungszweck dieser Funktion.

Syntax 1

HOURS (*timestamp-expression*)

Syntax 2

HOURS (*timestamp-expression*, *timestamp-expression*)

Syntax 3

HOURS (*time-or-timestamp-expression*, *integer-expression*)

Parameter

- **Zeit_oder_Zeitstempelausdruck** Ein Wert vom Typ TIME oder TIMESTAMP.
- **Zeitstempelausdruck** Ein Wert vom Typ TIMESTAMP.
- **Ganzzahlausdruck** Die Anzahl der Stunden, die zu *Zeit_oder_Zeitstempelausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl an Stunden von *Zeit_oder_Zeitstempelausdruck* abgezogen.

Hinweise zum Casting von Datentypen finden Sie unter [„CAST-Funktion \[Datentypkonvertierung\]“ auf Seite 330](#).

Rückgabe

INTEGER bei Syntax 1 und Syntax 2.

TIME oder TIMESTAMP bei Syntax 3.

Bemerkungen

Das Ergebnis der HOURS-Funktion hängt von deren Argumenten ab.

- **Syntax 1** Wenn Sie einen einzelnen *Zeitstempelausdruck* an die HOURS-Funktion übergeben, wird die Anzahl der Stunden zwischen Mitternacht am 0000-02-29 und *Zeitstempelausdruck* als INTEGER-Wert zurückgegeben.

Hinweis

"0000-02-29" gibt kein tatsächliches Datum wieder. Es ist der von der HOURS-Funktion standardmäßig verwendete TIMESTAMP-Wert.

- **Syntax 2** Wenn Sie zwei TIMESTAMP-Werte an die HOURS-Funktion übergeben, liefert die Funktion die als Ganzzahl ausgedrückte Anzahl der Stunden zwischen den beiden.
- **Syntax 3** Wenn Sie einen TIMESTAMP-Wert und einen INTEGER-Wert an die HOURS-Funktion übergeben, liefert die Funktion das TIMESTAMP-Ergebnis nach dem Addieren der durch die Ganzzahl ausgedrückten Anzahl an Stunden zum *Zeit_oder_Zeitstempelausdruck*-Argument. Wenn Sie einen TIME-Wert als erstes Argument übergeben, wird entsprechend ein TIME-Wert als Ergebnis zurückgegeben. Syntax 3 unterstützt keine implizite Konvertierung des ersten Arguments. Es kann erforderlich sein, das erste Argument explizit in einen DATE-, TIME- oder TIMESTAMP-Wert zu konvertieren. Wenn das erste Argument ein DATE-Wert ist, wird Mitternacht für die Zeitangabe angenommen.

Verwenden Sie statt Syntax 2 die DATEDIFF-Funktion. Verwenden Sie statt Syntax 3 die DATEADD-Funktion.

Siehe auch

- [„DATEDIFF-Funktion \[Datum und Uhrzeit\]“ auf Seite 344](#)
- [„DATEADD-Funktion \[Datum und Uhrzeit\]“ auf Seite 343](#)

Beispiel

Die folgenden Anweisungen geben den Wert "4" zurück, womit angezeigt wird, dass der zweite TIMESTAMP-Wert vier Stunden nach dem ersten liegt. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEDIFF).

```
SELECT HOURS( '1999-07-13 06:07:12', '1999-07-13 10:07:12' );  
SELECT DATEDIFF( hour, '1999-07-13 06:07:12', '1999-07-13 10:07:12' );
```

Die folgende Anweisung gibt den Wert 17517342 zurück:

```
SELECT HOURS( '1998-07-13 06:07:12' );
```

Die folgenden Anweisungen geben den DATETIME-Wert 1999-05-13 02:05:07.000 zurück. Es wird empfohlen, dass Sie das zweite Beispiel (DATEADD) verwenden.

```
SELECT HOURS( CAST( '1999-05-12 21:05:07' AS DATETIME ), 5 );  
SELECT DATEADD( hour, 5, '1999-05-12 21:05:07' );
```

IFNULL-Funktion [Verschiedene]

Wenn der erste Ausdruck Null ist, wird der Wert des zweiten Ausdrucks zurückgegeben. Wenn der erste Ausdruck nicht NULL ist, wird der Wert des dritten Ausdrucks zurückgegeben. Wenn der erste Ausdruck nicht NULL ist und es keinen dritten Ausdruck gibt, wird NULL zurückgegeben.

Syntax

IFNULL(*expression-1*, *expression-2* [, *expression-3*])

Parameter

- ***expression-1*** Der zu untersuchende Ausdruck. Sein Wert bestimmt, ob *expression-2* oder *expression-3* zurückgegeben wird.
- ***expression-2*** Der Rückgabewert, wenn *expression-1* NULL ist
- ***expression-3*** Der Rückgabewert, wenn *expression-1* nicht NULL ist

Rückgabe

Welcher Datentyp zurückgegeben wird, hängt vom Datentyp von *expression-2* und *expression-3* ab.

Beispiel

Die folgende Anweisung gibt den Wert -66 zurück:

```
SELECT IFNULL( NULL, -66 );
```

Die folgende Anweisung gibt NULL zurück, weil der erste Ausdruck nicht NULL ist und es keinen dritten Ausdruck gibt:

```
SELECT IFNULL( -66, -66 );
```

INSERTSTR-Funktion [Zeichenfolge]

Fügt an einer festgelegten Position eine Zeichenfolge in eine andere Zeichenfolge ein.

Syntax

INSERTSTR(*integer-expression*, *string-expression-1*, *string-expression-2*)

Parameter

- ***Ganzzahlausdruck*** Die Position, nach der die Zeichenfolge eingefügt werden soll. Verwenden Sie die Null, um eine Zeichenfolge am Beginn einzufügen.
- ***Zeichenfolgenausdruck_1*** Die Zeichenfolge, in die die andere Zeichenfolge eingefügt werden soll
- ***Zeichenfolgenausdruck_2*** Die einzufügende Zeichenfolge.

Rückgabe

LONG VARCHAR

Siehe auch

- „STUFF-Funktion [Zeichenfolge]“ auf Seite 407
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert backoffice zurück:

```
SELECT INSERTSTR( 0, 'office ', 'back' );
```

INTTOHEX-Funktion [Datentypkonvertierung]

Gibt die Zeichenfolge zurück, die das hexadezimale Äquivalent einer Ganzzahl darstellt.

Syntax

INTTOHEX(*integer-expression*)

Parameter

- **Ganzzahlausdruck** Die Ganzzahl, die in das hexadezimale Format konvertiert wird

Rückgabe

VARCHAR

Bemerkungen

Die Funktionen CAST, CONVERT, HEXTOINT und INTTOHEX können benutzt werden, um in und aus hexadezimalen Werten zu konvertieren.

Siehe auch

- [„HEXTOINT-Funktion \[Datentypkonvertierung\]“ auf Seite 357](#)
- [Konvertierung in und aus hexadezimalen Werten \[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)

Beispiel

Die folgende Anweisung gibt den Wert 0000009c zurück:

```
SELECT INTTOHEX( 156 );
```

ISDATE-Funktion [Datentypkonvertierung]

Testet, ob ein Zeichenfolgenargument in ein Datum konvertiert werden kann.

Syntax

ISDATE(*string*)

Parameter

- **string** Die Zeichenfolge, die analysiert wird, um herauszufinden, ob sie ein gültiges Datum darstellt.

Rückgabe

INT

Bemerkungen

Wenn eine Konvertierung möglich ist, gibt die Funktion 1 zurück, sonst 0. Wenn das Argument NULL ist, wird "0" zurückgegeben.

ISNULL-Funktion [Verschiedene]

Gibt den ersten Nicht-NULL-Ausdruck in einer Liste zurück. Diese Funktion ist mit der COALESCE-Funktion identisch.

Syntax

ISNULL(*expression*, *expression* [, ...])

Parameter

- **expression** Ein Ausdruck, der auf NULL getestet wird.

Mindestens zwei Ausdrücke müssen an die Funktion übergeben werden und alle Ausdrücke müssen vergleichbar sein.

Rückgabe

Der Rückgabotyp dieser Funktion hängt von den angegebenen Ausdrücken ab. Das bedeutet, dass der Datenbankserver beim Auswerten der Funktion zuerst nach einem Datentyp sucht, mit dem alle Ausdrücke verglichen werden können. Wenn er gefunden wird, vergleicht der Datenbankserver die Ausdrücke und gibt das Ergebnis in dem für den Vergleich verwendeten Datentyp zurück. Wenn der Datenbankserver keinen gemeinsamen Vergleichsdattentyp findet, wird ein Fehler zurückgegeben.

Siehe auch

- „[COALESCE-Funktion \[Verschiedene\]](#)“ auf Seite 336

Beispiel

Die folgende Anweisung gibt den Wert -66 zurück:

```
SELECT ISNULL( NULL , -66, 55, 45, NULL, 16 );
```

LCASE-Funktion [Zeichenfolge]

Konvertiert alle Zeichen in einer Zeichenfolge in Kleinbuchstaben.

Syntax

LCASE(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, die in Kleinbuchstaben konvertiert werden soll

Rückgabe

- CHAR
- LONG VARCHAR
- VARCHAR

Bemerkungen

Die LCASE-Funktion mit der LOWER-Funktion identisch.

Siehe auch

- „LOWER-Funktion [Zeichenfolge]“ auf Seite 369
- „UCASE-Funktion [Zeichenfolge]“ auf Seite 415
- „UPPER-Funktion [Zeichenfolge]“ auf Seite 415
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert `chocolate` zurück:

```
SELECT LCASE( 'ChoCoLatE' );
```

LEFT-Funktion [Zeichenfolge]

Gibt mehrere Zeichen ab dem Beginn einer Zeichenfolge zurück.

Syntax

LEFT(*string-expression*, *integer-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge.
- **Ganzzahlausdruck** Die Anzahl der zurückzugebenden Zeichen

Rückgabe

LONG VARCHAR

Bemerkungen

Wenn eine Zeichenfolge Mehrbytezeichen enthält und die korrekte Kollation verwendet wird, ist die Anzahl der zurückgegebenen Bytes möglicherweise größer als die festgelegte Anzahl von Zeichen.

Sie können einen *Ganzzahlausdruck* angeben, der größer ist als der Wert im Zeichenfolgenausdruck-Argument. In diesem Fall wird der gesamte Wert zurückgegeben.

Wenn in der Eingabezeichenfolge Zeichenlängensemantik verwendet wurde, wird der Rückgabewert soweit wie möglich mit Ausdrücken der Zeichenlängensemantik beschrieben.

Siehe auch

- „RIGHT-Funktion [Zeichenfolge]“ auf Seite 390
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt die ersten 5 Zeichen jedes Surname-Werts in der Tabelle "Customers" zurück:

```
SELECT LEFT( Surname, 5) FROM GROUPO.Customers;
```

LENGTH-Funktion [Zeichenfolge]

Gibt die Anzahl von Zeichen in der Zeichenfolge zurück.

Syntax

```
LENGTH( string-expression )
```

Parameter

- **Zeichenfolgenderausdruck** Die Zeichenfolge.

Rückgabe

INT

Bemerkungen

Mit dieser Funktion bestimmen Sie die Länge einer Zeichenfolge. Geben Sie beispielsweise einen Spaltennamen für *Zeichenfolgenderausdruck* an, um die Länge der Werte in der Spalte zu bestimmen.

Wenn eine Zeichenfolge Mehrbytezeichen enthält und die korrekte Kollation verwendet wird, gibt LENGTH die Anzahl der Zeichen und nicht die Byte-Anzahl zurück. Wenn der Zeichenfolgenderausdruck ein BINARY-Datentyp ist, verhält sich die LENGTH-Funktion wie die BYTE_LENGTH-Funktion.

Hinweis

Sie können die LENGTH-Funktion und die CHAR_LENGTH-Funktion gleichermaßen für CHAR-, VARCHAR- und LONG VARCHAR-Datentypen verwenden. Sie müssen allerdings die LENGTH-Funktion bei BINARY- und Bit-Array-Datentypen verwenden.

Siehe auch

- „BYTE_LENGTH-Funktion [Zeichenfolge]“ auf Seite 328
- „Internationale Sprachen und Zeichensätze“ [SQL Anywhere Server - Datenbankadministration]
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert 9 zurück:

```
SELECT LENGTH( 'chocolate' );
```

LESSER-Funktion [Verschiedene]

Gibt den kleineren von zwei Parameterwerten zurück.

Syntax

LESSER(*expression-1*, *expression-2*)

Parameter

- ***expression-1*** Der erste Parameterwert, der verglichen werden soll.
- ***expression-2*** Der zweite Parameterwert, der verglichen werden soll.

Rückgabe

Der Rückgabetyt dieser Funktion hängt von den angegebenen Ausdrücken ab. Das bedeutet, dass der Datenbankserver beim Auswerten der Funktion zuerst nach einem Datentyp sucht, mit dem alle Ausdrücke verglichen werden können. Wenn er gefunden wird, vergleicht der Datenbankserver die Ausdrücke und gibt das Ergebnis in dem für den Vergleich verwendeten Datentyp zurück. Wenn der Datenbankserver keinen gemeinsamen Vergleichsdattentyp findet, wird ein Fehler zurückgegeben.

Bemerkungen

Wenn die Parameter gleich sind, wird der erste Wert zurückgegeben.

Siehe auch

- „[GREATER-Funktion \[Verschiedene\]](#)“ auf Seite 357

Beispiel

Die folgende Anweisung gibt den Wert 5 zurück:

```
SELECT LESSER( 10, 5 ) FROM dummy;
```

LIST-Funktion [Aggregat]

Gibt eine durch Begrenzer-Zeichenfolgen getrennte Liste von Werten für jede Zeile in einer Gruppe zurück.

Syntax

```
LIST(  
[ DISTINCT ] string-expression  
[ , delimiter-string ] )
```

Parameter

- ***Zeichenfolgenausdruck*** Ein Zeichenfolgenausdruck, normalerweise ein Spaltenname. Für jede Zeile in der Gruppe wird der Wert von *Zeichenfolgenausdruck* zur Ergebniszeichenfolge hinzugefügt, wobei die Werte durch *Trennzeichenfolge* getrennt werden. Wenn **DISTINCT** angegeben wurde, werden nur eindeutige *Zeichenfolgenausdruck*-Werte hinzugefügt.

- ***delimiter-string*** Eine Trennzeichenfolge für die Listeneinträge. Die Standardeinstellung ist ein Komma. Wenn NULL oder eine leere Zeichenfolge angegeben wird, gibt es keinen Begrenzer. *delimiter-string* muss eine Konstante sein.

Rückgabe

LONG VARCHAR

Bemerkungen

Die LIST-Funktion gibt die Verkettung (mit Begrenzern) aller Nicht-NULL-Werte für X für jede Zeile der Gruppe zurück. Wenn in der Gruppe nicht wenigstens eine Zeile mit einem definierten X-Wert existiert, gibt LIST(X) eine leere Zeichenfolge zurück.

NULL-Werte und leere Zeichenfolgen werden von der LIST-Funktion ignoriert.

Eine LIST-Funktion kann nicht als Fensterfunktion verwendet werden, aber sie kann als Eingabe für eine Fensterfunktion verwendet werden.

Beispiele

Die folgende Anweisung gibt alle in der Tabelle "Employees" enthaltenen Straßen zurück:

```
SELECT LIST( Street ) FROM GROUPO.Employees;
```

LOCATE-Funktion [Zeichenfolge]

Gibt die Position einer Zeichenfolge in einer anderen zurück.

Syntax

LOCATE(*string-expression-1*, *string-expression-2* [, *integer-expression*])

Parameter

- ***Zeichenfolgenausdruck_1*** Die zu durchsuchende Zeichenfolge.
- ***Zeichenfolgenausdruck_2*** Die Zeichenfolge, nach der gesucht wird.
- ***Ganzzahlausdruck*** Die Zeichenposition in der Zeichenfolge, an der die Suche beginnen soll. Das erste Zeichen hat Position 1. Wenn das Start-Offset negativ ist, gibt die locate-Funktion anstatt des ersten den letzten passenden Zeichenfolgen-Offset zurück. Ein negativer Offset gibt an, wieviel vom Ende einer Zeichenfolge von der Suche auszunehmen ist. Die Anzahl der ausgeschlossenen Byte wird mit der folgenden Formel berechnet: $(-1 * \text{Offset}) - 1$.

Rückgabe

INT

Bemerkungen

Wenn *Ganzzahlausdruck* angegeben ist, beginnt die Suche an dieser Position der Zeichenfolge.

Die erste Zeichenfolge kann lang sein (länger als 255 Byte), die zweite ist aber auf 255 Byte begrenzt. Wenn eine lange Zeichenfolge als zweites Argument angegeben wird, gibt die Funktion NULL zurück.

Wenn die Zeichenfolge nicht gefunden wird, wird "0" zurückgegeben. Die Suche nach einer Zeichenfolge mit Länge null gibt "1" zurück. Falls eines der Argumente NULL ist, ist das Ergebnis NULL.

Wenn Mehrbytezeichen mit der richtigen Kollation verwendet werden, können sich die Startposition und der Rückgabewert von den *Byte*-Positionen unterscheiden.

Siehe auch

- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321
- „CHARINDEX-Funktion [Zeichenfolge]“ auf Seite 334

Beispiel

Die folgende Anweisung liefert den Wert 8:

```
SELECT LOCATE(  
    'office party this week - rsvp as soon as possible',  
    'party',  
    2 );
```

LOG-Funktion [Numerisch]

Gibt den natürlichen Logarithmus einer Zahl zurück. Nicht unterstützt von UltraLite Java Edition-Datenbanken.

Syntax

LOG(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Die Zahl.

Rückgabe

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltprecisem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück. Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Bemerkungen

Das Argument ist ein Ausdruck, der den Wert eines beliebigen integrierten numerischen Datentyps zurückgibt.

Siehe auch

- „LOG10-Funktion [Numerisch]“ auf Seite 369

Beispiel

Die folgende Anweisung gibt den natürlichen Logarithmus von 50 zurück:

```
SELECT LOG( 50 );
```

LOG10-Funktion [Numerisch]

Gibt den Logarithmus zur Basis 10 einer Zahl zurück. Nicht unterstützt von UltraLite Java Edition-Datenbanken.

Syntax

LOG10(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Die Zahl.

Rückgabe

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch. Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Bemerkungen

Das Argument ist ein Ausdruck, der den Wert eines beliebigen integrierten numerischen Datentyps zurückgibt.

Siehe auch

- „LOG-Funktion [Numerisch]“ auf Seite 368

Beispiel

Die folgende Anweisung gibt den Logarithmus zur Basis 10 für 50 zurück:

```
SELECT LOG10( 50 );
```

LOWER-Funktion [Zeichenfolge]

Konvertiert alle Zeichen in einer Zeichenfolge in Kleinbuchstaben. Diese Funktion ist mit der LCASE-Funktion identisch.

Syntax

LOWER(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, die in Kleinbuchstaben konvertiert werden soll

Rückgabe

CHAR, VARCHAR, oder LONG VARCHAR entsprechend dem Datentyp des Arguments.

Bemerkungen

Die LCASE-Funktion mit der LOWER-Funktion identisch.

Siehe auch

- „LCASE-Funktion [Zeichenfolge]“ auf Seite 363
- „UCASE-Funktion [Zeichenfolge]“ auf Seite 415
- „UPPER-Funktion [Zeichenfolge]“ auf Seite 415
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert `chocolate` zurück:

```
SELECT LOWER( 'chOCOLate' );
```

LTRIM-Funktion [Zeichenfolge]

Entfernt führende Leerzeichen aus der Zeichenfolge.

Syntax

```
LTRIM( string-expression )
```

Parameter

- **Zeichenfolgenausdruck** Die zu kürzende Zeichenfolge

Rückgabe

- VARCHAR
- LONG VARCHAR

Bemerkungen

Die tatsächliche Länge des Ergebnisses ist die Länge des Ausdrucks minus der Anzahl der entfernten Zeichen. Wenn alle Zeichen entfernt werden, ist das Ergebnis eine leere Zeichenfolge.

Wenn der Parameter NULL sein kann, kann das Ergebnis NULL sein.

Wenn der Parameter NULL ist, ist das Ergebnis NULL.

Siehe auch

- „RTRIM-Funktion [Zeichenfolge]“ auf Seite 391
- „TRIM-Funktion [Zeichenfolge]“ auf Seite 413
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert `Test Message` zurück, wobei alle führenden Leerzeichen entfernt werden:

```
SELECT LTRIM( '      Test Message' );
```

MAX-Funktion [Aggregat]

Gibt den maximalen Wert für den Ausdruck zurück, der in jeder Zeilengruppe gefunden wurde

Syntax 1

MAX([**DISTINCT**] *expression*)

Parameter

- **expression** Der Ausdruck, bei dem der Höchstwert berechnet werden soll. Das ist üblicherweise ein Spaltenname.
- **DISTINCT expression** Gibt denselben Wert wie **MAX(expression)** zurück, er ist nur der Vollständigkeit halber angeführt.

Rückgabe

Derselbe Datentyp wie das Argument.

Bemerkungen

Zeilen, bei denen *expression* NULL ist, werden ignoriert. Gibt NULL für eine Gruppe zurück, die keine Zeilen enthält.

Für einfache Vergleiche von zwei Ausdrücken können Sie auch die GREATER-Funktion verwenden.

Siehe auch

- „MIN-Funktion [Aggregat]“ auf Seite 371
- „Fehlerbehandlung: Aggregatfunktionen und äußere Referenzen“ [*SQL Anywhere 16 - Änderungen und Upgrades*]

Beispiel

Die folgende Anweisung gibt den Wert 138948,000 zurück, der das höchste Gehalt in der Tabelle "Employees" darstellt:

```
SELECT MAX( Salary )  
FROM GROUPO.Employees;
```

MIN-Funktion [Aggregat]

Liefert den minimalen Wert für den Ausdruck, der in jeder Zeilengruppe gefunden wurde

Syntax 1

MIN([**DISTINCT**] *expression*)

Parameter

- **expression** Der Ausdruck, bei dem der minimale Wert berechnet werden soll. Das ist üblicherweise ein Spaltenname.

- **DISTINCT *expression*** Gibt denselben Wert wie MIN(*expression*) zurück, er ist nur der Vollständigkeit halber angeführt.

Rückgabe

Derselbe Datentyp wie das Argument.

Bemerkungen

Zeilen, bei denen *expression* NULL ist, werden ignoriert. Gibt NULL für eine Gruppe zurück, die keine Zeilen enthält.

Für einfache Vergleiche von zwei Ausdrücken können Sie auch die LESSER-Funktion verwenden. Siehe „LESSER-Funktion [Verschiedene]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#).

Siehe auch

- „MAX-Funktion [Aggregat]“ auf Seite 371

Beispiel

Die folgende Anweisung gibt den Wert 24903,000 zurück, der das niedrigste Gehalt in der Tabelle "Employees" darstellt:

```
SELECT MIN( Salary )  
FROM GROUPO.Employees;
```

MINUTE-Funktion [Datum und Uhrzeit]

Gibt die Minutenkomponente eines TIMESTAMP-Werts zurück.

Syntax

MINUTE(*timestamp-expression*)

Parameter

- **Zeitstempelausdruck** Der TIMESTAMP-Wert.

Rückgabe

SMALLINT

Bemerkungen

Der zurückgegebene Wert ist der Minutenteil des TIMESTAMP-Ausdrucks, ein SMALLINT-Wert zwischen 0 und 59.

Beispiel

Die folgende Anweisung gibt den Wert 22 zurück:

```
SELECT MINUTE( '1998-07-13 12:22:34' );
```

MINUTES-Funktion [Datum und Uhrzeit]

Bearbeitet einen TIMESTAMP-Wert oder gibt die Anzahl von Minuten zwischen zwei TIMESTAMP-Werten zurück. Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Syntax 1

MINUTES(*timestamp-expression*)

Syntax 2

MINUTES(*timestamp-expression*, *timestamp-expression*)

Syntax 3

MINUTES(*timestamp-or-time-expression*, *integer-expression*)

Parameter

- **Zeitstempelausdruck** Ein Ausdruck vom Typ TIMESTAMP.
- **Zeitstempel-_oder_Zeitausdruck** Ein Ausdruck vom Typ TIME oder TIMESTAMP.
- **Ganzzahlausdruck** Die Anzahl der Minuten, die zu *Zeitstempel-_oder_Zeitausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl von Minuten von *Zeitstempel-_oder_Zeitausdruck* abgezogen.

Rückgabe

INTEGER bei Syntax 1 und Syntax 2.

TIME oder TIMESTAMP bei Syntax 3.

Bemerkungen

Das Ergebnis der MINUTES-Funktion hängt von deren Argumenten ab.

- **Syntax 1** Wenn Sie einen einzelnen *Zeitstempelausdruck* an die MINUTES-Funktion übergeben, wird die Anzahl der Minuten zwischen Mitternacht am 0000-02-29 und *Zeitstempelausdruck* als INTEGER-Wert zurückgegeben.

Hinweis

"0000-02-29" gibt kein tatsächliches Datum wieder. Es ist das von der MINUTES-Funktion verwendete Standarddatum.

- **Syntax 2** Wenn Sie zwei TIMESTAMP-Werte an die MINUTES-Funktion übergeben, liefert die Funktion die als Ganzzahl ausgedrückte Anzahl der Minuten zwischen den beiden.
- **Syntax 3** Wenn Sie einen TIMESTAMP-Wert und einen INTEGER-Wert an die MINUTES-Funktion übergeben, liefert die Funktion das TIMESTAMP-Ergebnis nach dem Addieren der durch die Ganzzahl ausgedrückten Anzahl an Minuten zum *Zeitstempelausdruck*-Argument. Entsprechend ist, wenn das erste Argument für MINUTES ein TIME-Wert ist, das Ergebnis ebenfalls ein TIME-Wert. Syntax 3 unterstützt keine implizite Konvertierung des ersten Arguments. Es kann erforderlich

sein, das erste Argument explizit in einen DATE-, TIME- oder TIMESTAMP-Wert zu konvertieren. Wenn das erste Argument ein DATE-Wert ist, wird Mitternacht für die Zeitangabe angenommen.

Da die MINUTES-Funktion eine Ganzzahl zurückgibt, kann es zu einem Überlauf kommen, wenn Syntax 1 mit TIMESTAMP-Werten ab 4083-03-23 02:08:00 benutzt wird.

Verwenden Sie statt Syntax 2 die DATEDIFF-Funktion. Verwenden Sie statt Syntax 3 die DATEADD-Funktion.

Siehe auch

- „DATEDIFF-Funktion [Datum und Uhrzeit]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- „DATEADD-Funktion [Datum und Uhrzeit]“ [\[SQL Anywhere Server - SQL-Referenzhandbuch\]](#)
- „CAST-Funktion [Datentypkonvertierung]“ [auf Seite 330](#)

Beispiel

Die folgenden Anweisungen geben den Wert "240" zurück, womit angezeigt wird, dass der zweite TIMESTAMP-Wert 240 Minuten nach dem ersten liegt. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEDIFF).

```
SELECT MINUTES( '1999-07-13 06:07:12',  
               '1999-07-13 10:07:12' );  
  
SELECT DATEDIFF( minute,  
               '1999-07-13 06:07:12',  
               '1999-07-13 10:07:12' );
```

Die folgende Anweisung gibt den Wert 1051040527 zurück:

```
SELECT MINUTES( '1998-07-13 06:07:12' );
```

Die folgenden Anweisungen geben den TIMESTAMP-Wert 1999-05-12 21:10:07.000 zurück. Die erste Anweisung erfordert eine explizite Umwandlung des literalen Zeichenfolgenparameters. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEADD).

```
SELECT MINUTES( CAST( '1999-05-12 21:05:07' AS TIMESTAMP ), 5 );  
  
SELECT DATEADD( minute, 5, '1999-05-12 21:05:07' );
```

ML_GET_SERVER_NOTIFICATION-Funktion [System]

Mit dieser Funktion können UltraLite-Benutzer den Lightweight-Polling-Modus verwenden, um einen Notifier auf einem MobiLink-Server für serverinitiierte Synchronisationsanforderungen abzufragen. Nicht unterstützt von UltraLite Java Edition-Datenbanken.

Syntax

ML_GET_SERVER_NOTIFICATION(*notifier*, *address*, *key*)

Parameter

- **notifier** Der Name des abzurufenden Notifiers auf dem MobiLink-Server.

- **address** Die Datenstromparameter, etwa:

```
tcpip{host=pc1;port=1234}
```

- **key** Optional. Der Anforderungsschlüssel für die Benachrichtigung.

Rückgabe

Gibt den Betreff und den Inhalt einer Benachrichtigungsanforderung für den angegebenen Anforderungsschlüssel zurück.

Bemerkungen

Wenn für den angegebenen Anforderungsschlüssel keine Anforderungen vorhanden sind oder der Notifier-Name auf dem MobiLink-Server nicht existiert, ist das Ergebnis NULL. Wenn für den Anforderungsschlüssel NULL angegeben wird, wird die entfernte ID des Benutzers als Anforderungsschlüssel verwendet. Wenn eine Anforderung vorhanden ist, wird die Meldung in der folgenden Form zurückgegeben: *[Betreff]Inhalt* (Beispiel: [sync]profile1).

Diese Funktion kommuniziert über das Netzwerk, wenn sie Antworten vom MobiLink-Server abrufen. Daher benötigt die Funktion möglicherweise eine sehr lange Ausführungszeit, da die Verarbeitungszeit im Netzwerk einbezogen werden muss. Während der Ausführung kann es zu Perioden kommen, während derer die Funktion im Hintergrund ausgeführt werden kann, sodass Verarbeitungen in der Laufzeit auf anderen Verbindungen durchgeführt werden können. Diese Perioden sind allerdings nicht garantiert und hängen von der Komplexität der SQL-Anweisungen ab. Benutzern wird empfohlen, für den Abruf einer MobiLink-Adresse, die in dieser Funktion verwendet werden soll, die Funktion `sync_profile_option_value` mit einem bestehenden Synchronisationsprofil zu verwenden, um den Wert der Profiloption **Stream** zu erhalten. Der von diesem Funktionsaufruf zurückgegebene Wert kann direkt als MobiLink-Adressenparameter verwendet werden.

Siehe auch

- „[SYNC_PROFILE_OPTION_VALUE-Funktion \[System\]](#)“ auf Seite 411

Beispiel

```
SELECT ML_GET_SERVER_NOTIFICATION('Notifier1',
'tcpip{host=sybase;port=1234}', 'MyKey');
```

MOD-Funktion [Numerisch]

Gibt den Rest zurück, wenn eine Ganzzahl durch eine andere dividiert wird.

Syntax

```
MOD( dividend, divisor )
```

Parameter

- **dividend** Der Dividend oder Zähler der Division.
- **divisor** Der Divisor oder Nenner der Division

Rückgabe

- SMALLINT
- INT
- NUMERIC

Bemerkungen

Eine Division mit einem negativen Dividenten ergibt eine negative Zahl oder eine Null. Das Vorzeichen des Divisors hat keine Auswirkung.

Siehe auch

- [„REMAINDER-Funktion \[Numerisch\]“ auf Seite 387](#)

Beispiel

Die folgende Anweisung gibt den Wert 2 zurück:

```
SELECT MOD( 5, 3 );
```

MONTH-Funktion [Datum und Uhrzeit]

Gibt den Monat des angegebenen Datums zurück.

Syntax

MONTH(*date-expression*)

Parameter

- **Datumsausdruck** Ein Wert vom Typ DATE.

Rückgabe

SMALLINT

Bemerkungen

Der zurückgegebene Wert ist eine Zahl zwischen 1 und 12, entsprechend dem Monat des angegebenen Datums.

Beispiel

Die folgende Anweisung gibt den Wert 7 zurück:

```
SELECT MONTH( '1998-07-13' );
```

MONTHNAME-Funktion [Datum und Uhrzeit]

Gibt den Monatsnamen eines Datums zurück.

Syntax

MONTHNAME(*date-expression*)

Parameter

- **Zeitstempelausdruck** Ein TIMESTAMP-Wert.

Rückgabe

VARCHAR

Bemerkungen

Die MONTHNAME-Funktion gibt eine Zeichenfolge zurück, auch wenn das Ergebnis numerisch ist, wie z.B. "2" für den Monat Februar.

Siehe auch

- [„DATEPART-Funktion \[Datum und Uhrzeit\]“ auf Seite 347](#)

Beispiel

Die folgende Anweisung gibt den Wert September zurück:

```
SELECT MONTHNAME( '1998-09-05' );
```

MONTHS-Funktion [Datum und Uhrzeit]

Bearbeitet einen TIMESTAMP-Wert oder gibt die Anzahl von Monaten zwischen zwei TIMESTAMP-Werten zurück. Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Syntax 1

MONTHS(*timestamp-expression*)

Syntax 2

MONTHS(*timestamp-expression*, *timestamp-expression*)

Syntax 3

MONTHS(*timestamp-expression*, *integer-expression*)

Parameter

- **Zeitstempelausdruck** Ein Datums- und Uhrzeitwert vom Typ TIMESTAMP.
- **Ganzzahlausdruck** Die als Ganzzahl ausgedrückte Anzahl der Monate (vom Typ SMALLINT), die zu *Zeitstempelausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl an Monaten von *Zeitstempelausdruck* abgezogen. Wenn Sie *Ganzzahlausdruck* angeben, muss *Zeitstempelausdruck* explizit als TIME-, DATE- oder TIMESTAMP-Wert festgelegt sein. Wenn *Zeitstempelausdruck* ein TIME-Wert ist, wird der laufende Monat angenommen.

Hinweise zum Casting von Datentypen finden Sie unter [„CAST-Funktion \[Datentypkonvertierung\]“ auf Seite 330](#).

Rückgabe

INTEGER bei Syntax 1 und Syntax 2.

TIMESTAMP bei Syntax 3.

Bemerkungen

Das Ergebnis der MONTHS-Funktion hängt von deren Argumenten ab. Die MONTHS-Funktion ignoriert Stunden, Minuten, und Sekunden in ihren Argumenten.

- **Syntax 1** Wenn Sie einen einzelnen *Zeitstempelausdruck* an die MONTHS-Funktion übergeben, wird die Anzahl der Monate zwischen 0000-02 und *Zeitstempelausdruck* als INTEGER-Wert zurückgegeben.

Hinweis

"0000-02" gibt kein tatsächliches Datum wieder. Es ist das von der MONTHS-Funktion verwendete Standarddatum.

- **Syntax 2** Wenn Sie zwei TIMESTAMP-Werte an die MONTHS-Funktion übergeben, liefert die Funktion die als Ganzzahl ausgedrückte Anzahl der Monate zwischen den beiden.
- **Syntax 3** Wenn Sie einen TIMESTAMP-Wert und einen INTEGER-Wert an die MONTHS-Funktion übergeben, liefert die Funktion das TIMESTAMP-Ergebnis nach dem Addieren der durch die Ganzzahl ausgedrückten Anzahl an Monaten zu *Zeitstempelausdruck*.

Verwenden Sie statt Syntax 2 die DATEDIFF-Funktion. Verwenden Sie statt Syntax 3 die DATEADD-Funktion.

Der Wert von MONTHS wird anhand der Anzahl von Monatsersten zwischen zwei Datumsangaben berechnet.

Siehe auch

- „DATEDIFF-Funktion [Datum und Uhrzeit]“ auf Seite 344
- „DATEADD-Funktion [Datum und Uhrzeit]“ auf Seite 343

Beispiel

Die folgende Anweisung gibt den Wert 2 zurück, womit angezeigt wird, dass das zweite Datum zwei Monate nach dem ersten liegt. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEDIFF).

```
SELECT MONTHS( '1999-07-13 06:07:12', '1999-09-13 10:07:12' );

SELECT DATEDIFF( month,
  '1999-07-13 06:07:12',
  '1999-09-13 10:07:12' );
```

Die folgende Anweisung gibt den Wert 23981 zurück:

```
SELECT MONTHS( '1998-07-13 06:07:12' );
```

Die folgenden Anweisungen geben den TIMESTAMP-Wert 1999-10-12 21:05:07.000 zurück. Es wird empfohlen, das zweite Beispiel zu verwenden (DATEADD):

```
SELECT MONTHS( CAST( '1999-05-12 21:05:07' AS DATETIME ), 5 );

SELECT DATEADD( month, 5, '1999-05-12 21:05:07' );
```

NEWID-Funktion [Verschiedene]

Generiert einen UUID-Wert (universell eindeutiger Bezeichner). Ein UUID-Wert entspricht dem GUID-Wert (global eindeutiger Bezeichner).

Syntax

NEWID()

Parameter

Der NEWID-Funktion sind keine Parameter zugeordnet.

Rückgabe

UNIQUEIDENTIFIER

Bemerkungen

Die NEWID-Funktion kann in einer DEFAULT-Klausel für eine Spalte angewendet werden.

UUIDs können verwendet werden, um Zeilen in einer Tabelle eindeutig zu identifizieren. Ein auf einem Computer erzeugter Wert stimmt mit dem Wert, der auf einem anderen Computer erzeugt wurde, nicht überein, und kann daher als Schlüssel in Synchronisations- und Replikationsumgebungen verwendet werden.

Siehe auch

- „Der Standardwert NEWID“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „STRTOUUID-Funktion [Zeichenfolge]“ auf Seite 407
- „UUIDTOSTR-Funktion [Zeichenfolge]“ auf Seite 416

Beispiel

Die folgende Anweisung erstellt eine Tabelle namens "mytab" mit zwei Spalten. Die Spalte pk hat einen eindeutig bezeichnenden Datentyp und ordnet die Funktion NEWID als den Standardwert zu. Die Spalte c1 hat einen Ganzzahl-Datentyp.

```
CREATE TABLE mytab(  
    pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),  
    c1 INT );
```

Die folgende Anweisung gibt einen eindeutigen Bezeichner als Zeichenfolge zurück:

```
SELECT UUIDTOSTR( NEWID() );
```

Beispiel: Der zurückgegebene Wert könnte "96603324-6FF6-49DE-BF7D-F44C1C7E6856" sein.

NOW-Funktion [Datum und Uhrzeit]

Gibt das aktuelle Datum und die aktuelle Zeit als TIMESTAMP Wert zurück. Die Genauigkeit ist durch die Genauigkeit der Systemuhr begrenzt.

Syntax

NOW([*])

Rückgabe

TIMESTAMP

Bemerkungen

NOW ist gleichwertig mit der GETDATE-Funktion und dem CURRENT TIMESTAMP-Spezialwert. NOW(*) und NOW() sind gleichwertige Konstruktionen.

Jede Instanz der NOW-Funktion in einer Anforderung wird höchstens einmal ausgewertet. Mehrere Instanzen von NOW in derselben Anforderung liefern möglicherweise identische oder auch unterschiedliche TIMESTAMP-Werte.

Siehe auch

- [„GETDATE-Funktion \[Datum und Uhrzeit\]“ auf Seite 356](#)
- [„CURRENT TIMESTAMP-Spezialwert“ auf Seite 275](#)

Beispiel

Die folgende Anweisung gibt das aktuelle Datum und die aktuelle Uhrzeit zurück:

```
SELECT NOW( * );
```

NULLIF-Funktion [Verschiedene]

Liefert einen verkürzten CASE-Ausdruck, indem Ausdrücke verglichen werden.

Syntax

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

Parameter

- ***expression-1*** Ein zu vergleichender Ausdruck.
- ***expression-2*** Ein zu vergleichender Ausdruck

Rückgabe

Datentyp des ersten Arguments.

Bemerkungen

NULLIF vergleicht die Werte der zwei Ausdrücke.

Wenn der erste Ausdruck gleich dem zweiten Ausdruck ist, gibt NULLIF den Wert NULL zurück.

Wenn der erste Ausdruck nicht gleich dem zweiten Ausdruck oder der zweite Ausdruck NULL ist, gibt NULLIF den ersten Ausdruck zurück.

Die NULLIF-Funktion bietet eine verkürzte Methode zum Schreiben bestimmter CASE-Ausdrücke.

Siehe auch

- „CASE-Ausdrücke“ auf Seite 280

Beispiel

Die folgende Anweisung gibt den Wert a zurück:

```
SELECT NULLIF( 'a', 'b' );
```

Die folgende Anweisung gibt NULL zurück:

```
SELECT NULLIF( 'a', 'a' );
```

PATINDEX-Funktion [Zeichenfolge]

Gibt eine Ganzzahl zurück, die die Startposition des ersten Auftretens eines Musters in einer Zeichenfolge darstellt.

Syntax

```
PATINDEX( '%pattern%', string-expression )
```

Parameter

- **pattern** Das Muster, nach dem gesucht wird. Wenn der führende Prozent-Platzhalter weggelassen wird, gibt die PATINDEX-Funktion "1" zurück, falls das Muster am Anfang der Zeichenfolge auftritt, und "0", falls nicht.

Das Muster für UltraLite verwendet die folgenden Platzhalter:

Platzhalterzeichen	Gefunden
_ (Unterstrich)	Ein Zeichen
% (Prozent)	Eine Zeichenfolge mit null oder mehr Zeichen
[]	Ein einzelnes Zeichen im angegebenen Bereich oder Menge
[^]	Ein einzelnes Zeichen außerhalb des angegebenen Bereichs oder Menge

- **Zeichenfolgenausdruck** Die Zeichenfolge, die nach dem Muster durchsucht werden soll

Rückgabe

INT

Bemerkungen

Die PATINDEX-Funktion gibt die Startposition des ersten Auftretens des Musters zurück. Wenn das Muster nicht gefunden wird, wird "0" zurückgegeben.

Siehe auch

- „LOCATE-Funktion [Zeichenfolge]“ auf Seite 367
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert 2 zurück:

```
SELECT PATINDEX( '%hoco%', 'chocolate' );
```

Die folgende Anweisung gibt den Wert 11 zurück:

```
SELECT PATINDEX( '%4_5_', '0a1A 2a3A 4a5A' );
```

Die folgende Anweisung gibt 14 zurück, also das erste, nicht alphanumerische Zeichen im Zeichenfolgenausdruck. Das Muster '%[^a-z0-9]%' kann anstelle von '%[^a-zA-Z0-9]%' verwendet werden, wenn die Datenbank die Groß- und Kleinschreibung nicht berücksichtigt.

```
SELECT PATINDEX( '%[^a-zA-Z0-9]%', 'SQLAnywhere16 has many new features' );
```

Die folgende Anweisung kann verwendet werden, um alle Zeichen bis einschließlich zum ersten nicht alphanumerischen Zeichen in einer Zeichenfolge abzurufen.

```
SELECT LEFT( @string, PATINDEX( '%[^a-zA-Z0-9]%', @string ) );
```

Die folgenden Anweisungen erstellen die Tabelle myTable und füllen sie mit verschiedenen Zeichenfolgen mit alphanumerischen Zeichen, Leerzeichen und nicht-alphanumerischen Zeichen. Danach zeigen die SELECT-Anweisung und die folgenden Ergebnisse, wie Sie PATINDEX verwenden können, um die Startposition von Leerstellen und nicht-alphanumerischen Zeichen in den Zeichenfolgen zu finden:

```
CREATE TABLE myTable( coll LONG VARCHAR );

INSERT INTO myTable (coll) VALUES( 'the quick brown fox jumped over the lazy dog' ),
( 'the quick brown fox $$$$ jumped over the lazy dog' ),
( 'the quick brown fox 0999 jumped over the lazy dog' ),
( 'the quick brown fox ** jumped over the lazy dog' ),
( 'thequickbrownfoxjumpedoverthelazydog' ),
( 'thequickbrownfoxjum999pedoverthelazydog' ),
( 'thequick$$$$brownfox' ),
( 'the quick brown fox$$ jumped over the lazy dog' );

SELECT coll,
  //position of first non-alphanumeric character or space:
  PATINDEX( '%[^a-z0-9]%', coll) AS blank_posn,
  //position of first non-alphanumeric char that isn't a space:
  PATINDEX( '%[^ a-z0-9]%', coll) AS non_alpha_char,
  //everything up to and including first non-alphanumeric char that isn't a space:
  LEFT ( coll, PATINDEX( '%[^ a-zA-Z0-9]%', coll) ) AS left_str,
  //first non-alphanumeric char that isn't a space, and everything to the right:
  SUBSTRING ( coll, PATINDEX( '%[^ a-zA-Z0-9]%', coll) ) AS sub_str
FROM myTable;
```

col1	blank_posn	non_alpha_char	left_str	sub_str
the quick brown fox jumped over the lazy dog	4	0		the quick brown fox jumped over the lazy dog
the quick brown fox \$\$\$\$ jumped over the lazy dog	4	21	the quick brown fox \$	\$\$\$\$ jumped over the lazy dog
the quick brown fox 0999 jumped over the lazy dog	4	0		the quick brown fox 0999 jumped over the lazy dog
the quick brown fox ** jumped over the lazy dog	4	21	the quick brown fox *	** jumped over the lazy dog
thequickbrownfoxjumpedoverthelazydog	0	0		thequickbrownfoxjumpedoverthelazydog
thequickbrownfox-jum999pedoverthelazydog	0	0		thequickbrownfox-jum999pedoverthelazydog
thequick\$\$\$\$brownfox	9	9	thequick\$	\$\$\$\$brownfox
the quick brown fox\$\$ jumped over the lazy dog	4	20	the quick brown fox \$	\$\$ jumped over the lazy dog

PI-Funktion [Nummerisch]

Gibt den numerischen Wert PI zurück.

Syntax

PI([*])

Rückgabe

DOUBLE

Bemerkungen

Die Funktion gibt einen DOUBLE-Wert zurück.

PI(*) und PI() sind semantisch gleichwertig.

Beispiel

Die folgende Anweisung gibt den Wert 3,141592653(...) zurück:

```
SELECT PI( * );
```

POWER-Funktion [Numerisch]

Berechnet die Potenz einer Zahl zur Basis einer anderen Zahl.

Syntax

POWER(*numeric-expression-1*, *numeric-expression-2*)

Parameter

- **Numerischer_Ausdruck_1** Die Basis.
- **Numerischer_Ausdruck_2** Der Exponent.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE und führt die Berechnung als doppelte genaue Gleitkommazahl durch. Wenn ein Argument NULL ist, ist das Ergebnis NULL.

Beispiel

Die folgende Anweisung gibt den Wert 64 zurück:

```
SELECT POWER( 2, 6 );
```

QUARTER-Funktion [Datum und Uhrzeit]

Gibt eine Zahl zurück, die das Quartal des Jahres im angegebenen TIMESTAMP-Ausdruck darstellt.

Syntax

QUARTER(*timestamp-expression*)

Parameter

- **Zeitstempelausdruck** Das Datum für das Quartal.

Rückgabe

INTEGER

Bemerkungen

Die Quartale sind die folgenden:

Quarter	Zeitraum (inklusive)
1	Januar bis 31. März
2	April bis 30. Juni
3	Juli bis 30. September
4	Oktober bis 31. Dezember

Beispiel

Die folgende Anweisung gibt den Wert 2 zurück:

```
SELECT QUARTER( '1987/05/02' );
```

RADIANS-Funktion [Numerisch]

Konvertiert eine Zahl von Grad in Bogenmaß.

Syntax

RADIANS(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Eine Zahl, in Grad. Dieser Winkel wird in Bogenmaß konvertiert.

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihr Argument in DOUBLE und führt die Berechnung als doppelgenaue Gleitkommazahl durch.

Beispiel

Die folgende Anweisung gibt einen Wert von ungefähr 0,5236 zurück:

```
SELECT RADIANS( 30 );
```

RAND-Funktion [Numerisch]

Gibt eine Zufallszahl im Intervall von 0 bis 1 zurück, mit einem optionalen Initialwert. Nicht unterstützt von UltraLite Java Edition-Datenbanken.

Syntax

RAND([*integer-expression*])

Parameter

- **Ganzzahlausdruck** Ein optionaler Initialwert zur Erstellung einer Zufallszahl. Dieses Argument ermöglicht Ihnen die Erstellung einer Zufallszahlensequenz.

Rückgabe

DOUBLE

Bemerkungen

Die RAND-Funktion ist ein multiplikativ-linear-kongruenter Zufallszahlengenerator. Siehe Park and Miller (1988), CACM 31(10), S. 1192-1201 and Press et al. (1992), Numerical Recipes in C (2. Ausgabe, Kapitel 7, S. 279). Das Ergebnis eines Aufrufs der RAND-Funktion ist eine Pseudo-Zufallszahl n . Dabei gilt: $0 < n < 1$ (weder 0,0 noch 1,0 kann das Ergebnis sein).

Wenn eine Verbindung zum Server hergestellt wird, setzt der Zufallszahlengenerator einen Anfangswert (Initialwert). Jede Verbindung erhält einen eindeutigen Initialwert und sieht daher eine andere Zufallssequenz von anderen Verbindungen. Sie können auch einen Initialwert (*Ganzzahlausdruck*) als Argument angeben. Normalerweise sollten Sie dies nur einmal durchführen, bevor Sie eine Sequenz von Zufallszahlen durch nachfolgende Aufrufe der RAND-Funktion anfordern. Wenn Sie den Initialwert mehr als einmal initialisieren, wird die Sequenz neu gestartet. Wenn Sie denselben Initialwert angeben, wird dieselbe Sequenz generiert. Initialwerte, deren Werte nahe beieinander liegen, generieren ähnliche Anfangssequenzen, deren Divergenzen im Laufe der Sequenz zunehmen.

Kombinieren Sie bei dem Versuch, statistische Zufallsergebnisse zu erhalten, nicht die von einem Initialwert generierte Sequenz mit einem von einem zweiten Initialwert generierten Sequenz. Anders gesagt: Sie dürfen den Initialwert während der Generierung einer Sequenz von Zufallswerten nicht zurücksetzen.

Die RAND-Funktion wird als eine nicht-deterministische Funktion angesehen. Der Abfrageoptimierer behält die Ergebnisse der Funktion RAND nicht im Cache.

Weitere Hinweise zu nicht-deterministischen Funktionen finden Sie unter [Caching von Funktionen \[SQL Anywhere Server - SQL-Benutzerhandbuch\]](#).

Beispiel

Die folgenden Anweisungen ergeben elf Zufallsergebnisse. Jeder nachfolgende Aufruf der RAND-Funktion, bei dem kein Initialwert angegeben ist, erzeugt weiterhin unterschiedliche Ergebnisse:

```
SELECT RAND( 1 );
SELECT RAND( ), RAND( ), RAND( ), RAND( ), RAND( );
SELECT RAND( ), RAND( ), RAND( ), RAND( ), RAND( );
```

Die folgende Anweisung erzeugt zwei Ergebnismengen mit identischen Sequenzen, da der Initialwert zweimal angegeben wird:

```
SELECT RAND( 1 ), RAND( ), RAND( ), RAND( ), RAND( );
SELECT RAND( 1 ), RAND( ), RAND( ), RAND( ), RAND( );
```

Das folgende Beispiel erzeugt fünf Ergebnisse, die ähnliche Werte enthalten und keine zufällige Verteilung aufweisen. Aus diesem Grund ist es nicht empfehlenswert, die RAND-Funktion mehr als einmal mit ähnlichen Initialwerten aufzurufen:

```
SELECT RAND( 1 ), RAND( 2 ), RAND( 3 ), RAND( 4 ), RAND( 5 );
```

Das folgende Beispiel erzeugt fünf identische Ergebnisse und sollte vermieden werden:

```
SELECT RAND( 1 ), RAND( 1 ), RAND( 1 ), RAND( 1 ), RAND( 1 );
```

REMAINDER-Funktion [Numerisch]

Gibt den Rest zurück, wenn eine Ganzzahl durch eine andere dividiert wird.

Syntax

REMAINDER(*dividend*, *divisor*)

Parameter

- ***dividend*** Der Dividend oder Zähler der Division.
- ***divisor*** Der Divisor oder Nenner der Division

Rückgabe

- INTEGER
- NUMERIC

Bemerkungen

Sie können auch die MOD-Funktion verwenden, um den Rest zurückzugeben.

Siehe auch

- „MOD-Funktion [Numerisch]“ auf Seite 375

Beispiel

Die folgende Anweisung gibt den Wert 2 zurück:

```
SELECT REMAINDER( 5, 3 );
```

REPEAT-Funktion [Zeichenfolge]

Verkettet eine Zeichenfolge in der angegebenen Häufigkeit.

Syntax

REPEAT(*string-expression*, *integer-expression*)

Parameter

- ***Zeichenfolgenausdruck*** Die zu wiederholende Zeichenfolge.
- ***Ganzzahlausdruck*** Die Anzahl, wie oft eine Zeichenfolge wiederholt werden soll. Wenn *Ganzzahlausdruck* negativ ist, wird eine leere Zeichenfolge zurückgegeben.

Rückgabe

LONG VARCHAR

Bemerkungen

Wenn die tatsächliche Länge der Ergebniszeichenfolge das Maximum für den Rückgabebetyp überschreitet, tritt ein Fehler auf. Das Ergebnis wird auf die maximal zulässige Zeichenfolgenreihe gekürzt.

Das Verhalten dieser Funktion ist identisch mit dem der REPLICATE-Funktion.

Siehe auch

- [„REPLICATE-Funktion \[Zeichenfolge\]“ auf Seite 389](#)
- [„UltraLite-Zeichenfolgenfunktionen“ auf Seite 321](#)

Beispiel

Die folgende Anweisung gibt den Wert repeatrepeatrepeat zurück:

```
SELECT REPEAT( 'repeat', 3 );
```

REPLACE-Funktion [Zeichenfolge]

Ersetzt eine Zeichenfolge mit einer anderen Zeichenfolge und gibt das neue Ergebnis zurück.

Syntax

REPLACE(*original-string*, *search-string*, *replace-string*)

Parameter

Wenn ein Argument NULL ist, gibt die Funktion NULL zurück.

- ***original-string*** Die zu durchsuchende Zeichenfolge. Sie kann eine beliebige Länge haben.
- ***search-string*** Die Zeichenfolge, nach der gesucht und die von *replace-string* ersetzt werden soll. Die Zeichenfolge ist auf 255 Byte beschränkt. Wenn *search-string* eine leere Zeichenfolge ist, wird die ursprüngliche Zeichenfolge zurückgegeben.
- ***replace-string*** Die Zeichenfolge, die *search-string* ersetzt. Sie kann eine beliebige Länge haben. Wenn *replace-string* eine leere Zeichenfolge ist, wird jedes Auftreten von *search-string* gelöscht.

Rückgabe

LONG VARCHAR

Bemerkungen

Diese Funktion ersetzt alle gefundenen Zeichenfolgen.

Vergleiche berücksichtigen die Groß-/Kleinschreibung, wenn die Datenbank auf die Berücksichtigung von Groß-/Kleinschreibung eingestellt ist.

Siehe auch

- „SUBSTRING-Funktion [Zeichenfolge]“ auf Seite 408
- „CHARINDEX-Funktion [Zeichenfolge]“ auf Seite 334
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert `xx.def.xx.ghi` zurück:

```
SELECT REPLACE( 'abc.def.abc.ghi', 'abc', 'xx' );
```

Die folgende Anweisung generiert eine Ergebnismenge, die ALTER PROCEDURE-Anweisungen enthält, welche bei der Ausführung gespeicherte Prozeduren ausbessern würde, die sich wiederum auf eine Tabelle beziehen, die umbenannt wurde. (Um nutzbar zu sein, muss der Tabellename eindeutig sein.)

```
SELECT REPLACE(
    REPLACE( proc_defn, 'OldTableName', 'NewTableName' ),
    'CREATE PROCEDURE',
    'ALTER PROCEDURE' )
FROM SYS.SYSPROCEDURE
WHERE proc_defn LIKE '%OldTableName%';
```

REPLICATE-Funktion [Zeichenfolge]

Verkettet eine Zeichenfolge in der angegebenen Häufigkeit.

Syntax

REPLICATE(*string-expression*, *integer-expression*)

Parameter

- **Zeichenfolgenausdruck** Die zu wiederholende Zeichenfolge.
- **Ganzzahlausdruck** Die Anzahl, wie oft eine Zeichenfolge wiederholt werden soll.

Rückgabe

LONG VARCHAR

Bemerkungen

Wenn die tatsächliche Länge der Ergebniszeichenfolge das Maximum für den Rückgabebetyp überschreitet, tritt ein Fehler auf. Das Ergebnis wird auf die maximal zulässige Zeichenfolgenreöße gekürzt.

Das Verhalten dieser Funktion ist identisch mit dem der REPEAT-Funktion.

Siehe auch

- „REPEAT-Funktion [Zeichenfolge]“ auf Seite 387
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert repeatrepeatrepeat zurück:

```
SELECT REPLICATE( 'repeat', 3 );
```

RIGHT-Funktion [Zeichenfolge]

Gibt die äußersten rechten Zeichen einer Zeichenfolge zurück.

Syntax

RIGHT(*string-expression*, *integer-expression*)

Parameter

- **Zeichenfolgenderausdruck** Die Zeichenfolge, für die die äußersten rechten Zeichen zurückgegeben werden.
- **Ganzzahlausdruck** Die Anzahl der Zeichen am Ende der Zeichenfolge, die zurückgegeben werden sollen

Rückgabe

LONG VARCHAR

Bemerkungen

Wenn eine Zeichenfolge Mehrbytezeichen enthält, ist die Anzahl der zurückgegebenen Bytes möglicherweise größer als die angegebene Anzahl von Zeichen.

Sie können einen *Ganzzahlausdruck* angeben, der größer ist als der Wert in der Spalte. In diesem Fall wird der gesamte Wert zurückgegeben.

Wenn in der Eingabezeichenfolge Zeichenlängensemantik verwendet wurde, wird der Rückgabewert soweit wie möglich mit Ausdrücken der Zeichenlängensemantik beschrieben.

Siehe auch

- „LEFT-Funktion [Zeichenfolge]“ auf Seite 364
- „Internationale Sprachen und Zeichensätze“ [*SQL Anywhere Server - Datenbankadministration*]
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt die letzten 5 Zeichen jedes Surname-Werts in der Tabelle "Customers" zurück:

```
SELECT RIGHT( Surname, 5 ) FROM GROUPO.Customers;
```

ROUND-Funktion [Numerisch]

Rundet *Numerischer_Ausdruck* auf die Anzahl von Stellen hinter dem Komma, die in *Ganzzahlausdruck* angegeben ist.

Syntax

ROUND(*numeric-expression*, *integer-expression*)

Parameter

- **Nummerischer_Ausdruck** Die zu rundende Zahl, die an die Funktion übergeben wurde.
- **Ganzzahlausdruck** Eine positive Ganzzahl bestimmt die Anzahl von signifikanten Stellen rechts vom Dezimalzeichen für die Rundung. Eine negative Ganzzahl bestimmt die Anzahl von signifikanten Stellen links vom Dezimalzeichen für die Rundung.

Rückgabe

NUMERIC

Bemerkungen

Das Ergebnis dieser Funktion ist entweder NUMERIC oder DOUBLE. Wenn es ein numerisches Ergebnis gibt und *Ganzzahlausdruck* ein negativer Wert ist, wird die Gesamtstellenzahl um 1 erhöht.

Siehe auch

- [„TRUNCNUM-Funktion \[Numerisch\]“ auf Seite 414](#)

Beispiel

Die folgende Anweisung gibt den Wert 123,200 zurück:

```
SELECT ROUND( 123.234, 1 );
```

RTRIM-Funktion [Zeichenfolge]

Entfernt nachgestellte Leerzeichen aus der Zeichenfolge.

Syntax

RTRIM(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die zu kürzende Zeichenfolge

Rückgabe

- VARCHAR
- LONG VARCHAR

Bemerkungen

Die tatsächliche Länge des Ergebnisses ist die Länge des Ausdrucks minus der Anzahl der entfernten Zeichen. Wenn alle Zeichen entfernt werden, ist das Ergebnis eine leere Zeichenfolge.

Wenn das Argument NULL ist, ist das Ergebnis NULL.

Siehe auch

- „TRIM-Funktion [Zeichenfolge]“ auf Seite 413
- „LTRIM-Funktion [Zeichenfolge]“ auf Seite 370
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert `Test Message` zurück, wobei alle nachgestellten Leerzeichen entfernt werden:

```
SELECT RTRIM( 'Test Message      ' );
```

SECOND-Funktion [Datum und Uhrzeit]

Gibt den Sekundenwert des `TIMESTAMP`-Arguments zurück.

Syntax

```
SECOND( timestamp-expression )
```

Parameter

- **Zeitstempelausdruck** Der `TIMESTAMP`-Wert.

Rückgabe

`SMALLINT`

Bemerkungen

Gibt eine Zahl von 0 bis 59 zurück, die der Sekundenkomponente des angegebenen `TIMESTAMP`-Arguments entspricht.

Beispiel

Die folgende Anweisung liefert den Wert 25.

```
SELECT SECOND( '1998-07-13 21:21:25' );
```

SECONDS-Funktion [Datum und Uhrzeit]

Bearbeitet einen `TIMESTAMP`-Wert oder gibt die Anzahl von Sekunden zwischen zwei `TIMESTAMP`-Werten zurück. Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Syntax 1

```
SECONDS( timestamp-expression )
```

Syntax 2

```
SECONDS( timestamp-expression, timestamp-expression )
```

Syntax 3

```
SECONDS( time-or-timestamp-expression, integer-expression )
```

Parameter

- **Zeitstempelausdruck** Ein TIMESTAMP-Wert.
- **Zeit-_oder_Zeitstempelausdruck** Ein Wert vom Typ TIME oder TIMESTAMP.
- **Ganzzahlausdruck** Die Anzahl der Sekunden, die zu *Zeit-_oder_Zeitstempelausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl an Sekunden von *Zeit-_oder_Zeitstempelausdruck* abgezogen. Wenn Sie einen Ganzzahlausdruck angeben, muss *Zeitstempelausdruck* explizit als Datentyp TIME, DATE oder TIMESTAMP festgelegt sein. Wenn *Zeit-_oder_Zeitstempelausdruck* vom Typ DATE ist, wird die Zeitangabe als Mitternacht angenommen.

Rückgabe

UNSIGNED BIGINT bei Syntax 1.

SIGNED BIGINT bei Syntax 2.

TIME oder TIMESTAMP bei Syntax 3.

Bemerkungen

Das Ergebnis der SECONDS-Funktion hängt von deren Argumenten ab.

- **Syntax 1** Wenn Sie einen einzelnen *Zeitstempelausdruck* an die SECONDS-Funktion übergeben, wird die Anzahl der Sekunden zwischen Mitternacht am 0000-02-29 und *Zeitstempelausdruck* als UNSIGNED BIGINT-Wert zurückgegeben.

Hinweis

"0000-02" gibt kein tatsächliches Datum wieder. Es ist das von der SECONDS-Funktion verwendete Standarddatum.

- **Syntax 2** Wenn Sie zwei TIMESTAMP-Werte an die SECONDS-Funktion übergeben, liefert die Funktion die als Ganzzahl ausgedrückte Anzahl der Sekunden zwischen den beiden.
- **Syntax 3** Wenn Sie einen TIMESTAMP-Wert und eine Ganzzahl an die SECONDS-Funktion übergeben, liefert die Funktion das TIMESTAMP-Ergebnis nach dem Addieren der durch die Ganzzahl ausgedrückten Anzahl an Sekunden zum *Zeit- oder Zeitstempelausdruck*-Argument. Wenn Sie einen TIME-Wert an die SECONDS-Funktion übergeben, liefert die Funktion entsprechend einen Wert vom Typ TIME.

Verwenden Sie statt Syntax 2 die DATEDIFF-Funktion. Verwenden Sie statt Syntax 3 die DATEADD-Funktion.

Siehe auch

- „CAST-Funktion [Datentypkonvertierung]“ auf Seite 330
- „DATEADD-Funktion [Datum und Uhrzeit]“ auf Seite 343
- „DATEDIFF-Funktion [Datum und Uhrzeit]“ auf Seite 344

Beispiel

Die folgenden Anweisungen geben den Wert "14400" zurück, womit angezeigt wird, dass der zweite TIMESTAMP-Wert 14400 Sekunden nach dem ersten liegt.

```
SELECT SECONDS( '1999-07-13 06:07:12',  
                '1999-07-13 10:07:12' );  
SELECT DATEDIFF( second,  
                '1999-07-13 06:07:12',  
                '1999-07-13 10:07:12' );
```

Die folgende Anweisung liefert den Wert 63062431632.

```
SELECT SECONDS( '1998-07-13 06:07:12' );
```

Die folgenden Anweisungen geben den TIMESTAMP-Wert "1999-05-12 21:05:12.000" zurück.

```
SELECT SECONDS( CAST( '1999-05-12 21:05:07' AS TIMESTAMP ), 5);  
SELECT DATEADD( second, 5, '1999-05-12 21:05:07' );
```

SHORT_PLAN-Funktion [Verschiedene]

Gibt eine kurze Beschreibung der UltraLite-Planoptimierungsstrategie einer SQL-Anweisung als Zeichenfolge zurück. Die Beschreibung ist dieselbe, die von der EXPLANATION-Funktion zurückgegeben wird.

Syntax

SHORT_PLAN(*string-expression*)

Bemerkungen

Bei bestimmten Abfragen kann der Ausführungsplan für UltraLite von dem für SQL Anywhere ausgewählten Plan abweichen.

Parameter

- **Zeichenfolgenausdruck** Die SQL-Anweisung, die gewöhnlich eine SELECT-Anweisung ist, aber auch eine UPDATE- oder DELETE-Anweisung sein kann

Rückgabe

LONG VARCHAR

Siehe auch

- [„EXPLANATION-Funktion \[Verschiedene\]“](#) auf Seite 355

Beispiel

Die folgende Anweisung übergibt eine SELECT-Anweisung als Zeichenfolgenparameter und gibt den Ausführungsplan für eine Abfrage zurück.

```
SELECT SHORT_PLAN(  
    'SELECT * FROM GROUPO.Departments WHERE DepartmentID > 100' );
```

SIGN-Funktion [Numerisch]

Gibt das Vorzeichen (positiv und negativ) für die angegebene Zahl.

Syntax

SIGN(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Die Zahl, deren Vorzeichen zurückgegeben werden soll. *Numerischer_Ausdruck* kann vom Typ INTEGER, DOUBLE oder NUMERIC sein.

Rückgabe

SMALLINT

Bemerkungen

Bei negativen Zahlen gibt die SIGN-Funktion "-1" zurück.

Bei Null gibt die SIGN-Funktion "0" zurück.

Bei positiven Zahlen gibt die SIGN-Funktion "1" zurück.

Beispiel

Die folgende Anweisung gibt den Wert "-1" zurück:

```
SELECT SIGN( -550 );
```

SIMILAR-Funktion [Zeichenfolge]

Gibt eine Zahl zurück, die die Ähnlichkeit zwischen zwei Zeichenfolgen angibt.

Syntax

SIMILAR(*string-expression-1*, *string-expression-2*)

Parameter

- **Zeichenfolgenausdruck_1** Die erste Zeichenfolge, die verglichen werden soll.
- **Zeichenfolgenausdruck_2** Die zweite Zeichenfolge, die verglichen werden soll

Rückgabe

SMALLINT

Bemerkungen

Die Funktion gibt eine Ganzzahl zwischen 0 und 100 zurück, die die Ähnlichkeit zwischen den beiden Zeichenfolgen repräsentiert. Das Ergebnis kann als Prozent der Zeichen interpretiert werden, die bei beiden Zeichenfolgen übereinstimmen. Ein Wert von 100 Prozent bedeutet, dass die beiden Zeichenfolgen identisch sind.

Diese Funktion kann zum Korrigieren einer Namensliste verwendet werden (wie zum Beispiel Kunden). Einige Kunden könnten zu der Liste mehrmals mit leicht unterschiedlichen Namen hinzugefügt worden sein. Sie können mit der SIMILAR-Funktion nach ähnlichen Kundennamen suchen, indem Sie die customer-Tabelle mit sich selbst verknüpfen und einen Bericht über alle Ähnlichkeiten erzeugen, die größer sind als 90 Prozent, aber kleiner als 100 Prozent.

Die bei der SIMILAR-Funktion durchgeführte Berechnung ist komplexer als nur die Anzahl der übereinstimmenden Zeichen.

Siehe auch

- [„UltraLite-Zeichenfolgenfunktionen“ auf Seite 321](#)

Beispiel

Die folgende Anweisung gibt den Wert "75" zurück, was angibt, dass zwei Werte einander zu 75% ähnlich sind.

```
SELECT SIMILAR( 'toast', 'coast' );
```

SIN-Funktion [Numerisch]

Gibt den Sinus einer Zahl zurück.

Syntax

SIN(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Der Winkel im Bogenmaß.

Rückgabe

DOUBLE

Bemerkungen

Die SIN-Funktion gibt den Sinus des Arguments zurück, wobei das Argument ein Winkel im Bogenmaß ist. Die SIN- und ASIN-Funktionen sind inverse Vorgänge.

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltgenauem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück.

Siehe auch

- [„ASIN-Funktion \[Numerisch\]“ auf Seite 325](#)
- [„COS-Funktion \[Numerisch\]“ auf Seite 339](#)
- [„COT-Funktion \[Numerisch\]“ auf Seite 339](#)
- [„TAN-Funktion \[Numerisch\]“ auf Seite 412](#)

Beispiel

Die folgende Anweisung gibt den SIN-Wert "0,52" zurück:

```
SELECT SIN( 0.52 );
```

SOUNDEX-Funktion [Zeichenfolge]

Gibt eine Zahl zurück, die den Klang der Zeichenfolge darstellt.

Syntax

SOUNDEX(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die auszuwertende Zeichenfolge

Rückgabe

SMALLINT

Bemerkungen

Der Wert der SOUNDEX-Funktion für eine Zeichenfolge basiert auf dem ersten Buchstaben und den nächsten drei Konsonanten außer H, Y und W. Vokale in *Zeichenfolgenausdruck* werden ignoriert, sofern sie nicht der erste Buchstabe der Zeichenfolge sind. Doppelte Buchstaben werden als ein Buchstabe gezählt. Beispiel: Das Wort "Apples" basiert auf den Buchstaben A, P, L und S.

Mehrbytezeichen werden von der SOUNDEX-Funktion ignoriert.

Wenn auch nicht ganz fehlerfrei, so gibt die SOUNDEX-Funktion üblicherweise dieselbe Zahl für Wörter zurück, die ähnlich klingen und die mit demselben Buchstaben beginnen.

Die SOUNDEX-Funktion funktioniert grundsätzlich nur mit englischen Wörtern. Für andere Sprachen ist sie nur beschränkt sinnvoll.

Siehe auch

- [„UltraLite-Zeichenfolgenfunktionen“ auf Seite 321](#)

Beispiel

Die folgende Anweisung gibt zwei identische Zahlen (3827) zurück, die den Klang der einzelnen Namen darstellen.

```
SELECT SOUNDEX( 'Smith' ), SOUNDEX( 'Smythe' );
```

SPACE-Funktion [Zeichenfolge]

Gibt eine angegebene Anzahl von Leerstellen zurück.

Syntax

SPACE(*integer-expression*)

Parameter

- **Ganzzahlausdruck** Die Anzahl von Leerstellen, die zurückgegeben werden sollen

Rückgabe

LONG VARCHAR

Bemerkungen

Wenn *Ganzzahlausdruck* negativ ist, wird eine Nullzeichenfolge zurückgegeben.

Siehe auch

- [„UltraLite-Zeichenfolgenfunktionen“ auf Seite 321](#)

Beispiel

Die folgende Anweisung gibt eine Zeichenfolge zurück, die 10 Leerstellen enthält.

```
SELECT SPACE( 10 );
```

SQRT-Funktion [Numerisch]

Gibt die Quadratwurzel einer Zahl zurück.

Syntax

SQRT(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Die Zahl, für die die Quadratwurzel berechnet werden soll

Rückgabe

DOUBLE

Bemerkungen

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltprecisem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück.

Beispiel

Die folgende Anweisung liefert den Wert 3.

```
SELECT SQRT( 9 );
```

ST_AsBinary-Funktion [Räumlich]

Gibt eine binäre Zeichenfolge zurück, die die angegebene Geometrie darstellt. Das Ausgabeformat ist WKB gemäß OGC SFS 1.1. Dieses Format enthält keine Z- und M-Werte.

Syntax

ST_AsBinary(*geometry-expression*)

Rückgabe

- **BINARY** Gibt die WKB-Darstellung von *Geometrieausdruck* zurück.

Beispiel

Die folgende Anweisung gibt das Ergebnis

0x010100000000000000000000f03f0000000000000040 zurück:

```
SELECT ST_AsBinary(ST_Point(1.0, 2.0, 4326))
```

Der Server ruft implizit die ST_AsBinary-Funktion auf, wenn Geometrien in BINARY konvertiert werden. Die folgende Anweisung gibt beispielsweise das Ergebnis

0x010100000000000000000000f03f0000000000000040 zurück:

```
SELECT CAST(ST_Point(1.0, 2.0, 4326) AS BINARY(50))
```

ST_AsExtText-Funktion [Räumlich]

Gibt eine binäre Zeichenfolge zurück, die die angegebene Geometrie darstellt. Das Ausgabeformat ist EWKT.

Syntax

ST_AsExtText(*geometry-expression*)

Rückgabe

- **VARCHAR** Gibt die EWKT-Darstellung von *Geometrieausdruck* zurück.

Beispiel

Die folgende Anweisung gibt das Ergebnis SRID=4326;Point (1 2) zurück, wobei die SRID als Präfix einbezogen wird:

```
SELECT ST_AsExtText(ST_Point(1.0, 2.0, 4326))
```

Die ST_AsExtText()-Funktion wird implizit aufgerufen, wenn Geometrien in VARCHAR-Datentypen konvertiert werden. Die folgende Anweisung gibt beispielsweise das Ergebnis SRID=4326;Point (1 2) zurück:

```
SELECT CAST(ST_Point(1.0, 2.0, 4326) AS VARCHAR(25))
```

ST_AsText-Funktion [Räumlich]

Gibt eine binäre Zeichenfolge zurück, die die angegebene Geometrie darstellt. Das Ausgabeformat ist WKT gemäß OGC SFS 1.1.

Syntax

ST_AsText(*geometry-expression*)

Rückgabe

- **VARCHAR** Gibt die WKT-Darstellung von *Geometrieausdruck* zurück.

Beispiel

Die folgende Anweisung gibt das Ergebnis `Point (1 2)` zurück:

```
SELECT ST_AsText(ST_Point(1.0, 2.0, 4326))
```

ST_Distance-Funktion [Räumlich]

Gibt die kleinste Entfernung zwischen zwei angegebenen Geometriewerten zurück. Wenn die Punkte in SRID 4326 sind, ist die Einheit Meter.

Syntax

```
ST_Distance( geo1,geo2 )
```

Parameter

Name	Typ	Beschreibung
<i>geo1</i>	ST_Geometry	Der erste Geometriewert, der verwendet werden soll, um die Entfernung zwischen zwei Geometriewerten zu berechnen.
<i>geo2</i>	ST_Geometry	Der zweite Geometriewert, der verwendet werden soll, um die Entfernung zwischen zwei Geometriewerten zu berechnen.

Rückgabe

- **DOUBLE** Gibt die kleinste Entfernung zwischen den angegebenen Geometriewerten zurück.

Beispiel

Die folgende Anweisung gibt das Ergebnis `3367142.4632130372` zurück:

```
SELECT ST_Distance(ST_Point(-79.38,43.65,4326),ST_Point(-123.1,49.28,4326))
```

ST_Equals-Funktion [Räumlich]

Testet, ob ein ST_Geometry-Wert räumlich gleich einem anderen ST_Geometry-Wert ist. Zwei Geometriewerte können als gleichwertig angesehen werden, wenn sie dieselben X- und Y-Koordinaten haben und sich im selben Bezugssystem befinden.

Der Test kann durch die Auflösung des räumlichen Bezugssystems oder die Präzision der Daten begrenzt sein.

Die ST_Equals-Funktion definiert die Semantik für die Vergleichsprädikate (= und <>), IN-Listenprädikate, DISTINCT und GROUP BY.

Syntax

ST_Equals(*geo1*,*geo2*)

Parameter

Name	Typ	Beschreibung
<i>geo1</i>	ST_Geometry	Der erste Geometriewert, der verglichen werden soll.
<i>geo2</i>	ST_Geometry	Der zweite Geometriewert, der verglichen werden soll

Rückgabe

- **BIT** Gibt 1 zurück, wenn die beiden Geometriewerte räumlich gleich sind, sonst 0.

Beispiel

Die folgende Anweisung gibt das Ergebnis 1 zurück:

```
SELECT ST_Equals(ST_Point(1,1,4326),ST_Point(1,1,4326))
```

ST_IntersectsRect-Funktion [Räumlich]

Testet, ob sich ein Punkt innerhalb des Rechtecks befindet, das durch die beiden als "min" und "max" angegebenen Punkte definiert wird.

Syntax

ST_IntersectsRect(*location*,*min*,*max*)

Parameter

Name	Typ	Beschreibung
<i>location</i>	ST_Geometry	Der Punkt, der getestet werden soll.
<i>min</i>	ST_Geometry	Der minimale Punktwert für die Definition des Felds.
<i>max</i>	ST_Geometry	Der maximale Punktwert für die Definition des Felds.

Rückgabe

- **BIT** Gibt den Wert 1 zurück, wenn die *location* eine Schnittmenge mit dem angegebenen Feld hat, sonst 0.

Beispiel

Die folgende Anweisung gibt das Ergebnis 1 zurück:

```
SELECT ST_IntersectsRect(ST_Point(1,1,4326),ST_Point(0,0,4326),
ST_Point(3,3,4326))
```

ST_Point-Funktion [Räumlich]

Konstruiert einen Punkt basierend auf X- und Y-Koordinaten.

Syntax

ST_Point(*x,y,SRID*)

Parameter

Name	Typ	Beschreibung
<i>x</i>	DOUBLE	Die X-Koordinate, die verwendet werden soll, um den Punkt zu konstruieren.
<i>y</i>	DOUBLE	Die Y-Koordinate, die verwendet werden soll, um den Punkt zu konstruieren.
<i>SRID</i>	INTEGER	Der dem Punkt zugeordnete SRID-Wert.

Rückgabe

- **ST_Point** Gibt einen ST_Geometry-Wert zurück, der aus der Eingabezeichenfolge erstellt wird.

Beispiel

Die folgende Anweisung erstellt einen Punkt bei (10.0,20.0) im Bezugssystem 2163:

```
SELECT ST_Point(10.0,20.0,2163)
```

ST_PointFromExtText-Funktion [Räumlich]

Gibt einen ST_Geometry-Wert zurück, der aus einem VARCHAR-Wert umgewandelt wurde, der eine EWKT-Darstellung eines ST_Geometry-Elements enthält.

Syntax

ST_PointFromText(*ewkt*)

Parameter

Name	Typ	Beschreibung
<i>ewkt</i>	VARCHAR	Die EWKT-Darstellung.

Rückgabe

- **ST_Geometry** Gibt einen ST_Geometry-Wert zurück, der aus der Eingabezeichenfolge erstellt wurde.

Beispiel

Die folgende Anweisung gibt das Ergebnis `SRID=4326;Point(10 20)` zurück, um anzuzeigen, dass ein Punkt bei (10,20) im Bezugssystem 4326 erstellt wurde:

```
SELECT ST_PointFromExtText('SRID=4326;Point(10 20)')
```

ST_PointFromText-Funktion [Spatial]

Gibt einen ST_Geometry-Wert zurück, der aus einem VARCHAR-Wert umgewandelt wurde, der eine WKT-Darstellung eines ST_Geometry-Elements enthält.

Syntax

ST_PointFromText(*wkt*, *srid*)

Parameter

Name	Typ	Beschreibung
<i>wkt</i>	VARCHAR	Die WKT-Darstellung.
<i>srid</i>	INT	Der Bezeichner für das räumliche Bezugssystem des Ergebnisses wird durch den SRID-Parameter angezeigt.

Rückgabe

- **ST_Geometry** Gibt einen ST_Geometry-Wert zurück, der aus der Eingabezeichenfolge erstellt wurde.

Die räumliche Referenz-ID des Ergebnisses ist die vom Parameter *srid* angegebene ID.

Beispiel

Die folgende Anweisung gibt das Ergebnis `SRID=4326;Point(10 20)` zurück, um anzuzeigen, dass ein Punkt bei (10,20) im Bezugssystem 4326 erstellt wurde:

```
SELECT ST_PointFromText('Point(10 20)',4326)
```

ST_PointFromWKB-Funktion [Spatial]

Gibt einen ST_Geometry-Wert zurück, der aus einem BINARY-Wert umgewandelt wurde, der eine WKB-Darstellung eines ST_Geometry-Elements enthält.

Syntax

ST_PointFromWKB(*wkb*, *srid*)

Parameter

Name	Typ	Beschreibung
<i>wkb</i>	BINARY	Die WKB-Darstellung.
<i>srid</i>	INTEGER	Der dem Punkt zugeordnete SRID-Wert.

Rückgabe

- **ST_Geometry** Gibt einen ST_Geometry-Wert zurück, der aus der Eingabezeichenfolge erstellt wurde.

Beispiel

Die folgende Anweisung gibt das Ergebnis (1.0 , 2.0 , 4326) zurück:

```
SELECT ST_PointFromWKB(0x01010000000000000000f03f00000000000040,4326)
```

ST_SRID-Funktion [Räumlich]

Ruft das dem Geometriewert zugeordnete räumliche Bezugssystem (SRID) ab.

Syntax

```
ST_SRID( geo1, srid )
```

Parameter

Name	Typ	Beschreibung
<i>geo1</i>	ST_Geometry	Der Punktwert.
<i>srid</i>	INTEGER	Der dem Punkt zugeordnete SRID-Wert.

Rückgabe

- **INT** Gibt die SRID der Geometrie zurück.

Beispiel

Die folgende Anweisung gibt das Ergebnis 4326 zurück:

```
SELECT ST_SRID( ST_Point ( 10, 20, 4326 ) );
```

ST_X-Funktion [Räumlich]

Gibt die X-Koordinate des ST_Geometry-Werts zurück.

Syntax

```
ST_X(geo1)
```

Parameter

Name	Typ	Beschreibung
<i>geo1</i>	ST_Geometry	Der ST_Geometry-Wert, aus dem die x-Koordinate ermittelt werden soll.

Rückgabe

- **DOUBLE** Gibt die X-Koordinate des ST_Geometry-Werts zurück.

Beispiel

Die folgende Anweisung gibt das Ergebnis 10.0 zurück:

```
SELECT ST_X(ST_Point(10.0,20.0,4326))
```

ST_Y-Funktion [Räumlich]

Gibt die Y-Koordinate des ST_Geometry-Werts zurück.

Syntax

```
.ST_Y(geo1)
```

Parameter

Name	Typ	Beschreibung
<i>geo1</i>	ST_Geometry	Der ST_Geometry-Wert, aus dem die y-Koordinate ermittelt werden soll.

Rückgabe

- **DOUBLE** Gibt die Y-Koordinate des ST_Geometry-Werts zurück.

Beispiel

Im folgenden Beispiel wird das Ergebnis 20.0 zurückgegeben:

```
SELECT ST_Y(ST_Point(10.0,20.0,4326))
```

STR-Funktion [Zeichenfolge]

Gibt die einer Zahl entsprechende Zeichenfolge zurück.

Syntax

```
STR( numeric-expression [, length [, decimal ] ] )
```

Parameter

- **Nummerischer_Ausdruck** Jeder angenähert numerische Ausdruck (float, real, double precision) zwischen -1E126 und 1E127.
- **length** Die Anzahl der zurückzugebenden Zeichen (einschließlich dem Dezimalzeichen, aller Stellen rechts und links vom Dezimalzeichen sowie Leerzeichen). Standardwert ist "10".
- **decimal** Die Anzahl der zurückzugebenden Dezimalstellen. Standardwert ist "0".

Rückgabe

VARCHAR

Bemerkungen

Wenn der Ganzzahlteil der Zahl nicht in die angegebene Länge passt, dann ist das Ergebnis eine Zeichenfolge mit der angegebenen Länge, die nur Sternchen enthält. Die folgende Anweisung liefert beispielsweise "***".

```
SELECT STR( 1234.56, 3 );
```

Hinweis

Die maximale unterstützte Länge ist 128. Jede Länge, die nicht zwischen 1 und 128 liegt, ergibt ein Ergebnis von NULL.

Siehe auch

- [„UltraLite-Zeichenfolgenfunktionen“ auf Seite 321](#)

Beispiel

Die folgende Anweisung gibt eine Zeichenfolge von sechs Leerstellen, gefolgt von "1235" zurück, bei einer Gesamtsumme von zehn Zeichen:

```
SELECT STR( 1234.56 );
```

Die folgende Anweisung liefert das Ergebnis "1234,6":

```
SELECT STR( 1234.56, 6, 1 );
```

STRING-Funktion [Zeichenfolge]

Verkettet eine oder mehrere Zeichenfolgen in eine große Zeichenfolge.

Syntax

```
STRING( string-expression [, ... ] )
```

Parameter

- **Zeichenfolgenausdruck** Die auszuwertende Zeichenfolge

Wenn nur ein Argument angegeben ist, wird es in einen einzelnen Ausdruck konvertiert. Wenn mehrere Argumente übergeben werden, werden sie zu einer einzelnen Zeichenfolge verkettet.

Rückgabe

- LONG VARCHAR
- LONG BINARY

Bemerkungen

Nummerische oder Datumsparameter werden vor der Verkettung in Zeichenfolgen konvertiert. Mit der STRING-Funktion kann auch ein einzelner Ausdruck in eine Zeichenfolge konvertiert werden, indem dieser Ausdruck als einziger Parameter angegeben wird.

Wenn alle Parameter NULL sind, gibt STRING NULL zurück. Wenn irgend welche Parameter nicht NULL sind, werden alle NULL-Parameter als leere Zeichenfolgen behandelt.

Siehe auch

- [„UltraLite-Zeichenfolgenfunktionen“ auf Seite 321](#)

Beispiel

Die folgende Anweisung liefert den Wert "testing123".

```
SELECT STRING( 'testing', NULL, 123 );
```

STRTOUUID-Funktion [Zeichenfolge]

Konvertiert einen Zeichenfolgenwert in den Wert eines eindeutigen Bezeichners (UUID oder GUID).

Hinweis

In einer Datenbank, die vor Version 9.0.2 erstellt wurde, werden die STRTOUUID- und UUIDTOSTR-Funktionen benötigt, um zwischen binären und Zeichenfolgen-Darstellungen von UUID-Werten zu konvertieren.

In Datenbanken, die mit Version 9.0.2 oder später erstellt wurden, ist der UNIQUEIDENTIFIER-Datentyp ein nativer Datentyp. Verwenden Sie bei diesen Versionen STRTOUUID und UUIDTOSTR nicht.

Weitere Hinweise finden Sie unter „[UltraLite, SQLDatentypen](#)“ auf Seite 296.

Syntax

```
STRTOUUID( string-expression )
```

Parameter

- **Zeichenfolgenerausdruck** Eine Zeichenfolge im Format `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`

Rückgabe

UNIQUEIDENTIFIER

Bemerkungen

Konvertiert eine Zeichenfolge des Formats `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`, wobei hier *x* für eine hexadezimale Ziffer steht, in einen eindeutig identifizierenden Wert.

Diese Funktion ist nützlich für das Einfügen von UUID-Werten in eine Datenbank.

Siehe auch

- „[UUIDTOSTR-Funktion \[Zeichenfolge\]](#)“ auf Seite 416
- „[NEWID-Funktion \[Verschiedene\]](#)“ auf Seite 379
- „[UltraLite-Zeichenfolgenfunktionen](#)“ auf Seite 321

STUFF-Funktion [Zeichenfolge]

Löscht mehrere Zeichen aus einer Zeichenfolge und ersetzt sie durch eine andere Zeichenfolge.

Syntax

```
STUFF( string-expression-1, start, length, string-expression-2 )
```

Parameter

- **Zeichenfolgenausdruck_1** Die Zeichenfolge, die durch die STUFF-Funktion geändert wird.
- **start** Die Zeichenposition, an der mit dem Löschen der Zeichen begonnen wird. Das erste Zeichen in der Zeichenfolge hat die Position 1.
- **length** Die Anzahl der zu löschenden Zeichen.
- **Zeichenfolgenausdruck_2** Die einzufügende Zeichenfolge. Zum Löschen eines Teils einer Zeichenfolge mithilfe der STUFF-Funktion verwenden Sie eine Ersetzungszeichenfolge von NULL.

Rückgabe

VARCHAR

Siehe auch

- [„INSERTSTR-Funktion \[Zeichenfolge\]“ auf Seite 361](#)
- [„UltraLite-Zeichenfolgenfunktionen“ auf Seite 321](#)

Beispiel

Die folgende Anweisung gibt den Wert "chocolate pie" zurück:

```
SELECT STUFF( 'chocolate cake', 11, 4, 'pie' );
```

SUBSTRING-Funktion [Zeichenfolge]

Gibt eine Teilzeichenfolge einer Zeichenfolge zurück.

Syntax

```
{ SUBSTRING | SUBSTR } ( string-expression, start  
[ , length ] )
```

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, aus der eine Teilzeichenfolge zurückgegeben werden soll.
- **start** Die Anfangsposition der zurückzugebenden Teilzeichenfolge, in Zeichen.
- **length** Die Länge der zurückzugebenden Teilzeichenfolge, in Zeichen. Wenn *length* angegeben ist, ist die Teilzeichenfolge auf diese Länge begrenzt.

Rückgabe

- LONG BINARY
- LONG VARCHAR

Bemerkungen

In UltraLite verfügt die Datenbank nicht über eine `ansi_substring`-Option, aber die SUBSTR-Funktion verhält sich so, als wäre `ansi_substring` standardmäßig aktiviert. Das Verhalten der Funktion entspricht dem ANSI/ISO SQL/2008-Verhalten:

- **Startwert** Das erste Zeichen in der Zeichenfolge ist auf Position 1. Ein negatives oder ein Null-Start-Offset wird so behandelt, als ob die Zeichenfolge links mit Nicht-Zeichen angefüllt ist.
- **Längenwert** Ein positiver Wert für *length* gibt an, dass die Teilzeichenfolge *length* Zeichen rechts von der Startposition endet.

Ein negativer Wert für *length* gibt einen Fehler zurück.

Ein Wert *length* von Null gibt eine leere Zeichenfolge zurück.

Wenn *Zeichenfolgenausdruck* ein binärer Datentyp ist, verhält sich die SUBSTRING-Funktion wie BYTE_SUBSTR.

Um Zeichen am Ende einer Zeichenfolge zu erhalten, benutzen Sie die Funktion RIGHT.

Wenn in der Eingabezeichenfolge Zeichenlängensemantik verwendet wurde, wird der Rückgabewert soweit wie möglich mit Ausdrücken der Zeichenlängensemantik beschrieben.

Siehe auch

- „[BYTE_SUBSTR-Funktion \[Zeichenfolge\]](#)“ auf Seite 329
- „[LEFT-Funktion \[Zeichenfolge\]](#)“ auf Seite 364
- „[RIGHT-Funktion \[Zeichenfolge\]](#)“ auf Seite 390
- „[CHARINDEX-Funktion \[Zeichenfolge\]](#)“ auf Seite 334
- „[UltraLite-Zeichenfolgenfunktionen](#)“ auf Seite 321

Beispiel

Die folgende Tabelle zeigt die Werte, die von der SUBSTRING-Funktion zurückgegeben werden.

Beispiel	Ergebnis
SUBSTRING('front yard', 1, 4)	fron
SUBSTRING('back yard', 6, 4)	yard
SUBSTR('abcdefgh', 0, -2)	Gibt einen Fehler zurück
SUBSTR('abcdefgh', -2, 2)	Gibt eine leere Zeichenfolge zurück

SUM-Funktion [Aggregat]

Gibt die Gesamtsumme des angegebenen Ausdrucks für jede Zeilengruppe zurück

Syntax 1

SUM([DISTINCT] *expression*)

Parameter

- ***expression*** Der Name des Ausdrucks, dessen Summe gebildet werden soll. Das ist üblicherweise ein Spaltenname.

- **DISTINCT *expression*** Berechnet die Summe der eindeutigen Werte von *expression* in jeder Gruppe.

Rückgabe

- INTEGER
- DOUBLE
- NUMERIC

Bemerkungen

Zeilen, in denen der angegebene Ausdruck NULL ist, werden nicht mit eingeschlossen.

Gibt NULL für eine Gruppe zurück, die keine Zeilen enthält.

Diese Funktion kann einen Überlauffehler erzeugen, was dazu führt, dass ein Fehler zurückgegeben wird. Sie können die CAST-Funktion auf *Nummerischer_Ausdruck* anwenden, um den Überlauffehler zu vermeiden. Siehe „[CAST-Funktion \[Datentypkonvertierung\]](#)“ auf Seite 330.

Siehe auch

- „[COUNT-Funktion \[Aggregat\]](#)“ auf Seite 340
- „[AVG-Funktion \[Aggregat\]](#)“ auf Seite 327

Beispiel

Die folgende Anweisung liefert den Wert 3749146.740.

```
SELECT SUM( Salary )  
FROM GROUPO.Employees;
```

SWITCHOFFSET-Funktion [Datum und Zeit]

Gibt einen TIMESTAMP WITH TIME ZONE-Wert zurück, der vom ursprünglichen Zeitzonen-Offset in den angegebenen Zeitzonen-Offset umgewandelt wurde.

Syntax

SWITCHOFFSET(*tmz-expression*, *time-zone-offset*)

Parameter

- **TMZ-Ausdruck** Der TIMESTAMP WITH TIME ZONE-Wert, der konvertiert werden soll.
- **time-zone-offset** Der Zeitzonen-Offset des Ergebnisses. Der Wert kann eine Ganzzahl sein, welche die Minuten vor oder nach der Coordinated Universal Time (UTC) darstellt, eine Zeichenfolge im Format { + | - } hh:nn oder "Z" für die Zulu-Zeitzone. Die Zulu-Zeitzone ist dieselbe Zeitzone wie UTC.

Rückgabe

TIMESTAMP WITH TIME ZONE

Siehe auch

- „TIMESTAMP WITH TIME ZONE-Datentyp“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „SYSDATETIMEOFFSET-Funktion [Datum und Zeit]“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Beispiel

Im folgenden Beispiel wird eine Zeitzonen-Offsetwert von -04:00 Stunden in -07:00 Stunden geändert. Der zurückgegebene Wert ist 2009-04-03 11:45:12.123-07:00.

```
SELECT CAST ( '2009-04-03 14:45:12.123-04:00' AS datetimeoffset ) AS EDT,  
SWITCHOFFSET( EDT, '-07:00' ) AS PDT;
```

SYNC_PROFILE_OPTION_VALUE-Funktion [System]

Gibt den Wert der Option zurück, der dem angegebenen Optionsnamen entspricht. Nicht unterstützt von UltraLite Java Edition-Datenbanken.

Syntax

SYNC_PROFILE_OPTION_VALUE(*profile_name*, *option_name*)

Parameter

- **profile_name** Der Name des Synchronisationsprofils, das geprüft werden soll.
- **option_name** Der Name der Option für den Abruf des entsprechenden Werts.

Rückgabe

Gibt den Wert der Option zurück, der dem angegebenen Optionsnamen entspricht.

Bemerkungen

Optionsnamen mit Punkten rufen Werte aus einer Teilliste ab, wobei der Name der Option vor dem Punkt steht, der Name der Teilliste danach.

Siehe auch

- „[ML_GET_SERVER_NOTIFICATION-Funktion \[System\]](#)“ auf Seite 374

Beispiel

Sehen Sie sich das folgende Profil als Beispiel an:

```
MobiLinkUid=joe;Stream=tcip{host=sybase;port=1234};Ping=1
```

- **MobiLinkUid** joe
- **Stream** tcip{host=sybase;port=1234}
- **Stream.host** sybase

- **Stream.port** 1234
- **Ping** 1

TAN-Funktion [Numerisch]

Gibt den Tangens einer Zahl zurück.

Syntax

TAN(*numeric-expression*)

Parameter

- **Numerischer_Ausdruck** Ein Winkel im Bogenmaß.

Rückgabe

DOUBLE

Bemerkungen

Die ATAN- und TAN-Funktionen sind inverse Vorgänge.

Diese Funktion konvertiert ihre Argumente in DOUBLE, führt die Berechnung mit doppeltprecisem Gleitkomma durch und gibt das Ergebnis als DOUBLE-Wert zurück.

Siehe auch

- „COS-Funktion [Numerisch]“ auf Seite 339
- „SIN-Funktion [Numerisch]“ auf Seite 396

Beispiel

Die folgende Anweisung gibt den Tangenswert für "0,52" zurück:

```
SELECT TAN( 0.52 );
```

TODATETIMEOFFSET-Funktion [Datum und Zeit]

Verwendet den angegebenen Zeitzone-Offset, um einen TIMESTAMP-Wert in einen TIME STAMP WITH TIME ZONE-Wert umzuwandeln.

Syntax

TODATETIMEOFFSET(*timestamp-expression*, *time-zone-offset*)

Parameter

- **Zeitstempelausdruck** Der zu konvertierende TIMESTAMP-Ausdruck.
- **time-zone-offset** Der Zeitzone-Offset. Der Wert kann eine Ganzzahl sein, welche die Minuten vor oder nach der Coordinated Universal Time (UTC) darstellt, eine VARCHAR-Zeichenfolge in der

Form { + | - }hh:nn oder die Zeichenfolge "Z" für die Zulu Time Zone. Die Zulu-Zeitzone ist dieselbe Zeitzone wie UTC.

Rückgabe

TIMESTAMP WITH TIME ZONE

Siehe auch

- „TIMESTAMP WITH TIME ZONE-Datentyp“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]

Beispiel

Im folgenden Beispiel wird TIMESTAMP-Wert in einen TIMESTAMP WITH TIME ZONE-Wert konvertiert.

```
SELECT CAST('2009-04-03 14:45:12.123' AS TIMESTAMP) AS orig,  
       TODATETIMEOFFSET (orig, '+11:00');
```

TODAY-Funktion [Datum und Uhrzeit]

Gibt das aktuelle Datum als DATE-Wert zurück.

Syntax

TODAY([*])

Rückgabe

DATE

Bemerkungen

TODAY(*) und TODAY() sind semantisch gleichwertig. TODAY ist gleichwertig mit dem CURRENT DATE-Spezialwert.

Jede Instanz der TODAY-Funktion in einer Anforderung wird höchstens einmal ausgewertet. Mehrere Instanzen von TODAY in derselben Anforderung liefern möglicherweise identische oder auch unterschiedliche DATE-Werte.

Beispiel

Die folgenden Anweisungen geben den aktuellen Tag gemäß der Systemuhr zurück.

```
SELECT TODAY( * );  
SELECT CURRENT DATE;
```

TRIM-Funktion [Zeichenfolge]

Entfernt führende und nachgestellte Leerzeichen aus einer Zeichenfolge.

Syntax

TRIM(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die zu kürzende Zeichenfolge

Rückgabe

- VARCHAR
- LONG VARCHAR

Siehe auch

- „LTRIM-Funktion [Zeichenfolge]“ auf Seite 370
- „RTRIM-Funktion [Zeichenfolge]“ auf Seite 391
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert "chocolate" ohne führende oder nachgestellte Leerzeichen zurück.

```
SELECT TRIM( '   chocolate   ' );
```

TRUNCNUM-Funktion [Nummerisch]

Kürzt eine Zahl an der angegebenen Anzahl von Dezimalstellen.

Syntax

```
{ TRUNCNUM | TRUNCATE }( numeric-expression, integer-expression )
```

Parameter

- **Nummerischer_Ausdruck** Die zu kürzende Zahl. Dieses Argument kann vom Typ NUMERIC oder DOUBLE sein.
- **Ganzzahlausdruck** Eine positive Ganzzahl bestimmt die Anzahl von signifikanten Stellen rechts vom Dezimalzeichen für die Rundung. Eine negative Ganzzahl bestimmt die Anzahl von signifikanten Stellen links vom Dezimalzeichen für die Rundung.

Rückgabe

NUMERIC oder DOUBLE

Bemerkungen

Wenn ein Parameter NULL ist, ist das Ergebnis NULL.

Sie sollten die TRUNCNUM-Funktion und nicht die TRUNCATE-Funktion zum Kürzen von Zahlen verwenden.

Die Verwendung der TRUNCATE-Funktion wird nicht empfohlen, weil das Wort "truncate" ein Schlüsselwort ist, was bedeutet, dass Sie entweder die `quoted_identifier`-Option auf OFF setzen oder das Wort "TRUNCATE" in Anführungszeichen setzen müssen.

Siehe auch

- [„ROUND-Funktion \[Numerisch\]“ auf Seite 390](#)

Beispiel

Die folgende Anweisung gibt den Wert "600" zurück:

```
SELECT TRUNCNUM( 655, -2 );
```

Die folgende Anweisung liefert den Wert 655.340.

```
SELECT TRUNCNUM( 655.348, 2 );
```

UCASE-Funktion [Zeichenfolge]

Konvertiert alle Zeichen in einer Zeichenfolge in Großbuchstaben.

Syntax

UCASE(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, die in Großbuchstaben konvertiert werden soll.

Rückgabe

CHAR, VARCHAR, oder LONG VARCHAR entsprechend dem Datentyp des Arguments.

Bemerkungen

Diese Funktion ist mit der UPPER-Funktion identisch.

Siehe auch

- [„UPPER-Funktion \[Zeichenfolge\]“ auf Seite 415](#)
- [„LCASE-Funktion \[Zeichenfolge\]“ auf Seite 363](#)
- [„UltraLite-Zeichenfolgenfunktionen“ auf Seite 321](#)

Beispiel

Die folgende Anweisung gibt den Wert "CHOCOLATE" zurück:

```
SELECT UCASE( 'ChocoLate' );
```

UPPER-Funktion [Zeichenfolge]

Konvertiert alle Zeichen in einer Zeichenfolge in Großbuchstaben.

Syntax

UPPER(*string-expression*)

Parameter

- **Zeichenfolgenausdruck** Die Zeichenfolge, die in Großbuchstaben konvertiert werden soll.

Rückgabe

CHAR, VARCHAR, oder LONG VARCHAR entsprechend dem Datentyp des Arguments.

Bemerkungen

Diese Funktion ist mit der UCASE-Funktion identisch.

Siehe auch

- „UCASE-Funktion [Zeichenfolge]“ auf Seite 415
- „LCASE-Funktion [Zeichenfolge]“ auf Seite 363
- „LOWER-Funktion [Zeichenfolge]“ auf Seite 369
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung gibt den Wert "CHOCOLATE" zurück:

```
SELECT UPPER( 'ChocoLate' );
```

UUIDTOSTR-Funktion [Zeichenfolge]

Konvertiert den Wert eines eindeutigen Bezeichners (UUID, auch bekannt als GUID) in einen Zeichenfolgenwert.

Hinweis

In einer Datenbank, die vor Version 9.0.2 erstellt wurde, werden die STRTOUUID- und UUIDTOSTR-Funktionen benötigt, um zwischen binären und Zeichenfolgen-Darstellungen von UUID-Werten zu konvertieren.

In Datenbanken, die mit Version 9.0.2 oder später erstellt wurden, ist der UNIQUEIDENTIFIER-Datentyp ein nativer Datentyp. Verwenden Sie bei diesen Versionen STRTOUUID und UUIDTOSTR nicht.

Weitere Hinweise finden Sie unter „UltraLite, SQLDatentypen“ auf Seite 296.

Syntax

```
UUIDTOSTR( uuid-expression )
```

Parameter

- **UUID-Ausdruck** Ein eindeutig bezeichnender Wert.

Rückgabe

VARCHAR

Bemerkungen

Konvertiert einen eindeutig bezeichnenden Wert in eine Zeichenfolge des Formats `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`, wobei hier x für eine hexadezimale Ziffer steht. Sollte der BINARY-Wert kein gültiger eindeutiger Bezeichner sein, wird NULL zurückgegeben.

Diese Funktion ist nützlich, um einen UUID-Wert anzuzeigen.

Siehe auch

- „NEWID-Funktion [Verschiedene]“ auf Seite 379
- „STRTOUUID-Funktion [Zeichenfolge]“ auf Seite 407
- „UltraLite-Zeichenfolgenfunktionen“ auf Seite 321

Beispiel

Die folgende Anweisung erstellt eine Tabelle namens "mytab" mit zwei Spalten. Spalte pk verfügt über einen eindeutig identifizierenden Datentyp, die Spalte c1 über einen Ganzzahl-Datentyp. Die Anweisung fügt dann zwei Zeilen mit den Werten "1" und "2" entsprechend in die Spalte c1 ein.

```
CREATE TABLE mytab(  
    pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),  
    c1 INT );  
INSERT INTO mytab( c1 ) values ( 1 );  
INSERT INTO mytab( c1 ) values ( 2 );
```

Die Ausführung der folgenden SELECT-Anweisung gibt alle Daten der neu erstellten Tabelle zurück.

```
SELECT * FROM mytab;
```

Sie werden eine Tabelle sehen, die aus zwei Spalten und zwei Zeilen besteht. Der Wert für die Spalte pk wird binary-Werte darstellen.

Wenn Sie die eindeutigen Bezeichnerwerte in ein lesbares Format konvertieren möchten, führen Sie die folgende Anweisung aus:

```
SELECT UUIDTOSTR(pk), c1 FROM mytab;
```

Die UUIDTOSTR-Funktion wird nicht für Datenbanken benötigt, die mit Version 9.0.2 oder höher erstellt wurden.

WEEKS-Funktion [Datum und Uhrzeit]

Bearbeitet einen TIMESTAMP-Wert oder gibt die Anzahl von Wochen zwischen zwei TIMESTAMP-Werten zurück. Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Syntax 1

```
WEEKS( timestamp-expression )
```

Syntax 2

```
WEEKS( timestamp-expression, timestamp-expression )
```

Syntax 3

WEEKS(*timestamp-expression*, *integer-expression*)

Parameter

- **Zeitstempelausdruck** Ein Datums- und Uhrzeitwert vom Typ TIMESTAMP.
- **Ganzzahlausdruck** Die Anzahl der Wochen, die zu *Zeitstempelausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl an Wochen von *Zeitstempelausdruck* abgezogen. Wenn Sie einen *Ganzzahlausdruck* angeben, muss *Zeitstempelausdruck* explizit als TIME-, DATE- oder TIMESTAMP-Wert festgelegt sein.

Rückgabe

INTEGER bei Syntax 1 und Syntax 2.

TIMESTAMP bei Syntax 3.

Bemerkungen

Wenn ein einzelnes Datum eingegeben wird (Syntax 1), liefert die WEEKS-Funktion die Anzahl der Wochen seit 0000-02-29.

Wenn zwei Daten eingegeben werden (Syntax 2), liefert die WEEKS-Funktion die Anzahl der Wochen zwischen den beiden Daten. Die WEEKS-Funktion ähnelt der DATEDIFF-Funktion, allerdings ist die Methode zur Berechnung der Anzahl der Wochen zwischen den Daten nicht identisch und kann ein anderes Ergebnis bringen. Der Rückgabewert für WEEKS wird festgelegt, indem die Anzahl von Tagen zwischen den beiden Daten durch 7 geteilt und abgerundet wird. DATEDIFF verwendet jedoch in seiner Berechnung die Anzahl der überschrittenen Wochengrenzen. Daher können die beiden Funktionen unterschiedliche Werte zurückgeben. Beispiel: Wenn das erste Datum ein Freitag ist und das zweite der nachfolgende Montag, gibt die WEEKS-Funktion den Unterschied 0 zurück, die DATEDIFF-Funktion dagegen den Unterschied 1. Obwohl keine Methode besser ist als die andere, sollten Sie den Unterschied im Auge behalten, wenn Sie sich für WEEKS oder DATEDIFF entscheiden.

Weitere Hinweise zur DATEDIFF-Funktion finden Sie unter „[DATEDIFF-Funktion \[Datum und Uhrzeit\]](#)“ auf Seite 344.

Wenn ein Datum und eine Ganzzahl eingegeben werden (Syntax 3), addiert die WEEKS-Funktion die mit der Ganzzahl ausgedrückte Anzahl von Wochen zum angegebenen *Zeitstempelausdruck*. Bei Syntax 3 müssen Sie *Zeitstempelausdruck* explizit als Datentyp TIME, DATE oder TIMESTAMP festlegen. Wenn *Zeitstempelausdruck* ein TIME-Wert ist, wird das aktuelle Datum angenommen. Verwenden Sie statt Syntax 3 die DATEADD-Funktion.

Weitere Hinweise zur DATEADD-Funktion finden Sie unter „[DATEADD-Funktion \[Datum und Uhrzeit\]](#)“ auf Seite 343.

Siehe auch

Hinweise zum Casting von Datentypen finden Sie unter „[CAST-Funktion \[Datentypkonvertierung\]](#)“ auf Seite 330.

Beispiel

Die folgende Anweisung gibt den Wert 8 zurück. Das bedeutet, dass 2008-09-13 10:07:12 acht Wochen später ist als 2008-07-13 06:07:12.

```
SELECT WEEKS( '2008-07-13 06:07:12', '2008-09-13 10:07:12' );
```

Die folgende Anweisung gibt den Wert 104792 zurück. Das bedeutet, dass das Datum 104792 nach 0000-02-29 liegt.

```
SELECT WEEKS( '2008-07-13 06:07:12' );
```

Die folgende Anweisung gibt den TIMESTAMP-Wert 2008-06-16 21:05:07.0 zurück. Dies steht für das Datum und die Uhrzeit fünf Wochen nach 2008-05-12 21:05:07.

```
SELECT WEEKS( CAST( '2008-05-12 21:05:07' AS TIMESTAMP ), 5 );
```

YEAR-Funktion [Datum und Uhrzeit]

Gibt die Jahreskomponente des TIMESTAMP-Arguments zurück.

Syntax

```
YEAR( timestamp-expression )
```

Parameter

- **Zeitstempelausdruck** Ein TIMESTAMP-Wert.

Rückgabe

SMALLINT

Bemerkungen

Der zurückgegebene Wert ist die Jahreskomponente des angegebenen TIMESTAMP-Werts, in Form eines SMALLINT-Werts.

Beispiel

Das folgende Beispiel gibt den Wert "2001" zurück.

```
SELECT YEAR( '2001-09-12' );
```

YEARS-Funktion [Datum und Uhrzeit]

Bearbeitet einen TIMESTAMP-Wert oder gibt die Anzahl von Jahren zwischen zwei TIMESTAMP-Werten zurück. Weitere Hinweise finden Sie im folgenden Abschnitt "Bemerkungen".

Syntax 1

```
YEARS( timestamp-expression )
```

Syntax 2

```
YEARS( timestamp-expression, timestamp-expression )
```

Syntax 3

YEARS(*timestamp-expression*, *integer-expression*)

Parameter

- **Zeitstempelausdruck** Ein Datums- und Uhrzeitwert vom Typ TIMESTAMP.
- **Ganzzahlausdruck** Die Anzahl der Jahre (als SMALLINT-Wert), die zu *Zeitstempelausdruck* addiert werden sollen. Wenn *Ganzzahlausdruck* negativ ist, wird die entsprechende Anzahl an Jahren von *Zeitstempelausdruck* abgezogen. Wenn Sie einen *Ganzzahlausdruck* angeben, muss *Zeitstempelausdruck* explizit als DATE-, TIME- oder TIMESTAMP-Wert festgelegt sein. Wenn *Zeitstempelausdruck* ein TIME-Wert ist, wird das laufende Jahr angenommen.

Hinweise zum Casting von Datentypen finden Sie unter [„CAST-Funktion \[Datentypkonvertierung\]“](#) auf Seite 330.

Rückgabe

SMALLINT bei Syntax 1 und Syntax 2.

TIMESTAMP bei Syntax 3.

Bemerkungen

Der Wert von YEARS wird anhand der Anzahl der Neujahrstage zwischen zwei Datumsangaben berechnet.

Siehe auch

- [„DATEDIFF-Funktion \[Datum und Uhrzeit\]“](#) auf Seite 344
- [„DATEADD-Funktion \[Datum und Uhrzeit\]“](#) auf Seite 343

Beispiel

Die nachstehenden Anweisungen geben beide -4 zurück.

```
SELECT YEARS( '1998-07-13 06:07:12',  
              '1994-03-13 08:07:13' );
```

```
SELECT DATEDIFF( year,  
                '1998-07-13 06:07:12',  
                '1994-03-13 08:07:13' );
```

Die folgenden Anweisungen geben "1998" zurück:

```
SELECT YEARS( '1998-07-13 06:07:12' )  
SELECT DATEPART( year, '1998-07-13 06:07:12' );
```

Die folgenden Anweisungen geben das eingegebene Datum plus 300 Jahre zurück:

```
SELECT YEARS( CAST( '1998-07-13 06:07:12' AS TIMESTAMP ), 300 )  
  
SELECT DATEADD( year, 300, '1998-07-13 06:07:12' );
```

YMD-Funktion [Datum und Uhrzeit]

Gibt einen Datumswert zurück, der dem Jahr, Monat und Monatstag entspricht. Argumente sind SMALLINT-Werte von -32768 bis 32767.

Syntax

YMD(*smallint-expression1*, *smallint-expression2*, *smallint-expression3*)

Parameter

- **SMALLINT-Ausdruck_1** Die Jahreszahl.
- **SMALLINT-Ausdruck_2** Die Monatszahl. Das Jahr wird angepasst, wenn der Monat außerhalb des Bereichs 1-12 liegt.
- **SMALLINT-Ausdruck_3** Die Tagesnummer. Der Tag kann jede Ganzzahl sein, das Datum wird entsprechend angepasst.

Rückgabe

DATE

Beispiel

Die folgende Anweisung liefert den Wert 1998-06-12.

```
SELECT YMD( 1998, 06, 12 );
```

Wenn die Werte außerhalb des normalen Bereichs liegen, wird das Datum entsprechend angepasst. Die folgende Anweisung gibt beispielsweise den DATE-Wert "2000-03-01" zurück.

```
SELECT YMD( 1999, 15, 1 );
```

UltraLite-SQL-Anweisungen

Die von UltraLite SQL unterstützten SQL-Anweisungen sind eine Teilmenge der Anweisungen, die von SQL Anywhere-Datenbanken unterstützt werden.

Bevor Sie beginnen

- Die Tabellen in UltraLite unterstützen das Konzept des Eigentümers nicht. Als Erleichterung für bestehenden SQL-Code und programmtechnisch generierten SQL-Code gestattet UltraLite die Syntax *Eigentümer.Tabellenname*. Der Eigentümer wird jedoch nicht überprüft, da Tabelleneigentümer in UltraLite nicht unterstützt werden.
- Die Dokumentationen der UltraLite SQL-Anweisungen befolgen dieselben Syntaxkonventionen, die auch von SQL Anywhere-Anweisungen verwendet werden. Sie sollten diese Konventionen kennen und wissen, wie sie in der SQL-Syntax verwendet werden.
- Bei der Verwendung von UltraLite SQL wird eine Transaktion erstellt. Eine Transaktion besteht aus allen Änderungen (INSERT-, UPDATE- und DELETE-Anweisungen) seit der letzten ROLLBACK- bzw. COMMIT-Anweisung.

Diese Änderungen können durch COMMIT dauerhaft gemacht werden. Eine ROLLBACK-Anweisung bewirkt, dass die Änderungen entfernt werden.

Siehe auch

- „SQL-Anweisungen“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Syntaxkonventionen“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „UltraLite-Transaktionsverarbeitung“ auf Seite 486
- „COMMIT-Anweisung [UltraLite]“ auf Seite 432
- „ROLLBACK-Anweisung [UltraLite]“ auf Seite 457

Kategorien der UltraLite-Anweisungen

SQL-Anweisungen werden durch das erste Wort in einer Anweisung identifiziert, bei dem es sich zumeist um ein Verb handelt. Durch diese handlungsorientierte Syntax wird die Sprache zu einer Gruppe imperativer Anweisungen (Befehle) für die Datenbank. In UltraLite können die unterstützten SQL-Anweisungen wie folgt klassifiziert werden:

- **Datenabfrageanweisungen** Diese Anweisungen, die auch als Abfragen bezeichnet werden, geben Ihnen die Möglichkeit, Zeilen von Datenausdrücken aus Tabellen auszuwählen. Die Datenabfrage wird mit der SELECT-Anweisung durchgeführt.
- **Datenmanipulationsanweisungen** Diese Anweisungen ermöglichen es Ihnen, den Inhalt der Datenbank zu ändern. Die Änderung der Daten wird mit folgenden Anweisungen durchgeführt:
 - INSERT
 - UPDATE
 - DELETE
- **Datendefinitionsanweisungen** Diese Anweisungen geben Ihnen die Möglichkeit, die Struktur oder das Schema der Datenbank festzulegen. Das Schema kann mit den folgenden Anweisungen geändert werden:
 - ALTER DATABASE SCHEMA FROM FILE-Anweisung
 - CREATE INDEX
 - CREATE TABLE
 - DROP INDEX
 - DROP TABLE
 - ALTER TABLE
 - TRUNCATE TABLE
- **Transaktionssteuerungsanweisungen** Mit diesen Anweisungen können Sie Transaktionen innerhalb der UltraLite-Anweisung steuern. Die Transaktionssteuerung wird mit folgenden Anweisungen erreicht:
 - CHECKPOINT
 - COMMIT
 - ROLLBACK

- **Synchronisationsverwaltung** Diese Anweisungen gestatten es Ihnen, die Synchronisation mit einem MobiLink-Server temporär zu steuern. Die Synchronisationsverwaltung wird mit folgenden Anweisungen durchgeführt:
 - START SYNCHRONIZATION DELETE
 - STOP SYNCHRONIZATION DELETE
 - CREATE PUBLICATION
 - ALTER PUBLICATION
 - DROP PUBLICATION

Siehe auch

- „Ausdrücke in UltraLite“ auf Seite 277
- „Operatoren in UltraLite“ auf Seite 293
- „SELECT-Anweisung [UltraLite]“ auf Seite 457
- „INSERT-Anweisung [UltraLite]“
- „UPDATE-Anweisung [UltraLite]“
- „DELETE-Anweisung [UltraLite]“
- „ALTER DATABASE SCHEMA FROM FILE-Anweisung [UltraLite]“
- „CREATE INDEX-Anweisung [UltraLite]“
- „CREATE TABLE-Anweisung [UltraLite]“
- „DROP INDEX-Anweisung [UltraLite]“
- „DROP TABLE-Anweisung [UltraLite]“
- „ALTER TABLE-Anweisung [UltraLite]“
- „TRUNCATE TABLE-Anweisung [UltraLite]“
- „CHECKPOINT-Anweisung [UltraLite]“
- „COMMIT-Anweisung [UltraLite]“
- „ROLLBACK-Anweisung [UltraLite]“
- „START SYNCHRONIZATION DELETE-Anweisung [UltraLite]“
- „STOP SYNCHRONIZATION DELETE-Anweisung [UltraLite]“
- „CREATE PUBLICATION-Anweisung [UltraLite]“
- „ALTER PUBLICATION-Anweisung [UltraLite]“
- „DROP PUBLICATION-Anweisung [UltraLite]“

ALTER DATABASE SCHEMA FROM FILE-Anweisung [UltraLite]

Ändert die Schemadefinition einer bestehenden UltraLite-Datenbank mithilfe eines SQL-Skripts.

Hinweis

Diese Anweisung wird von UltraLite Java Edition-Datenbanken nicht unterstützt.

Syntax

ALTER DATABASE SCHEMA FROM FILE *filename*

Parameter

filename Legt den Namen und Pfad eines SQL-Skripts fest, das für das Upgrade des Schemas einer bestehenden UltraLite-Datenbank verwendet wird.

Bemerkungen

Verwenden Sie `ulinit` oder `ulunload` zum Extrahieren der für Ihr Skript erforderlichen DDL-Anweisungen. Mit diesen Dienstprogrammen stellen Sie sicher, dass die DDL-Anweisungen syntaktisch korrekt sind. Verwenden Sie `ulinit` (Option `-l logfile`) oder `ulunload` (unter Verwendung der Optionen `-n -s output-file`).

Sichern Sie die Datenbank, bevor Sie diese Anweisung ausführen.

Der Zeichensatz der SQL-Skriptdatei muss mit dem Zeichensatz der Datenbank übereinstimmen, bei der Sie das Upgrade durchführen wollen.

Stellen Sie sicher, dass Ihr Gerät nicht zurückgesetzt wird, während die Anweisung ausgeführt wird. Wenn Sie das Gerät während eines Schema-Upgrades zurücksetzen, wird die UltraLite-Datenbank unbrauchbar.

Zeilen, die nicht ins Schema passen, werden gelöscht (wenn z.B. eine Eindeutigkeits-Integritätsregel hinzugefügt wird und es mehrere Zeilen mit denselben Werten gibt, werden alle Zeilen bis auf eine gelöscht). In diesem Fall wird die Warnung `SQL_ROW_DROPPED_DURING_SCHEMA_UPGRADE` generiert. Sie können anhand dieser Warnung den Fehler ermitteln und die Datenbank von der Sicherungsversion wiederherstellen.

Siehe auch

- „UltraLite-Datenbankschemas“ auf Seite 49
- „Deployment von UltraLite-Datenbankschema-Upgrades“ auf Seite 125
- „UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (`ulinit`)“ auf Seite 214
- „UltraLite-Dienstprogramm zum Entladen von Datenbanken (`ulunload`)“ auf Seite 233

Beispiel

Die folgende Anweisung ändert das Schema der Datenbank mithilfe des SQL-Skripts *MySchema.sql*:

```
ALTER DATABASE SCHEMA FROM FILE 'MySchema.sql'
```

ALTER PUBLICATION-Anweisung [UltraLite]

Ändert eine Publikation.

Syntax

```
ALTER PUBLICATION publication-name alterpub-clause
```

alterpub-clause :

```
ADD TABLE table-name [ WHERE search-condition ]  
| ALTER TABLE table-name [ WHERE search-condition ]  
| { DROP | DELETE } TABLE table-name  
| RENAME publication-name
```

Bemerkungen

Eine Publikation identifiziert Daten in einer entfernten Datenbank, die synchronisiert werden sollen.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Suchbedingungen in UltraLite“ auf Seite 283
- „UltraLite-Clientsynchronisationsplanung“ auf Seite 74
- „CREATE PUBLICATION-Anweisung [UltraLite]“ auf Seite 435
- „DROP PUBLICATION-Anweisung [UltraLite]“ auf Seite 447
- „START SYNCHRONIZATION DELETE-Anweisung [UltraLite]“ auf Seite 460
- „STOP SYNCHRONIZATION DELETE-Anweisung [UltraLite]“ auf Seite 461

Beispiel

Die folgende Anweisung ALTER PUBLICATION fügt die Tabelle 'Customers' zur pub_contact-Publikation hinzu.

```
ALTER PUBLICATION pub_contact  
ADD TABLE Customers
```

ALTER SYNCHRONIZATION PROFILE-Anweisung [UltraLite]

Ändert ein UltraLite-Synchronisationsprofil.

Syntax

```
ALTER SYNCHRONIZATION PROFILE sync-profile-name  
MERGE sync-option [; ... ]
```

sync-option :
sync-option-name = *sync-option-value*

sync-option-name : *string*

sync-option-value : *string*

Parameter

- **sync-profile-name** Der Name des Synchronisationsprofils.
- **MERGE-Klausel** Verwenden Sie diese Klausel, um bestehende Optionen in einem Synchronisationsprofil zu ändern oder ihm neue hinzuzufügen.
- **sync-option** Eine Zeichenfolge mit mindestens einem Paar "Option=Wert", getrennt durch Semikola. Beispiel: 'option1=value1;option2=value2'.
- **sync-option-name** Der Name der Synchronisationsprofiloption

- **sync-option-value** Der Wert der Synchronisationsprofiloption.

Bemerkungen

Synchronisationsprofile legen fest, wie eine UltraLite-Datenbank mit dem MobiLink-Server synchronisiert wird.

Sie können die MERGE-Klausel verwenden, um Änderungen an einem bestehenden Synchronisationsprofil durchzuführen. Wenn diese Klausel verwendet wird, werden nur die Synchronisationsoptionen geändert, die in der MERGE-Klausel angegeben sind. Um eine Synchronisationsoption aus einem Synchronisationsprofil zu entfernen, sollte die Synchronisationsoptions-Zeichenfolge aussehen wie 'option1=' (um die Option auf einen leeren Wert zu setzen).

Die Synchronisationsprofiloption STREAM unterscheidet sich von den anderen Optionen, weil ihr Wert eine Unterliste enthält. Zum Beispiel: 'STREAM=TCPIP{host=192.168.1.1;port=1234}'. In diesem Fall ist 'host=192.168.1.1;port=1234' die Unterliste. Um einen Unterlistenwert hinzuzufügen oder zu entfernen, verwenden Sie einen Punkt zwischen dem STREAM-Synchronisationsoptionsnamen und dem Unterlisten-Optionsnamen. Beispiel: MERGE 'stream.port=5678;stream.host=;compression=zlib' ergibt das Synchronisationsprofil stream=TCPIP{port=5678;compression=zlib}. Der Versuch, den Datenstrom auf einen neuen Wert zu setzen, ersetzt den gesamten Datenstromwert. Beispiel: MERGE 'stream=HTTPS' ergibt folgendes Synchronisationsprofil: stream=HTTPS{ }.

Nebenwirkungen

Keine.

Siehe auch

- „Synchronisationsprofiloptionen“ auf Seite 230
- „DROP SYNCHRONIZATION PROFILE-Anweisung [UltraLite]“ auf Seite 448
- „SYNCHRONIZE-Anweisung [UltraLite]“ auf Seite 462

Beispiel

Das Folgende ist ein Beispiel für die Anweisung ALTER SYNCHRONIZATION PROFILE...REPLACE:

```
CREATE SYNCHRONIZATION PROFILE myProfile1;  
ALTER SYNCHRONIZATION PROFILE myProfile1  
    REPLACE 'publications=p1;uploadonly=on';
```

Das Folgende ist ein Beispiel für die Anweisung ALTER SYNCHRONIZATION PROFILE...MERGE:

```
CREATE SYNCHRONIZATION PROFILE myProfile2 'publications=p1;  
ALTER SYNCHRONIZATION PROFILE myProfile2  
    MERGE 'publications=p2;uploadonly=on';
```

Das folgende Beispiel zeigt die Änderungen, die auftreten, nachdem eine Sequenz von ALTER SYNCHRONIZATION PROFILE-Befehlen mit verschiedenen Optionen ausgeführt wurde.

Es gilt folgende Annahme: myProfile1='MobiLinkUID=mary;ScriptVersion=default'.

Nach der Ausführung von `ALTER SYNCHRONIZATION PROFILE myProfile1 REPLACE 'MobiLinkPwd=sql;ScriptVersion=1'` ist myProfile
'MobiLinkPwd=sql;ScriptVersion=1'.

Nach der Ausführung von `ALTER SYNCHRONIZATION PROFILE myProfile1 MERGE 'MobiLinkUID=mary;STREAM=tcPIP'` ist myProfile
'MobiLinkPwd=sql;ScriptVersion=1;MobiLinkUID=mary;STREAM=tcPIP'.

Nach der Ausführung von `ALTER SYNCHRONIZATION PROFILE myProfile1 MERGE 'MobiLinkUID=;STREAM.host=192.168.1.1;STREAM.port=1234;ScriptVersion=;'`
' ist myProfile 'MobiLinkPwd=sql;STREAM=tcPIP{192.168.1.1;port=1234}'.

Nach der Ausführung von `ALTER SYNCHRONIZATION PROFILE myProfile1 MERGE 'MobiLinkPwd=;Ping=yes;STREAM =HTTP'` ist myProfile 'Ping=yes;STREAM=HTTP'.

Nach der Ausführung von `ALTER SYNCHRONIZATION PROFILE myProfile1 MERGE 'STREAM=HTTP{host=192.168.1.1}'` ist myProfile
'Ping=yes;STREAM=HTTP{host=192.168.1.1}'.

ALTER TABLE-Anweisung [UltraLite]

Ändert eine Tabellendefinition.

Syntax

```
ALTER TABLE table-name {  
  add-clause  
  | modify-clause  
  | drop-clause  
  | rename-clause  
}
```

add-clause :
ADD { *column-definition* | *table-constraint* }

modify-clause :
ALTER *column-definition* | *sync-constraint*

drop-clause :
DROP { *column-name* | **CONSTRAINT** *constraint-name* }

rename-clause :
RENAME {
 new-table-name
 | [*old-column-name* **TO**] *new-column-name*
 | **CONSTRAINT** *old-constraint-name* **TO** *new-constraint-name* }

column-definition :
column-name *data-type*
 [[**NOT**] **NULL**]
 [**DEFAULT** *column-default*]
 [**UNIQUE**]

column-default :

GLOBAL AUTOINCREMENT [(*number*)]

AUTOINCREMENT

CURRENT DATE

CURRENT TIME

CURRENT TIMESTAMP

NULL

NEWID()

constant-value

table-constraint :

[**CONSTRAINT** *constraint-name*]

{ *fkey-constraint* | *unique-key-constraint* }

[**WITH MAX HASH SIZE** *integer*]

fkey-constraint :

[**NOT NULL**] **FOREIGN KEY** [*role-name*] (*ordered-column-list*)

REFERENCES *table-name* (*column-name*, ...)

[**CHECK ON COMMIT**]

unique-key-constraint :

UNIQUE (*ordered-column-list*)

ordered-column-list :

(*column-name* [**ASC** | **DESC**], ...)

sync-constraint : **SYNCHRONIZE** { **ON** | **OFF** | **ALL** | **DOWNLOAD** }

Parameter

add-Klausel Fügt einer Tabelle eine neue Spalte oder Tabellenintegritätsregel hinzu:

Fügt einer Tabelle eine neue Spalte oder Tabellenintegritätsregel hinzu:

- **ADD *column-definition*** Fügt einer Tabelle eine neue Spalte hinzu. Wenn die Spalte einen Standardwert hat, werden alle Zeilen der neuen Spalte mit diesem Standardwert gefüllt. Eine Beschreibung der Schlüsselwörter und Unterklauseln für diese Klausel finden Sie unter „[CREATE TABLE-Anweisung \[UltraLite\]](#)“ auf Seite 438.
- **ADD *table-constraint*** Fügt einer Tabelle eine Integritätsregel hinzu. Mit dem optionalen Integritätsregelnamen können Sie später einzelne Integritätsregeln ändern oder löschen, anstatt die gesamte Tabellen-Integritätsregel zu löschen. Eine Beschreibung der Schlüsselwörter und Unterklauseln für diese Klausel finden Sie unter „[CREATE TABLE-Anweisung \[UltraLite\]](#)“ auf Seite 438.

Beim Hinzufügen einer neuen Eindeutigkeits-Integritätsregel müssen alle Integritätsregelspalten nicht nullwertfähig sein. Um Eindeutigkeits-Integritätsregel hinzuzufügen, ändern Sie die Spalte auf NOT NULL.

Hinweis

Es ist nicht möglich, einen Primärschlüssel in UltraLite hinzuzufügen.

modify-Klausel Ändert eine einzelne Spaltendefinition. Sie können keine Primärschlüssel in der *column-definition* verwenden, wenn sie Teil einer ALTER-Anweisung sind. Wenn nötig werden die

Daten in der geänderten Spalte in einen neuen Datentyp konvertiert. Wenn ein Konvertierungsfehler auftritt, schlägt der Vorgang fehl und die Tabelle bleibt unverändert. Eine umfassende Erklärung der *table-definition* finden Sie unter „[CREATE TABLE-Anweisung \[UltraLite\]](#)“ auf Seite 438.

drop-Klausel Löscht eine Spalten- oder Tabellenintegritätsregel.

- **DROP *column-name*** Löscht die Spalte aus der Tabelle. Wenn die Spalte in einem Index, einer Eindeutigkeits-Integritätsregel, einem Fremdschlüssel oder Primärschlüssel enthalten ist, muss das Objekt gelöscht werden, *bevor* die Spalte gelöscht werden kann.
- **DROP CONSTRAINT *table-constraint*** Löscht die genannte Integritätsregel aus der Tabellendefinition. Eine umfassende Erklärung der *table-constraint* finden Sie unter „[CREATE TABLE-Anweisung \[UltraLite\]](#)“ auf Seite 438.

Hinweis

Es ist nicht möglich, einen Primärschlüssel in UltraLite zu löschen.

rename-Klausel Ändert den Namen einer Tabelle, Spalte oder Integritätsregel:

- **RENAME *new-table-name*** Ändert den Namen einer Tabelle in *new-table-name*. Alle Anwendungen, die den alten Tabellennamen benutzen, müssen geändert werden. Fremdschlüssel, denen automatisch der alte Tabellename zugeordnet wurde, ändern nicht ihre Namen.
- **RENAME *old-column-name* TO *new-column-name*** Ändert den Namen der Spalte in *new-column-name*. Alle Anwendungen, die den alten Spaltennamen benutzen, müssen geändert werden.
- **RENAME *old-constraint-name* TO *new-constraint-name*** Ändert den Namen der Integritätsregel in *new-constraint-name*. Alle Anwendungen, die den alten Integritätsregelnamen benutzen, müssen geändert werden.

Hinweis

Es ist nicht möglich, einen Primärschlüssel in UltraLite umzubenennen.

Column-constraint (Spaltenintegritätsregel) Eine Spaltenintegritätsregel schränkt die Werte ein, die eine Spalte enthalten kann, um die Integrität der Daten in der Datenbank sicherzustellen. Eine Spaltenintegritätsregel kann nur UNIQUE sein.

UNIQUE Identifiziert eine oder mehrere Spalten, die jede Zeile einer 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 Eindeutigkeits-Integritätsregel besitzen.

Synchronisations-Integritätsregel-Klausel Geben Sie eine Synchronisations-Integritätsregel an, um zu ermitteln, ob eine Tabelle synchronisiert werden kann oder nicht, und ob alle Upload-Zeilen vorhanden sind, nur Änderungen in der Tabelle übertragen werden, oder keine Änderungen in der Tabelle übertragen werden.

- **SYNCHRONIZE ON** Standardeinstellung: Die Tabelle kann synchronisiert werden und nur die Änderungen in der Tabelle werden im Upload übertragen.

- **SYNCHRONIZE OFF** Die Tabelle kann nicht synchronisiert werden, und es wird ein Fehler gemeldet, wenn die Tabelle in eine Publikation aufgenommen wird.
- **SYNCHRONIZE ALL** Die Tabelle kann synchronisiert werden und alle Zeilen in der Tabelle werden im Upload übertragen. Diese Integritätsregel wird von UltraLite Java Edition-Datenbanken nicht unterstützt.
- **SYNCHRONIZE DOWNLOAD** Die Tabelle kann mit Änderungen in der konsolidierten Datenbank synchronisiert werden, jedoch werden lokale Änderungen im Upload übertragen.

Bemerkungen

Nur eine Tabellenintegritätsregel *table-constraint* oder Spaltenintegritätsregel *column-constraint* kann in einer ALTER TABLE-Anweisung hinzugefügt, geändert oder gelöscht werden.

Der Rollenname ist der Name des Fremdschlüssels. Die Hauptfunktion des Rollennamens *role-name* liegt in der Unterscheidung von zwei Fremdschlüsseln für dieselbe Tabelle. Alternativ dazu können Sie den Fremdschlüssel mit CONSTRAINT *constraint-name* benennen. Verwenden Sie jedoch nicht beide Methoden zur Benennung eines Fremdschlüssels.

Sie können eine Tabellen- oder Spaltenintegritätsregel nicht mit MODIFY ändern. Wenn Sie eine Integritätsregel ändern möchten, löschen Sie die alte Integritätsregel mit DELETE und fügen die neue Integritätsregel mit ADD hinzu.

ALTER TABLE kann nicht ausgeführt werden, wenn eine Anweisung, die die Tabelle betrifft, bereits von einer anderen Anforderung oder Abfrage referenziert wurde. UltraLite verarbeitet auch keine Anforderungen, die eine Tabelle referenzieren, wenn diese Tabelle gerade geändert wird. Außerdem ist es nicht möglich, ALTER TABLE auszuführen, wenn die Datenbank aktive Abfragen oder nicht festgeschriebene Transaktionen enthält.

Für UltraLite.NET-Benutzer: Sie können diese Anweisung nur ausführen, wenn Sie auch die Dispose-Methode für alle Datenobjekte ausführen (z.B. ULDataReader). Siehe [ULBulkCopy.Dispose-Methode \[UltraLite.NET\]](#) [*UltraLite - .NET-Programmierung*].

Anweisungen werden nicht freigegeben, wenn gleichzeitig Schemaänderungen initiiert werden.

Siehe auch

- [UltraLite-Datenbankschemas auf Seite 49](#)
- [„CREATE TABLE-Anweisung \[UltraLite\]“ auf Seite 438](#)
- [„DROP TABLE-Anweisung \[UltraLite\]“ auf Seite 448](#)
- [„UltraLite, SQLDatentypen“ auf Seite 296](#)
- [„Tabellenänderung“ \[*SQL Anywhere Server - SQL-Benutzerhandbuch*\]](#)
- [„Tabellen- und Spalten-Integritätsregeln“ \[*SQL Anywhere Server - SQL-Benutzerhandbuch*\]](#)
- [„Partitionsgrößen“ auf Seite 73](#)
- [„Methoden zum Finden des zuletzt zugewiesenen GLOBAL AUTOINCREMENT-Werts“ auf Seite 72](#)

Beispiele

Die folgende Anweisung löscht die Street-Spalte aus einer fiktiven Tabelle namens MyEmployees.

```
ALTER TABLE MyEmployees
DROP Street
```

Das folgende Beispiel ändert die Street-Spalte der fiktiven Tabelle MyCustomers, damit sie ca. 50 Zeichen enthalten kann.

```
ALTER TABLE MyCustomers
ALTER Street CHAR(50)
```

ALTER USER-Anweisung [UltraLite]

Ändert Benutzereinstellungen.

Hinweis

Diese Anweisung wird von UltraLite Java Edition-Datenbanken nicht unterstützt.

Syntax 1

```
ALTER USER user-name [ IDENTIFIED BY password ]
```

Parameter

user-name Der Name des Benutzers.

IDENTIFIED BY-Klausel Das Kennwort für den Benutzer

Bemerkungen

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

Nebenwirkungen

Keine.

Siehe auch

- „CREATE USER-Anweisung [UltraLite]“ auf Seite 444
- „DROP USER-Anweisung [UltraLite]“ auf Seite 449

Beispiel

Die nachstehende Anweisung ändert einen Benutzer namens SQLTester. Das Kennwort wird auf "welcome" eingestellt.

```
ALTER USER SQLTester IDENTIFIED BY welcome
```

CHECKPOINT-Anweisung [UltraLite]

Setzt Checkpoints in der Datenbank.

Hinweis

Diese Anweisung wird von UltraLite Java Edition-Datenbanken nicht unterstützt.

Syntax

CHECKPOINT

Bemerkungen

Sie können die CHECKPOINT-Anweisung als Auslöser für eine Bereinigung von Festschreibungen verwenden. Bei einer Bereinigung von Festschreibungen werden nicht festgeschriebene Transaktionen in den Speicher geschrieben.

Wenn Sie die Embedded SQL-API verwenden, können Sie auch die Methode `ULCheckpoint` verwenden. Wenn Sie eine C++-Komponentenanwendung schreiben, können Sie auch die Methode `Checkpoint` bei einem Verbindungsobjekt verwenden. Alle anderen APIs müssen diese Anweisung verwenden.

Nebenwirkungen

Diese Anweisung schreibt alle ausstehenden festgeschriebenen Transaktionen in den Speicher, jedoch keine aktuellen Transaktionen.

Siehe auch

- „Bereinigen einzelner oder gruppierter Transaktionen“ auf Seite 487
- „COMMIT-Anweisung [UltraLite]“ auf Seite 432
- „UltraLite-Verbindungsparameter `COMMIT_FLUSH`“ auf Seite 175
- `ULCheckpoint`-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]
- `ULConnection.Checkpoint`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]

Beispiel

Die folgende Anweisung setzt einen Checkpoint in der Datenbank:

```
CHECKPOINT
```

COMMIT-Anweisung [UltraLite]

Macht die Änderungen in der Datenbank dauerhaft.

Syntax

COMMIT [WORK]

Bemerkungen

Bei der Verwendung von UltraLite SQL wird eine Transaktion erstellt. Eine Transaktion besteht aus allen Änderungen (INSERT-, UPDATE- und DELETE-Anweisungen) seit dem letzten ROLLBACK- bzw. COMMIT-Vorgang. Die COMMIT-Anweisung beendet die aktuelle Transaktion und macht alle Änderungen in der Datenbank dauerhaft, die während der Transaktion vorgenommen wurden.

Änderungen, die an den Datenbankobjekten mit ALTER-, CREATE- und DROP-Anweisungen durchgeführt wurden, werden automatisch festgeschrieben.

Siehe auch

- „CHECKPOINT-Anweisung [UltraLite]“ auf Seite 432
- „ROLLBACK-Anweisung [UltraLite]“ auf Seite 457

Beispiel

Die folgende Anweisung macht die Änderungen der aktuellen Transaktion in der Datenbank dauerhaft:

```
COMMIT
```

CREATE INDEX-Anweisung [UltraLite]

Erstellt einen Index für eine angegebene Tabelle.

Syntax

```
CREATE [ UNIQUE ] INDEX [ IF NOT EXISTS ] [ index-name ]  
ON table-name ( ordered-column-list )  
[ WITH MAX HASH SIZE integer ]
```

```
ordered-column-list :  
( column-name [ ASC | DESC ], ... )
```

Parameter

UNIQUE Das UNIQUE-Attribut stellt sicher, dass in der Tabelle keine zwei Zeilen vorhanden sind, die in allen Indexspalten identische Werte aufweisen. Jeder Indexschlüssel muss eindeutig sein oder NULL in mindestens einer Spalte enthalten.

Es besteht ein Unterschied zwischen einer Eindeutigkeits-Integritätsregel in einer Tabelle und einem eindeutigen Index. Spalten mit einem eindeutigen Index lassen NULL zu, nicht aber Spalten in einer Eindeutigkeits-Integritätsregel. Ein Fremdschlüssel kann entweder einen Primärschlüssel oder eine Eindeutigkeits-Integritätsregel referenzieren, aber keinen eindeutigen Index, da dieser mehrere Instanzen von NULL enthalten kann.

Wenn die Spalten in einer eindeutigen Integritätsregel während einer Aktualisierung geändert werden und ein Fremdschlüssel diese eindeutige Integritätsregel referenziert, werden Zeilen, die keine Zeilen in der eindeutigen Regel referenzieren, aus der entfernten Datenbank gelöscht.

IF NOT EXISTS-Klausel Wenn das Attribut IF NOT EXISTS angegeben wurde und der benannte Index bereits vorhanden ist, werden keine Änderungen vorgenommen und es wird kein Fehler zurückgegeben.

ordered-column-list Eine sortierte Liste von Spalten. Spaltenwerte im Index können in aufsteigender oder absteigender Reihenfolge sortiert werden.

WITH MAX HASH SIZE Legt die Hash-Größe (in Byte) für diesen Index fest. Dieser Wert setzt den Standardwert der Eigenschaft MaxHashSize außer Kraft, der für die Datenbank gilt. Weitere Hinweise zur Standardgröße finden Sie unter [„Lesen der Datenbankeigenschaften“ auf Seite 39](#).

Bemerkungen

UltraLite erstellt automatisch Indizes für Primärschlüssel und Eindeutigkeits-Integritätsregeln.

Indizes können die Abfrageperformance steigern, indem sie es UltraLite ermöglichen, bestimmte Zeilen schnell zu finden. Im Gegensatz dazu können Indizes die Synchronisation sowie INSERT-, DELETE- und UPDATE-Anweisungen verlangsamen, da sie verwaltet werden müssen.

Indizes werden automatisch verwendet, um die Performance von Abfragen zu verbessern, die von der Datenbank ausgegeben werden, und um Abfragen mithilfe einer ORDER BY-Klausel zu sortieren. Sobald ein Index erstellt wurde, wird er nie wieder in einer SQL-Anweisung referenziert, außer wenn er mit DROP INDEX gelöscht wird.

Indizes brauchen Speicherplatz in der Datenbank. Der zusätzliche Aufwand für die Verwaltung der Indizes kann die Performance der Datenänderungsvorgänge beeinträchtigen. Aus diesem Grund sollten Sie nur Indizes erstellen, die die Abfrageperformance verbessern.

UltraLite verarbeitet keine Anforderungen oder Abfragen, die die Tabelle referenzieren, während die Anweisung CREATE INDEX ausgeführt wird. Außerdem ist es nicht möglich, CREATE INDEX auszuführen, wenn die Datenbank aktive Abfragen oder nicht festgeschriebene Transaktionen enthält.

UltraLite kann für Entwickler Ausführungspläne zur Optimierung von Abfragen zur Verfügung stellen. Siehe [„Ausführungspläne in UltraLite“ auf Seite 477](#).

Für UltraLite.NET-Benutzer: Sie können diese Anweisung nur ausführen, wenn Sie auch die Dispose-Methode für alle Datenobjekte ausführen (z.B. ULDataReader). Siehe [ULBulkCopy.Dispose-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#).

Anweisungen werden nicht freigegeben, wenn gleichzeitig Schemaänderungen initiiert werden. Siehe [UltraLite-Datenbankschemas auf Seite 49](#).

Nebenwirkungen

- Automatisches Festschreiben (Autocommit).

Siehe auch

- [„UltraLite Performance-Tipps“ auf Seite 469](#)
- [„DROP INDEX-Anweisung \[UltraLite\]“ auf Seite 446](#)
- [„UltraLite-Erstellungsparameter max_hash_size“ auf Seite 151](#)
- [„UltraLite-Indizes“ auf Seite 56](#)

Beispiel

Im folgenden Beispiel wird ein zweispaltiger Index für die Tabelle Employee erstellt.

```
CREATE INDEX employee_name_index
ON Employees ( Surname, GivenName )
```

Die folgende Anweisung erstellt einen Index auf der SalesOrderItems-Tabelle für die ProductID-Spalte.

```
CREATE INDEX item_prod
ON SalesOrderItems ( ProductID )
```

Das folgende Szenario zeigt die Auswirkungen von MAX HASH SIZE in einer UltraLite Java Edition-Datenbank, sofern eine Tabelle "Employees" eine Spalte "Initials" mit dem Typ VARCHAR(3) sowie eine Spalte EmployeeID mit dem Typ TINY enthält.

Die folgende Anweisung zerlegt alle Werte vollständig, wenn nur ASCII7-Zeichen benutzt werden:

```
CREATE INDEX ascii_a ON Employees( Initials ) WITH MAX HASH SIZE 3
```

Die folgende Anweisung zerlegt alle Werte vollständig, unabhängig von den enthaltenen Zeichen:

```
CREATE INDEX unicode_a ON Employees( Initials ) WITH MAX HASH SIZE 9
```

Die folgende Anweisung zerlegt nur die "Initials"-Werte, auch wenn nur ASCII-Zeichen benutzt werden, da die ersten 9 Byte für "Initials" reserviert sind:

```
CREATE INDEX compound_1 ON Employees( Initials, EmployeeID ) WITH MAX HASH
SIZE 9
```

Die folgende Anweisung zerlegt sowohl die Werte in "Initials" als auch in "EmployeeID":

```
CREATE INDEX compound_2 ON Employees( Initials, EmployeeID ) WITH MAX HASH
SIZE 10
```

CREATE PUBLICATION-Anweisung [UltraLite]

Erstellt eine Publikation.

Syntax

```
CREATE PUBLICATION [ IF NOT EXISTS ] publication-name
( TABLE table-name [ WHERE search-condition ], ... )
```

Parameter

- **IF NOT EXISTS-Klausel** Wenn die IF NOT EXISTS-Klausel angegeben wurde und die benannte Publikation bereits vorhanden ist, werden keine Änderungen vorgenommen und es wird kein Fehler zurückgegeben.
- **TABLE-Klausel** Verwenden Sie die TABLE-Klausel, um eine Tabelle in die Publikation aufzunehmen. Es gibt keine Beschränkung für die Anzahl der TABLE-Klauseln.
- **WHERE-Klausel** Wenn eine WHERE-Klausel angegeben ist, werden während der Synchronisation nur Zeilen für den Upload aus der zugeordneten Tabelle berücksichtigt, die die *search-condition* erfüllen.

Wenn Sie keine WHERE-Klausel angeben, wird jede Zeile in der Tabelle, die in UltraLite seit der letzten Synchronisation geändert wurde, beim Upload berücksichtigt.

Bemerkungen

Eine Publikation identifiziert synchronisierte Daten in einer entfernten UltraLite-Datenbank.

Eine Publikation richtet Tabellen ein, die während eines einzelnen Synchronisationsvorgangs synchronisiert werden, und legt fest, welche Daten zum MobiLink-Server hochgeladen werden. Der MobiLink-Server sendet möglicherweise Zeilen für diese (und nur diese) Tabellen während seiner Download-Sitzung zurück. Heruntergeladene Zeilen müssen allerdings nicht der WHERE-Klausel für eine Tabelle entsprechen.

Nur ganze Tabellen können veröffentlicht werden. Es ist nicht möglich, spezifische Spalten aus einer Tabelle in UltraLite zu publizieren.

Nebenwirkungen

- Automatisches Festschreiben (Autocommit).

Siehe auch

- „Suchbedingungen in UltraLite“ auf Seite 283
- „UltraLite-Clients“ auf Seite 69
- „DROP PUBLICATION-Anweisung [UltraLite]“ auf Seite 447
- „ALTER PUBLICATION-Anweisung [UltraLite]“ auf Seite 424
- „Suchbedingungen in UltraLite“ auf Seite 283

Beispiel

Mit der folgenden Anweisung werden alle Spalten und Zeilen von zwei Tabellen publiziert.

```
CREATE PUBLICATION pub_contact (  
    TABLE Contacts,  
    TABLE Customers  
)
```

Mit der folgenden Anweisung werden nur die Zeilen der Customers-Tabelle publiziert, bei denen die State-Spalte MN enthält

```
CREATE PUBLICATION pub_customer (  
    TABLE Customers  
    WHERE State = 'MN'  
)
```

CREATE SYNCHRONIZATION PROFILE-Anweisung [UltraLite]

Erstellt oder ersetzt ein UltraLite-Synchronisationsprofil.

Syntax

CREATE [OR REPLACE] SYNCHRONIZATION PROFILE *sync-profile-name sync-option* [;...]

sync-option :

sync-option-name = *sync-option-value*

sync-option-name : *string*

sync-option-value : *string*

Parameter

- **OR REPLACE-Klausel** Wenn ein benanntes Synchronisationsprofil bereits vorhanden ist, wird es ersetzt. Wenn das Profil nicht existiert, wird es erstellt.
- **sync-profile-name** Der Name des Synchronisationsprofils.
- **sync-option** Eine Zeichenfolge mit mindestens einem Paar "Option=Wert", getrennt durch Semikola. Beispiel: 'option1=value1;option2=value2'.
- **sync-option-name** Der Name der Synchronisationsprofiloption
- **sync-option-value** Der Wert der Synchronisationsprofiloption.

Bemerkungen

Synchronisationsprofile legen fest, wie eine UltraLite-Datenbank mit dem MobiLink-Server synchronisiert wird.

Sie können die REPLACE-Klausel verwenden, um Änderungen an einem bestehenden Synchronisationsprofil durchzuführen. Diese Klausel ersetzt den Inhalt des Synchronisationsprofils mit dem, was in der neuen Synchronisationsoptions-Zeichenfolge enthalten ist. Dieser Ansatz entspricht einem Löschen des Synchronisationsprofils, um danach eines mit demselben Namen, aber unter Verwendung der neuen Zeichenfolge zu erstellen. Ein Synchronisationsprofil muss daher keine vollständige Synchronisationsdefinition enthalten, weil Parameter während der Synchronisation zusammengeführt oder überschrieben werden können.

Die Synchronisationsprofiloption STREAM unterscheidet sich von den anderen Optionen, weil ihr Wert eine Unterliste enthält. Zum Beispiel: 'STREAM=TCPIP{host=192.168.1.1;port=1234}'. In diesem Fall ist 'host=192.168.1.1;port=1234' die Unterliste. Um einen Unterlistenwert hinzuzufügen oder zu entfernen, verwenden Sie einen Punkt zwischen dem STREAM-Synchronisationsoptionsnamen und dem Unterlisten-Optionsnamen. Beispiel: MERGE 'stream.port=5678;stream.host=;compression=zlib' ergibt das Synchronisationsprofil stream=TCPIP{port=5678;compression=zlib}. Der Versuch, den Datenstrom auf einen neuen Wert zu setzen, ersetzt den gesamten Datenstromwert. Beispiel: MERGE 'stream=HTTPS' ergibt folgendes Synchronisationsprofil: stream=HTTPS{ }.

Nebenwirkungen

Keine.

Siehe auch

- „Synchronisationsprofiloptionen“ auf Seite 230
- „ALTER SYNCHRONIZATION PROFILE-Anweisung [UltraLite]“ auf Seite 425
- „DROP SYNCHRONIZATION PROFILE-Anweisung [UltraLite]“ auf Seite 448
- „SYNCHRONIZE-Anweisung [UltraLite]“ auf Seite 462

Beispiel

Das folgende Beispiel erstellt ein Synchronisationsprofil namens Test1.

```
CREATE SYNCHRONIZATION PROFILE Test1  
'MobiLinkId=mary;Stream=TCPIP{host=192.168.1.1;port=1234}'
```

CREATE TABLE-Anweisung [UltraLite]

Erstellt eine neue Tabelle.

Syntax

```
CREATE TABLE [ IF NOT EXISTS ] table-name (  
  { column-definition | table-constraint | sync-constraint }, ...  
)
```

```
column-definition :  
column-name data-type  
[ [ NOT ] NULL ]  
[ DEFAULT column-default ]  
[ STORE AS FILE (file-name-column) [ CASCADE DELETE ]  
[ column-constraint ]
```

```
column-default :  
AUTOFILENAME(prefix,extension)  
| GLOBAL AUTOINCREMENT [ ( number ) ]  
| AUTOINCREMENT  
| CURRENT DATE  
| CURRENT TIME  
| CURRENT TIMESTAMP  
| CURRENT UTC TIMESTAMP  
| NULL  
| NEWID( )  
| constant-value
```

```
file-name  
"filename"
```

```
column-constraint :  
PRIMARY KEY  
| UNIQUE
```

```
table-constraint :  
{ [ CONSTRAINT constraint-name ]  
  pkey-constraint  
  | fkey-constraint  
  | unique-key-constraint }  
[ WITH MAX HASH SIZE integer ]
```

```
pkey-constraint :  
PRIMARY KEY [ ordered-column-list ]
```

```
fkey-constraint :  
[ NOT NULL ] FOREIGN KEY [ role-name ] ( ordered-column-list )
```

REFERENCES *table-name* (*column-name*, ...)
 [**CHECK ON COMMIT**]

unique-key-constraint :
UNIQUE (*ordered-column-list*)

ordered-column-list :
 (*column-name* [**ASC** | **DESC**], ...)

sync-constraint : **SYNCHRONIZE** { **ON** | **OFF** | **ALL** | **DOWNLOAD** }

Parameter

IF NOT EXISTS-Klausel Mit dieser Klausel erstellen Sie eine Tabelle. Existiert die angegebene Tabelle bereits, werden keine Änderungen durchgeführt und keine Fehlermeldung ausgegeben.

column-definition Legt eine Spalte in einer Tabelle fest. Für diese Klausel sind folgende Parameter verfügbar:

- **column-name** Der Spaltenname ist ein Bezeichner. Mehrere Spalten in derselben Tabelle können nicht denselben Namen haben. Siehe „[Bezeichner in UltraLite](#)“ auf Seite 271.

UltraLite Java Edition-Datenbanken unterstützen die Partitionierung von Datenbankdateien, sodass externe Dateien jetzt verwendet werden können, um große BLOB-Werte zu speichern, wobei die Dateien in zwei Spalten referenziert werden: Die erste speichert den Dateinamen und muss über den Datentyp CHAR(Größe)... AUTOFILENAME(...) verfügen, die zweite wird verwendet, um auf den Inhalt der Datei zuzugreifen. Sie ist als Datentyp LONG BINARY STORE AS... definiert. Die Spalte mit dem Dateiinhalt ist schreibgeschützt.

- **data-type** Der Datentyp der Spalte. Siehe „[UltraLite, SQLDatentypen](#)“ auf Seite 296.
- **[NOT] NULL** Wenn NOT NULL angegeben ist, oder wenn die Spalte in einer UNIQUE- oder PRIMARY KEY-Integritätsregel vorkommt, kann die Spalte in keiner Zeile NULL enthalten. Andernfalls ist NULL erlaubt.
- **column-default** Setzt den Standardwert für die Spalte. Wenn ein DEFAULT-Wert angegeben ist, wird er als Wert für die Spalte in jeder INSERT-Anweisung benutzt, die für diese Spalte keinen Wert angibt. Wenn DEFAULT nicht angegeben wird, entspricht er DEFAULT NULL. Folgende Standardoptionen sind verfügbar:

- **AUTOFILENAME** Diese Klausel unterstützt das Speichern von externen BLOB-Dateien in einer partitionierten UltraLite Java Edition-Datenbank.

Beim Partitionieren der Datenbank muss die Spalte, die zum Speichern der Dateinamen bestimmt ist, die AUTOFILENAME(Präfix,Erweiterung)-Klausel enthalten. Diese Klausel gibt an, wie neue Dateinamen für eingelesene BLOB-Werte generiert werden sollen. Die Werte für Präfix und Erweiterung sind Zeichenliteral-Konstanten. Siehe [Connection.OPTION_BLOB_FILE_BASE_DIR-Variable \[BlackBerry\] \[UltraLiteJ\] \[UltraLite® – Java-Programmierung\]](#).

- **AUTOINCREMENT** Wenn AUTOINCREMENT verwendet wird, muss die Spalte einer der Ganzzahl-Datentypen oder ein numerisch exakter Typ sein. Ist beim Einfügen in die Tabelle

kein Wert für die AUTOINCREMENT-Spalte vorgegeben, wird ein eindeutiger Wert erstellt, der größer ist als alle Werte in der Spalte. Wenn eine INSERT-Anweisung einen Wert für die Spalte festlegt, der größer als der aktuelle Maximalwert für die Spalte ist, wird dieser Wert als Startpunkt für nachfolgende Einfügungen verwendet.

Tipp

In UltraLite wird der autoincrement-Wert nicht auf 0 gesetzt, wenn die Tabelle erstellt wird, und AUTOINCREMENT generiert negative Zahlen, wenn ein Datentyp mit Vorzeichen für die Spalte verwendet wird. Sie sollten daher AUTOINCREMENT-Spalten als Integer-Datentyp ohne Vorzeichen deklarieren, um die Verwendung negativer Werte zu verhindern.

- **GLOBAL AUTOINCREMENT** Ähnlich wie AUTOINCREMENT, mit dem Unterschied, dass die Domäne partitioniert ist. Jede Teilmenge enthält dieselbe Anzahl von Werten. Sie ordnen jeder Kopie der Datenbank eine eindeutige Datenbank-Identifizierungsnummer zu. UltraLite liefert Standardwerte in einer Datenbank nur von der Teildomäne, die eindeutig durch diese Datenbanknummer gekennzeichnet ist.

Tipp

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.

Siehe „[GLOBAL AUTOINCREMENT-Spalten in UltraLite deklarieren](#)“ auf Seite 71 und „[UltraLite-Option global_database_id](#)“ auf Seite 198.

- **[NOT] NULL** Steuert, ob die Spalte NULL-Werte enthalten kann.
- **NEWID()** Eine Funktion, die einen eindeutigen bezeichnenden Wert generiert. Siehe „[NEWID-Funktion \[Verschiedene\]](#)“ auf Seite 379.
- **CURRENT TIMESTAMP** Kombiniert CURRENT DATE und CURRENT TIME zu einem TIMESTAMP-Wert, der Jahr, Monat, Tag, Stunde, Minute, Sekunde und Sekundenbruchteile enthält. Der Sekundenbruchteil wird mit 3 Dezimalstellen gespeichert. Die Genauigkeit ist durch die Genauigkeit der Systemuhr begrenzt. Siehe „[CURRENT TIMESTAMP-Spezialwert](#)“ auf Seite 275.
- **| CURRENT UTC TIMESTAMP** Ein TIMESTAMP WITH TIMEZONE Wert, der die Coordinated Universal Time (UTC) mit Jahr, Monat, Tag, Stunde, Minute, Sekunde, Sekundenbruchteil und Zeitzone enthält. Der Sekundenbruchteil wird mit 3 Dezimalstellen gespeichert. Die Genauigkeit ist durch die Genauigkeit der Systemuhr begrenzt. Siehe „[CURRENT UTC TIMESTAMP-Spezialwert](#)“ auf Seite 276.
- **CURRENT DATE** Speichert das aktuelle Jahr sowie den aktuellen Monat und Tag. Siehe „[CURRENT DATE-Spezialwert](#)“ auf Seite 274.
- **CURRENT TIME** Speichert die aktuelle Stunde, Minute, Sekunde und Sekundenbruchteile. Siehe „[CURRENT TIME-Spezialwert](#)“ auf Seite 274.

- **Konstantenwert** Eine Konstante für den Datentyp der Spalte. Die Konstante ist gewöhnlich eine Zahl oder eine Zeichenfolge.
- **STORE AS FILE (*file-name-column*) [CASCADE DELETE]** Nur von UltraLite Java Edition-Datenbanken unterstützt.

Legen Sie fest, dass eine LONG BINARY-Spalte extern gespeichert werden soll (Partitionierung der Datenbank) und legen Sie **file-name-column** fest, um die Spalte zu benennen, die verwendet wird, um die Dateinamen der extern gespeicherte BLOB-Werte aufzunehmen. Eine Spalte mit dieser Klausel muss vom Typ LONG BINARY sein und verhält sich wie eine schreibgeschützte Spalte.

- **column-constraint-Klausel** Geben Sie eine Spaltenintegritätsregel an, um die in einer Spalte zulässigen Werte einzuschränken. Eine Spaltenintegritätsregel kann Folgendes sein:
 - **PRIMARY KEY** Wenn sie Teil einer *column-constraint* ist, legt die PRIMARY KEY-Klausel die Spalte als Primärschlüssel für die Tabelle fest. Primärschlüssel kennzeichnen eindeutig jede Zeile in einer Tabelle. Standardmäßig lassen in Primärschlüsseln aufgenommene Spalten nicht NULL zu.
 - **UNIQUE** Identifiziert eine oder mehrere Spalten, die jede Zeile einer 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 Eindeutigkeits-Integritätsregel besitzen. NULL ist nicht zulässig.

Tabellenintegritätsregel-Klausel Geben Sie eine Tabellenintegritätsregel an, um die Werte zu beschränken, die eine oder mehrere Spalten in der Tabelle enthalten können. Verwenden Sie die CONSTRAINT-Klausel, um einen Bezeichner für die Tabellenintegritätsregel anzugeben. Tabellenintegritätsregeln können eine Primärschlüssel-Integritätsregel, eine Fremdschlüssel-Integritätsregel oder eine Eindeutigkeitsintegritätsregel sein, wie unten beschrieben:

- **Primärschlüssel-Integritätsregel-Klausel** Legt die angegebene(n) Spalte(n) als Primärschlüssel für die Tabelle fest. Primärschlüssel kennzeichnen eindeutig jede Zeile in einer Tabelle. Spalten, die in Primärschlüsseln enthalten sind, dürfen NULL-Werte nicht zulassen.
- **Fremdschlüssel-Integritätsregel-Klausel** Geben Sie eine Fremdschlüssel-Integritätsregel an, um die Werte von einer oder mehreren Spalten zu beschränken, die mit den Werten in einem Primärschlüssel (oder einer Eindeutigkeits-Integritätsregel) einer anderen Tabelle übereinstimmen müssen.
 - **NOT NULL-Klausel** Geben Sie NOT NULL an, um NULL-Werte in den Fremdschlüsselspalten nicht zuzulassen. NULL in einem Fremdschlüssel bedeutet, dass dieser Zeile in der Fremdtabelle keine Zeile in der Primärtabelle entspricht. Wenn mindestens ein Wert in einem mehrspaltigen Fremdschlüssel NULL ist, werden die Werte nicht beschränkt, die in anderen Spalten des Schlüssels enthalten sein können.
 - **Rollenname-Klausel** Geben Sie einen *role-name* an, um den Fremdschlüssel zu benennen. *role-name* wird verwendet, um Fremdschlüssel in derselben Tabelle voneinander zu unterscheiden. Alternativ dazu können Sie den Fremdschlüssel mit CONSTRAINT *constraint-name* benennen. Verwenden Sie jedoch nicht beide Methoden zur Benennung eines Fremdschlüssels.

- **REFERENCES-Klausel** Geben Sie die REFERENCES-Klausel an, um eine oder mehrere Spalten in der Primärtabelle festzulegen, die als Fremdschlüssel-Integritätsregel verwendet werden sollen. Jeder *column-name*, der von Ihnen in einer REFERENCES-Spaltenintegritätsregel angegeben wird, muss eine Spalte in der Primärtabelle sein, für die eine Eindeutigkeits-Integritätsregel oder Primärschlüssel-Integritätsregel gilt.
- **CHECK ON COMMIT** Nicht unterstützt von UltraLite Java Edition-Datenbanken. Geben Sie CHECK ON COMMIT an, damit der Datenbankserver auf ein COMMIT wartet, bevor er Fremdschlüssel-Integritätsregeln erzwingt. Standardmäßig werden Fremdschlüssel-Integritätsregeln während Einfügings-, Lösche- und Aktualisierungsvorgängen sofort erzwungen. Wenn allerdings CHECK ON COMMIT gesetzt ist, können Datenbankänderungen in beliebiger Reihenfolge durchgeführt werden, auch wenn sie Fremdschlüssel-Integritätsregeln verletzen, sofern inkonsistente Daten vor dem nächsten COMMIT aufgelöst werden.
- **Eindeutigkeits-Integritätsregel-Klausel** Geben Sie eine Eindeutigkeits-Integritätsregel an, um eine oder mehrere Spalten festzulegen, die jede Zeile in der Tabelle eindeutig identifizieren. Es kann in einer Tabelle nicht mehrere Zeilen mit denselben Werten in der bzw. allen benannten Spalte(n) geben. Eine Tabelle kann mehr als eine Eindeutigkeits-Integritätsregel besitzen.
- **WITH MAX HASH SIZE** Legt die Hash-Größe (in Byte) für diesen Index fest. Dieser Wert setzt den Standardwert der Eigenschaft MaxHashSize außer Kraft, der für die Datenbank gilt. Hinweise zur Standardgröße finden Sie unter [„Lesen der Datenbankeigenschaften“ auf Seite 39](#). Siehe auch [„UltraLite-Erstellungsparameter max_hash_size“ auf Seite 151](#).

Synchronisations-Integritätsregel-Klausel Geben Sie eine Synchronisations-Integritätsregel an, um zu ermitteln, ob eine Tabelle synchronisiert werden kann oder nicht, und ob alle Upload-Zeilen vorhanden sind, nur Änderungen in der Tabelle übertragen werden, oder keine Änderungen in der Tabelle übertragen werden.

- **SYNCHRONIZE ON** Standardeinstellung: Die Tabelle kann synchronisiert werden und nur die Änderungen in der Tabelle werden im Upload übertragen.
- **SYNCHRONIZE OFF** Die Tabelle kann nicht synchronisiert werden, und es wird ein Fehler gemeldet, wenn die Tabelle in eine Publikation aufgenommen wird.
- **SYNCHRONIZE ALL** Die Tabelle kann synchronisiert werden und alle Zeilen in der Tabelle werden im Upload übertragen. Diese Integritätsregel wird von UltraLite Java Edition-Datenbanken nicht unterstützt.
- **SYNCHRONIZE DOWNLOAD** Die Tabelle kann mit Änderungen in der konsolidierten Datenbank synchronisiert werden, jedoch werden lokale Änderungen im Upload übertragen.

Bemerkungen

Normalerweise werden Spaltenintegritätsregeln verwendet, es sei denn, die Integritätsregel referenziert mehr als nur eine Spalte in der Tabelle. In diesen Fällen muss eine Tabellenintegritätsregel verwendet werden. Wenn eine Anweisung eine Verletzung einer Integritätsregel bewirkt, wird die Ausführung der Anweisung nicht abgeschlossen. Alle Änderungen, die von der Anweisung vor der Fehlererkennung durchgeführt wurden, werden zurückgesetzt und ein Fehler wird gemeldet.

Jede Zeile in der Tabelle besitzt einen eindeutigen Primärschlüsselwert.

Wenn kein Rollenname angegeben ist, wird der Rollenname wie folgt zugeordnet:

1. Wenn es keinen Fremdschlüssel mit einem Rollennamen gibt, der genauso lautet wie der Tabellename, wird der Tabellename als Rollenname zugeordnet.
2. Wenn der Tabellename bereits vergeben ist, wird der Rollenname der Tabellename, der aus einer für diese Tabelle eindeutigen dreistelligen und mit Nullen aufgefüllten Zahl zusammengesetzt ist.

Schemaänderungen Anweisungen werden nicht freigegeben, wenn gleichzeitig Schemaänderungen initiiert werden. Siehe [UltraLite-Datenbankschemas auf Seite 49](#).

UltraLite verarbeitet keine Anforderungen oder Abfragen, die die Tabelle referenzieren, während die Anweisung CREATE TABLE ausgeführt wird. Außerdem ist es nicht möglich, CREATE TABLE auszuführen, wenn die Datenbank aktive Abfragen oder nicht festgeschriebene Transaktionen enthält.

Für UltraLite.NET-Benutzer: Sie können diese Anweisung nur ausführen, wenn Sie auch die Dispose-Methode für alle Datenobjekte ausführen (z.B. ULDataReader). Siehe [ULBulkCopy.Dispose-Methode \[UltraLite.NET\] \[UltraLite - .NET-Programmierung\]](#).

Synchronisation von externen BLOB-Spalten (nur UltraLiteJ) In der konsolidierten Datenbank wird die Dateinamensspalte als reguläre CHAR-Spalte und die BLOB-Dateispalte als reguläre BLOB (LONG BINARY)-Spalte gespeichert. Bei einem Download wird die Dateinamensspalte ignoriert. Es wird ein neuer Dateiname anhand der Datenbankoption (Connection.OPTION_BLOB_FILE_BASE_DIR), des Präfixes und der in der DEFAULT AUTOFILENAME-Klausel angegebenen Erweiterungszeichenfolgen generiert. Für J2SE verwendet der in der Spalte mit dem Dateinamen gespeicherte Wert die Syntax <Datenbankoption-Blobdatei-Basisverzeichnis><Präfix><Autogenerierter_Ganzzahlwert>.<Erweiterung> und für BlackBerry verwendet der gespeicherte Wert die Syntax <Präfix><Autogenerierter_Ganzzahlwert>.<Erweiterung>. Aus diesem Grund sind für BlackBerry generierte Dateinamen immer relativ.

Zugriff auf externe BLOB-Spalten (nur UltraLiteJ) Dateien mit externen BLOB-Werten werden nur geöffnet, wenn die Clientanwendung versucht, die Spalte mit den Werten zu lesen. Zu diesem Zeitpunkt muss der Dateiname, der in der durch die STORE AS FILE-Klausel bezeichneten Spalte gespeichert ist, ein gültiger Dateiname sein. Für den BlackBerry werden relative Dateinamen anhand der Datenbankoption OPTION_BLOB_FILE_BASE_DIR aufgelöst. Wenn UltraLite ermittelt, dass ein Dateiname nicht mit dem Präfix "file://" beginnt, wird dem Dateinamen der in der OPTION_BLOB_FILE_BASE_DIR-Option enthaltene Wert vorangestellt, bevor versucht wird, die Datei zu öffnen.

Eine BLOB-Datei wird in die Datenbank eingefügt, indem ein Dateiname für die **file-name-column**-Spalte festgelegt wird. Der eingefügt Dateiname muss ein gültiger Dateiname sein und bei BlackBerry das Format für einen vollqualifizierten, absoluten Pfaddateinamen gemäß der Beschreibung im Datei-URL-Format in IETF RFCs 1738 & 2396 einhalten (siehe Paketbeschreibung javax.microedition.io.file, in der BlackBerry JDE API-Dokumentation). Beim Einfügen kann ein Wert für die STORE AS FILE-Spalte nicht angegeben werden.

Sobald eine Datei in die Datenbank eingefügt wurde, übernimmt die Datenbank die vollständige Kontrolle über die Datei und geht davon aus, dass außerhalb keine Änderungen vorgenommen werden.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Ausdrücke in UltraLite“ auf Seite 277
- „DROP TABLE-Anweisung [UltraLite]“ auf Seite 448
- „CREATE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „UltraLite, SQLDatentypen“ auf Seite 296
- „Partitionsgrößen“ auf Seite 73

Beispiel

Mit der folgenden Anweisung wird eine Tabelle für eine Bibliotheksdatenbank zur Aufnahme von Bücherdaten erstellt.

```
CREATE TABLE library_books (  
    isbn CHAR(20)          PRIMARY KEY,  
    copyright_date        DATE,  
    title                  CHAR(100),  
    author                 CHAR(50),  
    location               CHAR(50),  
    FOREIGN KEY location REFERENCES room  
)
```

Die folgende Anweisung erstellt eine Tabelle für eine Bibliotheksdatenbank zur Aufnahme von Angaben über ausgeliehene Bücher. Der Standardwert für date_borrowed zeigt an, dass das Buch an dem Tag ausgeliehen wurde, an dem der Eintrag erfolgte. Die Spalte date_returned ist NULL, bis das Buch zurückgegeben wird.

```
CREATE TABLE borrowed_book (  
    loaner_name    CHAR(100) PRIMARY KEY,  
    date_borrowed DATE NOT NULL DEFAULT CURRENT DATE,  
    date_returned  DATE,  
    book           CHAR(20),  
    FOREIGN KEY (book) REFERENCES library_books (isbn)  
)
```

Mit der folgenden Anweisung werden Tabellen für die Datenbank einer Verkaufsabteilung zur Aufnahme von Angaben zu Aufträgen und Auftragspositionen erstellt.

```
CREATE TABLE Orders (  
    order_num INTEGER NOT NULL PRIMARY KEY,  
    date_ordered DATE,  
    name CHAR(80)  
);  
CREATE TABLE Order_item (  
    order_num    INTEGER NOT NULL,  
    item_num     SMALLINT NOT NULL,  
    PRIMARY KEY (order_num, item_num),  
    FOREIGN KEY (order_num)  
    REFERENCES Orders (order_num)  
)
```

CREATE USER-Anweisung [UltraLite]

Erstellt einen Datenbankbenutzer oder eine Gruppe.

Hinweis

Diese Anweisung wird von UltraLite Java Edition-Datenbanken nicht unterstützt.

Syntax

CREATE USER *user-name* **IDENTIFIED BY** *password*

Parameter

user-name Der Name des Benutzers, den Sie erstellen.

password Das Kennwort für den Benutzer, den Sie erstellen.

Bemerkungen

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

Nebenwirkungen

Keine.

Siehe auch

- „ALTER USER-Anweisung [UltraLite]“ auf Seite 431
- „DROP USER-Anweisung [UltraLite]“ auf Seite 449

Beispiel

Das folgende Beispiel erstellt einen Benutzer namens SQLTester mit dem Kennwort "welcome".

```
CREATE USER SQLTester IDENTIFIED BY welcome
```

DELETE-Anweisung [UltraLite]

Löscht Zeilen aus einer Tabelle in der Datenbank.

Syntax

```
DELETE [ FROM ] table-name[[AS] correlation-name]  
[ WHERE search-condition ]
```

Parameter

correlation-name Ein Bezeichner, der verwendet wird, wenn eine Referenzierung der Tabelle an anderer Stelle in der Anweisung erfolgt.

WHERE-Klausel Wenn eine WHERE-Klausel angegeben ist, werden nur die Zeilen gelöscht, die die *search-condition* erfüllen.

Die WHERE-Klausel unterstützt keine nicht-deterministischen Funktionen wie z.B. RAND oder Variablen. Diese Klausel beschränkt auch keine Spalten. Spalten müssen möglicherweise eine andere Tabelle referenzieren, wenn sie in einer Unterabfrage verwendet werden.

Bemerkungen

Die Art und Weise, in der UltraLite den Zeilenzustand protokolliert, ist eindeutig. Stellen Sie sicher, dass Sie die Auswirkungen von Löschungen und dem Zeilenzustand verstehen.

Siehe auch

- „Suchbedingungen in UltraLite“ auf Seite 283
- „Verwaltung des Zeilenstatus in einer UltraLite-Datenbank“ auf Seite 485
- „START SYNCHRONIZATION DELETE-Anweisung [UltraLite]“ auf Seite 460
- „STOP SYNCHRONIZATION DELETE-Anweisung [UltraLite]“ auf Seite 461

Beispiel

Die folgende Anweisung entfernt Mitarbeiter 105 aus der Employees-Tabelle.

```
DELETE
FROM Employees
WHERE EmployeeID = 105
```

Die folgende Anweisung entfernt alle Daten vor dem Jahr 2000 aus der Tabelle FinancialData.

```
DELETE
FROM FinancialData
WHERE Year < 2000
```

DROP INDEX-Anweisung [UltraLite]

Löscht einen Index.

Syntax

```
DROP INDEX[ IF EXISTS ] [ table-name.]index-name
```

Bemerkungen

Sie können den Primärindex einer Tabelle nicht löschen.

UltraLite verarbeitet keine Anforderungen oder Abfragen, die die Tabelle referenzieren, während die Anweisung DROP INDEX ausgeführt wird. Außerdem ist es nicht möglich, DROP INDEX auszuführen, wenn die Datenbank aktive Abfragen oder nicht festgeschriebene Transaktionen enthält.

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP INDEX-Anweisung versucht, einen Index zu entfernen, der nicht existiert.

Wenn Sie die IF EXISTS-Klausel angeben und die benannte Tabelle nicht gefunden werden kann, wird ein Fehler zurückgegeben.

Für UltraLite.NET-Benutzer: Sie können diese Anweisung nur ausführen, wenn Sie auch die Dispose-Methode für alle Datenobjekte ausführen (z.B. ULDataReader). Siehe [ULBulkCopy.Dispose-Methode \[UltraLite.NET\]](#) [[UltraLite - .NET-Programmierung](#)].

Anweisungen werden nicht freigegeben, wenn gleichzeitig Schemaänderungen initiiert werden. Siehe [UltraLite-Datenbankschemas auf Seite 49](#).

Siehe auch

- „CREATE INDEX-Anweisung [UltraLite]“ auf Seite 433
- „UltraLite-Indizes“ auf Seite 56

Beispiel

Die folgende Anweisung löscht einen fiktiven Index, fin_codes_idx, in der Tabelle FinancialData:

```
DROP INDEX FinancialData.fin_codes_idx
```

DROP PUBLICATION-Anweisung [UltraLite]

Löscht Publikationen.

Syntax

```
DROP PUBLICATION[ IF EXISTS ] publication-name, ...
```

Bemerkungen

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP PUBLICATION-Anweisung versucht, eine Publikation zu entfernen, die nicht existiert.

Siehe auch

- „UltraLite-Clientsynchronisationsplanung“ auf Seite 74
- „ALTER PUBLICATION-Anweisung [UltraLite]“ auf Seite 424
- „CREATE PUBLICATION-Anweisung [UltraLite]“ auf Seite 435

Beispiel

Mit der folgenden Anweisung wird die Publikation pub_contact gelöscht.

```
DROP PUBLICATION pub_contact
```

DROP SYNCHRONIZATION PROFILE-Anweisung [UltraLite]

Löscht ein Synchronisationsprofil.

Syntax

DROP SYNCHRONIZATION PROFILE [IF EXISTS] *sync-profile-name*

Parameter

- **sync-profile-name** Der Name des Synchronisationsprofils.

Bemerkungen

Synchronisationsprofile definieren, wie eine UltraLite- oder UltraLite Java Edition-Datenbank mit dem MobiLink-Server synchronisiert wird.

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP SYNCHRONIZATION PROFILE-Anweisung versucht, ein Synchronisationsprofil zu entfernen, das nicht existiert.

Nebenwirkungen

Keine.

Siehe auch

- „CREATE SYNCHRONIZATION PROFILE-Anweisung [UltraLite]“ auf Seite 436
- „ALTER SYNCHRONIZATION PROFILE-Anweisung [UltraLite]“ auf Seite 425
- „SYNCHRONIZE-Anweisung [UltraLite]“ auf Seite 462

Beispiel

Das folgende Beispiel zeigt die Syntax für das Löschen eines Synchronisationsprofils namens Test1.

```
DROP SYNCHRONIZATION PROFILE Test1
```

DROP TABLE-Anweisung [UltraLite]

Entfernt eine Tabelle und alle ihre Daten aus einer Datenbank.

Syntax

DROP TABLE [IF EXISTS] *table-name*

Bemerkungen

Die DROP TABLE-Anweisung löscht die angegebene Tabelle aus der Datenbank. Alle Daten in der Tabelle sowie Indizes und Schlüssel werden ebenfalls entfernt.

UltraLite verarbeitet keine Anforderungen oder Abfragen, die die Tabelle oder ihre Indizes referenzieren, während die Anweisung DROP TABLE ausgeführt wird. Außerdem ist es nicht möglich, DROP TABLE auszuführen, wenn es aktive Abfragen oder nicht festgeschriebene Transaktionen gibt.

Verwenden Sie die IF EXISTS-Klausel, um zu vermeiden, dass ein Fehler zurückgegeben wird, wenn die DROP TABLE-Anweisung versucht, eine Tabelle zu entfernen, die nicht existiert.

Für UltraLite.NET-Benutzer: Sie können diese Anweisung nur ausführen, wenn Sie auch die Dispose-Methode für alle Datenobjekte ausführen (z.B. ULDataReader). Siehe [ULBulkCopy.Dispose-Methode \[UltraLite.NET\]](#) [*UltraLite - .NET-Programmierung*].

Anweisungen werden nicht freigegeben, wenn gleichzeitig Schemaänderungen initiiert werden. Siehe [UltraLite-Datenbankschemas auf Seite 49](#).

Siehe auch

- „ALTER TABLE-Anweisung [UltraLite]“ auf Seite 427
- „CREATE TABLE-Anweisung [UltraLite]“ auf Seite 438

Beispiel

Die folgende Anweisung löscht eine fiktive Tabelle, EmployeeBenefits, aus der Datenbank.

```
DROP TABLE EmployeeBenefits
```

DROP USER-Anweisung [UltraLite]

Löscht einen Benutzer.

Hinweis

Diese Anweisung wird von UltraLite Java Edition-Datenbanken nicht unterstützt.

Syntax

```
DROP USER userid IDENTIFIED BY password
```

Parameter

- **userid** Die Benutzer-ID des Benutzers, den Sie löschen.
- **password** Das Kennwort für den Benutzer

Bemerkungen

Keine.

Nebenwirkungen

Keine.

Siehe auch

- „ALTER USER-Anweisung [UltraLite]“ auf Seite 431
- „CREATE USER-Anweisung [UltraLite]“ auf Seite 444

Beispiel

Das folgende Beispiel löscht den Benutzer SQLTester aus einer Datenbank.

```
DROP USER SQLTester
```

FROM-Klausel [UltraLite]

Mit dieser Klausel geben Sie die Tabellen oder Ansichten an, die in einer SELECT-Anweisung verwendet werden.

Hinweis

Diese Anweisung wird von UltraLite Java Edition-Datenbanken nicht unterstützt.

Syntax

FROM *table-expression*, ...

table-expression :

table-name [[**AS**] *correlation-name*]

| (*select-list*) [**AS**] *derived-table-name* (*column-name*, ...)

| (*table-expression*)

| *table-expression* *join-operator* *table-expression* [**ON** *search-condition*] ...

join-operator :

|
| **INNER JOIN**
| **CROSS JOIN**
| **LEFT OUTER JOIN**
| **JOIN**

Parameter

table-name Eine Basistabelle oder temporäre Tabelle. Tabellen können nicht verschiedenen Benutzern in UltraLite gehören. Wenn Sie Tabellen mit der Benutzer-ID qualifizieren, wird die ID ignoriert.

correlation-name Ein Bezeichner, der verwendet wird, wenn eine Referenzierung der Tabelle an anderer Stelle in der Anweisung erfolgt. Beispiel: In der folgenden Anweisung ist a als der Korrelationsname für die Tabelle Contacts festgelegt, und b ist der Korrelationsname für die Tabelle Customers.

```
SELECT *  
FROM Contacts a, Customers b  
WHERE a.CustomerID=b.ID
```

derived-table-name Eine abgeleitete Tabelle ist eine verschachtelte SELECT-Anweisung in der FROM-Klausel.

Elemente aus der SELECT-Liste der abgeleiteten Tabelle werden vom (optionalen) abgeleiteten Tabellennamen referenziert, der von einem Punkt (.) und dem Spaltennamen gefolgt wird. Sie können den Spaltennamen auch alleine verwenden, wenn er eindeutig ist.

Es ist nicht möglich, abgeleitete Tabellen innerhalb der SELECT-Anweisung zu referenzieren.

join-operator Geben Sie den Typ des Joins an. Wenn Sie ein Komma (,) oder CROSS JOIN angeben, können Sie keine ON-Unterklausel angeben. Wenn Sie JOIN angeben, müssen Sie eine ON-Unterklausel angeben. Bei INNER JOIN und LEFT OUTER JOIN ist die ON-Klausel optional.

Bemerkungen

Wenn keine FROM-Klausel vorhanden ist, müssen die Ausdrücke in der SELECT-Anweisung ein konstanter Ausdruck sein.

Hinweis

Obwohl sich diese Beschreibung auf Tabellen bezieht, gilt sie, sofern nicht anders angegeben, auch für abgeleitete Tabellen.

Die FROM-Klausel erstellt eine Ergebnismenge, die aus allen Spalten aller angegebenen Tabellen besteht. Anfänglich sind alle Zeilenkombinationen in den angegebenen Tabellen in der Ergebnismenge enthalten und die Anzahl der Kombinationen wird im Allgemeinen durch JOIN-Bedingungen bzw. WHERE-Klauseln verringert.

Wenn Sie keinen Join-Typ angeben und stattdessen die Tabellen in einer durch Komma getrennte Liste anführen, wird standardmäßig ein CROSS JOIN verwendet.

Bei INNER-Joins gibt die Beschränkung der Ergebnisse des Joins mit einer ON-Klausel oder WHERE-Klausel gleichwertige Ergebnisse zurück. Bei OUTER-Joins sind die beiden nicht gleichwertig.

Hinweis

UltraLite unterstützt weder KEY JOINS noch NATURAL JOINS.

Siehe auch

- „Unterabfragen in Ausdrücken“ auf Seite 281
- „Joins: Daten aus mehreren Tabellen abrufen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „DELETE-Anweisung [UltraLite]“ auf Seite 445
- „SELECT-Anweisung [UltraLite]“ auf Seite 457
- „UPDATE-Anweisung [UltraLite]“ auf Seite 466

Beispiel

Die folgenden Klauseln sind gültige FROM-Klauseln:

```
...
FROM Employees
...

...
FROM Customers
CROSS JOIN SalesOrders
CROSS JOIN SalesOrderItems
CROSS JOIN Products
...
```

Die folgende Abfrage verwendet eine abgeleitete Tabelle, um die Namen der Kunden in der Tabelle Customers zurückzugeben, die mehr als drei Bestellungen in der Tabelle SalesOrders haben:

```
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
```

INSERT-Anweisung [UltraLite]

Fügt Zeilen in eine Tabelle ein.

Syntax

```
INSERT [ INTO ]
table-name [ ( column-name, ... ) ]
{ VALUES ( expression, ... ) | select-statement }
```

Bemerkungen

Die INSERT-Anweisung kann verwendet werden, um eine einzelne Zeile einzufügen, oder um mehrere Zeilen einer Abfrageresultmenge einzufügen.

Wenn Spalten angegeben sind, werden die Werte einzeln in die angegebenen Spalten eingefügt. Wird keine Liste für die Spaltennamen definiert, werden die Werte in der Reihenfolge in die Tabellenspalten eingefügt, in der sie in der Tabelle erscheinen (dieselbe Reihenfolge wie mit SELECT * abgerufen). Zeilen werden in die Tabelle in einer beliebigen Reihenfolge eingefügt.

In Tabellen eingefügte Zeichenfolgen werden immer in der gleichen Schreibung (groß oder klein) gespeichert, in der sie eingegeben wurden, unabhängig davon, ob die Datenbank die Groß- und Kleinschreibung berücksichtigt.

Siehe auch

- [„SELECT-Anweisung \[UltraLite\]“ auf Seite 457](#)

Beispiel

Die folgende Anweisung fügt der Datenbank die Abteilung Eastern Sales hinzu.

```
INSERT
INTO Departments ( DepartmentID, DepartmentName )
VALUES ( 230, 'Eastern Sales' )
```

Die folgende Anweisung fügt die Werte von a und b aus othertable in 'mytable' ein, wenn der Wert von c in othertable höher als 10 ist.

```
INSERT INTO mytable( col1, col2 ) SELECT a, b FROM othertable WHERE c > 10
```

LOAD TABLE-Anweisung [UltraLite]

Importiert Massendaten aus einer externen Datei in eine Datenbanktabelle.

Syntax

```
LOAD [ INTO ] TABLE [ owner.]table-name
( column-name, ... )
```

FROM *stringfilename*
 [*load-option* ...]

load-option :

CHECK CONSTRAINTS { ON | OFF }
COMPUTES { ON | OFF }
DEFAULTS { ON | OFF }
DELIMITED BY *string*
ENCODING *encoding*
ESCAPES { ON }
FORMAT { ASCII | TEXT }
ORDER { ON | OFF }
QUOTES { ON | OFF }
SKIP *integer*
STRIP { ON | OFF | BOTH }
WITH CHECKPOINT { ON | OFF }

comment-prefix : string

encoding : string

Parameter

- **column-name** Mit dieser Klausel geben Sie eine oder mehrere Spalten an, in die Daten geladen werden sollen. Alle nicht in der Spaltenliste vorhandenen Spalten werden NULL, wenn DEFAULTS auf OFF gesetzt ist. Wenn DEFAULTS auf ON gesetzt ist und die Spalte einen Standardwert hat, wird dieser Wert benutzt. Wenn DEFAULTS auf OFF gesetzt ist und eine nicht nullwertfähige Spalte aus der Spaltenliste ausgelassen wird, versucht der Datenbankserver, die leere Zeichenfolge in den Datentyp der Spalte zu konvertieren.

Wenn eine Spaltenliste angegeben ist, werden die in der Datei erwarteten Spalten sowie die Reihenfolge, in der sie erwartungsgemäß erscheinen werden, aufgelistet. Spaltennamen können nicht wiederholt werden.

- **FROM string-filename** Damit geben Sie die Datei an, aus der die Daten geladen werden sollen. *string-filename* wird an den Datenbankserver als Zeichenfolge übergeben. Die Zeichenfolge unterliegt daher denselben Formatierungsanforderungen für die Datenbank wie auch andere SQL-Zeichenfolgen. Speziell sind folgende Punkte zu beachten:

- Bei Angabe des Verzeichnissuchpfads muss das Backslashzeichen (\) durch zwei Backslashes dargestellt werden. Die Anweisung zum Laden der Daten aus der Datei *c:\temp\input.dat* in die Tabelle Employees lautet wie folgt:

```
LOAD TABLE Employees
FROM 'c:\\temp\\input.dat' ...
```

- Der Pfadname ist relativ zum Datenbankserver, nicht aber zur Clientanwendung.
- Sie können eine UNC-Pfadangabe verwenden, um Daten aus Dateien von anderen Computern als dem Datenbankserver zu laden.
- **load-option** Es gibt mehrere Ladeoptionen, mit denen Sie steuern können, wie Daten geladen werden sollen. Die folgende Liste enthält die unterstützten Ladeoptionen:

- **CHECK CONSTRAINTS-Klausel** Diese Klausel steuert, ob Integritätsregeln während des Ladens überprüft werden. CHECK CONSTRAINTS ist standardmäßig auf ON gesetzt, das Dienstprogramm Unload (ulunload) schreibt aber LOAD TABLE-Anweisungen mit der Option CHECK CONSTRAINTS auf OFF. Wenn Sie CHECK CONSTRAINTS auf OFF setzen, deaktivieren Sie Prüf-Integritätsregeln, was z.B. während des Datenbank-Neuaufbaus nützlich sein kann.
- **COMPUTES-Klausel** Diese Option wird von UltraLite verarbeitet, aber ignoriert.
- **DEFAULTS-Klausel** Standardmäßig hat DEFAULTS die Einstellung OFF. Wenn DEFAULTS auf OFF gesetzt ist, wird allen Spalten, die in der Spaltenliste nicht vorhanden sind, NULL zugeordnet. Wenn DEFAULTS auf OFF gesetzt ist und eine nicht nullwertfähige Spalte aus der Liste ausgelassen wird, versucht der Datenbankserver, die leere Zeichenfolge in den Datentyp der Spalte zu konvertieren. Wenn DEFAULTS auf ON gesetzt ist und die Spalte einen Standardwert hat, wird dieser Wert benutzt.
- **DELIMITED BY-Klausel** Verwenden Sie diese Klausel, um die Spaltentrennzeichenfolge anzugeben. Das Standardtrennzeichen für Spalten ist ein Komma. Sie können jedoch auch eine beliebige Zeichenfolge mit einer Länge von bis zu 255 Byte verwenden (z.B. . . . DELIMITED BY '###' . . .). Es gelten die Formatierungsanforderungen anderer SQL-Zeichenfolgen. Wenn Sie durch Tabulatoren getrennte Werte angeben möchten, könnten Sie z.B. die hexadezimale Escapesequenz für das Tabulatorzeichen (9) verwenden, . . . DELIMITED BY '\x09' . . .
- **ENCODING-Klausel** Diese Klausel gibt die Zeichenkodierung an, die bei den Daten verwendet wird, die in die Datenbank geladen werden. UltraLite nimmt keine Zeichensatz-Konvertierung vor: Die Kodierung der Datendatei muss mit der Datenbank übereinstimmen.
- **ESCAPES-Klausel** ESCAPES ist immer ON, daher werden die Zeichen nach dem Backslashzeichen vom Datenbankserver als Sonderzeichen erkannt und interpretiert. Zeilenumbruch-Zeichen können als Kombination \n eingefügt werden, und andere Zeichen können als hexadezimale ASCII-Codes in die Daten eingefügt werden, wie zum Beispiel als \x09 für das Tabulatorzeichen. Eine Sequenz von zwei Backslashes (\\) wird als ein einzelner Backslash interpretiert. Ein Backslash gefolgt von einem beliebigen Zeichen außer n, x, X oder \ wird als zwei separate Zeichen interpretiert. Zum Beispiel werden mit \q ein Backslash und der Buchstabe q eingefügt.
- **FORMAT-Klausel** Diese Klausel gibt das Format der Datenquelle an, von der Sie Daten laden. Mit TEXT werden Eingabezeilen als Zeichen vorausgesetzt (wie von der ENCODING-Option festgelegt), eine Zeile pro Ausgabezeile, wobei die Werte durch die Spalten-Trennzeichenfolge voneinander getrennt werden. ASCII wird auch unterstützt.
- **QUOTES-Klausel** Diese Klausel gibt an, ob Zeichenfolgen in Anführungszeichen gesetzt werden. UltraLite unterstützt nur ON, daher erwartet die LOAD TABLE-Anweisung, dass Zeichenfolgen in Anführungszeichen gesetzt sind. Das Anführungszeichen ist ein Apostroph (einfaches Anführungszeichen). Das erste Zeichen dieser Art, das in der Eingabedatei gefunden wird, wird als Zeichen für Anführungszeichen für die Zeichenfolge behandelt. Die Zeichenfolgen müssen durch ein jeweils passendes Anführungszeichen abgeschlossen werden.

Spalten-Trennzeichen können in Spaltenwerte aufgenommen werden. Es wird auch vorausgesetzt, dass Apostrophe oder Anführungszeichen nicht Teil des Wertes sind. Daher wird die folgende

Zeile wie zwei Werte behandelt und nicht wie drei, obwohl es ein Komma in der Adresse gibt. Außerdem werden die Anführungszeichen um die Adresse nicht in die Datenbank eingefügt.

```
'123 High Street, Anytown',(715)398-2354
```

Wenn Sie ein Anführungszeichen in einen Wert einbeziehen möchten, müssen Sie zwei Anführungszeichen verwenden. Die folgende Zeile enthält in der dritten Spalte einen Wert, der ein Apostrophzeichen ist:

```
'123 High Street, Anytown','(715)398-2354',''''
```

- **SKIP-Klausel** Mit dieser Klausel geben Sie an, ob Zeilen am Anfang einer Datei ignoriert werden sollen. Das Argument *integer* gibt die Anzahl der zu überspringenden Zeilen an. Sie können diese Klausel verwenden, um beispielsweise Zeilen mit Spaltenüberschriften zu überspringen.
- **STRIP-Klausel** Diese Klausel wird verarbeitet, aber ignoriert. Diese Klausel gibt an, ob bei Werten ohne Anführungszeichen vorangestellte oder nachgestellte Leerzeichen entfernt werden sollen, bevor die Werte eingefügt werden. Die Option STRIP kann mit den folgenden Optionen verwendet werden:
 - **STRIP ON** Führende Leerzeichen werden entfernt.
 - **STRIP OFF** Voran- bzw. nachgestellte Leerzeichen werden nicht entfernt.
 - **STRIP BOTH** Sowohl führende als auch nachgestellte Leerzeichen werden entfernt.
- **WITH CHECKPOINT-Klausel** Mit dieser Klausel geben Sie an, ob ein Checkpoint gesetzt werden soll. Die Standardeinstellung ist OFF. Wenn diese Klausel auf ON gesetzt ist, wird ein Checkpoint gesetzt, nachdem die Anweisung erfolgreich abgeschlossen ist.

Bemerkungen

Diese Anweisung stellt auch Unterstützung für die Behandlung der Ausgabe des SQL Anywhere-Dienstprogramms dbunload (die Datei reload.sql) bereit. LOAD TABLE ist nur bei Verwendung von DBISQL unter Windows verfügbar.

Hinweis

Diese Anweisung wird von UltraLite Java Edition-Datenbanken nicht unterstützt.

Die empfohlene Methode für das Entladen und Neuladen von UltraLite-Datenbanken ist die Verwendung der Dienstprogramme ulunload und ulload. Beachten Sie auch, dass das Dienstprogramm ulinit Schema und Daten direkt aus einer SQL Anywhere-Datenbank laden kann.

LOAD TABLE ermöglicht das effiziente Einfügen großer Datenmengen aus einer Datei in die Datenbanktabelle. Die Anweisung ist hauptsächlich dazu bestimmt, die Ausgabe des SQL Anywhere-Dienstprogramms dbunload (die Datei reload.sql) zu unterstützen.

Bei FORMAT TEXT wird ein NULL-Wert angezeigt, indem gar kein Wert angegeben wird. Beispiel: Wenn drei Werte erwartet werden und die Datei 1 , , 'Fred' , enthält, dann sind die eingefügten Werte 1, NULL und Fred. Wenn die Datei 1 , 2 , enthält, werden die Werte 1, 2 und NULL eingefügt. Werte,

die nur aus Leerstellen bestehen, werden ebenfalls als NULL angesehen. Beispiel: Wenn die Datei 1, , 'Fred' , enthält, dann werden die Werte 1, NULL und Fred eingefügt. Alle anderen Werte werden als Nicht-NULL angesehen. Beispiel: " (Apostroph gefolgt von Apostroph) ist eine leere Zeichenfolge. "NULL" ist eine Zeichenfolge, die vier Buchstaben enthält.

Wenn eine Spalte, die mit LOAD TABLE geladen wird, NULL nicht zulässt und der Dateiwert NULL ist, erhalten numerische Werte den Wert "0" (Null) und Zeichenspalten eine leere Zeichenfolge (") zugeordnet. Wenn eine Spalte, die mit LOAD TABLE geladen wird, NULL zulässt und der Dateiwert NULL ist, dann ist der Spaltenwert NULL (bei allen Typen).

Wenn die Tabelle die Spalten a, b und c enthält und die Eingabedaten a, b und c enthalten, aber die LOAD-Anweisung nur a und b als Spalten angibt, in die Daten geladen werden sollen, werden die folgenden Werte in die Spalte c eingefügt:

- Wenn DEFAULT ON angegeben ist und Spalte c einen Standardwert hat, wird der Standardwert verwendet.
- Wenn Spalte c keinen Standardwert hat und NULL zulässig ist, wird NULL verwendet.
- Wenn Spalte c keinen Standardwert hat und NULL nicht zulässig ist, wird entweder eine Null (0) oder eine leere Zeichenfolge (") verwendet bzw. ein Fehler zurückgegeben, abhängig vom Datentyp der Spalte.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „INSERT-Anweisung [UltraLite]“ auf Seite 452
- „UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload)“ auf Seite 233
- „Dienstprogramm zum Entladen (dbunload)“ [[SQL Anywhere Server - Datenbankadministration](#)]

Beispiel

Hier folgt ein Beispiel für LOAD TABLE. Zuerst wird eine Tabelle erstellt, in die dann Daten geladen werden, wobei eine Datei mit dem Namen *input.txt* verwendet wird.

```
CREATE TABLE t( a CHAR(100) primary key, let_me_default INT DEFAULT 1, c
CHAR(100) )
```

Nachfolgend wird der Inhalt der Datei *input.txt* gezeigt:

```
'this_is_for_column_c', 'this_is_for_column_a', ignore_me
```

Die folgende LOAD-Anweisung lädt die Datei *input.txt*:

```
LOAD TABLE T ( c, a ) FROM 'input.txt' FORMAT TEXT DEFAULTS ON
```

Der Befehl `SELECT * FROM t` ergibt folgende Ergebnismenge:

a	let_me_default	c
this_is_for_column_a	1	this_is_for_column_c

ROLLBACK-Anweisung [UltraLite]

Beendet eine Transaktion und setzt alle Änderungen zurück, die an den Daten seit der letzten Ausführung einer COMMIT- oder ROLLBACK-Anweisung durchgeführt wurden.

Syntax

ROLLBACK [WORK]

Bemerkungen

Bei der Verwendung von UltraLite SQL wird eine Transaktion erstellt. Eine Transaktion besteht aus allen Änderungen (INSERT-, UPDATE- und DELETE-Anweisungen) seit dem letzten ROLLBACK- bzw. COMMIT-Vorgang. Die ROLLBACK-Anweisung beendet die aktuelle Transaktion und setzt alle Änderungen zurück, die seit dem vorhergehenden COMMIT oder ROLLBACK gemacht wurden.

Siehe auch

- „COMMIT-Anweisung [UltraLite]“ auf Seite 432

Beispiel

Die folgende Anweisung setzt die Datenbank in den Zustand zurück, in dem sie bei der vorherigen Festschreibung war.

```
ROLLBACK
```

SELECT-Anweisung [UltraLite]

Ruft Informationen aus der Datenbank ab

Syntax

```
SELECT [ DISTINCT ] [ row-limitation ]
select-list
[ FROM table-expression, ... ]
[ WHERE search-condition ]
[ GROUP BY group-by-expression, ... ]
[ ORDER BY order-by-expression, ... ]
[ FOR { UPDATE | READ ONLY } ]
[ OPTION ( FORCE ORDER ) ]
```

row-limitation :

```
FIRST
| TOP n [ START AT m ]
```

select-list :

```
expression [ [ AS ] alias-name ], ...
```

order-by-expression :
{ *integer* | *expression* } [**ASC** | **DESC**]

Parameter

DISTINCT-Klausel Geben Sie DISTINCT an, um Duplikatzeilen in den Ergebnissen zu eliminieren. Wenn Sie DISTINCT nicht festlegen, werden alle Zeilen zurückgegeben, die die Klauseln der SELECT-Anweisung erfüllen, einschließlich Duplikatzeilen. Da die Ausführung vieler Anweisungen bedeutend länger dauert, wenn DISTINCT angegeben ist, sollten Sie DISTINCT nur dann verwenden, wenn es unbedingt notwendig ist.

row-limitation Verwenden Sie Zeilenbeschränkungen, um eine Teilmenge des Ergebnisses zurückzugeben. Beispiel: Geben Sie FIRST an, um die erste Zeile einer Ergebnismenge abzurufen. Verwenden Sie TOP *n*, um die ersten *n* Zeilen der Ergebnisse zurückzugeben. Geben Sie START AT *m* an, um die Position des Ausgangspunkts festzulegen, wenn die TOP *n* Zeilen abgerufen werden. Um diese Zeilen zu sortieren, damit sie sinnvolle Ergebnisse zurückgeben, geben Sie eine ORDER BY-Klausel für die SELECT-Anweisung an.

select-list Eine Liste von Ausdrücken, die angibt, was aus der Datenbank abgerufen werden soll. Gewöhnlich sind Ausdrücke in einer Auswahlliste Spaltennamen. Sie können allerdings auch andere Arten von Ausdrücken sein, wie z.B. Funktionen. Verwenden Sie einen Stern (*), um sämtliche Spalten aller Tabellen auszuwählen, die in der FROM-Klausel aufgelistet sind. Optional können Sie ein Alias für jeden Ausdruck in der *select-list* festlegen. Mit einem Alias können Sie die Ausdrücke in der *select-list* von anderer Stelle aus in der Abfrage referenzieren, wie z.B. innerhalb der WHERE- und ORDER BY-Klauseln.

FROM-Klausel Zeilen werden aus den in *table-expression* angegebenen Tabellen und Ansichten abgerufen. Siehe „[FROM-Klausel \[UltraLite\]](#)“ auf Seite 450.

WHERE-Klausel Wenn eine WHERE-Klausel angegeben ist, werden nur die Zeilen ausgewählt, die die *search-condition* erfüllen. Siehe „[Suchbedingungen in UltraLite](#)“ auf Seite 283.

GROUP BY-Klausel Das Ergebnis der Abfrage, die eine GROUP BY-Klausel hat, enthält eine Zeile für jede unterschiedliche Menge von Werten im GROUP BY-Ausdruck. Die sich ergebenden Zeilen werden oftmals auch Gruppen genannt, da es im Ergebnis eine Zeile für jede Gruppe von Zeilen aus der Tabellenliste gibt. Auf die Zeilen in diesen Gruppen können Aggregatfunktionen angewendet werden. NULL wird als eindeutiger Wert betrachtet.

ORDER BY-Klausel Diese Klausel sortiert die Ergebnisse einer Abfrage entsprechend des in der Klausel angegebenen Ausdrucks. Jeder Ausdruck in der ORDER BY-Klausel kann in aufsteigender (ASC) Reihenfolge oder absteigender (DESC) Reihenfolge (Standardwert) sortiert werden. Wenn der Ausdruck eine Ganzzahl *n* ist, werden die Abfrageergebnisse nach dem *n*-ten Ausdruck in der Auswahlliste sortiert.

Die einzige Möglichkeit, sicherzustellen, dass Zeilen in einer bestimmten Reihenfolge zurückgegeben werden, ist die Verwendung von ORDER BY. Ohne die Klausel ORDER BY gibt UltraLite die Zeilen in der jeweils effizientesten Reihenfolge zurück.

UltraLite unterstützt das Sortieren von LONG VARCHAR- oder LONG BINARY-Werten nicht.

FOR-Klausel Diese Klausel hat zwei Variationen, die das Abfrageverhalten steuern:

- **FOR READ ONLY** Diese Klausel gibt an, dass die Abfrage nicht für Updates verwendet wird. Sie sollten diese Klausel möglichst angeben, da UltraLite in einigen Fällen bessere Ergebnisse erzielen kann, wenn bekannt ist, dass eine Abfrage nicht für Aktualisierungen verwendet wird. UltraLite könnte z.B. einen direkten Table-Scan ausführen, wenn es erfährt, dass ein schreibgeschützter Zugriff erforderlich ist. FOR READ ONLY ist das Standardverhalten. Siehe „[Direkte Page-Scans](#)“ auf Seite 479.
- **FOR UPDATE** Diese Klausel ermöglicht die Verwendung der Abfrage für Aktualisierungen. Diese Klausel muss explizit angegeben werden, weil sonst Aktualisierungen nicht zulässig sind (FOR READ ONLY ist das Standardverhalten).

OPTION (FORCE ORDER)-Klausel Diese Klausel wird nicht für den allgemeinen Gebrauch empfohlen. Sie ersetzt die UltraLite-Reihenfolge für den Tabellenzugriff und legt fest, dass UltraLite auf die Tabellen in der Reihenfolge zugreift, in der sie in der Abfrage aufgeführt sind. Verwenden Sie diese Klausel nur, wenn die Abfragereihenfolge effizienter ist als die UltraLite-Reihenfolge.

UltraLite kann auch Ausführungspläne zur Optimierung von Abfragen verwenden. Siehe „[Ausführungspläne in UltraLite](#)“ auf Seite 477.

Bemerkungen

Vergewissern Sie sich immer, dass die Abfrage geschlossen wird. Andernfalls kann kein Speicher freigegeben werden und die Anzahl der temporären Tabellen kann unnötig zunehmen.

Siehe auch

- „[UltraLite Performance-Tipps](#)“ auf Seite 469
- „[SELECT-Anweisung](#)“ [[SQL Anywhere Server - SQL-Referenzhandbuch](#)]
- „[Abfragen](#)“ [[SQL Anywhere Server - SQL-Benutzerhandbuch](#)]

Beispiel

Die folgende Anweisung stellt die Anzahl der Mitarbeiter in der Tabelle Employees fest.

```
SELECT COUNT(*)
FROM Employees
```

Die folgende Anweisung wählt 10 Zeilen aus der Tabelle Employees aus. Der Beginn ist bei der 40. Zeile und das Ende bei der 49. Zeile.

```
SELECT TOP 10 START AT 40 * FROM Employees
```

SET OPTION-Anweisung [UltraLite]

Ändert die Werte von Datenbankoptionen.

Syntax

```
SET OPTION option-name=[option-value]
```

option-name: identifier

option-value: string, identifier, or number

Bemerkungen

Mit dieser Anweisung können Sie Optionen bei einer UltraLite-Datenbank einstellen. Die meisten UltraLite-Optionen werden festgelegt, wenn Sie die Datenbank erstellen, und können später nicht mehr geändert werden.

Es ist nicht möglich festzulegen, ob eine Option dauerhaft ist. UltraLite legt fest, ob es sich um eine dauerhafte oder temporäre Option handelt. Dauerhafte Optionen werden in der Datenbank gespeichert. Temporäre Optionen werden nur verwendet, bis die Verbindung oder die Datenbank gestoppt wird.

UltraLite führt eine Festschreibung durch, wenn dauerhafte Optionen festgelegt werden: **global_database_id** und **ml_remote_id**. UltraLite führt keine Festschreibung für temporäre oder verbindungsbasierte Optionen durch.

Die einzige Datenbankoption, die zurückgesetzt werden kann, ist **ml_remote_id**. Beispiel:

```
SET OPTION ml_remote_id=
```

Das Ergebnis ist, dass die ID auf NULL gesetzt wird. Wenn dies der Fall ist, generiert UltraLite automatisch einen neuen Wert bei der nächsten Synchronisation.

Siehe auch

- „UltraLite-Option global_database_id“ auf Seite 198
- „UltraLite ml_remote_id-Option“ auf Seite 199
- „UltraLite-Datenbankoptionen“ auf Seite 195
- „DB_PROPERTY-Funktion [System]“ auf Seite 351

Beispiel

Die folgenden Anweisung setzt die Option global_database_id auf 100:

```
SET OPTION global_database_id=100
```

START SYNCHRONIZATION DELETE-Anweisung [UltraLite]

Startet die Protokollierung von gelöschten Zeilen für die MobiLink-Synchronisation neu.

Syntax

```
START SYNCHRONIZATION DELETE
```

Bemerkungen

UltraLite-Datenbanken protokollieren automatisch alle Änderungen an Zeilen, die synchronisiert werden müssen. UltraLite überträgt bei der nächsten Synchronisation diese Änderungen in die konsolidierte Datenbank. Mit dieser Anweisung können Sie die Protokollierung von gelöschten Zeilen neu starten, die davor durch eine Anweisung STOP SYNCHRONIZATION DELETE gestoppt wurde.

Bei der Ausführung einer STOP SYNCHRONIZATION DELETE-Anweisung wird kein Löschvorgang synchronisiert, der über diese Verbindung ausgeführt wird. Die Wirkung hält so lange an, bis eine START SYNCHRONIZATION DELETE-Anweisung ausgeführt wird.

Verwenden Sie START SYNCHRONIZATION DELETE nicht, wenn Ihre Anwendung keine Daten synchronisiert.

Die Art und Weise, in der UltraLite den Zeilenzustand protokolliert, ist eindeutig. Stellen Sie sicher, dass Sie die Auswirkungen von Löschungen und dem Zeilenzustand verstehen.

Siehe auch

- „Verwaltung des Zeilenstatus in einer UltraLite-Datenbank“ auf Seite 485
- „STOP SYNCHRONIZATION DELETE-Anweisung [UltraLite]“ auf Seite 461

Beispiel

Mit der folgenden Sequenz von SQL-Anweisungen wird veranschaulicht, wie START SYNCHRONIZATION DELETE und STOP SYNCHRONIZATION DELETE eingesetzt werden.

```
STOP SYNCHRONIZATION DELETE;  
DELETE FROM PROPOSAL  
  WHERE last_modified < months( CURRENT TIMESTAMP, -1 );  
START SYNCHRONIZATION DELETE;  
COMMIT;
```

STOP SYNCHRONIZATION DELETE-Anweisung [UltraLite]

Stoppt die Protokollierung von gelöschten Zeilen für die MobiLink-Synchronisation.

Syntax

STOP SYNCHRONIZATION DELETE

Bemerkungen

UltraLite-Datenbanken protokollieren automatisch alle Änderungen an Zeilen, die synchronisiert werden müssen. UltraLite überträgt bei der nächsten Synchronisation diese Änderungen in die konsolidierte Datenbank. Mit dieser Anweisung können Sie die Protokollierung von gelöschten Zeilen stoppen, die davor durch eine Anweisung START SYNCHRONIZATION DELETE gestartet wurde. Dieser Befehl kann hilfreich sein, wenn Sie Korrekturen in einer entfernten Datenbank vornehmen. Er sollte aber mit Vorsicht verwendet werden, da er die MobiLink-Synchronisation auf effiziente Weise deaktiviert. Sie sollten die Lösungsprotokollierung nur temporär stoppen.

Bei der Ausführung einer STOP SYNCHRONIZATION DELETE-Anweisung wird kein weiterer Löschvorgang synchronisiert, der über diese Verbindung ausgeführt wird. Die Wirkung hält so lange an, bis eine START SYNCHRONIZATION DELETE-Anweisung ausgeführt wird.

Verwenden Sie STOP SYNCHRONIZATION DELETE nicht, wenn Ihre Anwendung keine Daten synchronisiert.

Die Art und Weise, in der UltraLite den Zeilenzustand protokolliert, ist eindeutig. Stellen Sie sicher, dass Sie die Auswirkungen von Löschungen und dem Zeilenzustand verstehen.

Siehe auch

- „Verwaltung des Zeilenstatus in einer UltraLite-Datenbank“ auf Seite 485
- „START SYNCHRONIZATION DELETE-Anweisung [UltraLite]“ auf Seite 460

Beispiel

Mit der folgenden Sequenz von SQL-Anweisungen wird veranschaulicht, wie START SYNCHRONIZATION DELETE und STOP SYNCHRONIZATION DELETE eingesetzt werden.

```
STOP SYNCHRONIZATION DELETE;  
DELETE FROM PROPOSAL  
WHERE last_modified < months( CURRENT TIMESTAMP, -1 );  
START SYNCHRONIZATION DELETE;  
COMMIT;
```

SYNCHRONIZE-Anweisung [UltraLite]

Synchronisiert eine UltraLite- oder UltraLite Java Edition-Datenbank über den MobiLink-Server.

Syntax

```
SYNCHRONIZE {  
  PROFILE sync-profile-name [ MERGE sync-option [ ;... ] ]  
  | USING sync-option [ ;... ]  
}
```

sync-option :
sync-option-name = *sync-option-value*

sync-option-name : *string*

sync-option-value : *string*

Parameter

- **sync-profile-name** Der Name des Synchronisationsprofils.
- **MERGE-Klausel** Verwenden Sie diese Klausel, wenn Sie Optionen hinzufügen oder aufheben wollen, die im Synchronisationsprofil bereitgestellt werden.
- **USING-Klausel** Verwenden Sie diese Klausel, wenn Sie Synchronisationsoptionen ohne Referenzierung eines Synchronisationsprofils angeben wollen.
- **sync-option** Eine Zeichenfolge mit mindestens einem Paar "Option=Wert", getrennt durch Semikola. Beispiel: 'option1=value1;option2=value2'.
- **sync-option-name** Der Name der Synchronisationsoption
- **sync-option-value** Der Wert der Synchronisationsoption.

Bemerkungen

Die Synchronisation wird entsprechend der Parameter im Synchronisationsprofil konfiguriert, oder die Parameter können in der Anweisung selbst angegeben werden.

Indem sie das Zusammenführen von Synchronisationsoptionen zulassen, können Entwickler es vermeiden, einige Optionen (z.B. MobiLinkPwd) in der Datenbank zu speichern.

Wenn die Callback-Funktion einer Synchronisation in UltraLite definiert und registriert wird, wird bei jedem Ausführen einer SYNCHRONIZE-Anweisung der Verarbeitungsfortschritt dieser Synchronisation an die Callback-Funktion übergeben. Wenn kein Callback registriert wurde, werden die Informationen über den Verarbeitungsfortschritt unterdrückt.

Nebenwirkungen

Keine.

Siehe auch

- „CREATE SYNCHRONIZATION PROFILE-Anweisung [UltraLite]“ auf Seite 436
- „ALTER SYNCHRONIZATION PROFILE-Anweisung [UltraLite]“ auf Seite 425
- „DROP SYNCHRONIZATION PROFILE-Anweisung [UltraLite]“ auf Seite 448
- ULSetSynchronizationCallback-Methode [UltraLite Embedded SQL] [*UltraLite - C- und C++-Programmierung*]

Beispiel

Das folgende Beispiel zeigt die Syntax für die Synchronisation eines Synchronisationsprofils namens Test1, wobei MobiLinkPwd nicht als Teil des Profils gespeichert wurde:

```
SYNCHRONIZE PROFILE Test1 MERGE 'MobiLinkPwd=sql'
```

Das folgende Beispiel zeigt die erforderliche Syntax, um die Publikation und UploadOnly-Optionen einem Synchronisationsprofil namens Test1 hinzuzufügen.

```
SYNCHRONIZE PROFILE Test1  
MERGE 'publications=p2;uploadonly=on'
```

Das folgende Beispiel zeigt, wie Sie USING verwenden.

```
SYNCHRONIZE USING  
'MobiLinkUid=joe;MobiLinkPwd=sql;ScriptVersion=1;Stream=TCPIP{host=localhost}'
```

Das folgende Beispiel zeigt die Syntax für die Synchronisation der Publikation und UploadOnly-Optionen.

```
SYNCHRONIZE  
USING 'publications=p2;uploadonly=on'
```

TRUNCATE TABLE-Anweisung [UltraLite]

Löscht alle Zeilen aus einer Tabelle, ohne die Tabelle zu löschen

Syntax

```
TRUNCATE TABLE table-name
```

Bemerkungen

Die Anweisung TRUNCATE TABLE löscht alle Zeilen aus einer Tabelle und der MobiLink-Server wird bei der nachfolgenden Synchronisation nicht über ihre Löschung informiert. Dies entspricht der Ausführung der folgenden Anweisungen:

```
STOP SYNCHRONIZATION DELETE;  
DELETE FROM TABLE;  
START SYNCHRONIZATION DELETE;
```

Hinweis

Diese Anweisung muss für eine Datenbank, die an einer Synchronisation oder Replikation beteiligt ist, mit großer Umsicht verwendet werden. Da der MobiLink-Server nicht informiert wird, kann dieser Löschvorgang zu Inkonsistenzen führen, die bewirken können, dass die Synchronisation oder Replikation fehlschlägt.

Nach einer TRUNCATE TABLE-Anweisung bleiben die Tabellenstruktur, alle Indizes und die Integritätsregeln und Spaltendefinitionen weiter bestehen. Nur Daten werden gelöscht.

TRUNCATE TABLE kann nicht ausgeführt werden, wenn eine Anweisung, die die Tabelle betrifft, bereits von einer anderen Anforderung oder Abfrage referenziert wurde. UltraLite verarbeitet auch keine Anforderungen, die eine Tabelle referenzieren, wenn diese Tabelle gerade geändert wird. Außerdem ist es nicht möglich, TRUNCATE TABLE auszuführen, wenn die Datenbank aktive Abfragen oder nicht festgeschriebene Transaktionen enthält.

Für UltraLite.NET-Benutzer: Sie können diese Anweisung nur ausführen, wenn Sie auch die Dispose-Methode für alle Datenobjekte ausführen (z.B. ULDataReader). Siehe [ULBulkCopy.Dispose-Methode \[UltraLite.NET\]](#) [*UltraLite - .NET-Programmierung*].

Schemaänderungen Anweisungen werden nicht freigegeben, wenn gleichzeitig Schemaänderungen initiiert werden.

Nebenwirkungen

Wenn die Tabelle eine als DEFAULT AUTOINCREMENT oder DEFAULT GLOBAL AUTOINCREMENT definierte Spalte enthält, setzt TRUNCATE TABLE den nächsten verfügbaren Wert für die Spalte zurück.

Sobald die Zeilen mit TRUNCATE TABLE als gelöscht markiert sind, sind sie für den Benutzer, der diese Aktion durchgeführt hat, nicht mehr zugreifbar, außer der Benutzer gibt eine Anweisung ROLLBACK aus. Sie bleiben jedoch für andere Verbindungen zugreifbar. Verwenden Sie COMMIT, um die Löschungen dauerhaft zu machen, wodurch von keiner Verbindung mehr auf die Daten zugegriffen werden kann.

Wenn Sie die gekürzte Tabelle synchronisieren, haben alle INSERT-Anweisungen, die auf die Tabelle angewendet werden, Vorrang vor der TRUNCATE TABLE-Anweisung.

Siehe auch

- [UltraLite-Datenbankschemas auf Seite 49](#)
- [„DELETE-Anweisung \[UltraLite\]“ auf Seite 445](#)
- [„START SYNCHRONIZATION DELETE-Anweisung \[UltraLite\]“ auf Seite 460](#)
- [„STOP SYNCHRONIZATION DELETE-Anweisung \[UltraLite\]“ auf Seite 461](#)

Beispiel

Die folgende Anweisung löscht alle Zeilen aus der Tabelle Departments.

```
TRUNCATE TABLE Departments
```

Wenn Sie dieses Beispiel ausführen, achten Sie darauf, dass Sie anschließend eine ROLLBACK-Anweisung ausführen, um Ihre Änderungen zurückzusetzen.

UNION-Anweisung [UltraLite]

Kombiniert die Ergebnisse von zwei oder mehr SELECT-Anweisungen.

Hinweis

Diese Anweisung wird von UltraLite Java Edition-Datenbanken nicht unterstützt.

Syntax

```
select-statement-without-ordering
[ UNION [ ALL | DISTINCT ] select-statement-without-ordering ]...
[ ORDER BY [ number [ ASC | DESC ], ... ]
```

Bemerkungen

Die Ergebnisse mehrerer SELECT-Anweisungen können mit UNION zu größeren Ergebnismengen kombiniert werden. Jede SELECT-Anweisung muss dieselbe Anzahl von Ausdrücken in ihren jeweiligen Auswahllisten haben und darf keine ORDER BY-Klausel enthalten.

Die Ergebnisse von UNION ALL sind die kombinierten Ergebnisse der verbundenen SELECT-Anweisungen. Geben Sie UNION oder UNION DISTINCT an, um Ergebnisse ohne Duplikatzellen zu erhalten. Das Entfernen von Duplikatzellen erhöht jedoch die Gesamt-Ausführungszeit der Anweisung. Geben Sie UNION ALL an, um Duplikatzellen zuzulassen.

Wenn versucht wird, entsprechende Ausdrücke zu kombinieren, die unterschiedliche Datentypen haben, versucht UltraLite, einen Datentyp zu finden, in dem die kombinierten Werte dargestellt werden können. Wenn dies nicht möglich ist, schlägt der Vereinigungsvorgang fehl und ein Fehler wird ausgegeben (z.B. "Umwandeln von 'Nachname' auf numerisch nicht möglich").

Die in den Ergebnissen angezeigten Spaltennamen sind die Spaltennamen (oder Aliase), die bei der ersten SELECT-Anweisung verwendet wurden.

ORDER BY für UNION ist auf Ganzzahlen beschränkt. Die ORDER BY-Klausel verwendet Ganzzahlen, um die Reihenfolge festzulegen, wobei die Ganzzahl den Abfrageausdruck angibt, nach dem die Ergebnisse sortiert werden sollen.

Siehe auch

- „[SELECT-Anweisung \[UltraLite\]](#)“ auf Seite 457

Beispiel

Das folgende Beispiel listet alle unterschiedlichen Zunamen auf, die in den kombinierten Tabellen Employees und Customers gefunden werden.

```
SELECT Surname FROM Employees
UNION
SELECT Surname FROM Customers
```

UPDATE-Anweisung [UltraLite]

Ändert Zeilen in einer Tabelle:

Syntax

```
UPDATE table-name[[AS] correlation-name]
SET column-name = expression, ...
[ WHERE search-condition ]
```

Parameter

table-name *table-name* gibt den Namen der Tabelle an, die aktualisiert werden soll. Nur eine einzige Tabelle ist zulässig.

correlation-name Ein Bezeichner, der verwendet wird, wenn eine Referenzierung der Tabelle an anderer Stelle in der Anweisung erfolgt.

SET-Klausel Jede benannte Spalte wird auf den Wert des Ausdrucks auf der rechten Seite des Gleichheitszeichens gesetzt. Es gibt keine Einschränkungen für den Ausdruck. Wenn der Ausdruck ein *column-name* ist, wird der alte Wert verwendet.

Nur der Wert der in der SET-Klausel angegebenen Spalten kann geändert werden. Es ist nicht möglich, den Wert einer Spalte mit UPDATE auf seinen Standardwert zu setzen.

WHERE-Klausel Wenn eine WHERE-Klausel angegeben ist, werden nur die Zeilen aktualisiert, die die *search-condition* erfüllen.

Bemerkungen

Mit der UPDATE-Anweisung werden die Werte in einer Tabelle geändert.

In Tabellen eingefügte Zeichenfolgen werden immer in der gleichen Schreibung (groß oder klein) gespeichert, in der sie eingegeben wurden, unabhängig davon, ob die Datenbank die Groß- und Kleinschreibung berücksichtigt.

Siehe auch

- „INSERT-Anweisung [UltraLite]“ auf Seite 452
- „DELETE-Anweisung [UltraLite]“ auf Seite 445
- „Suchbedingungen in UltraLite“ auf Seite 283

Beispiel

Das folgende Beispiel transferiert den Mitarbeiter Philip Chin (employee 129) von der Verkaufsabteilung in die Marketingabteilung (department 400).

```
UPDATE Employees
SET DepartmentID = 400
WHERE EmployeeID = 129
```

Ein Beispiel, in dem ein *correlation-name* verwendet wird.

```
UPDATE Employee E
SET salary = salary * 1.05
WHERE EXISTS( SELECT 1 FROM Sales S HAVING E.Sales > Avg( S.sales)
GROUP BY S.dept_no )
```

UltraLite Performance-Tipps

Dieser Abschnitt behandelt Möglichkeiten zur Optimierung verschiedener UltraLite Funktionen, um die Performance von Abfragen, Einfügungen oder Aktualisierungen zu verbessern.

Cachegrößenanpassung für eine UltraLite-Datenbank

Obwohl eine explizite Anpassung der Cachegröße nicht erforderlich ist, kann sie sinnvoll sein, wenn Ihre UltraLite-Datenbankanwendung vom Betriebssystem auf mobilen Geräten gezwungen wird, ihre Speicherbelegung zu reduzieren.

UltraLite-Datenbank Cachegrößen wachsen dynamisch aufgrund der Datentransaktionen und innerhalb des mit Parametern festgelegten verfügbaren Speichers auf dem Gerät. Normalerweise brauchen Sie keine Parameter anzugeben. Wenn Ihre Datenbank groß ist (z. B. 400 MB), kann es sinnvoll sein, den Parameter `CACHE_MAX_SIZE` vom Standardwert auf die maximal zulässige Größe einzustellen. UltraLite weist basierend auf der maximalen Cachegröße Datenstrukturen zu. Der Standardwert ist nicht sehr groß: Sie müssen daher explizit einen hohen Maximalwert einstellen, um diesen zusätzlichen Speicher nutzen zu können. Eine maximale Cachegröße festzulegen, die viel größer ist als die maximale tatsächliche Datenbankdatei, bringt keine Vorteile.

UltraLite verkleinert den Cache nicht automatisch. Die Datenbank-Cachegröße kann nur mit der `cache_allocation`-Datenbankoption in Ihrer Anwendung explizit gesteuert werden. Als Reaktion auf die Meldung des Betriebssystems, dass zu wenig Speicher zur Verfügung steht, passen Sie die Datenbankoption `cache_allocation` an, nachdem Sie sich mit der Datenbank verbunden haben.

Hinweis

Dynamische Cachedimensionierung wird von UltraLite Java Edition-Datenbanken nicht unterstützt. Weitere Hinweise zu Cachegrößen von UltraLite Java Edition-Datenbanken finden Sie unter [„Cachegrößen der UltraLite Java Edition-Datenbank“](#) auf Seite 470.

Beispiele

Das folgende UltraLite C++-Codebeispiel zeigt, wie die maximale Cachegröße durch Aktualisierung der Verbindungszeichenfolge auf 100 MB gesetzt wird:

```
static ul_char const * ConnectionParms =  
    "UID=DBA;PWD=sql;DBF=sample.udb;CACHE_MAX_SIZE=100m";
```

Das folgende UltraLite C++-Code-Beispiel zeigt, wie die Cachezuweisung auf die Hälfte reduziert wird, um die Größe des Cache neu festzulegen:

```
ULConnection * conn = ULDatabaseManager::OpenConnection(ConnectionParms);  
ul_u_long percent;  
percent = conn->GetDatabasePropertyInt( "cache_allocation" );  
conn->SetDatabaseOptionInt( "cache_allocation", percent / 2 );
```

Siehe auch

- „UltraLite-Option `cache_allocation`“ auf Seite 195
- „UltraLite-Verbindungsparameter `CACHE_SIZE`“ auf Seite 172
- „UltraLite-Verbindungsparameter `CACHE_MIN_SIZE`“ auf Seite 171
- „UltraLite-Verbindungsparameter `CACHE_MAX_SIZE`“ auf Seite 170

Cachegrößen der UltraLite Java Edition-Datenbank

UltraLite-Datenbanken werden auf einer Gruppe von Seiten auf allen Plattformen gespeichert. Im Cache wird ein Arbeitssatz von Seiten untergebracht und über ein First-in-First-out-Schema (FIFO) verwaltet. Aktuell verwendete Seiten werden im Cache gesperrt, damit sie nicht ausgelagert werden.

Bei größeren Datenbanken kann die Datenbank jedes Mal, wenn sie geöffnet wird, so konfiguriert werden, dass die Anzahl der Zeilen und Indexseiten, die sich zu einem bestimmten Zeitpunkt im Speicher befinden, begrenzt ist.

Sie können die UltraLite Java Edition-Datenbankperformance mit Benchmarktests grafisch aufzeichnen, um die optimale Cachegröße für Ihre Geschäftslösung zu ermitteln.

Sie können Tests bei verschiedenen Cachegrößen durchführen und nach abrupten Performance-Schwankungen Ausschau halten. Ihr Cache sollte groß genug sein, damit genügend Arbeitsseiten zur Verfügung stehen. Ziehen Sie die folgenden Vorschläge zur Cacheauslastung in Betracht:

- Erstellen Sie mehrere Indizes für die Tabelle und fügen Sie Fremdschlüssel hinzu.
- Fügen Sie Zeilen in zufälliger Reihenfolge ein (nicht wie in der Indexsortierung).
- Erstellen Sie umfangreiche Zeilen, zumindest 25 % der Seitengröße der Datenbank.
- Setzen Sie den Index-Hash auf einen anderen Wert als 0. Diese Vergrößerung erhöht auch die erforderlichen Seitenzugriffe.
- Beginnen Sie die grafische Darstellung der Performance bei der geringsten Cachegröße. Beispiel: 256 KB unter Windows (der kleinste zulässige Cache für diese Plattform) oder 64 KB auf allen anderen Plattformen.
- Verschlüsseln oder verschleiern Sie die Datenbank. Die Verschleierung verwendet weniger Code, verglichen mit der starken Verschlüsselung, und führt eine geringere Anzahl von Berechnungen aus. Die Performance sollte bei einfacher Verschlüsselung nur marginal langsamer als bei gar keiner Verschlüsselung sein. Allerdings sind es Ihre Sicherheitsanforderungen, die letztlich bestimmen, ob Sie die starke Verschlüsselung verwenden.

Siehe auch

- „`Index-Hash-Methode`“ auf Seite 472

Tipps für die Abfrageperformance

Dieser Abschnitt enthält Methoden zur Verbesserung der Performance von Abfragen in UltraLite-Datenbanken.

Index-Scan erstellen und verwalten

Sie können einen oder mehrere Indizes erstellen, um die Performance Ihrer Abfragen zu verbessern oder um abhängig vom Typ des erstellten Indexes sicherzustellen, dass Zeilenwerte eindeutig bleiben.

Ein Index liefert eine Sortierfolge der Zeilen einer Tabelle anhand der Werte in einigen oder allen Spalten. Beim Erstellen von Indizes bestimmt die Reihenfolge, in der Sie die zu indizierenden Spalten auswählen, auch die Reihenfolge, in der die Spalten im Index erscheinen. Indizes können bei strategisch richtiger Verwendung die Performance von Suchvorgängen in Spalten mit Indizes deutlich verbessern.

Verwenden Sie die folgenden empfohlenen Verfahren zur Verbesserung der Abfrageperformance:

- Erstellen Sie einen Index für Spalten, für die Folgendes zutrifft:
 - Sie enthalten Werte, nach denen häufig gesucht wird.
 - Sie werden von der Abfrage verwendet, um Tabellen zu verknüpfen.
 - Sie werden häufig in ORDER BY-, GROUP BY- oder WHERE-Klauseln verwendet.
- Erstellen Sie einen zusammengesetzten Index und stellen Sie sicher, dass die erste Spalte des Indexes am häufigsten vom Prädikat der Abfrage verwendet wird, wenn Sie ihn erstellen.
- Stellen Sie sicher, dass der Overhead der Aktualisierungsverwaltung für einen Index nicht zu hoch für den Speicher des Geräts ist.
- Erstellen und verwalten Sie keine überflüssigen Indizes. Indizes müssen aktualisiert werden, wenn die Daten in einer Spalte geändert werden, sodass alle Einfüge-, Aktualisierungs- und Löschvorgänge auch für die Indizes ausgeführt werden.
- Erstellen Sie einen Index für große Tabellen.
- Erstellen Sie keine redundanten Indizes. Wenn Sie z.B. einen Index für Tabelle T mit den Spalten (x,y) erstellen, können Sie eine Redundanz schaffen, wenn es bereits einen anderen Index für T mit den Spalten (x, y, z) gibt.

Siehe auch

- „UltraLite-Indizes“ auf Seite 56
- „Verwalten von temporären Tabellen“ auf Seite 478
- „Direkte Page-Scans“ auf Seite 479
- „Einen Ausführungsplan anzeigen“ auf Seite 480
- Zusammengesetzte Indizes auf Seite 57
- „EXPLANATION-Funktion [Verschiedene]“ auf Seite 355
- `ULPreparedStatement.GetPlan`-Methode [UltraLite C++] [*UltraLite - C- und C++-Programmierung*]
- `ULCommand.Plan`-Eigenschaft [UltraLite.NET] [*UltraLite - .NET-Programmierung*]
- `PreparedStatement.getPlan`-Methode [UltraLiteJ] [*UltraLite® – Java-Programmierung*]

Index-Hash-Methode

Sie können die Performance Ihrer Abfragen optimieren, indem Sie eine bestimmte Größe für den maximalen **Hash**-Wert wählen. Ein Hash-Schlüssel repräsentiert die tatsächlichen Werte der indizierten Spalte. Mit einem Index-Hash soll der ressourcenintensive Vorgang beim Suchen, Laden und Entpacken von Zeilen zur Ermittlung des indizierten Werts vermieden werden. Er vermeidet diese Vorgänge, indem eine ausreichende Menge der tatsächlichen Zeilendaten mit der Zeilen-ID aufgenommen werden.

Eine Zeilen-ID gestattet es UltraLite, die tatsächlichen Zeilendaten in der Datenbankdatei zu finden. Wenn Sie die Hash-Größe auf 0 setzen (was den Index-Hash deaktiviert), enthält der Indexeintrag nur diese Zeilen-ID. Wenn Sie die Hash-Größe auf einen anderen Wert als 0 setzen, wird auch ein Hash-Schlüssel verwendet. Ein Hash-Schlüssel kann alle umgewandelten Daten in der Zeile oder einen Teil davon enthalten und wird zusammen mit der Zeilen-ID auf der Indexseite gespeichert.

Wie viele Zeilendaten der Hash-Schlüssel enthält, wird wie folgt bestimmt:

- Von der konfigurierten Eigenschaft der maximalen Hash-Größe.
- Von der Anzahl der Zeilendaten, die tatsächlich für den Datentyp der Spalte erforderlich sind.

Ein Hash-Beispiel

Ein Index-Hash-Wert verwaltet die Reihenfolge der tatsächlichen Zeilendaten von indizierten Spalten. Wenn Sie z.B. die Spalte `LastName` für eine Tabelle namens `Employees` indiziert haben, werden möglicherweise vier Namen wie folgt angezeigt:

- Anders
- Anderseck
- Andersen
- Anderson

Wenn Sie die ersten sechs Buchstaben in den Hash-Wert einbeziehen, werden Ihre Hash-Schlüssel für diese Zeilenwerte wie folgt angezeigt:

- Anders

- Anders
- Anders
- Anders

Obwohl diese Einträge gleich aussehen, wird der erste "Anders" in der Liste verwendet, um den tatsächlichen Wert von **Anders** darzustellen. Mit dem letzten "Anders" in der Liste wird hingegen der tatsächliche Zeilenwert **Anderson** dargestellt.

Sehen Sie sich die folgende Anweisung an:

```
SELECT *  
FROM Employees  
WHERE LastName = 'Andersen'
```

Wenn die Employees-Tabelle einen sehr hohen Anteil von Namen ähnlich "Andersen" enthält, ist der Hash-Schlüssel möglicherweise nicht eindeutig genug, um einen Performancevorteil zu erzielen. In diesem Fall kann UltraLite nicht ermitteln, ob einer der Hash-Schlüssel die Bedingungen dieser Anweisung erfüllt. Wenn mehrere gleiche Index-Hash-Schlüssel vorhanden sind, muss UltraLite dennoch Folgendes durchführen:

1. Die Tabellenteile suchen, die der betreffenden Zeilen-ID entspricht.
2. Die Daten entladen und entpacken, damit der Wert ausgewertet werden kann.

Ein Nutzen für die Performance wird nur erzielt, wenn UltraLite eine erhebliche Anzahl von eindeutigen Hash-Schlüsseln erkennen kann, damit sich die Berechnung der Abfragebedingung unmittelbar auf den Index auswirkt. Wenn die Tabelle Employees z.B. Tausende Namen enthält, bietet der Hash-Schlüssel mit sechs Buchstaben immer noch einen gewissen Performancevorteil. Wenn die Employees-Tabelle jedoch nur eine relativ große Anzahl von Namen enthält, die mit Anders* beginnen, sollte der Hash-Wert mindestens sieben Buchstaben umfassen, damit anteilmäßig mehr eindeutige Hash-Schlüssel verwendet werden. Daher werden die ursprünglichen vier Namen am Beginn dieses Beispiels nun mit diesen Hash-Schlüsseln dargestellt:

- Anders
- Anderse
- Anderse
- Anderso

In diesem Beispiel müssten nur zwei der vier Zeilenwerte entpackt und ausgewertet werden, und nicht alle vier.

Siehe auch

- [„Optimale Hash-Größengrenze“ auf Seite 474](#)
- [„Hinzufügen eines UltraLite-Indexes“ auf Seite 59](#)
- [„UltraLite-Erstellungsparameter max_hash_size“ auf Seite 151](#)

Optimale Hash-Größengrenze

Die maximale Hash-Standardgröße von 4 Byte in UltraLite ist für die meisten Deployments geeignet. Sie können die Größe erhöhen, um mehr Daten zusammen mit der Zeilen-ID einzubeziehen. Diese Änderung kann jedoch die Größe des Indexes erhöhen und ihn zwischen mehreren Seiten fragmentieren. Diese Änderung könnte auch die Größe der Datenbank erhöhen. Die Auswirkung einer erhöhten Hash-Größe hängt von der Anzahl der Zeilen in der Tabelle ab. Wenn Sie z.B. nur ein paar Zeilen haben, passt ein großer Index-Hash-Schlüssel immer noch auf die Indexseite. In diesem Fall kommt es nicht zu einer Indexfragmentierung.

Um eine optimale Hash-Größe auszuwählen, sollten Sie den Datentyp, die Zeilendaten und die Datenbankgröße berücksichtigen (v.a. wenn eine Tabelle viele Zeilen enthält).

Die einzige Möglichkeit zu überprüfen, ob Sie eine optimale Hash-Größe gewählt haben, besteht darin, auf dem Zielgerät Benchmark-Tests mit der UltraLite-Clientanwendung auszuführen. Verschiedene Hash-Größen wirken sich auf die Anwendung und die Abfrageperformance aus, zusätzlich zu den Änderungen in der Datenbank selbst.

Die Index-Hash-Methode verbessert Einfügungen, Aktualisierungen, Löschungen und Suchen, wenn die Spalten, die indiziert werden, über eine gute Verteilung von Werten verfügen, wie etwa Zeichenfolgen, die kein gemeinsames Präfix haben, was jedoch zu größeren Indexstrukturen führt. Hash-Indizes ermitteln Zeilen zunächst mit dem Hash und dann mit direktem Zeilenvergleich, um Zeilen mit dem Hash-Wert zu unterscheiden. Wenn die Hash-Größe ausreichend groß ist, identifiziert die Hash-Methode eindeutig eine Zeile, ohne sie zu lesen und zu vergleichen. Wenn die Hash-Größe jedoch zu groß ist und die Seitengröße klein, benötigt der Index möglicherweise zu viele Datenbankseiten.

Datentyp

Um die Hash-Methode auf den gesamten Wert in einer Spalte anzuwenden, müssen Sie die von den verschiedenen Datentypen erforderte Hash-Größe berücksichtigen, die in der folgenden Tabelle aufgelistet ist. UltraLite verwendet die maximale Hash-Größe nur dann, wenn dies erforderlich ist, und überschreitet nie die festgelegte maximale Hash-Größe. UltraLite verwendet eine kleinere Hash-Größe, wenn der Spaltentyp nicht den vollständigen oberen Byte-Grenzwert verwendet.

Datentyp	Verwendete Byte, um die Hash-Methode auf den gesamten Wert anzuwenden
LONG VARCHAR, DOUBLE, FLOAT, REAL, LONG BINARY, ST_GEO- METRY	Kein Hash-Wert
BIT, TINYINT	1
SMALLINT	2
INTEGER, DATE	4

Datentyp	Verwendete Byte, um die Hash-Methode auf den gesamten Wert anzuwenden
BIGINT, DATETIME, TIME, TIMESTAMP, TIMESTAMP WITH TIMEZONE	8
DECIMAL, NUMERIC	Ungefähr die Gesamtstanzahl dividiert durch zwei.
CHAR, VARCHAR	<p>Um die Hash-Methode auf die gesamte Zeichenfolge anzuwenden, muss die maximale Hash-Größe in Byte mit der deklarierten Größe der Spalte übereinstimmen. Multiplizieren Sie in einer UTF-8-kodierten Datenbank immer die deklarierte Größe mit dem Faktor 2, jedoch nur bis zur maximal erlaubten Größe von 32 Byte.</p> <p>Wenn Sie z.B. eine Spalte in einer nicht UTF-8-kodierten Datenbank als VARCHAR(10) deklarieren, ist die erforderliche Größe 10 Byte. Wenn Sie dieselbe Spalte jedoch in einer UTF-8-kodierten Datenbank deklarieren, beträgt die Hash-Größe für die gesamte Zeichenfolge 20 Byte.</p> <p>Bei UltraLite Java Edition-Datenbanken wird für jedes VARCHAR-Zeichen die Hash-Methode auf seine UTF-8-Darstellung, die 1 - 3 Zeichen lang sein kann, angewendet. UltraLite Java Edition verwendet $(3 * n)$ Byte, um die Hash-Methode auf eine VARCHAR (n)-Spalte anzuwenden. Sie können jedoch eine Hash-Größe von n Byte angeben, um die Hash-Methode auf alle Werte anzuwenden, wenn die VARCHAR(n)-Spalte nur ASCII7-Zeichen enthält.</p>
BINARY, VARBINARY	<p>Die maximale Hash-Größe in Byte muss mit der deklarierten Größe der Spalte übereinstimmen.</p> <p>Wenn Sie z.B. eine Spalte als BINARY(30) deklarieren, ist die erforderliche Größe 30 Byte.</p>
UNIQUEIDENTIFIER	16

Wenn Sie beispielsweise die maximale Hash-Größe für einen zweispaltigen zusammengesetzten Index, den Sie als INTEGER bzw. BINARY(20) deklariert haben, auf 6 Byte setzen, geschieht basierend auf den Größenanforderungen des Datentyps Folgendes:

- Der gesamte Wert der Zeile in der INTEGER-Spalte erhält einen Hash-Wert und wird im Index gespeichert, da für INTEGER-Datentypen nur 4 Byte erforderlich sind.
- Nur die ersten 2 Byte der BINARY-Spalte erhalten einen Hash-Wert und werden im Index gespeichert, da die ersten 4 Byte von der INTEGER-Spalte verwendet werden. Wenn die verbleibenden 2 Byte keinen ausreichenden Teil der BINARY-Spalte im Hash-Index speichern, erhöhen Sie die maximale Hash-Größe.

Die Zeilendaten

Die Zeilenwerte der Daten, die in der Datenbank gespeichert werden, beeinflussen ebenfalls die Effizienz eines Hash-Indexes.

Wenn Sie beispielsweise ein gemeinsames Präfix für Einträge einer bestimmten Spalte verwenden, bleibt der Hash wirkungslos, sobald eine Größe gewählt wird, die nur die Präfixe einbezieht. Wählen Sie in diesem Fall eine Größe, die sicherstellt, dass mehr als das gemeinsame Präfix im Hash-Index gespeichert wird. Wenn das gemeinsame Präfix lang ist, empfiehlt es sich möglicherweise, überhaupt keinen Hash für die Werte zu verwenden.

Wenn ein nicht eindeutiger Hash-Index viele doppelte Werte speichert und UltraLite nicht den gesamten Wert im Hash-Index halten kann, verbessert der Hash die Performance wahrscheinlich nicht.

Datenbankgröße

Jede Indexseite hat einen gewissen festen Overhead, doch der Großteil des Platzes einer Seite wird von den tatsächlichen Indexeinträgen verwendet. Eine höhere Hash-Größe bedeutet, dass die einzelnen Indexeinträge größer sind, weshalb weniger Einträge auf eine Seite passen. Bei großen Tabellen verwenden Indizes mit großen Hash-Werten mehr Seiten als Indizes mit kleinen oder gar keinen Hash-Werten. Je mehr Seiten erforderlich sind, desto größer wird die Datenbank und desto geringer wird die Performance. Letzteres geschieht üblicherweise, weil der Cache nur eine bestimmte Anzahl von Seiten enthalten kann, sodass UltraLite-Seiten ein- und ausgelagert werden.

Die folgende Tabelle enthält Annäherungswerte dafür, wie die Hash-Größe beeinflussen kann, wie viele Seiten zum Speichern von Daten in einem Index erforderlich sind:

Tabelle	Seitengröße	Hash-Größe	Anzahl von Einträgen	Erforderliche Seiten
Tabelle A	4 kB	0	1200	3 Seiten
Tabelle B	4 kB	32 Byte	116	3 Seiten
Tabelle C	4 kB	32 Byte	1200 Einträge	11 Seiten

Festlegen der Hash-Größe

Sie können die maximale Hash-Größe auf zwei Weisen festlegen:

- Um einen Datenbank-Standardwert für die maximale Größe zu speichern, können Sie bei der Datenbankerstellung den Erstellungsparameter `max_hash_size` angeben. Wenn Sie standardmäßig keinen Hash-Index verwenden wollen, setzen Sie diesen Wert auf 0. Andernfalls können Sie ihn auf jeden beliebigen Wert bis maximal 32 Byte festlegen oder den UltraLite-Standardwert von 4 Byte beibehalten.
- Überschreibt den Standardwert durch Festlegen einer bestimmten Hash-Größe, wenn Sie einen neuen Index erstellen. Verwenden Sie eine der folgenden Methoden:
 - In Sybase Central legen Sie die Eigenschaft "Maximale Hashgröße" fest, wenn Sie einen neuen Index erstellen.

- In SQL verwenden Sie die Klausel WITH MAX HASH SIZE in der Anweisung CREATE TABLE oder CREATE INDEX.

Siehe auch

- „Hinzufügen eines UltraLite-Indexes“ auf Seite 59
- „UltraLite, SQLDatentypen“ auf Seite 296
- „UltraLite-Erstellungsparameter max_hash_size“ auf Seite 151
- „CREATE INDEX-Anweisung [UltraLite]“ auf Seite 433
- „CREATE TABLE-Anweisung [UltraLite]“ auf Seite 438

Ausführungspläne in UltraLite

UltraLite-Ausführungspläne zeigen an, wie bei der Ausführung einer Abfrage auf Tabellen und Indizes zugegriffen wird. UltraLite verfügt über einen **Abfrageoptimierer**. Der Optimierer ist eine interne Komponente der UltraLite-Laufzeitbibliothek, der versucht, einen effizienten Plan für die Abfrage zu erstellen. Er versucht, die Verwendung von temporären Tabellen für die Speicherung von Zwischenergebnissen zu vermeiden und sicherzustellen, dass nur auf die erforderliche Untermenge einer Tabelle zugegriffen wird, wenn eine Abfrage zwei Tabellen verknüpft.

Den Optimierer außer Kraft setzen

Der Optimierer zielt immer auf den effizientesten Zugriffsplan ab. Dieses Ziel ist jedoch nicht immer garantiert, vor allem bei einer komplexen Abfrage, bei der eine Vielzahl von Möglichkeiten besteht. In extremen Fällen können Sie die gewählte Tabellenreihenfolge außer Kraft setzen, indem Sie einer Abfrage die Klausel `OPTION (FORCE ORDER)` hinzufügen. Sie zwingt UltraLite dazu, auf die Tabellen in der Reihenfolge zuzugreifen, in der sie in der Abfrage aufgeführt sind. *Diese Option wird nicht für den allgemeinen Gebrauch empfohlen.* Wenn die Performance niedrig ist, ist es gewöhnlich vorzuziehen, geeignete Indizes zu erstellen, um die Ausführung zu beschleunigen.

Tip

Wenn Sie Daten nicht mithilfe der Abfrage aktualisieren, sollten Sie die `FOR READ ONLY`-Klausel in Ihrer Abfrage verwenden. Diese Klausel bietet möglicherweise eine bessere Performance. Siehe „[SELECT-Anweisung \[UltraLite\]](#)“ auf Seite 457.

Vom Optimierer verwendete Zugriffsmethode bestimmen

Der UltraLite-Optimierer verwendet leistungsfähige Optimierungsstrategien für die Auswahl eines Indexes für die Abfrageoptimierung. Bei einfachen Abfragen kann nicht leicht vorab ermittelt werden, welchen Index der Optimierer verwendet, um die Abfrageperformance zu optimieren, oder ob überhaupt ein Index verwendet wird. Bei zunehmender Komplexität hängt die Auswahl des Indexes von den Klauseln ab, die für Ihre Abfrage erforderlich sind. Normalerweise kann das Vorhandensein einer `FOR READ ONLY`-Klausel bewirken, dass der Optimierer einen direkten Table-Scan anstatt eines Indexes auswählt, um eine bessere Abfrageperformance zu erzielen.

Beim Optimieren einer Abfrage überprüft der Optimierer anhand der Anforderungen der Abfrage, ob Indizes vorhanden sind, die zur Verbesserung der Performance verwendet werden können. Wenn die

Performance nicht mithilfe eines Indexes verbessert werden kann, durchsucht der Optimierer keinen Index. Stattdessen wird eine temporäre Tabelle oder ein direkter Page-Scan verwendet. Aus diesem Grund müssen Sie möglicherweise mit den Indizes experimentieren und die generierten Ausführungspläne häufig überprüfen, um Folgendes sicherzustellen:

- Sie verwalten nur Indizes, die vom Optimierer verwendet werden.
- Sie minimieren die Anzahl der erstellten temporären Tabellen.

Bei komplexen Abfragen ist es noch schwieriger vorherzusehen, welcher Index verwendet wird. Wenn eine Abfrage z.B. ein WHERE-Prädikat sowie eine GROUP BY-Klausel zusätzlich zu einer ORDER BY-Klausel enthält, erfüllt ein einzelner Index möglicherweise nicht die Suchbedingungen der betreffenden Abfrage. Wenn Sie daher einen Index erstellt haben, um die Selektivitätsanforderungen des WHERE-Prädikats zu erfüllen, stellen Sie möglicherweise fest, dass der Optimierer ihn gar nicht verwendet. Stattdessen verwendet der Optimierer einen Index, der eine bessere Performance für die ORDER BY-Bedingungen bietet, da diese Klausel den größten Verarbeitungsaufwand hat.

Ausführungsplan überprüfen

Sie können den Ausführungsplan programmtechnisch mit dem geeigneten API-Aufruf oder in der Plananzeige in Interactive SQL überprüfen:

- **Wenn kein Index verwendet wird** wird der folgende Ausführungsplan eingeblendet:

```
scan(T)
```

- **Wenn eine temporäre Tabelle verwendet wird** wird der folgende Ausführungsplan eingeblendet:

```
temp [scan(T)]
```

- **Wenn ein Index verwendet wird** ist der Indexname im Ausführungsplan enthalten:

```
scan (T, index_name)
```

Siehe auch

- „Verwalten von temporären Tabellen“ auf Seite 478

Verwalten von temporären Tabellen

Im Allgemeinen versucht der Optimierer immer, die Erstellung von temporären Tabellen für die Rückgabe von Abfrageergebnissen zu vermeiden, da die gesamte temporäre Tabelle aufgefüllt werden muss, bevor die erste Zeile zurückgegeben werden kann. Wenn ein Index existiert, versucht der Optimierer, den Index zuerst zu verwenden, bevor er als letzte Möglichkeit eine temporäre Tabelle erstellt.

Ein Zugriffsplan verwendet eine temporäre Tabelle, um Daten während seiner Ausführung in einer flüchtigen bzw. temporären Arbeitstabelle zu speichern. Diese Tabelle ist nur vorhanden, während der Zugriffsplan ausgeführt wird. Üblicherweise werden temporäre Tabellen verwendet, wenn Zwischenergebnisse zu groß für den verfügbaren Speicher sind, wie zum Beispiel:

- Wenn Unterabfragen im Rahmen des Zugriffsplans früh ausgewertet werden müssen.
- Wenn Daten in einer temporären Tabelle nur für eine einzelne Verbindung gehalten werden.
- Wenn eine Abfrage eine ORDER BY-Klausel für eine Spalte enthält, bei der es sich nicht um einen Index handelt.
- Wenn eine Abfrage eine GROUP BY-Klausel für eine Spalte enthält, bei der es sich nicht um einen Index handelt.

Es ist schwierig vorherzusehen, ob ein Index die Erstellung temporärer Tabellen vermeidet. Sie sollten daher die Pläne für eine Abfrage überprüfen, um sicherzustellen, dass die erstellten Indizes vom UltraLite-Abfrageoptimierer tatsächlich verwendet werden.

Sie können die Verwendung von temporären Tabellen vermeiden, indem Sie für die in der ORDER BY- oder GROUP BY-Klausel verwendeten Spalten einen Index nutzen.

Siehe auch

- „UltraLite-Verbindungsparameter TEMP_DIR“ auf Seite 189
- „Vom Optimierer verwendete Zugriffsmethode bestimmen“ auf Seite 477
- „So lesen Sie Ausführungspläne“ auf Seite 481
- „Gründe für das Anzeigen von Ausführungsplänen“ auf Seite 480

Direkte Page-Scans

UltraLite verwendet direkte Page-Scans als Alternative zu Index-Scans, wenn es effizienter ist, auf Informationen direkt über die Datenbankseite zuzugreifen. Ein direkter Page-Scan wird nur verwendet, nachdem der Optimierer Folgendes bestätigt hat:

- Kein bereits bestehender Index kann die Ergebnisse effizienter zurückgeben.
- Sie verwenden die Abfrage nicht, um Aktualisierungen vorzunehmen. Beispiel: Sie haben die Anweisung als FOR READ ONLY deklariert (die Standardeinstellung, wenn keine FOR-Klausel angegeben wurde) oder die Abfrage so geschrieben, dass es offensichtlich ist, dass keine Daten aktualisiert werden.

Da UltraLite die Zeilen direkt von den Seiten einliest, auf denen die Zeilen gespeichert sind, werden Abfrageergebnisse unsortiert zurückgegeben. Die Reihenfolge von nachfolgenden Abfrageergebnissen ist nicht vorhersehbar. Wenn die Reihenfolge der Zeilen vorhersehbar und deterministisch sein soll, müssen Sie eine ORDER BY-Klausel verwenden, um Ergebnisse in einer konsistenten Reihenfolge zu erhalten. Wenn die Reihenfolge dagegen nicht von Bedeutung ist, können Sie die ORDER BY-Klausel weglassen, um die Abfrageperformance zu verbessern.

Hinweis

Es ist nicht möglich, direkte Page-Scans zu verwenden, wenn Sie eine ULTable-Klasse in einer UltraLite-API verwenden, um Ihre Anwendung zu programmieren.

Sie können überprüfen, wann UltraLite eine Seite direkt durchsucht oder welcher Index für die Rückgabe von Ergebnissen verwendet wurde.

Siehe auch

- „Vom Optimierer verwendete Zugriffsmethode bestimmen“ auf Seite 477
- „Index-Scan erstellen und verwalten“ auf Seite 471

Gründe für das Anzeigen von Ausführungsplänen

Sie können einen Ausführungsplan in Interactive SQL anzeigen, um folgende Informationen zu erhalten:

- Welcher Index für die Rückgabe der Ergebnisse verwendet wird. Ein Index Scan-Objekt enthält den Namen der Tabelle und den Index für die verwendete Tabelle.
- Ob eine temporäre Tabelle verwendet wird, um die Ergebnisse zurückzugeben. Temporäre Tabellen werden in die temporäre Datei von UltraLite geschrieben.
- In welcher Reihenfolge Tabellen verknüpft werden. Anhand dieser Informationen können Sie ermitteln, wie die Performance betroffen ist.
- Warum eine Abfrage langsam ausgeführt wird oder um sicherzustellen, dass eine Abfrage nicht langsam ausgeführt wird.

Siehe auch

- „UltraLite-Verbindungsparameter TEMP_DIR“ auf Seite 189

Einen Ausführungsplan anzeigen

Verwenden Sie Interactive SQL, um einen UltraLite-Plan anzuzeigen, der eine Zusammenfassung darüber enthält, wie eine vorbereitete Anweisung auszuführen ist. Der Textplan wird in der Interactive SQL-Plananzeige angezeigt.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

In UltraLite ist ein Ausführungsplan lediglich eine kurze Textzusammenfassung des Plans. Es werden keine anderen Plantypen unterstützt. Da es sich jedoch um einen kurzen Plan handelt, können Sie Pläne schnell vergleichen, weil die Informationen in einer einzigen Zeile zusammengefasst sind.

Aufgabe

1. Klicken Sie auf **Extras** » Plananzeige.
2. Geben Sie im Fensterausschnitt **SQL** eine Abfrage ein.
3. Klicken Sie auf **Plan abrufen**, um einen Plan für die angegebenen SQL-Anweisungen zu generieren.

Ergebnisse

Der Textplan erscheint im unteren Fensterausschnitt der Plananzeige.

Beispiel

Sehen Sie sich die folgende Anweisung an:

```
SELECT I.inv_no, I.name, T.quantity, T.prod_no
FROM Invoice I, Transactions T
WHERE I.inv_no = T.inv_no
```

Diese Anweisung kann den folgenden Plan ergeben:

```
join[scan(Invoice,primary),index-scan(Transactions,secondary)]
```

Der Plan zeigt an, dass der Join-Vorgang abgeschlossen wurde, indem alle Zeilen aus der Rechnungstabelle gelesen wurden (dem Index namens primary folgend). Dann wird der Index namens secondary aus der Transaktionstabelle verwendet, um nur die Zeilen zu lesen, deren inv_no-Spalten übereinstimmen.

Siehe auch

- „Interactive SQL-Dienstprogramm (dbisql)“ [[SQL Anywhere Server - Datenbankadministration](#)]
- „So lesen Sie Ausführungspläne“ auf Seite 481

So lesen Sie Ausführungspläne

Da die kurzen UltraLite-Pläne Textzusammenfassungen des Zugriffs auf eine Abfrage sind, müssen Sie wissen, wie die Vorgänge eines Join- oder Scan-Vorgangs einer Tabelle implementiert werden.

- **Für Scan-Vorgänge** Scan-Vorgänge werden durch einen einzelnen Operanden dargestellt, der sich auf eine einzelne Tabelle bezieht und einen Index verwendet. Der Tabellename und der Indexname werden in runden Klammern ((,)) im Anschluss an den Vorgangsnamen angezeigt.
- **Für andere Vorgänge** Andere Vorgänge werden durch einen oder mehrere Operanden dargestellt, die auch selbst Pläne sein können. In UltraLite sind diese Operanden durch Kommas getrennte Listen, die zwischen eckigen Klammern ([]) stehen.

Vorgangsliste

In der folgenden Liste sind die Vorgänge aufgeführt, die von UltraLite unterstützt werden.

Vorgang	Beschreibung
count(*)	Zählt die Anzahl der Zeilen in einer Tabelle

Vorgang	Beschreibung
distinct [<i>plan</i>]	Implementiert den DISTINCT-Aspekt einer Abfrage, um doppelte Zeilen zu vergleichen und zu eliminieren. Der Vorgang wird verwendet, wenn der zugrunde liegende Plan Zeilen so sortiert, dass aufeinander folgende doppelte Zeilen eliminiert werden. Wenn zwei aufeinander folgende Zeilen übereinstimmen, wird nur die erste Zeile zur Ergebnismenge hinzugefügt.
dummy	Es wird kein Vorgang ausgeführt. Der Vorgang wird nur in zwei Fällen verwendet: <ul style="list-style-type: none"> • Wenn Sie DUMMY in einer FROM-Klausel angeben. • Wenn die FROM-Klausel in der Abfrage fehlt.
filter [<i>plan</i>]	Führt für jede Zeile, die vom zugrunde liegenden Plan bereitgestellt wird, eine Suchbedingung aus. Nur die als TRUE ausgewerteten Zeilen werden an die Ergebnismenge übergeben.
group-by [<i>plan</i>]	Erstellt ein Aggregat von GROUP BY-Ergebnissen, um mehrere Zeilen von gruppierten Daten zu sortieren. Zeilen werden in der vorgefundenen Reihenfolge aufgelistet und durch den Vergleich der aufeinander folgenden Zeilen gruppiert.
group-single [<i>plan</i>]	Erstellt ein Aggregat von GROUP BY-Ergebnissen, jedoch nur, wenn bekannt ist, dass eine einzelne Zeile zurückgegeben wird.
keyset [<i>plan</i>]	Zeichnet auf, welche Zeilen verwendet werden, um Zeilen in einer temporären Tabelle zu erstellen, sodass UltraLite die Ausgangszeilen aktualisieren kann. Wenn diese Zeilen nicht aktualisiert werden sollen, verwenden Sie die FOR READ ONLY-Klausel in der Abfrage, um diesen Vorgang zu eliminieren.
index-scan (<i>table-name</i> , <i>index-name</i>)	Liest nur einen Teil der Tabelle. Der Index wird verwendet, um die Startzeile zu finden.
join [<i>plan</i> , <i>plan</i>]	Führt einen Inner-Join zwischen zwei Plänen aus
lojoin [<i>plan</i> , <i>plan</i>]	Führt einen Links-Outer-Join zwischen zwei Plänen aus
like-scan (<i>table-name</i> , <i>index-name</i>)	Liest nur einen Teil einer Tabelle. Der Index wird verwendet, um die Startzeile durch Mustervergleich zu finden.

Vorgang	Beschreibung
rowlimit [<i>plan</i>]	Führt den Zeilenbegrenzungsvorgang für übertragene Zeilen aus. Zeilenbegrenzungen werden mit der TOP n- bzw. FIRST-Klausel der SELECT-Anweisung gesetzt.
scan (<i>table-name</i> , <i>index-name</i>)	Liest eine gesamte Tabelle unter Befolgung der Reihenfolge, die durch den Index vorgegeben ist.
sub-query [<i>plan</i>]	Markiert den Start einer Unterabfrage
temp [<i>plan</i>]	<p>Erstellt eine temporäre Tabelle aus den Zeilen im zugrunde liegenden Plan. UltraLite verwendet eine temporäre Tabelle, wenn zugrunde liegende Zeilen sortiert werden müssen und kein Index gefunden wurde, um diese Reihenfolge aufzubauen.</p> <p>Sie können einen Index hinzufügen, um die Erstellung der temporären Tabelle zu vermeiden. Jeder zusätzliche Index erhöht jedoch die Dauer für das Einfügen oder Synchronisieren von Zeilen in der Tabelle, auf die sich der Index bezieht.</p>
union-all [<i>plan</i> , ..., <i>plan</i>]	Führt einen UNION ALL-Vorgang für die Zeilen aus, die vom zugrunde liegenden Plan generiert werden

Performance-Tipps für Einfügen und Aktualisieren

Dieser Abschnitt beschreibt Methoden zur Verbesserung der Performance bei der Ausführung von Einfügungen und Aktualisierungen in UltraLite-Datenbanken.

Transaktions- und Zeilenstatusverwaltung

UltraLite verwaltet Statusinformationen zusammen mit den Daten in der Datenbank. UltraLite protokolliert und speichert Statusinformationen für die Verwaltung folgender Komponenten:

- Gleichzeitige Verbindungen, sodass UltraLite Ressourcen gemeinsam verwenden kann, wenn erforderlich.
- Status des Synchronisationsfortschritts, um sicherzustellen, dass die Synchronisation erfolgreich durchgeführt wird.
- Zeilenzustand, um die Datenintegrität zu gewährleisten. Hierbei wird protokolliert, wie sich Daten zwischen Synchronisationen geändert haben.
- Transaktionen, um festzustellen, wann und wie Daten festgeschrieben werden. In UltraLite wird eine Transaktion entweder vollständig oder überhaupt nicht abgearbeitet.

- Informationen über Wiederherstellung und Sicherung, um Daten vor Betriebssystemabstürzen und Endbenutzeraktionen zu schützen, wie etwa vor dem Entfernen von Speicherkarten oder dem Zurücksetzen des Geräts, während UltraLite läuft.

Siehe auch

- „UltraLite-Parallelität“ auf Seite 484
- „Funktionen des UltraLite-Synchronisationsclients“ auf Seite 2
- „Verwaltung des Zeilenstatus in einer UltraLite-Datenbank“ auf Seite 485
- „UltraLite-Transaktionsverarbeitung“ auf Seite 486
- „Sichern und Wiederherstellen einer UltraLite- und UltraLite Java Edition-Datenbank“ auf Seite 47

UltraLite-Parallelität

UltraLite verwendet automatisch die folgenden Methoden zur Verwaltung paralleler Datenbankzugriffe:

- **Mehrere UltraLite-Datenbankzugriffe** Eine einzelne Anwendung kann Verbindungen zu mehreren Datenbanken öffnen. UltraLite Java Edition bietet keine Unterstützung für den gleichzeitigen Zugriff.
- **Mehrere Anwendungen** Eine UltraLite- oder UltraLite Java Edition-Datenbank kann jeweils nur durch einen einzelnen Prozess geöffnet werden.
- **Mehrere Threads** UltraLite unterstützt Anwendungen mit mehreren Threads. Eine einzelne Anwendung kann so geschrieben werden, dass sie mehrere Threads verwendet, von denen jeder eine Verbindung zu einer Datenbank oder auch zu unterschiedlichen Datenbanken herstellen kann.
- **Mehrere Transaktionen bzw. Anforderungen** Über jede Verbindung kann jeweils eine einzelne Transaktion aktiv sein. Transaktionen können aus einer einzelnen Anforderung oder aus mehreren Anforderungen bestehen. Datenänderungen während einer Transaktion sind in der Datenbank erst dann dauerhaft, wenn die Transaktion festgeschrieben wurde. Entweder werden alle in einer Transaktion durchgeführten Datenänderungen festgeschrieben oder alle werden zurückgesetzt.
- **Synchronisation** Während der Upload- und Download-Phase ist der Lese- und Schreibzugriff auf die Datenbank gestattet. Wenn eine Anwendung jedoch eine Zeile ändert, die anschließend auch vom Download geändert werden soll, schlägt der Download fehl und wird zurückgesetzt. Verwenden Sie den Synchronisationsparameter `DisableConcurrency`, um den Zugriff auf Daten während der Synchronisation zu deaktivieren.

Wenn die Synchronisation fehlschlägt, unterstützt UltraLite wieder aufnehmbare Downloads auf allen Plattformen.

Siehe auch

- „Fehlgeschlagene Downloads“ [*MobiLink - Serveradministration*]
- „Einschränkungen für UltraLite- und UltraLite Java Edition-Datenbanken“ auf Seite 9
- „UltraLite-Transaktionsverarbeitung“ auf Seite 486
- „UltraLite-Clients“ auf Seite 69
- „Synchronisationsparameter Additional Parameters“ auf Seite 91

Verwaltung des Zeilenstatus in einer UltraLite-Datenbank

Die Verwaltung von Zeilenstatusinformationen ist eine leistungsfähige Funktion von UltraLite-Datenbanken. Die Protokollierung des Tabellen- und Zeilenzustands ist für die Datensynchronisation besonders wichtig.

Hinweis

UltraLite Java Edition unterstützt die Zeilenstatusverwaltung nicht. UltraLite Java Edition-Datenbanken verwenden Transaktionslogs, um Änderungen zu protokollieren, die synchronisiert werden müssen.

Um den Zeilenzustand in einer UltraLite-Datenbank zu protokollieren, wird eine interne Markierung verwendet. Zeilenzustände steuern Transaktionsverarbeitung, Wiederherstellung und Synchronisation. Wenn eine Anwendung eine Zeile einfügt, aktualisiert oder löscht, ändert UltraLite den Zustand der Zeile, um den Vorgang und die Verbindung, die den betreffenden Vorgang ausgeführt hat, widerzuspiegeln. Wenn eine Transaktion festgeschrieben wird, werden die Zustände aller Zeilen, die von der Transaktion betroffen sind, geändert, um die Festschreibung anzuzeigen. Wenn während des Festschreibens ein unerwarteter Fehler eintritt, wird die gesamte Transaktion zurückgesetzt. In der folgenden Liste sind die jeweiligen Verhaltensweisen beschrieben:

- **Beim Start eines Löschvorgangs** Der Zustand der einzelnen betroffenen Zeilen wird geändert, um ihre Löschung widerzuspiegeln. Wenn ein Löschvorgang durch ein Zurücksetzen rückgängig gemacht wird, wird der ursprüngliche Zeilenstatus wiederhergestellt.
- **Beim Festschreiben eines Löschvorgangs** Die betroffenen Zeilen werden nicht immer aus dem Speicher entfernt. Wenn die Zeile noch nie synchronisiert wurde, wird sie entfernt. Wenn die Zeile synchronisiert wurde, wird sie nicht entfernt, da der Löschvorgang zunächst mit der konsolidierten Datenbank synchronisiert werden muss. Nach der nächsten Synchronisation wird die Zeile dann aus dem Speicher gelöscht.
- **Beim Aktualisieren einer Zeile** Es wird eine neue Version der Zeile erstellt. Die Zustände der alten und der neuen Zeile werden so gesetzt, dass anstelle der alten Zeile die neue Zeile zu sehen ist.
- **Beim Festschreiben einer Zeilenaktualisierung** Wenn eine Transaktion festgeschrieben wird, werden die Zustände aller Zeilen, die von der Transaktion betroffen sind, geändert, um die Festschreibung anzuzeigen. Wenn eine Änderung synchronisiert wird, müssen die alte und die neue Zeile vorhanden sein, um eine Konflikterkennung und -lösung zuzulassen. Die alten Zeilen werden dann aus der Datenbank gelöscht und die neue Zeile wird zu einer normalen Zeile.
- **Beim Hinzufügen einer Zeile** Die Zeile wird der Datenbank hinzugefügt und als nicht festgeschrieben markiert.
- **Beim Festschreiben einer hinzugefügten Zeile** Die Zeile wird als festgeschrieben markiert und erhält die Kennzeichnung, dass eine Synchronisation mit der konsolidierten Datenbank erforderlich ist.

Siehe auch

- „Sichern und Wiederherstellen einer UltraLite- und UltraLite Java Edition-Datenbank“ auf Seite 47
- „Bereinigen einzelner oder gruppierter Transaktionen“ auf Seite 487
- „UltraLite-Transaktionsverarbeitung“ auf Seite 486

UltraLite-Transaktionsverarbeitung

Eine Transaktion ist eine logische Gruppe von Vorgängen, die automatisch durchgeführt werden. Daher werden entweder alle Vorgänge der Transaktion in der Datenbank gespeichert oder gar kein Vorgang. Der Zugriff einer UltraLite-Anwendung auf die UltraLite-Laufzeitanwendung ist serialisiert. Wenngleich mehrere Transaktionen gleichzeitig geöffnet sein können, kann UltraLite jeweils nur eine einzelne Transaktion verarbeiten. Das bedeutet, dass Folgendes für eine Anwendung nicht möglich ist:

- Blockierte Transaktionen (so genannte Deadlocks). UltraLite blockiert nie eine Anforderung basierend auf einer vorhandenen Zeilensperre. In diesem Fall gibt UltraLite sofort einen Fehler zurück.
- Überschreiben von ausstehenden Änderungen. Eine Transaktion kann die ausstehenden Änderungen einer anderen Transaktion nicht überschreiben. Wenn eine Transaktion eine Zeile ändert, sperrt UltraLite diese Zeile, bis die Transaktion **festgeschrieben** oder **zurückgesetzt** wird. Die Sperre verhindert, dass andere Transaktionen die Zeile ändern, aber sie können die Zeile trotzdem lesen.

Tipp

Alle UltraLite-APIs, ausgenommen die UltraLiteJ- und C++-APIs, können im **autocommit**-Modus verwendet werden.

Im Autocommit-Modus führt UltraLite nach jedem Vorgang ein COMMIT aus. Einige APIs verwenden Autocommit standardmäßig. Wenn Sie eine dieser Schnittstellen einsetzen, müssen Sie die Autocommit-Funktion ausschalten, um die Vorteile von Transaktionen mit mehreren Vorgängen zu nutzen. Wie Sie die Autocommit-Funktion ausschalten, hängt von der verwendeten Programmierschnittstelle ab. Bei den meisten Schnittstellen ist diese Funktion eine Eigenschaft des Verbindungsobjekts.

Siehe:

- UltraLite C++: „Transaktionsverwaltung“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite.NET: „Transaktionsverwaltung“ [[UltraLite - .NET-Programmierung](#)]
- UltraLiteJ: „Zeilenvorgänge verwalten“ [[UltraLite® – Java-Programmierung](#)]

Beispiel: Zwei Anwendungen, A und B, lesen dieselbe Zeile aus der Datenbank und berechnen jeweils basierend auf den gelesenen Daten neue Werte für eine ihrer Spalten. Wenn A die Zeile mit ihrem neuen Wert aktualisiert und B dann versucht, dieselbe Zeile zu ändern, erhält B einen Fehler. Ein Versuch, eine gesperrte Zeile zu ändern, verursacht den SQLCODE-Fehler `SQL_LOCKED`, während ein Versuch, eine gelöschte Zeile zu ändern, den Fehler `SQL_NOTFOUND` hervorruft. Sie sollten daher Ihre Anwendung so programmieren, dass sie nach dem Versuch, die Daten zu ändern, den SQLCODE-Wert prüft.

Siehe auch

- UltraLite C++: „Fehlerbehandlung“ [[UltraLite - C- und C++-Programmierung](#)]
- UltraLite.NET: „Fehlerbehandlung“ [[UltraLite - .NET-Programmierung](#)]
- UltraLiteJ: „Fehlerbehandlung“ [[UltraLite® – Java-Programmierung](#)]

Bereinigen einzelner oder gruppierter Transaktionen

Sie können einen Wiederherstellungspunkt in UltraLite wählen, indem Sie die Bereinigung von festgeschriebenen Transaktionen verzögern. Mit dem Wiederherstellungspunkt kann kontrolliert werden, wann eine Teilmenge von SQL-Anweisungen in einer Transaktion zusätzlichen operativen Overhead verursacht, wenn UltraLite die Festschreibungen auf den Massenspeicher schreibt.

UltraLite verwendet standardmäßig eine vorgangsbasierte Standardeinstellung, bei der einzelne Transaktionen beim Festschreiben sofort auf den Massenspeicher geschrieben werden. Bei einigen Deployments können diese häufigen Vorgänge zu aufwändig sein und den Transaktionsdurchsatz einschränken. Um die durch diese Standardeinstellung bewirkten Performancekosten zu verringern, können Sie einen statusbasierten Ansatz auswählen. Diese Methode verzögert besonders bei Anwendungen, die Autocommit-Vorgänge verwenden, den zusätzlichen Aufwand, der entsteht, wenn die festgeschriebenen Transaktionen auf den Massenspeicher geschrieben werden.

- **Bei Checkpoint** Sie können einen eigenen Checkpoint festlegen und ihn verwenden, um die bis dahin ausgeführten Vorgänge zu übernehmen. Sie können eine beliebige Anzahl von Checkpoints verwenden, entweder innerhalb einer einzelnen oder mehrerer Transaktionen.
- **Gruppirt** Sie können einen Transaktionsschwellenwert bzw. einen Timeoutwert für die Freigabe der ausgeführten Vorgänge wählen.

Das statusbasierte Verzögern der Bereinigung von Festschreibungen ermöglicht eine bessere Performance und ein klar strukturiertes Design der Anwendung, weil Anwendungen nicht auf eine Antwort von UltraLite warten müssen. Die verzögerte Bereinigung von Festschreibungen bietet auch mehr Sicherheit für Transaktionen, da Sie eine bessere Kontrolle über die Daten erhalten, deren Bearbeitung noch nicht abgeschlossen wurde. In einer Vertriebsanwendung kann z.B. eine Bestellung für eine zweite Anwendung verfügbar sein, bevor alle Elemente hinzugefügt oder genehmigt werden.

Es ist jedoch wichtig, die Wiederherstellbarkeit einer Transaktion, deren Übernahme auf den Massenspeicher verzögert wurde, zu beachten. Nicht übernommene Transaktionen können nicht wiederhergestellt werden. Sie müssen daher die Vor- und Nachteile von Datenintegrität der Anwendung einerseits und Performance andererseits abwägen.

Siehe auch

- „UltraLite-Verbindungsparameter COMMIT_FLUSH“ auf Seite 175
- „UltraLite-Option commit_flush_count [temporär]“ auf Seite 196
- „UltraLite-Option commit_flush_timeout [temporär]“ auf Seite 197
- „CHECKPOINT-Anweisung [UltraLite]“ auf Seite 432
- ULCheckpoint-Methode [UltraLite Embedded SQL] [[UltraLite - C- und C++-Programmierung](#)]
- ULConnection.Checkpoint-Methode [UltraLite C++] [[UltraLite - C- und C++-Programmierung](#)]

Benchmarktipps für UltraLite

Benchmarktestaktivitäten sollten im Allgemeinen vor dem Erreichen des Produktionsstadiums in der Anwendungsentwicklung durchgeführt werden. In dieser Phase testen Sie die UltraLite-Datenbank und die Anwendung, um sicherzustellen, dass beide Komponenten so effizient wie möglich zusammenarbeiten. Wenn Sie durch Ihre Tests feststellen, dass die Performance nicht optimal ist, können Sie Ihre Datenbank bzw. Anwendung optimieren, um die Benchmark-Ergebnisse zu verbessern.

Hinweis

Wenn Ihr UltraLite-Deployment Teil einer MobiLink-Synchronisationsumgebung ist, vergessen Sie nicht, auch die Synchronisationsperformance zu testen. Siehe „[MobiLink-Optimierung der Performance](#)“ [[MobiLink - Serveradministration](#)].

Arten von Benchmarktests

Sie können Benchmarktests ausführen, um die Performance von folgenden Elementen zu testen:

- SQL-Anweisungen

Die UltraLite-Datenbank wurde optimiert, um SQL-Abfragen effizient zu verarbeiten und schnell Ergebnisse zurückzugeben. Trotzdem sollten Sie herausfinden, wie gut größere Abfragen ausgeführt werden, um die Datenbankperformance zu verbessern.

- Synchronisation

Der Schlüssel zum optimalen Durchsatz während der MobiLink-Synchronisation besteht darin, mehrere Synchronisationen gleichzeitig und effizient auszuführen.

- Indizes

- Tabellendesign

- Anwendungscode

- Gerätekonfiguration

Vergleichen Sie beispielsweise einen externen Flash-Speicher mit dem internen Speicher des Geräts als Deployment-Speicherort für UltraLite.

- Datenbankkonfiguration

Probieren Sie z. B. verschiedene Cachegrößen, Seitengrößen, Reservierungsgrößen, Indizes, Hash-Größen etc., aus.

- Datendurchsatz (auf Transaktionen pro Sekunde basierend)

Auch wenn UltraLite nicht als Datenbank für die Verarbeitung von Massendaten vorgesehen ist, ist dies eine Benchmark, die Sie je nach Ihren Unternehmensanforderungen testen sollten.

- Softwareänderungen

Sie sollten die Auswirkungen auf Softwareänderungen bei zwei verschiedenen Versionen von UltraLite oder verschiedenen Versionen einer Anwendung testen.

Siehe auch

- „SQL-Abfrage testen“ auf Seite 489
- „MobiLink-Optimierung der Performance“ [*MobiLink - Serveradministration*]

SQL-Abfrage testen

SQL-Abfragen können einfach oder auch komplex sein. Abhängig von der Natur und der Wichtigkeit Ihrer Abfragen können Sie unter zwei Benchmarktests wählen.

Repräsentative SQL-Benchmarks

Diese Art von Tests erfordern, dass Sie eine Auswahl von Anweisungen testen, die für typische Transaktionen repräsentativ sind, die die Anwendung während alltäglicher Vorgänge durchführt. Verschiedene Anwendungen erfordern unterschiedliche Benchmarktests, weil für jede Anwendung andere Unternehmensanforderungen gelten. Eine Anwendung für die Ablesung von Zählern würde z.B. nur eine einzige INSERT-Anweisung testen. Eine Anwendung für den mobilen Außendienst hingegen sollte mehrere INSERT-Anweisungen testen, zusätzlich zu mehreren SELECT- und möglicherweise einer UPDATE-Anweisung.

Der Umfang Ihrer Abfragen in Ihrer Anwendung kann die Möglichkeiten für einen realistischen Test deutlich einschränken. Wenn in Ihrer Anwendung viele Abfragen verarbeitet werden müssen, werden Sie möglicherweise darauf beschränkt, die spezifischen SQL-Benchmarktests durchzuführen.

Gezielte SQL-Benchmarks

Wenn Ihre Anwendungen mit einer großen Anzahl von Anweisungen arbeiten, kann es sinnvoll sein, den Umfang der Tests auf die folgenden Elemente zu beschränken:

- Die am häufigsten benutzten Anweisungen.
- Die Anweisungen, die große Datenvolumen verarbeiten.
- Die Anweisungen mit besonders zeitkritischen Anforderungen.
- Die Anweisungen, die für die Geschäftsanforderungen Ihrer Anwendung besonders wichtig sind.
- Die komplexesten Anweisungen, wie z. B. jene, die die größte Anzahl von Tabellen-Joins haben oder eine Menge von Unterabfragen verwenden. Diese Anweisungen können die Gerätere Ressourcen stark belasten. Auch wenn sie nicht regelmäßig verwendet werden, kann es sinnvoll sein, zu überprüfen, ob sie nicht die Kapazität des Geräts überschreiten.
- Die Anweisungen, die nicht auf einem Index basieren.
- Die Anweisungen, die einen großen Anteil an Speicherressourcen beanspruchen.

Methoden

1. Die Vorbereitungsphase

Ermöglicht es Ihnen, Ihr Datenbankdesign abzuschließen und einen stabilen Punkt in Ihrer Anwendungsentwicklung zu erreichen, bevor Sie mit den Benchmarktests beginnen.

2. Die Erstellungsphase

Ermöglicht es Ihnen, ein angepasstes Programm zu erstellen, das das Endbenutzerverhalten wiedergibt, das Sie für Ihr UltraLite-Deployment erwarten.

3. Die Ausführungs- und Analysephase

Ermöglicht es Ihnen, die verschiedenen Elemente Ihrer Datenbank aufeinander abzustimmen und die Ergebnisse dieser Änderungen aufzuzeichnen, damit Sie sie analysieren können. Die Tests werden wiederholt, bis der maximale Nutzen der einzelnen Änderungen erreicht ist.

Die Vorbereitungsphase

Während der Vorbereitungsphase bringen Sie die Datenbank und die Anwendung in einen Zustand, in dem sie zu erfolgreichen Benchmark-Kandidaten werden, und ermitteln, welche Ziele Sie mit Ihren Tests erreichen wollen. Führen Sie Folgendes durch:

1. Schließen Sie das logische Design der Datenbank ab.

Vergewissern Sie sich, dass folgende Aufgaben ausgeführt wurden:

- Sie haben Tabellen erstellt und sie mit repräsentativen Daten gefüllt.
- Sie haben Indizes erstellt, um Daten in diesen Tabellen effizienter abrufen zu können.

2. Bereiten Sie die physische Deployment-Umgebung sowohl der Datenbank als auch der Anwendung vor. Die Deployment-Umgebung muss die endgültige Produktionsumgebung genau darstellen. Das bedeutet, dass die Labor- und Produktionsumgebungen dieselben Speicher- und Festplattenkonfigurationen auf demselben Plattform-/Gerätetyp haben sollten.

3. Stellen Sie sicher, dass Sie einen stabilen Punkt in der Phase Ihrer Anwendungsprogrammierung erreicht haben. Vergessen Sie nicht, dass Sie nach Performanceoptimierungen und nicht nach Fehlern suchen, aber als Folge der Tests werden möglicherweise auch Fehler erkannt.

Alle Abfragen müssen auf die erforderlichen Daten zugreifen können und ein entsprechendes Datenvolumen zurückgeben. Wenn die Produktionsumgebung eine Datensortierung erfordert, achten Sie darauf, dass Abfragen auch solche Daten erfassen. Anderenfalls kann die Anwendung repräsentative Speichererfordernisse nicht testen.

4. Führen Sie ein Deployment einer Kopie der Datenbank und der Anwendung auf der geplanten Festplattenkonfiguration (einschließlich Speicherort) vor.

5. Legen Sie fest, welches Element der Datenbankperformance Sie überprüfen und ggf. optimieren wollen.

Sie können nun Benchmark-Tests mit der Datenbank und der Anwendung durchführen.

Siehe auch

- „Arten von Benchmarktests“ auf Seite 488
- „Methoden“ auf Seite 490

Die Erstellungsphase

Sie müssen Tests erstellen, die zu verlässlichen Ergebnissen führen. Ansonsten ist es nicht möglich, die Ergebnisse im Laufe der Zeit zu vergleichen.

Die folgenden Merkmale machen einen Benchmarktest effektiv und verlässlich:

- **Ziel** Suchen Sie nach einer Kennzahl für die Performance oder möchten Sie die Dauer messen, die für die Verarbeitung eines Befehls in der Datenbank erforderlich ist? Im ersten Fall, also wenn Sie die SQL-Performance testen, kann es sinnvoll sein, eine oder mehrere Anweisungen wiederholt auszuführen, bis eine bestimmte Zeitspanne abgelaufen ist. Dieser Test liefert Ihnen die Durchsatzrate, die folgendermaßen zusammengefasst werden kann:

`statement-number / time-interval = throughput ratio`

- **Umgebung** Richten Sie eine Testumgebung als Ausgangsbasis (Baseline) ein und zeichnen Sie ihr Design und ihren Geltungsbereich auf. Wenn Sie denselben Test nicht unter denselben Bedingungen wiederholen können, ist ein Vergleich der Ergebnisse dieses Tests nicht wirklich möglich. Zusätzlich sollten die Hard- und Software, die Sie im Labor als Teil Ihres Benchmarktests verwenden, mit denen Ihrer Produktionsumgebung übereinstimmen.
- **Status** Verlässliche Benchmarktests beginnen jede Wiederholung mit derselben Aktion. Legen Sie fest, ob Anwendungen von Drittherstellern gleichzeitig mit UltraLite ausgeführt werden sollen. Wenn sich solche Anwendungen auf die Performance auswirken, sollten Sie sie Ihrem Benchmarktestdesign hinzufügen. Anwendungen von Drittherstellern, die nicht laufen sollen, müssen stets vollständig beendet werden, da auch minimierte oder inaktive Anwendungen bzw. Prozesse die Ergebnisse verfälschen können, weil weiterhin Speicher beansprucht wird.
- **Ergebnisse** Die Ergebnisse von Benchmarktests müssen nach jeder Testwiederholung in einheitlicher Weise erfasst werden. Im Lauf der Zeit ergibt sich aus den Ergebnissen ein Trend, anhand dessen Sie ermitteln können, welche Änderungen zu einer Verbesserung der UltraLite-Performance führen - entweder in der Datenbank oder in der Anwendung (oder in beiden).
- **Zeitnahmungsverfahren** Benchmarktests simulieren Benutzeraktionen, daher protokollieren Sie auch die Ausführungszeiten dieser Aktionen. Achten Sie darauf, dass Ihr Zeitnahmungsverfahren systematisch ist, damit Ausführungszeiten in den Ergebnissen Ihrer Tests präzise wiedergegeben werden.

Siehe auch

- „Methoden“ auf Seite 490
- „SQL-Abfrage testen“ auf Seite 489

Die Ausführungs- und Analysephase

Die Durchführung der Benchmarktests ist jene Phase, in der Sie Ihre Datenbank durch wiederholtes Ausführen eines Tests optimieren, indem Sie jeweils Änderungen in der Datenbank durchführen (z.B. den Wert einer oder mehrerer Datenbankeigenschaften oder Verbindungsparameter ändern) und dann den Test erneut durchführen, um die Auswirkung der Änderungen zu beobachten.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Die folgende Prozedur setzt voraus, dass Sie verschiedene Datenbankeigenschaften bzw. Verbindungsparameter testen, um den maximalen Nutzen zu ermitteln. Wiederholen Sie diese Prozedur, bis alle betroffenen Parameter getestet sind.

Tipp

Wählen Sie nur solche Eigenschaften oder Parameter aus, die sich signifikant auf die Arbeitslast auswirken und wichtig für die Ziele Ihres UltraLite-Deployments sind.

Aufgabe

1. Erstellen Sie einen Basiswert (Baseline), indem Sie den ersten Durchgang des Tests ausführen. Da Sie verschiedene Datenbankeigenschaften bzw. Verbindungsparameter testen, würden Sie in diesem Fall soweit wie möglich UltraLite-Standardeinstellungen verwenden.
2. Beginnen Sie jetzt mit Ihren normalen Testläufen, indem Sie nur jeweils eine Datenbankeigenschaft bzw. Verbindungsparameter ändern. Diese Einschränkung stellt sicher, dass die von Ihnen gesammelten Ergebnisse systematisch sind, und ermittelt werden kann, wann Sie den maximalen Nutzen aus Ihren Optimierungsaktivitäten erzielt haben.
3. Die Ausgabe des Benchmarkprogramms sollte Folgendes enthalten:
 - Einen Bezeichner oder ein Label für jeden Test
 - Die Zahl der Wiederholungen der Programmausführung
 - Den Namen des überprüften Elements und der daran durchgeführten Änderungen
 - Die aufgezeichnete verstrichene Zeit

Beispiel: Auch wenn Sie andere Datenbankparameter testen könnten, nehmen wir an, dass Sie Ihre Tests auf unterschiedliche Seitengrößen, Cachegrößen und Reservierungsgrößen beschränken. Die Ausgabe könnte in einer Tabelle gespeichert werden, die der folgenden ähnelt:

PROP/PARM	VALUES		
TEST NUMBER	001	002	003
page_size	1	2	8
CACHE_SIZE	128	256	512
RESERVE_SIZE	128	256	512

STMT ID	EXECUTION (seconds)		
01	01.55	01.50	01.49
02	02.01	02.20	01.59
03	00.33	00.55	00.44

4. Wenn Sie eine Wiederholung abgeschlossen haben, setzen Sie die Datenbank auf ihre Basiswerte (Baseline-Status) zurück, um sicherzustellen, dass Sie nicht unbeabsichtigt die Ergebnisse von nachfolgenden Durchgängen verfälschen.

Ergebnisse

Abhängig von den Ergebnissen des Benchmarktests führen Sie eine der folgenden Aktionen aus:

- Wenn sich die Performance verbessert, ändern Sie den Wert derselben Eigenschaft bzw. desselben Parameters und führen den Test erneut aus. Fahren Sie fort, diesen Wert zu optimieren, bis sich die Performance nicht weiter verbessert.
- Wenn sich die Performance verschlechtert, setzen Sie den Wert der Eigenschaft bzw. des Parameters auf den vorherigen Wert zurück.

Nächste Schritte

Testen Sie eine weitere neue Eigenschaft bzw. einen weiteren neuen Parameter.

Siehe auch

- [„Methoden“ auf Seite 490](#)

UltraLite-Fehlerbehandlung

Starten der UltraLite-Engine nicht möglich

Symptom

Sie müssen den Verbindungsparameter START verwenden, um die UltraLite-Engine mit der folgenden Definition zu starten, der Client gibt jedoch SQLE_UNABLE_TO_CONNECT_OR_START zurück.

```
START="\Program Files\ulengl6.exe"
```

Erklärung

Die Position der Anführungszeichen ist inkorrekt.

Empfehlung

Damit dieser Parameter funktioniert, muss das erste Anführungszeichen dem Zeichen \ folgen. Sie können Leerstellen in diesem Pfad z.B. folgendermaßen begrenzen:

```
START=\"Program Files\ulengl6.exe"
```

oder

```
START='\"Program Files\ulengl6.exe"'
```

Verbinden mit Datenbanken nach Upgrade nicht möglich

Symptom

Sie haben ein Upgrade von UltraLite durchgeführt. Sie stellen fest, dass Sie in der Lage sind, eine leere Datenbank mit den Administrationstools zu erstellen. Wenn Sie allerdings versuchen, eine Verbindung zu dieser oder zu einer anderen Datenbank (einschließlich *CustDB.udb*) mit Sybase Central herzustellen, erhalten Sie eine Fehlermeldung. Das Herstellen einer Verbindung zu SQL Anywhere-Datenbanken funktioniert hingegen problemlos.

Erklärung

Sie haben nicht alle SQL Anywhere-Anwendungen und -Prozesse geschlossen. Daher wurden Ihre UltraLite-Plug-Ins nicht korrekt installiert.

Empfehlung

Entfernen Sie SQL Anywhere und installieren Sie es erneut.

1. Schließen Sie Sybase Central, Interactive SQL und sonstige laufenden Datenbank-Engines.
2. Führen Sie die folgenden Befehle aus:

```
dbisql -terminate
```

`scjview -terminate`

3. Öffnen Sie den Windows **Task-Manager** und beenden Sie ggf. die Prozesse *scjview.exe* und *dbisql.exe*.
4. Installieren Sie die aktuellste Version von UltraLite erneut.

Siehe auch

- „UltraLite-Upgrades“ [[SQL Anywhere 16 - Änderungen und Upgrades](#)]

UltraLite-Datenbankbeschädigung

Symptom

Ihre UltraLite-Datenbank ist möglicherweise beschädigt, wenn folgende Probleme auftreten:

- Sie generiert die folgenden Fehler:
 - `SQLE_DEVICE_ERROR`
 - `SQLE_DATABASE_ERROR` (kann auch ein Symptom für andere Probleme sein)
 - `SQLE_MEMORY_ERROR` (kann auch ein Symptom für andere Probleme sein)
- Sie stürzt ab oder gibt ungültige Abfrageergebnisse zurück.

Erklärung

Es gibt zwei Hauptursachen für eine Beschädigung:

- Die häufigste Ursache ist, dass das Gerät Schwierigkeiten hat, die Datei zu speichern, und dabei fälschlicherweise ihren Inhalt verändert. Dieses Problem bewirkt üblicherweise, dass die UltraLite-Datenbank sehr bald nicht mehr funktionsfähig ist.
- Die zweite Ursache ist ein Fehler im UltraLite-Code, der verhindert, dass der Index korrekt aufrechterhalten wird. Diese Probleme können lange Zeit unbemerkt bleiben, weil die Änderung in den Ergebnissen einer Abfrage schwer zu erkennen sind.

Empfehlung

Prüfsummen werden verwendet, um Offline-Beschädigungen in einer UltraLite-Datenbank zu ermitteln, was die Wahrscheinlichkeit verringert, dass andere Daten aufgrund einer beschädigten kritischen Seite in Mitleidenschaft gezogen werden. Wenn eine Prüfsummenvalidierung beim Laden einer Seite in der UltraLite-Datenbank fehlschlägt, stoppt UltraLite die Datenbank sofort und meldet einen schwerwiegenden Fehler. Dieser Fehler kann nicht korrigiert werden. Stattdessen müssen Sie folgende Maßnahmen treffen:

1. Melden Sie den Fehler. Es ist hilfreich, wenn Sie die Abfolge der Ereignisse kennen, die die Beschädigung bewirkt haben, und wenn der Fehler reproduzierbar ist.
2. Wenn Sie die Daten benötigen, entladen Sie den Inhalt der UltraLite-Datenbank in eine Datei.

3. Erstellen Sie eine neue, leere UltraLite-Datenbank.
4. Fügen Sie die Daten wieder ein, indem Sie entweder eine Synchronisation ausführen oder die entladenen Daten laden.

Siehe auch

- [„UltraLite-Erstellungsparameter checksum_level“ auf Seite 144](#)

Datenbankgröße nicht stabil

Symptom

Ihre Anwendung sammelt eine Menge von BLOBs von mehreren Clientgeräten, synchronisiert diese Informationen mit einer konsolidierten Datenbank. Anschließend werden die synchronisierten Daten von allen Clientgeräten gelöscht. Die Datenbankgröße bleibt jedoch umfangreich, obwohl die Daten aus der Datenbank entfernt werden. Dies ist ein Problem, weil die Dateigröße auf Grund der beschränkten Ressourcen auf dem Gerät umsichtig verwaltet werden muss.

Erklärung

Die Datenbank sollte nur größer werden, wenn Ihre Daten in der Datenbank zunehmen. Wenn eine Datenbankdatei einmal größer geworden ist, bleibt diese Größe erhalten und wird nicht automatisch vermindert. Freier Speicherplatz wird intern in der Datei verwaltet.

Empfehlung

Achten Sie darauf, bei Tabellen, die nicht synchronisiert werden, die Anweisungen STOP, SYNCHRONIZATION, DELETE oder TRUNCATE nicht zu verwenden. Verwenden Sie stattdessen die DELETE-Anweisung mit einer FROM *Tabellenname*-Klausel für diese Tabellen.

Erstellen Sie die Datenbank nach der Synchronisation erneut:

1. Erstellen Sie die Datenbank, für die ein Deployment auf die Geräte vorgenommen wird.
2. Erstellen Sie ein SQL-Skript von DDL-Anweisungen, die das Schema festlegen, das von den Clientgeräten benötigt wird. Siehe [„Deployment von UltraLite-Datenbankschema-Upgrades“ auf Seite 125](#).
3. Synchronisieren Sie die Daten.
4. Löschen Sie die Datenbank.
5. Erstellen Sie eine neue, leere Datenbank und verwenden Sie das Standard-Datenbankschema mit der Anweisung ALTER DATABASE SCHEMA FROM FILE.

Siehe auch

- „STOP SYNCHRONIZATION DELETE-Anweisung [UltraLite]“ auf Seite 461
- „TRUNCATE TABLE-Anweisung [UltraLite]“ auf Seite 463
- „DELETE-Anweisung [UltraLite]“ auf Seite 445
- „ALTER DATABASE SCHEMA FROM FILE-Anweisung [UltraLite]“ auf Seite 423

Importieren von ASCII-Daten in eine neue UltraLite-Datenbank

Symptom

Sie haben eine neue UltraLite-Datenbank erstellt und verfügen über eine ASCII-Datendatei im .csv-Format, die Sie nicht importieren können.

Erklärung

Das .csv-Format wird von keinem der UltraLite-Administrationstools unterstützt.

Empfehlung

Sie können eine der folgenden Methoden versuchen:

- Verwenden Sie Interactive SQL (dbisql), um die Daten zu importieren. Verbinden Sie sich mit der UltraLite-Datenbank und klicken Sie auf **Daten » Daten importieren**. Alternativ dazu verbinden Sie sich mit der UltraLite-Datenbank und führen die INPUT-Anweisung aus (diese Anweisung kann nicht in einem UltraLite PreparedStatement-Objekt verwendet werden).

Hinweis

UltraLite erfordert Primärschlüssel. Auch wenn Interactive SQL die Tabellen erstellen kann, werden für sie nicht automatisch die Primärschlüssel erstellt. Stellen Sie immer eine Verbindung zu einer leeren UltraLite-Datenbank her, die Sie für diesen Zweck erstellt haben.

- Wenn Sie diese Funktionalität als Teil eines Batch-Prozesses verwenden wollen, müssen Sie Ihren eigenen Code schreiben.

Siehe auch

- „INPUT-Anweisung [Interactive SQL]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Interactive SQL-Dienstprogramm für UltraLite (dbisql)“ auf Seite 201

Dienstprogramme laufen weiterhin in der vorherigen Version

Symptom

Sie haben gerade UltraLite 16 installiert. Wenn Sie jedoch versuchen, eines der UltraLite-Dienstprogramme auszuführen, startet die vorherige Version.

Erklärung

Wenn mehrere Versionen von UltraLite auf Ihrem Computer installiert sind, müssen Sie bei der Verwendung der Administrationstools auf den Systempfad achten. Da bei der Installation das Programmverzeichnis der zuletzt installierten Version am Ende Ihres Systempfads angefügt wird, ist es möglich, eine neue Version der Software zu installieren und dennoch, ohne es zu bemerken, mit der früheren Version zu arbeiten.

Empfehlung

Es gibt mehrere Behelfslösungen für dieses Problem.

Siehe auch

- „Sicherstellen, dass Sie die richtige Version der Dienstprogramme ausführen, wenn mehrere Versionen installiert sind“ [[SQL Anywhere 16 - Änderungen und Upgrades](#)]

Ergebnismenge ändert sich unvorhersehbar

Symptom

Sie führen eine Abfrage aus und die von Ihnen erwartete Ergebnismenge ändert sich bei jeder Ausführung.

Erklärung

Überprüfen Sie genau die Ergebnismenge, die Sie erhalten. Sind die Ergebnisse in der Menge wirklich verschieden? Oder werden sie bloß jedes Mal in der effizientesten Reihenfolge zurückgegeben? Die ausgewählte Reihenfolge kann sich bei jeder Ausführung der Abfrage ändern, abhängig von Ihrem letzten Zugriff auf die Zeile und von anderen Faktoren.

Empfehlung

Wenn Ihre Ergebnismenge in einer vorhersehbaren und konsistenten Reihenfolge zurückgegeben werden soll, stellen Sie sicher, dass die SELECT-Anweisung eine ORDER BY-Klausel enthält. Wenn die Abfrage dennoch weiterhin Ergebnisse inkorrekt zurückgibt, ist möglicherweise Ihre Datenbank beschädigt.

Siehe auch

- „SELECT-Anweisung [UltraLite]“ auf Seite 457
- „UltraLite-Datenbankbeschädigung“ auf Seite 496

UltraLite-Engine-Client schlägt mit Fehler -764 fehl

Gilt für

Windows Mobile

Symptom

Sie führen die UltraLite-Engine auf einem Windows Mobile-Gerät aus und der Client gibt einen Fehler mit der Nummer -764 zurück.

Erklärung

Ein Fehler mit der Nummer -764 bedeutet, dass die Engine nicht gefunden wurde und nicht gestartet werden konnte.

Empfehlung

Versuchen Sie Folgendes:

- Sie könnten ein erneutes Deployment der Engine am empfohlenen Deployment-Speicherort, dem Verzeichnis `\Windows`, vornehmen. UltraLite sucht automatisch nach den Engine-Dateien an diesem Speicherort.
- Wenn Sie die Engine an einem anderen Speicherort installiert haben, stellen Sie sicher, dass Ihr Verbindungscode den Verbindungsparameter `START` verwendet.
- Wenn Sie den Verbindungsparameter `START` angegeben haben und Sie sicher sind, dass der Pfad zur Engine korrekt ist, vergewissern Sie sich, dass Sie die korrekte Escapesequenz für Sonderzeichen im Pfadnamen verwendet haben.

Möglicherweise müssen Sie z.B. diesen Code ändern:

```
ULConnection conn = new ULConnection(@"dbf=\Program Files\HelloEngine
\HelloEngine.udb;
START=\Windows\ulengl6.exe")
```

Ändern Sie den Code zu etwas Ähnlichem wie folgt:

```
ULConnection conn = new ULConnection(@"dbf=\\\"Program Files \"\
\HelloEngine\HelloEngine.udb;
START=\\Windows\ulengl6.exe");
```

Siehe auch

- „Start der UltraLite-Engine“ auf Seite 128
- „UltraLite-Verbindungsparameter `START`“ auf Seite 189

Index

Symbole

%, Operator

UltraLite, modulo-Funktion, 375

&

UltraLite-Bit-Operator, 294

-, Kommentarindikator

UltraLite, Info, 272

--case, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--checksum_level, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--collation, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--databaseid, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--datacopy, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--date, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--date_order, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--dba, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--exactschema, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--exclude, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--fips, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--help, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--identity-file, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--identity--password, Option

UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (uload), 223

--identity-file, Option

UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (uload), 223

--identity-password, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--insertforupload, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--key, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--list_collation, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--log, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

UltraLite-Synchronisationsdienstprogramm (ulsync), 227

--max_hash_size, Option

UltraLite, Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--mirror, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--no_warnings, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--obfuscate, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--overwrite, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--page_size, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--precision, Option

UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit), 214

--prompt, Option

- UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- publication, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- quiet, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- reserve, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- rootcert, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- SAconnect, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- scale, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- sql, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- sync_publication, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- time_format, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- timestamp_increment, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- timestamp_with_time_zone_format, Option
 - UltraLite, Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- utf8, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- ? Option Z
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- a, Option
 - Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload),254
 - ulinit-Erstellungsparameter,25
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
- b, Option
 - Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload),256
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
- c, Option
 - Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload),256
 - Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload),254
 - UltraLite Java Edition-Dienstprogramm für Datenbankinformationen (uljinfo),252
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - UltraLite-Dienstprogramm zum Validieren von Datenbanken (ulvalid),237
 - UltraLite-Informationsdienstprogramm (ulinfo),213
 - UltraLite-Synchronisationsdienstprogramm (ulsync),227
- d, Option
 - Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload),256
 - Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload),254
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
- d1, Option
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202
- data, Option

-
- UltraLite-Dienstprogramm zum Validieren von Datenbanken (ulvalid),237
 - E, Option
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - e, Option
 - Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload),256
 - UltraLite Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - UltraLite-Dienstprogramm zum Validieren von Datenbanken (ulvalid),237
 - ek, Option
 - Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload),256
 - Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload),254
 - UltraLite Java Edition-Dienstprogramm für Datenbankinformationen (uljinfo),252
 - f, Option
 - Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload),256
 - Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload),254
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - g, Option
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - h, Option
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - i, Option
 - Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload),254
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - K, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - k, Option
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Löschen der Datenbank (ulerase),212
 - l, Option
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - m, Option
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - n, Option
 - Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload),256
 - Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload),254
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - nearest_century, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - nogui, Option
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202

- o, Option
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
- onerror, Option
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202
- p, Option
 - Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload),256
 - Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload),254
 - UltraLite Java Edition-Dienstprogramm für Datenbankinformationen (uljinfo),252
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - UltraLite-Dienstprogramm zum Löschen der Datenbank (ulerase),212
 - UltraLite-Synchronisationsdienstprogramm (ulsync),227
- q, Option
 - Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload),256
 - Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload),254
 - UltraLite Java Edition-Dienstprogramm für Datenbankinformationen (uljinfo),252
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - UltraLite-Dienstprogramm zum Löschen der Datenbank (ulerase),212
 - UltraLite-Dienstprogramm zum Validieren von Datenbanken (ulvalid),237
 - UltraLite-Informationsdienstprogramm (ulinfo),213
 - UltraLite-Synchronisationsdienstprogramm (ulsync),227
- r, Option
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Synchronisationsdienstprogramm (ulsync),227
- S, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- s, Option
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
- SQL, Befehl
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202
- t, Option
 - Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload),256
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
- u, Option
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Löschen der Datenbank (ulerase),212
- ud, Option
 - UltraLite Engine, Dienstprogramm (ulengl6),210
- ul, Option
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202
- v, Option
 - Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload),256

-
- Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload),254
 - UltraLite Java Edition-Dienstprogramm für Datenbankinformationen (uljinfo),252
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - UltraLite-Dienstprogramm zum Validieren von Datenbanken (ulvalid),237
 - UltraLite-Synchronisationsdienstprogramm (ulsync),227
 - version
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202
 - w, Option
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - x, Option
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - y, Option
 - Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload),256
 - Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload),254
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - Z, Option
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - z, Option
 - UltraLite-Dienstprogramm für den SQL-Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - /*, Kommentarindikator
 - UltraLite, Info,272
 - //, Kommentarindikator
 - UltraLite, Info,272
 - 10054
 - UltraLite, Fehler im Synchronisationsdatenstrom,105
 - 130, Fehler
 - SQL-Code für UltraLite-Schema-Upgrade,125
 - 256-Bit, starke Verschlüsselung
 - UltraLite, Verwendung,29
 - UltraLite-Verbindungsparameter,179
 - ?
 - UltraLite-Eingabeparameter,282
 - @data-Option
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - UltraLite-Dienstprogramm zum Löschen der Datenbank (ulerase),212
 - UltraLite-Informationsdienstprogramm (ulinfo),213
 - UltraLite-Synchronisationsdienstprogramm (ulsync),227
 - ^
 - UltraLite-Bit-Operator,294
 - |
 - UltraLite-Bit-Operator,294
 - ~
 - UltraLite-Bit-Operator,294
 - A**
 - Abfragen
 - UltraLite, Fehlerbehandlung, unvorhersehbare Ergebnismengen,499
 - UltraLite-Optimierung,458
 - Abfrageoptimierer
 - (Siehe auch Optimierer)
 - UltraLite,477
 - Abfrageoptimierung
 - (Siehe auch Optimierer)
 - UltraLite SQL,477

- Abgeleitete Tabellen
 - UltraLite SQL,281
 - UltraLite, FROM-Klausel,450
- Abrufe
 - UltraLite,44
- Abrufen, Zeile
 - UltraLite, Parallelität,44
- ABS-Funktion
 - UltraLite-Syntax,323
- ACOS-Funktion
 - UltraLite-Syntax,323
- ActiveSync
 - Anwendungen für UltraLite-Clients registrieren,130
 - Deployment von ActiveSync-Providern für UltraLite ,128
 - MobiLink-Deployment von UltraLite-Anwendungen,88
 - UltraLite, Deployment von Providerdateien,128
- ActiveSync-Provider-Installationsprogramm (mlasinst)
 - Anwendungen für UltraLite-Clients registrieren,130
- Additional Parameters
 - UltraLite, Synchronisationsparameter,91
- Administrationstools
 - UltraLite, Fehlerbehandlung,495
- AES, Verschlüsselungsalgorithmus
 - UltraLite, fips,29
 - UltraLite, fips-Erstellungsparameter,150
 - UltraLite, Verwendung,29
- AES-Verschlüsselungsalgorithmus
 - UltraLite, Entwicklungsschritte,19
- Aggregatausdrücke
 - UltraLite, SQL-Syntax,281
- Aggregatfunktionen
 - UltraLite, alphabetische Liste,317
- Aktualisieren
 - UltraLite, Zeilen aktualisieren,466
- Aktualisierungen
 - UltraLite-Datenbanken,485
- Aktuelle Zeile
 - UltraLite, Parallelität,484
- Aliase
 - UltraLite-Entsprechungen,316
 - UltraLite-Spalten,458
- ALL-Suchbedingung
 - UltraLite SQL,287
- AllowDownloadDupRows
 - UltraLite, Synchronisationsparameter,91
- allsync, Tabellen
 - UltraLite-Datenbanken synchronisieren,78
- ALTER DATABASE SCHEMA FROM FILE-Anweisung
 - UltraLite, Auswirkung von Schemaänderungen,49
 - UltraLite, Syntax,423
 - Verwendungszweck,125
- ALTER PUBLICATION-Anweisung
 - UltraLite, Syntax,424
- ALTER SYNCHRONIZATION PROFILE-Anweisung
 - UltraLite, Syntax,425
- ALTER TABLE-Anweisung
 - UltraLite, Syntax,427
- ALTER USER-Anweisung
 - UltraLite, Syntax,431
- AND
 - UltraLite, logische Operatoren,286
 - UltraLite-Bit-Operatoren,294
- Ändern
 - UltraLite, ALTER PUBLICATION-Anweisung,424
 - UltraLite, ALTER TABLE-Anweisung,427,431
 - UltraLite-Spalten,427
 - UltraLite-Spaltenmethoden,52
 - UltraLite-Tabellen,427
- Ändern, UltraLite-Spaltendefinitionen
 - Info,52
- Android
 - Verschlüsselung und Verschleierung,29
- Anforderungen
 - UltraLite-Parallelität,484
 - UltraLite-Verwaltung,484
- Anweisungen
 - UltraLite,421
 - UltraLite, Anweisungsperformance optimieren,489
 - UltraLite, Typen,422
 - UltraLite, vorbereitete Eingabeparameter,282
- Anwendungen
 - (*Siehe auch* UltraLite-Anwendungen)
- Anwendungsprofilerstellung
 - UltraLite, Benchmarktests,488
- ANY-Suchbedingung
 - UltraLite SQL,288
- Anzeigen
 - UltraLite-Ausführungspläne,480

- UltraLite-Tabellenmethoden,54
- Anzeigen, UltraLite-Datenbankeinstellungen
Info,39
- ApplyFile, Methode
 - UltraLite, Ersatz für Schema-Upgrade ,125
- ARGN-Funktion
 - UltraLite-Syntax,324
- Arithmetische Operatoren
 - UltraLite, SQL-Syntax,293
 - UltraLite-Operatoren,293
- Arkuskosinus-Funktion
 - UltraLite, ACOS-Funktion,323
- Arkussinus-Funktion
 - UltraLite, ASIN-Funktion,325
- Arkustangens-Funktion
 - UltraLite, ATAN-Funktion,326
- Artikel
 - UltraLite, Kopiermethode,55
 - UltraLite-Datenbanken,78
- ASCII
 - UltraLite, Sortierung,26
 - UltraLite-Syntax,325
- ASCII, Dateien
 - UltraLite, importieren,498
- ASIN-Funktion
 - UltraLite-Syntax,325
- Assistent zum Erstellen einer Tabelle
 - UltraLite, Verwendungszweck,51
- Assistent zum Erstellen von Benutzern
 - UltraLite, Verwendungszweck,66
- Assistent zum Erstellen von Indizes
 - UltraLite verwenden ,59
- Assistent zum Erstellen von Publikationen
 - UltraLite, Verwendungszweck,86
- ATAN-Funktion
 - UltraLite-Syntax,326
- ATAN2-Funktion
 - UltraLite-Syntax,327
- Auf dem Gerät erstellen
 - Info,25
- Auffüllen
 - UltraLite-Datenbanken, mit ulinit erstellt,222
- Auffüllen mit Nullen
 - mit timestamp_with_time_zone_format-Option steuern,167
 - UltraLite-Erstellungsparameter date_format,148
 - UltraLite-Erstellungsparameter timestamp_format,164

- Ausdrücke
 - UltraLite SQL,277
 - UltraLite, CASE-Ausdrücke,280
 - UltraLite, IF-Ausdrücke,279
 - UltraLite, SQL-Operator-Vorrang,295
 - UltraLite-Aggregat,281
 - UltraLite-Eingabeparameter,282
 - UltraLite-Konstanten,278
 - UltraLite-Spaltennamen,278
 - UltraLite-Unterabfragen,281
- Ausdrücke in UltraLite
 - Info,277
- Äußere Referenzen
 - UltraLite-Unterabfragen,281
- Ausführen
 - UltraLite, CustDB-Anwendung,133
- Ausführungspläne
 - UltraLite, anzeigen,480
 - UltraLite, arbeiten mit,477
 - UltraLite, außer Kraft setzen,477
 - UltraLite, Indexverwendung prüfen,477
 - UltraLite, lesen,481
 - UltraLite, Text von,480
 - UltraLite, Vorgänge,481
- Auswählen
 - UltraLite, SELECT-Anweisung,457
 - UltraLiteJ, SELECT-Anweisung,457
- Auswählen, Indextyp
 - UltraLite, Info,59
- Authentifizieren von UltraLite-Benutzern
 - Info,61
- Authentifizierung
 - UltraLite-Benutzer,61
- Authentifizierungsparameter
 - UltraLite-Synchronisationsparameter,93
- Authentifizierungsstatus
 - UltraLite-Synchronisationsparameter,94
- Authentifizierungswert
 - UltraLite-Synchronisationsparameter,95
- Autocommit
 - UltraLite-Transaktionsübersicht ,486
- AUTOINCREMENT
 - UltraLite, Syntax,439
 - UltraLiteJ-Syntax,439
- AVG-Funktion
 - UltraLite-Syntax,327

B

- Basis 10, Logarithmus
 - UltraLite, LOG10-Funktion, 369
- Bearbeiten
 - UltraLite, Tabellenmethoden, 54
- Bedingungen
 - UltraLite, ALL-Bedingungen, 287
 - UltraLite, ANY, 288
 - UltraLite, BETWEEN, 288
 - UltraLite, EXISTS, 289
 - UltraLite, IN, 289
 - UltraLite-Suche, 283
- Beendigungscodes
 - Interactive SQL (dbisql), Dienstprogramm für UltraLite, 206
 - UltraLite-Dienstprogramm zum Entladen von Daten in XML (ulunload), 201
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload), 201
 - UltraLite-Dienstprogramm zur Datenbanksynchronisation (ulsync), 201
- Befehlseingaben
 - Interactive SQL-Modus, 201
- Befehlszeilen-Dienstprogramme
 - UltraLite SQL-Präprozessor (sqlpp), Syntax, 207
 - UltraLite, Informationen (ulinfo), Syntax, 213
 - UltraLite, Interactive SQL (dbisql), Syntax, 201
 - UltraLite, XML in Datenbank laden (ulload), Syntax, 222
 - UltraLite-Datenbank entladen (ulunload), Syntax, 233
 - UltraLite-Datenbank initialisieren (ulinit), Syntax, 214
 - UltraLite-Datenbank löschen (ulerase), Syntax, 212
 - UltraLite-Datenbank validieren (ulvalid), Syntax, 237
 - UltraLite-Engine (uleng16) starten, Syntax, 210
 - UltraLite-Engine stoppen (ulstop), Syntax, 211
 - UltraLite-Synchronisation (ulsync), Syntax, 227
- Begrenzungen
 - UltraLite, 9
- Beispielanwendungen
 - (*Siehe auch* Beispiele)
 - CustDB in UltraLite, 15
- Beispiele
 - (*Siehe auch* Beispielanwendung)
 - (*Siehe auch* praktische Einführungen)
- Benchmarktests
 - UltraLite SQL, testen, 489
 - UltraLite, Datenbanken, 488
 - UltraLite, Merkmale von verlässlichen Tests, 491
 - UltraLite, Methoden-Überblick, 490
 - UltraLite, Testausführung und Analyse, 492
 - UltraLite, Testerstellung, 491
 - UltraLite, Testvorbereitung, 490
 - UltraLite, Typen, 488
- Benutzer
 - für UltraLite-Datenbanken erstellen, 66
 - in UltraLite-Datenbanken verwalten, 61
 - UltraLite, DROP USER-Anweisung, 449
 - UltraLite, ALTER USER-Anweisung, 431
- Benutzer-IDs
 - maximale Länge für UltraLite, 190
 - UltraLite-Datenbanken, 61
- Benutzeranmeldung
 - PWD, UltraLite-Verbindungsparameter, 186
- Benutzerauthentifizierung
 - Authentifizierungswert, Synchronisationsparameter in UltraLite, 95
 - benutzerdefiniert, MobiLink, 93
 - Password, Synchronisationsparameter in UltraLite, 101
 - Statusberichte, UltraLite-Synchronisation, 94
 - UltraLite, 61
 - UltraLite, benutzerdefiniert für Synchronisation, 99
 - UltraLite, getUserName-Methode, 112
 - UltraLite-Synchronisation, user_name, 112
- Benutzerdefinierte Datentypen
 - UltraLite-Entsprechungen, 316
 - von UltraLite nicht unterstützt, 296
- Berechnete Spalten
 - UltraLite-Einschränkungen, 4
- Bereich
 - UltraLite-Datentyp, 296
- Bereinigen
 - UltraLite-Datenbanken, 196
- Bereinigung festschreiben
 - UltraLite, Konfiguration, 196
- Bereinigung, Anzahl
 - UltraLite-Formatierung, 196
- Bereinigung, Zeitüberschreitung
 - UltraLite-Formatierung, 197
- Bereitstellen, UltraLite-Verbindungsparameter
 - Info, 35
- Berücksichtigung der Groß-/Kleinschreibung

- UltraLite-Erstellungsparameter case,143
- Berücksichtigung von Groß- und Kleinschreibung
 - LIKE-Suchbedingung,292
 - UltraLite-Eigenschaft case,191
- Berücksichtigung von Groß- und Kleinschreibung, Hinweise
 - UltraLite, Info,143
- Beständige Speicherung
 - UltraLite-Datenbank-Speicherung,37
- BETWEEN-Suchbedingung
 - UltraLite SQL,288
- Bezeichner
 - UltraLite SQL,271
- Bibliotheken
 - UltraLite FIPS-aktivierte Anwendungen,151
 - UltraLite, Deployment von uleng auf Windows Mobile,19
 - UltraLite-Auswahl,19
- BIGINT-Datentyp
 - UltraLite,298
- Binär
 - UltraLite-Sortierung,26
- Binärdatentypen
 - UltraLite, maximale Größe,9
- BINARY-Datentypen
 - UltraLite,311
- BIT-Datentyp
 - UltraLite,299
- Bit-Operatoren
 - UltraLite SQL-Syntax,294
- BlackBerry
 - Dienstprogramme,252
 - Einschränkungen des Objektspeichers,14
 - uljdbtserv-Dienstprogramm,258
 - uljinfo-Dienstprogramm,252
 - uljload-Dienstprogramm,253
 - uljunload-Dienstprogramm,256
 - UltraLite Java Edition-Systemtabellen,263
 - Verschlüsselung und Verschleierung,29
- blob_file_base_dir-Datenbankoption
 - UltraLite Java Edition-Beschreibung,247
- blob_file_base_dir-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
- BYTE_LENGTH-Funktion
 - UltraLite-Syntax,328
- BYTE_SUBSTR-Funktion
 - UltraLite-Syntax,329

C

- C++, Anwendungen
 - (*Siehe auch* UltraLite, C/C++)
- C++, APIs
 - (*Siehe auch* UltraLite, C/C++-API)
- Cache
 - UltraLite cache_allocation-Eigenschaft,191
 - UltraLite-Anpassungen,469
- cache_allocation-Datenbankoption
 - UltraLite,195
- CACHE_MAX_SIZE, Verbindungsparameter
 - UltraLite, Syntax,170
- CACHE_MIN_SIZE, Verbindungsparameter
 - UltraLite, Syntax,171
- CACHE_SIZE, Verbindungsparameter
 - UltraLite, Syntax,172
- Cachegröße
 - UltraLite, Performance,469
 - UltraLite, Verwendung,155
- Cachezuweisung
 - UltraLite, Konfiguration,195
- Callback
 - UltraLite, Schema-Upgrade-Fehler,127
- CASE, Ausdruck
 - UltraLite, SQL-Syntax,280
- case, Eigenschaft
 - UltraLite, Beschreibung,191
- case, Erstellungsparameter
 - UltraLite, Beschreibung,143
- CASE-Ausdruck
 - UltraLite, NULLIF-Funktion,380
- CAST-Funktion
 - UltraLite-Syntax,330
- Casting
 - UltraLite, Datentypenliste,330
- CE_FILE, Verbindungsparameter
 - UltraLite, Syntax,174
- CEILING-Funktion
 - UltraLite-Syntax,332
- Certicom
 - UltraLite, TLS-aktivierte Synchronisation,117
 - UltraLite, Verschlüsselungsmodul,29
- CHAR-Datentyp
 - UltraLite,296
- CHAR-Funktion
 - UltraLite-Syntax,333
- CHAR_LENGTH-Funktion

- UltraLite-Syntax,334
- char_set, Eigenschaft
 - UltraLite, Beschreibung,191
- CHARINDEX-Funktion
 - UltraLite-Syntax,334
- CHECK CONSTRAINTS-Klausel
 - UltraLite, LOAD TABLE-Anweisung,454
- CHECK, Integritätsregeln
 - UltraLite-Einschränkungen,4
- CHECKPOINT-Anweisung
 - UltraLite, Syntax,432
- Checkpoints
 - UltraLite, Performance-Optimierung,487
- Checkpoints setzen
 - UltraLite, CHECKPOINT-Syntax,432
- CheckpointStore
 - UltraLite, Synchronisationsparameter,91
- checksum_level, Eigenschaft
 - UltraLite, Beschreibung,191
- checksum_level, Erstellungsparameter
 - UltraLite, Beschreibung,144
- Chiffrierschlüssel
 - UltraLite, ändern,150
- Client, Datenbanken
 - UltraLite-Optionen,90
- Clients
 - UltraLite MobiLink-Clients,69
- COALESCE-Funktion
 - UltraLite-Syntax,336
- Codepunkte
 - UltraLite,26
- collation-Eigenschaft
 - UltraLite, Beschreibung,191
- collation-Erstellungsparameter
 - UltraLite, Beschreibung,145
- COMMIT-Anweisung
 - UltraLite, Syntax,432
- COMMIT_FLUSH, Verbindungsparameter
 - UltraLite, Syntax,175
- commit_flush_count, Datenbankoption
 - UltraLite,196
- commit_flush_count, Eigenschaft
 - UltraLite, Beschreibung,191
- commit_flush_timeout, Datenbankoption
 - UltraLite,197
- commit_flush_timeout, Eigenschaft
 - UltraLite, Beschreibung,191
- COMPUTES-Klausel
 - UltraLite, LOAD TABLE-Anweisung,454
- CON, Verbindungsparameter
 - UltraLite, Syntax,177
- connCount-Eigenschaft
 - UltraLite-Beschreibung,191
- CONVERT-Funktion
 - UltraLite-Syntax,336
- Coordinated Universal Time
 - UltraLite, CURRENT UTC TIMESTAMP,276
- COS-Funktion
 - UltraLite-Syntax,339
- COT-Funktion
 - UltraLite-Syntax,339
- count, Vorgang
 - UltraLite, Ausführungspläne,481
- COUNT-Funktion
 - UltraLite-Syntax,340
- COUNT_UPLOAD_ROWS-Funktion
 - Syntax,341
- CPU
 - UltraLite-Begrenzungen,9
- CREATE INDEX-Anweisung
 - UltraLite, Beispiel,59
 - UltraLite, Syntax,433
 - UNIQUE-Parameter,433
- CREATE PUBLICATION-Anweisung
 - UltraLite, Syntax,435
- CREATE SYNCHRONIZATION PROFILE-Anweisung
 - UltraLite, Syntax,436
- CREATE TABLE-Anweisung
 - UltraLite, Syntax ,438
 - UltraLiteJ, Syntax ,438
- CREATE USER-Anweisung
 - UltraLite, Dynamic SQL-Syntax,444
- CROSS JOIN-Klausel
 - UltraLite, Syntax,450
- CSV, Dateien
 - UltraLite, importieren,498
- CURRENT DATE-Funktion
 - UltraLite, TODAY-Funktion,413
- CURRENT DATE-Spezialwert
 - UltraLite-Syntax,274
- CURRENT TIME-Spezialwert
 - UltraLite-Syntax,274
- CURRENT TIMESTAMP-Spezialwert
 - UltraLite-Funktionsvergleich,4
 - UltraLite-Syntax,275

- CURRENT UTC TIMESTAMP-Spezialwert
 - UltraLite-Syntax,276
- CURRENT_TIMESTAMP-Spezialwert
 - UltraLite-Syntax,275
- Cursor
 - UltraLite, aktuelle Zeile,484
 - UltraLite, Dirty Reads,44
- CustDB
 - UltraLite, Info,15
 - UltraLite-Anwendung, Readme-Dateien,17
- CustDB, UltraLite-Beispiel
 - Dateispeicherorte,17
- custdb.db
 - Speicherort,17
- custdb.sql
 - Synchronisationsskripten aufrufen,133
- custdb.udb
 - UltraLite-Speicherort,17

D

- Daemon
 - UltraLite-Engine ausführen,210
- database_name-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
- DATALENGTH-Funktion
 - UltraLite-Syntax,341
- DATE-Datentyp
 - UltraLite,306
- DATE-Funktion
 - UltraLite-Syntax,343
- date_format, Eigenschaft
 - UltraLite, Beschreibung,191
- date_format, Erstellungsparameter
 - UltraLite, Beschreibung,146
- date_format-Datenbankoption
 - UltraLite Java Edition-Beschreibung,248
- date_format-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
- date_order, Eigenschaft
 - UltraLite, Beschreibung,191
- date_order, Erstellungsparameter
 - UltraLite, Beschreibung,149
- date_order-Datenbankoption
 - UltraLite Java Edition-Beschreibung,248
- date_order-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
- DATEADD-Funktion

- UltraLite-Syntax,343
- DATEDIFF-Funktion
 - UltraLite-Syntax,344
- DATEFORMAT-Funktion
 - UltraLite-Syntax,346
- Dateien
 - mit MLFileTransfer übertragen,83
 - UltraLite, ActiveSync-Provider,128
 - UltraLite, Speicherort des CustDB-Beispiels,17
- Dateien übertragen
 - UltraLite-Dateien mit MLFileTransfer,83
- Dateigröße
 - UltraLite-Datenbank, Fehlerbehandlung,497
- Dateinamen
 - UltraLite, Verbindungsparameter,37
- Dateisysteme
 - (*Siehe auch* VFS)
- Daten
 - UltraLite, entladen,233
 - UltraLite, laden,222
 - UltraLite, Methoden anzeigen,54
 - UltraLite, Zeilen auswählen,457
 - UltraLiteJ, Zeilen auswählen,457
- Daten entfernen
 - UltraLite, Auswirkung der Dateigröße auf,497
- Daten importieren
 - UltraLite, Fehlerbehandlung,498
- Daten laden
 - UltraLite, LOAD TABLE-Anweisung,452
- Daten löschen
 - UltraLite, Auswirkung der Dateigröße auf,497
- Daten synchronisieren
 - UltraLite, Auswirkung der Dateigröße auf,497
- DATENAME-Funktion
 - UltraLite-Syntax,347
- Datenbank erstellen, Assistent
 - UltraLite,21
- Datenbank, Hinweise zur Seitengröße
 - UltraLite, Info,155
- Datenbank-Dienstprogramme
 - UltraLite-Datenverbindungen,35
- Datenbank-Validierung
 - UltraLite,46
- Datenbankdateien
 - (*Siehe auch* UltraLite-Datenbanken)
 - UltraLite, maximale Größe,9
 - UltraLite, Verbindungsparameter,37
 - UltraLite, verschlüsseln,179

- Datenbankeigenschaften
 - (*Siehe auch* Datenbankeigenschaften (UltraLite))
 - UltraLite, 191
 - UltraLite Java Edition, 245
- Datenbankeigenschaften (UltraLite)
 - durchsuchen, 39
 - Erstellungsparameter, 25
 - fips, Verwendung, 29
 - obfuscate, Verwendung, 29
 - utf8_encoding, Verwendung, 27
- Datenbanken
 - (*Siehe auch* UltraLite-Datenbanken)
 - mit UltraLite-Plug-In erstellen, 21
 - UltraLite, Massendaten einfügen, 452
 - Vergleich zwischen UltraLite und SQL Anywhere, 4
- Datenbanken erstellen
 - UltraLite-Erstellungsparameter, 141
- Datenbankgrößen
 - UltraLite-Begrenzungen, 9
- Datenbankobjekte
 - UltraLite, Kopiermethode, 55
- Datenbankoptionen
 - (*Siehe auch* Datenbankoptionen (UltraLite))
 - UltraLite, 195
 - UltraLite Java Edition, 246
- Datenbankoptionen (UltraLite)
 - cache_allocation, 195
 - commit_flush_count, 196
 - commit_flush_timeout, 197
 - DB_PROPERTY-Funktion, 351
 - durchsuchen, 40
 - global_database_id, 198
 - isolation_level, 199
 - ml_remote_id, 199
 - SET OPTION, Anweisung, 459
- Datenbankoptionen, Fenster zugreifen, 40
- Datenbankschemas
 - UltraLite-Systemtabellen, 238
- Datenbankverwaltung
 - UltraLite-Ebenen, 1
- Datenkonsistenz
 - UltraLite, Dirty Reads, 44
- Datenmanager
 - UltraLite, Datenbank-Speicherung, 37
- Datenquellen
 - UltraLite-Beispiel, praktische Einführung mit CustDB, 133
- Datenspeicherung
 - UltraLite-Begrenzungen, 9
- Datenstromfehler
 - UltraLite, Synchronisationsparameter, 105
- Datenstromparameter
 - UltraLite, Synchronisationsparameter, 108
- Datenstromsynchronisationsparameter
 - UltraLite, Synchronisationsparameter, 108
- Datenträgerfehler
 - UltraLite-Datenbanken, Übersicht, 486
- Datentypen
 - räumlich (UltraLite), 314
 - UltraLite, Aliasentsprechungen, 316
 - UltraLite, Casting von Werten, 330
 - UltraLite, Info, 296
 - UltraLite, SQL-Konvertierungsfunktionen, 317
 - UltraLite-Einschränkungen, 9
 - UltraLite-Spezialwert, 273
- Datentypen in UltraLite
 - Info, 296
- Datentypkonvertierungsfunktionen
 - UltraLite, Info, 317
- Datenverwaltung
 - UltraLite, Statusinformationen, 483
 - UltraLite-Beschreibung, 1
 - UltraLite-Komponenten, 19
- DATEPART-Funktion
 - UltraLite-Syntax, 347
- DATETIME-Datentyp
 - UltraLite, 307
- DATETIME-Funktion
 - UltraLite-Syntax, 348
- Datum/Zeit
 - UltraLite, Konvertierungsfunktionen, 317
- Datumsangaben
 - Abfrage des aktuellen Systemdatums, 357
 - UltraLite Java Edition-Eigenschaft date_format, 245
 - UltraLite Java Edition-Eigenschaft date_order, 245
 - UltraLite, Formatierung, 146
 - UltraLite, Konvertierungsfunktionen, 317
 - UltraLite, Reihenfolge, 149
 - UltraLite, Überschreitungslimit, 153
 - UltraLite, zweideutige Zeichenfolgenkonvertierung, 153
 - UltraLite-Eigenschaft date_format, 191

- UltraLite-Eigenschaft date_order,191
- Datumsangaben, Hinweise
 - UltraLite, Info,146
- Datumsfunktionen
 - UltraLite, alphabetische Liste,317
- Datumsteile
 - Info,317
 - verfügbar in UltraLite,146
- DAY-Funktion
 - UltraLite-Syntax,349
- DAYNAME-Funktion
 - UltraLite-Syntax,349
- DAYS-Funktion
 - UltraLite-Syntax,350
- DB_PROPERTY-Funktion
 - UltraLite-Syntax,351
- DBF, Verbindungsparameter
 - UltraLite, Syntax,177
- dbisql, Dienstprogramm
 - UltraLite, Beendigungscodes,206
 - UltraLite, Fehlerbehandlung, Datenimporte,498
 - UltraLite, Syntax,201
- dbisql.com
 - Info,206
- dbisql.exe
 - Info,206
 - vor Installation herunterfahren,495
- DBKEY, Verbindungsparameter
 - UltraLite, Syntax,179
- dbngen16.dll
 - ActiveSync-Conduit-Deployment in UltraLite,128
- DBN, Verbindungsparameter
 - UltraLite, Syntax,180
- DDL
 - UltraLite-Schemaänderungen,49
- Deadlocks
 - UltraLite, verhindern,486
- Deaktivieren, Parallelität
 - UltraLite-Synchronisationsparameter, Überblick,484
- DECIMAL-Datentyp
 - UltraLite,300
- DEFAULT TIMESTAMP, Spalten
 - UltraLite, Syntax,439
 - UltraLiteJ, Syntax,439
- DEFAULTS-Klausel
 - UltraLite, LOAD TABLE-Anweisung,454
- DEGREES-Funktion
 - Syntax,352
- DELETE-Anweisung
 - Syntax, Dynamic SQL für UltraLite,445
 - UltraLiteJ, Dynamic SQL-Syntax,445
- DELIMITED BY-Klausel
 - UltraLite LOAD TABLE Anweisung,454
- Demonstrationen
 - (*Siehe auch* praktische Einführungen)
- Deployment
 - Änderungen in UltraLite-Datenbankdateien,117
 - UltraLite auf Geräten,117
 - UltraLite FIPS-aktivierte Anwendungen,151
 - UltraLite, ActiveSync-Providerdateien,128
 - UltraLite-Datenbanken,125
 - UltraLite-Engine, Fehlerbehandlung,499
 - UltraLite-Schema-Upgrade,125
 - UltraLite-Upgrades auf Geräten,117
- Deployment durchführen
 - Anwendungen, die ActiveSync für UltraLite-Clients verwenden,88
- Desktop-PC
 - UltraLite-Datenbanken,38
- desktop-Verbindungsparameter
 - UltraLite, Syntax,181
- device-Verbindungsparameter
 - UltraLite, Syntax,182
- Devices
 - Deployment von UltraLite,117
- Dezimale Gesamtstellenzahl
 - UltraLite, precision-Erstellungsparameter,157
- Dezimalzahlen
 - UltraLite,273
- Dezimalzeichen, Hinweise zur Position
 - UltraLite, precision-Erstellungsparameter,157
 - UltraLite, scale-Erstellungsparameter,158
- Diagnoseprogramm
 - UltraLite, Benchmarktests,488
- Diagramm der Architektur
 - UltraLite,1
- Dienstprogramm zum Löschen der Datenbank
 - UltraLite, Syntax,212
- Dienstprogramme
 - Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload), Syntax,253
 - UltraLite,200
 - UltraLite Java Edition-Datenbankinformationen (uljinfo), Syntax,252

- UltraLite Java Edition-Datenbankübertragung (uljdbtserv) Syntax,258
 - UltraLite Java Edition-Dienstprogramm zum Entladen (uljunload), Syntax,256
 - UltraLite SQL-Präprozessor (sqlpp), Syntax,207
 - UltraLite, Fehlerbehandlung,498
 - UltraLite, Fehlercodes,201
 - UltraLite, Interactive SQL (dbisql), Syntax,201
 - UltraLite, XML in Datenbank laden (ulload), Syntax,222
 - UltraLite-Datenbank entladen (ulunload), Syntax,233
 - UltraLite-Datenbank initialisieren (ulinit), Syntax,214
 - UltraLite-Datenbank löschen (ulerase), Syntax,212
 - UltraLite-Datenbank validieren (ulvalid), Syntax,237
 - UltraLite-Engine (uleng16) starten, Syntax,210
 - UltraLite-Engine stoppen (ulstop), Syntax,211
 - UltraLite-Informationen (ulinfo), Syntax,213
 - UltraLite-Synchronisation (ulsync), Syntax,227
 - Windows Mobile-Datenbankverwaltung auf Gerät,38
 - DIFFERENCE, Differenzmenge
 - UltraLite-Syntax,353
 - Direkte Page-Scans
 - UltraLite, Info,479
 - Dirty Reads
 - UltraLite, Isolationsstufen,44
 - DisableConcurrency
 - UltraLite, Synchronisationsparameter,91
 - DISTINCT, Schlüsselwort
 - UltraLite SQL,458
 - distinct, Vorgang
 - UltraLite, Ausführungspläne,481
 - Doppel-Bindestrich
 - UltraLite, Kommentarindikator,272
 - Doppel-Schrägstrich
 - UltraLite, Kommentarindikator,272
 - DOUBLE-Datentyp
 - UltraLite,301
 - DOW-Funktion
 - UltraLite-Syntax,353
 - download only
 - UltraLite-Synchronisationsparameter,96
 - download only, Synchronisation
 - download_only, Synchronisationsparameter in UltraLite ,96
 - download_only, Synchronisationsparameter
 - UltraLite-Referenz,96
 - Downloadbestätigung senden
 - UltraLite, Synchronisationsparameter,104
 - Downloadbestätigungen
 - send_download_ack, Synchronisationsparameter in UltraLite,104
 - DROP INDEX-Anweisung
 - UltraLite, Syntax,446
 - UltraLiteJ, Syntax,446
 - DROP PUBLICATION-Anweisung
 - UltraLite, Syntax,447
 - DROP SYNCHRONIZATION PROFILE-Anweisung
 - UltraLite, Syntax,448
 - UltraLiteJ, Syntax,448
 - DROP TABLE-Anweisung
 - UltraLite, Syntax,448
 - UltraLiteJ, Syntax,448
 - DROP USER-Anweisung
 - UltraLite, Syntax,449
 - dummy, Vorgang
 - UltraLite, Ausführungspläne,481
 - Durchschnittsfunktion
 - UltraLite, AVG-Funktion,327
 - Durchsuchen
 - UltraLite, Tabelleninformationen,54
 - UltraLite-Tabellenmethoden,54
 - Dynamic SQL
 - UltraLite, arithmetische Operatoren,293
 - UltraLite, logische Operatoren,286
 - UltraLite, Operator-Vorrang,295
 - UltraLite-Bit-Operatoren,294
 - UltraLite-Zeichenfolgenoperatoren,294
- ## E
- Eigenschaften
 - DB_PROPERTY-Funktion,351
 - UltraLite durchsuchen,39
 - UltraLite Java Edition-Datenbanken,245
 - UltraLite Java Edition-Systemtabellen,269
 - UltraLite, Datenbank-Erstellungsparameter,25
 - UltraLite-Datenbanken,191
 - Eigenschaften (UltraLite)
 - DB_PROPERTY-Funktion,351
 - Eindeutige Indizes
 - UltraLite, Indexerstellung,59
 - UltraLite, Merkmale,57

-
- UltraLite, UNIQUE SQL-Parameter,433
 - Eindeutige Schlüssel
 - UltraLite, Indexerstellung,59
 - UltraLite, Merkmale,57
 - Eindeutigkeit von Primärschlüsseln erhalten
 - UltraLite-Clients in MobiLink-Systemen,70
 - Eindeutigkeits-Integritätsregel
 - UltraLite, Kopiermethode,55
 - Eindeutigkeits-Integritätsregeln
 - UltraLite, Merkmale,57
 - Eindeutigkeits-Integritätsregeln
 - UltraLite, CREATE TABLE-Anweisung,439
 - UltraLiteJ, CREATE TABLE-Anweisung,439
 - Einfügen
 - UltraLite, Daten mit LOAD TABLE-Anweisung,452
 - UltraLite, INSERT-Anweisung,452
 - UltraLite, Zeilen in Massendaten,452
 - UltraLiteJ, INSERT-Anweisung,452
 - Einführung in das UltraLite-Datenbank-Managementsystem
 - Info,1
 - Eingabeparameter
 - UltraLite, Info,282
 - Einschränkungen
 - UltraLite,9
 - UltraLite-Datentypen,296
 - ELSE
 - UltraLite, CASE-Ausdruck,280
 - UltraLite, IF-Ausdrücke,279
 - Embedded SQL
 - (*Siehe auch* UltraLite Embedded SQL)
 - UltraLite NULL-Werte,273
 - Embedded SQL, Bibliotheksfunktionen (*Siehe* UltraLite Embedded SQL, Bibliotheksfunktionen)
 - ENCODING-Klausel
 - UltraLite, LOAD TABLE-Anweisung,454
 - encryption, Eigenschaft
 - UltraLite, Beschreibung,191
 - END
 - UltraLite, CASE-Ausdruck,280
 - ENDIF
 - UltraLite, IF-Ausdrücke,279
 - Engines
 - (*Siehe auch* UltraLite-Engine)
 - Entfernen
 - UltraLite, Indizes,60
 - UltraLite-Benutzer mit SQL-Anweisungen,65
 - UltraLite-Benutzer mit Sybase Central,67
 - UltraLite-Benutzer mit Verbindungsparametern,63
 - Entfernte Datenbanken
 - UltraLite-Clients erstellen,70
 - UltraLite-Synchronisationszähler,3
 - Entfernte IDs
 - in UltraLite-Datenbanken festlegen,199
 - Entity-Relationship, Registerkarte
 - UltraLite, Verwendung,56
 - Entity-Relationship-Diagramme
 - UltraLite, Info,56
 - Entlade-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljunload)
 - Syntax,256
 - Entladen
 - UltraLite-Datenbanken,233
 - UltraLite-Datenbanken mit ulunload,233
 - Entladen, Dienstprogramm
 - Syntax,233
 - Entpackte Zeilen
 - UltraLite, Info,50
 - ER-Diagramm, Registerkarte
 - UltraLite, Info,56
 - Ereignisbenachrichtigungen
 - UltraLite, arbeiten mit,41
 - Ergebnismengen
 - UltraLite, Fehlerbehandlung, unvorhersehbare Änderungen,499
 - Ergebnismengen sortieren
 - UltraLite, Fehlerbehandlung,499
 - ERRORLEVEL, Umgebungsvariable
 - Interactive SQL-Rückgabecode für UltraLite,206
 - Erstellen
 - entfernte UltraLite-Datenbanken,70
 - Referenzdatenbanken für UltraLite,33
 - UltraLite, CREATE PUBLICATION-Anweisung,435
 - UltraLite, CREATE TABLE-Anweisung,438,439
 - UltraLite, Tabellenmethoden,53
 - UltraLite-Benutzer mit SQL-Anweisungen,65
 - UltraLite-Benutzer mit Sybase Central,66
 - UltraLite-Benutzer mit Verbindungsparametern,63
 - UltraLite-CustDB-Anwendung,133
 - UltraLite-Datenbank anhand von XML,23
 - UltraLite-Datenbanken,21
 - UltraLite-Datenbanken mit ulinit,214
 - UltraLite-Datenbanken über die Eingabeaufforderung,22

- UltraLite-Datenbanken unter Verwendung der zentralen Administration von entfernten Datenbanken,23
- UltraLite-Datenbanken, anhand MobiLink-Synchronisationsmodell,23
- UltraLite-Indizes,59
- UltraLite-Publikationen,78
- UltraLite-Publikationen, Tabellen,86
- UltraLite-Tabellenmethoden,51
- Erstellen von Datenbanken
 - entfernte UltraLite-Datenbank,70
- Erstellen, Datenbanken
 - Sybase Central, UltraLite-Plug-In,21
- Erstellen, entfernte Datenbanken
 - UltraLite-Clients,70
- Erstellen, Publikationen
 - UltraLite-Datenbanken mit MobiLink-Anwendungen,78
- Erstellen, Publikationsassistent
 - UltraLite-Zeilen publizieren,87
- Erstellen, UltraLite-Datenbanken
 - Info,21
- Erstellen, UltraLite-Tabellen
 - Info,51
- Erstellungsparameter (UltraLite)
 - case,143
 - checksum_level,144
 - collation,145
 - date_format,146
 - date_order,149
 - fips-Erstellungsparameter,150
 - Info,141
 - max_hash_size,151
 - nearest_century,153
 - obfuscate,154
 - page_size,155
 - precision ,157
 - scale,158
 - time_format,160
 - timestamp_format,162
 - timestamp_increment,164
 - utf8_encoding,167
- ESCAPES-Klausel
 - UltraLite, LOAD TABLE-Anweisung,454
- Escapesequenzen
 - UltraLite-Engine, Pfade,499
- ESQL (*Siehe* Embedded SQL) (*Siehe* UltraLite SQL)
- EXISTS-Suchbedingung

- UltraLite SQL,289
- UltraLite-Suchbedingungen,283
- Exklusiv-OR
 - UltraLite-Bit-Operator,294
- EXP-Funktion
 - UltraLite-Syntax,354
- EXPLANATION, Funktion
 - UltraLite-Syntax,355
- Exponenten
 - UltraLite,273
- Exponentialfunktion
 - UltraLite, EXP-Funktion,354
- Export-Tools
 - UltraLite, ulunload-Dienstprogramm,233
- Exportieren
 - UltraLite-Datenbanken mit ulunload,233

F

- Federal Information Processing Standards (*Siehe* FIPS)
- Fehler
 - UltraLite, Client-Fehler -764,499
 - UltraLite, Schema-Upgrade-Fehler,127
 - UltraLite, Speicherkapazitätsfehler verhindern,187
 - UltraLite, SQLE_DATABASE_ERROR,496
 - UltraLite, SQLE_DEVICE_ERROR,496
 - UltraLite, SQLE_MEMORY_ERROR,496
- Fehler-Callback
 - UltraLite, Schema-Upgrade-Fehler,127
- Fehlerbehandlung
 - globale UltraLite-ID-Nummern,71
 - Ping, UltraLite-Synchronisationsparameter,102
 - stream_error, UltraLite-Synchronisationsparameter,105
 - Sync Result, UltraLite-Synchronisationsparameter ,109
 - Synchronisationsprobleme mit Fremdschlüsselzyklen vermeiden,80
 - UltraLite, getUpload-Methode ,110
 - UltraLite, Synchronisationsparameter upload_ok,110
 - UltraLite, Wert für GLOBAL AUTOINCREMENT abrufen,72
 - UltraLite, wiederaufnehmbare Downloads implementieren,81
 - UltraLite-Datensicherungsanwendung,69
 - UltraLite-Prüfsummenfehler,144

- UltraLite-Verbindungsparameter, Vorrang,36
- Fehlerbehebung
 - UltraLite, Zeitstempel verwalten und Zeitstempelerhöhungen,162
- Fehlercodes
 - UltraLite-Dienstprogramm zum Entladen von Daten in XML (ulunload),201
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),201
 - UltraLite-Dienstprogramm zur Datenbanksynchronisation (ulsync),201
 - UltraLite-Dienstprogramme,201
- Festgeschriebene Anweisungen lesen
 - UltraLite-Isolationsstufen,43
- Festlegen, Hash-Größe
 - Info,476
- Festschreiben
 - UltraLite, COMMIT-Syntax,432
 - UltraLite, Datenbankzeilen,485
 - UltraLite, Transaktionsübersicht ,486
- file, Eigenschaft
 - UltraLite, Beschreibung,191
- filter, Vorgang
 - UltraLite, Ausführungspläne,481
- FIPS
 - UltraLite, Eigenschaft fips verwenden,29
 - UltraLite, fips-Erstellungsparameter,150
 - UltraLite, Setup und Deployment,151
 - UltraLite, TLS-aktivierte Synchronisation,117
 - UltraLite, verschlüsselte Datenbanken entwickeln,19
- fips, Datenbankeigenschaft
 - UltraLite, Entwicklungsschritte,19
 - UltraLite, Verwendung,29
- FIPS, Erstellungsparameter
 - UltraLite, Beschreibung,150
- fips, Netzwerkprotokolloption
 - UltraLite, TLS-aktivierte Synchronisation,117
- FIRST-Klausel
 - SELECT-Anweisung, für UltraLite,457
 - UltraLiteJ, SELECT-Anweisung,457
- FLOAT-Datentyp
 - UltraLite,302
- FLOOR-Funktion
 - UltraLite-Syntax,355
- FOR READ-Klausel
 - UltraLite, direkte Page-Scans,479
- FOR-Klausel
 - UltraLite, SELECT-Anweisung,458
- FORCE ORDER-Klausel
 - UltraLite, SELECT-Anweisung,459
- FORMAT-Klausel
 - UltraLite, LOAD TABLE-Anweisung,454
- Formatoptionen (UltraLite)
 - utf8_encoding, Verwendung,27
- Fortschrittszähler
 - Offset-Konflikt in UltraLite ,3
- Fremdschlüssel
 - UltraLite Java Edition-Systemtabelle,265
 - UltraLite, Fremdschlüssel,439
 - UltraLite, Kopiermethode,55
 - UltraLite, Merkmale,57
 - UltraLite, unbenannt,439
 - UltraLiteJ, Fremdschlüssel,439
 - UltraLiteJ, unbenannt,439
- Fremdschlüsselspalten
 - UltraLite Java Edition-Systemtabelle,266
- Fremdschlüsselzyklen
 - UltraLite, Info,80
 - UltraLite, Probleme,80
- Fremdserver
 - UltraLite, CREATE TABLE-Anweisung,438
- FROM-Klausel
 - UltraLite, LOAD TABLE-Anweisung,453
 - UltraLite, SELECT-Anweisung,458
 - UltraLite, Syntax,450
- Funktionen
 - Funktionstypen für UltraLite,316
 - NULL zurückgeben, wenn NULL-Argument angegeben wird,316
 - UltraLite, Datentypkonvertierung, SQL,317
 - UltraLite, Datum und Zeit,317
 - UltraLite, Einführung,316
 - UltraLite, numerische,320
 - UltraLite, System, SQL,322
 - UltraLite, Vergleichsliste,4
 - UltraLite, verschiedene,319
 - UltraLite, Zeichenfolge,321
 - UltraLite-Aggregat,317
- Funktionen, Aggregat
 - Info,317
 - UltraLite, AVG,327
 - UltraLite, COUNT,340
 - UltraLite, LIST,366
 - UltraLite, MAX,371
 - UltraLite, MIN,371

- UltraLite, SUM,409
- Funktionen, Datentypkonvertierung
 - UltraLite, CAST,330
 - UltraLite, CONVERT,336
 - UltraLite, HEXTOINT,357
 - UltraLite, Info,317
 - UltraLite, INTTOHEX,362
 - UltraLite, ISDATE,362
 - UltraLite, ISNULL,363
- Funktionen, Datum und Uhrzeit
 - SWITCHOFFSET,410
 - TODATETIMEOFFSET,412
 - UltraLite, DATE,343
 - UltraLite, DATEADD,343
 - UltraLite, DATEDIFF,344
 - UltraLite, DATEFORMAT,346
 - UltraLite, DATENAME,347
 - UltraLite, DATEPART,347
 - UltraLite, DATETIME,348
 - UltraLite, DAY,349
 - UltraLite, DAYNAME,349
 - UltraLite, DAYS,350
 - UltraLite, DOW,353
 - UltraLite, GETDATE,356
 - UltraLite, HOUR,358
 - UltraLite, HOURS,359
 - UltraLite, Info,317
 - UltraLite, MINUTE,372
 - UltraLite, MINUTES,373
 - UltraLite, MONTH,376
 - UltraLite, MONTHNAME,376
 - UltraLite, MONTHS,377
 - UltraLite, NOW,379
 - UltraLite, QUARTER,384
 - UltraLite, SECOND,392
 - UltraLite, SECONDS,392
 - UltraLite, TODAY,413
 - UltraLite, WEEKS,417
 - UltraLite, YEAR,419
 - UltraLite, YEARS,419
 - UltraLite, YMD,421
- Funktionen, numerisch
 - COUNT_UPLOAD_ROWS,341
 - DEGREES,352
 - RAND,385
 - UltraLite TRUNCNUM,414
 - UltraLite, ABS,323
 - UltraLite, ACOS,323
 - UltraLite, ASIN,325
 - UltraLite, ATAN,326
 - UltraLite, ATAN2,327
 - UltraLite, CEILING,332
 - UltraLite, COS,339
 - UltraLite, COT,339
 - UltraLite, EXP,354
 - UltraLite, FLOOR,355
 - UltraLite, Info,320
 - UltraLite, LOG,368
 - UltraLite, LOG10,369
 - UltraLite, MOD,375
 - UltraLite, PI,383
 - UltraLite, POWER,384
 - UltraLite, RADIANS,385
 - UltraLite, REMAINDER,387
 - UltraLite, ROUND,390
 - UltraLite, SIGN,395
 - UltraLite, SIN,396
 - UltraLite, SQRT,398
 - UltraLite, TAN,412
 - UltraLite, TRUNCATE,414
- Funktionen, System
 - ML_GET_SERVER_NOTIFICATION,374
 - SYNC_PROFILE_OPTION_VALUE,411
 - UltraLite, DATALENGTH,341
 - UltraLite, DB_PROPERTY,351
- Funktionen, verschiedene
 - UltraLite NEWID,379
 - UltraLite, ARGN,324
 - UltraLite, COALESCE,336
 - UltraLite, EXPLANATION,355
 - UltraLite, GREATER,357
 - UltraLite, IFNULL,360
 - UltraLite, Info,319
 - UltraLite, LESSER,366
 - UltraLite, NULLIF,380
 - UltraLiteSHORT_PLAN,394
- Funktionen, Zeichenfolge
 - UltraLite PATINDEX,381
 - UltraLite REPLACE,388
 - UltraLite STRTOUUID,407
 - UltraLite SUBSTRING,408
 - UltraLite UCASE,415
 - UltraLite UUIDTOSTR,416
 - UltraLite, ASCII,325
 - UltraLite, BYTE_LENGTH,328
 - UltraLite, BYTE_SUBSTR,329

-
- UltraLite, CHAR,333
 - UltraLite, CHAR_LENGTH,334
 - UltraLite, CHARINDEX,334
 - UltraLite, DIFFERENCE,353
 - UltraLite, Info,321
 - UltraLite, INSERTSTR,361
 - UltraLite, LCASE,363
 - UltraLite, LEFT,364
 - UltraLite, LENGTH,365
 - UltraLite, LOCATE,367
 - UltraLite, LOWER,369
 - UltraLite, LTRIM,370
 - UltraLite, REPEAT,387
 - UltraLite, REPLICATE,389
 - UltraLite, RIGHT,390
 - UltraLite, SIMILAR,395
 - UltraLite, SOUNDEX,397
 - UltraLite, SPACE,397
 - UltraLite, STR,405
 - UltraLite, STRING,406
 - UltraLite, STUFF,407
 - UltraLite, TRIM,413
 - UltraLite, UPPER,415
 - UltraLiteRTRIM,391
- G**
- Ganze Tabellen
 - UltraLite, veröffentlichen,78
 - Gepackte Zeilen
 - UltraLite, Info,50
 - Geräte
 - UltraLite, Deployment-Techniken,117
 - UltraLite, mehrere Verbindungsparameter,37
 - Gespeicherte Prozeduren
 - UltraLite-Einschränkungen,4
 - GETDATE-Funktion
 - Abfrage des aktuellen Systemdatums,357
 - UltraLite-Syntax,356
 - GetLastIdentity, Methode
 - UltraLite-Synchronisation,72
 - getScriptVersion, Methode
 - UltraLite, Beispiel,113
 - getStream, Methode
 - UltraLite, Beispiel,107
 - getUploadOK, Methode
 - UltraLite, Beispiel,110
 - Global autoincrement
 - in UltraLite einstellen,71
 - Standards in UltraLite einstellen,74
 - global autoincrement
 - Bereich ausgeschöpft in UltraLite,71
 - UltraLite-Clients in MobiLink-Systemen ,70
 - global_database_id, Eigenschaft
 - UltraLite, Beschreibung,191
 - global_database_id, Option
 - in UltraLite einstellen,71
 - UltraLite,198
 - UltraLite, CREATE TABLE-Anweisung,439
 - global_database_id-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
 - global_database_id-Option
 - UltraLiteJ, CREATE TABLE-Anweisung,439
 - Globale Datenbank-ID, Hinweise
 - UltraLite, Info,198
 - Globalen Datenbankbezeichner festlegen
 - UltraLite-Clients in MobiLink-Systemen,71
 - Globaler Datenbankbezeichner
 - in UltraLite einstellen,71
 - UltraLite, global_database_id,198
 - Globales Autoinkrement
 - UltraLite, global_database_id ,198
 - Globally Unique Identifiers (global eindeutige Bezeichner)
 - UltraLiteSQL-Syntax für die NEWID-Funktion,379
 - Globally unique identifiers (global eindeutige Bezeichner)
 - UltraLite-Clients in MobiLink-Systemen,70
 - Grafische Pläne
 - UltraLite, nicht unterstützt,480
 - GREATER-Funktion
 - UltraLite-Syntax,357
 - Groß- und Kleinschreibung berücksichtigen
 - SQL-Zeichenfolgen,272
 - UltraLite-Vergleichsoperatoren,285
 - Groß-/Kleinschreibung, Berücksichtigung
 - UltraLite, case-Erstellungsparameter ,143
 - Großbuchstaben
 - UltraLite, UPPER-Funktion,415
 - Großbuchstaben-Zeichenfolgen
 - UltraLite UCASE-Funktion,415
 - UltraLite, UPPER-Funktion,415
 - GROUP BY-Klausel
 - UltraLite, SELECT-Anweisung,458
 - group-by, Vorgang

UltraLite, Ausführungspläne,481

Grundlagen

UltraLite-Datenbankverwaltung,483

GUIDs

UltraLite SQL-Syntax für die STRTOUUID-Funktion,407

UltraLite-Clients in MobiLink-Systemen,70

UltraLiteSQL-Syntax für die NEWID-Funktion,379

UltraLiteSQL-Syntax für die UUIDTOSTR-Funktion,416

H

Hash

UltraLite, Größe konfigurieren,476

UltraLite, Hinweise zur Größe,151

UltraLite, optimale Größe,474

Hash-Methode

UltraLite, Indizes,151

UltraLite-Eigenschaft max_hash_size,191

HAVING-Klausel

UltraLite-Suchbedingungen,283

Hexadezimale Zeichenfolgen

UltraLite, Info,357

HEXTOINT-Funktion

UltraLite-Syntax,357

Hinzufügen

UltraLite in Spalten,427

UltraLite, Indizes,59

UltraLite, Spaltenmethoden,52

UltraLite-Benutzer mit SQL-Anweisungen,65

UltraLite-Benutzer mit Sybase Central,66

UltraLite-Benutzer mit Verbindungsparametern,63

Hinzufügen, Synchronisation

UltraLite-Anwendungen,81

Hinzufügen, UltraLite-Indizes

Info,59

Host

UltraLite, ULSynchronize-Argumente,108

HOURL-Funktion

UltraLite-Syntax,358

HOURS-Funktion

UltraLite-Syntax,359

I

IDENTIFIED BY Kennwort-Klausel

ALTER USER-Anweisung,431

IDs

ml_remote_id-Option,199

UltraLite, globale Datenbank-IDs,198

IF NOT EXISTS-Klausel

CREATE PUBLICATION-Anweisung [UltraLite],435

DROP INDEX-Anweisung [UltraLite] [UltraLiteJ],446

DROP PUBLICATION-Anweisung [UltraLite] [UltraLiteJ],446

DROP SYNCHRONIZATION PROFILE-Anweisung [UltraLite] [UltraLiteJ],446

DROP TABLE-Anweisung [UltraLite] [UltraLiteJ],446

UltraLite, CREATE TEXT INDEX-Anweisung,433

IF-Ausdrücke

UltraLite, SQL-Syntax,279

IFNULL-Funktion

UltraLite-Syntax,360

ignored rows

UltraLite-Synchronisationsparameter,97

ignored_rows, Synchronisationsparameter

UltraLite-Referenz,97

Importieren, Daten in Datenbanken

UltraLite, uload-Dienstprogramm,222

IN-Suchbedingung

UltraLite SQL,289

index-scan, Vorgang

UltraLite, Ausführungspläne,481

Index-Scans

UltraLite,471

Indexbasierte UltraLite-Optimierungen

UltraLite, Info,471

Indexperformance, Hinweise

UltraLite, Info,151

Indikatoren

UltraLite-SQL-Kommentare,272

Indizes

arbeiten mit, UltraLite,56

sysindex, UltraLite Java Edition-Systemtabelle,266

sysindexcolumn, UltraLite Java Edition-Systemtabelle,267

UltraLite, durchsuchten Index bestimmen,477

UltraLite, eindeutige Indexmerkmale,59

UltraLite, eindeutige Schlüsselmerkmale,59

UltraLite, Einführung,471

UltraLite, erstellen,59

-
- UltraLite, Erstellung,59
 - UltraLite, Hash-Hinweise,151
 - UltraLite, Hash-Wert,151
 - UltraLite, Kopiermethode,55
 - UltraLite, löschen,60
 - UltraLite, nicht eindeutige Indexmerkmale,59
 - UltraLite, page_size, Verwendung,155
 - UltraLite, Performancesteigerung,472
 - UltraLite, Primärschlüssel,33
 - UltraLite, sysindex-Systemtabelle,240
 - UltraLite, sysixcol-Systemtabelle,241
 - UltraLite, Typen,59
 - UltraLite, UNIQUE SQL-Parameter,433
 - UltraLite, verwenden,56,58
 - UltraLite, Verwendung umgehen,479
 - UltraLite-Einschränkungen,9
 - Initialisieren, Datenbanken
 - Sybase Central, UltraLite-Plug-In,21
 - Initialisierung
 - UltraLite-Datenbanken mit ulinit,214
 - INNER JOIN-Klausel
 - UltraLite, Syntax,450
 - Innere Referenzen
 - UltraLite-Unterabfragen,281
 - INPUT-Anweisung
 - UltraLite, Fehlerbehandlung,498
 - INSERT-Anweisung
 - UltraLite, Interactive SQL-Beispiel,55
 - UltraLite, Syntax,452
 - UltraLiteJ, Syntax,452
 - INSERTSTR-Funktion
 - UltraLite-Syntax,361
 - Installation
 - UltraLite, Fehlerbehandlung,495
 - Installieren
 - UltraLite auf Geräten,117
 - INTEGER-Datentyp
 - UltraLite,302
 - Integrität
 - UltraLite, CREATE TABLE-Anweisung,439
 - UltraLiteJ, CREATE TABLE-Anweisung,439
 - Integritätsregel
 - UltraLite, Verwendung,439
 - UltraLiteJ, Verwendung,439
 - Integritätsregeln
 - UltraLite, ALTER TABLE-Anweisung,427
 - UltraLite, referenzielle Integrität,80
 - UltraLite, umbenennen,427
 - Interactive SQL
 - Befehlszeile,206
 - UltraLite, Befehlszeile,201
 - UltraLite, Fehlerbehandlung, Datenimporte,498
 - UltraLite, Pläne anzeigen,477
 - UltraLite, Planinterpretation,481
 - UltraLite, Planvorgänge,481
 - UltraLite, Textpläne,480
 - Interactive SQL (dbisql), Dienstprogramm
 - UltraLite, Beendigungs_codes,206
 - UltraLite, Syntax,201
 - INTTOHEX-Funktion
 - UltraLite-Syntax,362
 - iPad (*Siehe* iOS)
 - iPhone (*Siehe* iOS)
 - IS
 - UltraLite, logische Operatoren,286
 - ISDATE-Funktion
 - UltraLite-Syntax,362
 - ISNULL-Funktion
 - UltraLite-Syntax,363
 - isolation_level-Option
 - UltraLite,199
 - Isolationsstufen
 - UltraLite, Dirty Reads,44
 - UltraLite, Info,43
 - UltraLite-Isolationsstufen,43
 - J**
 - join, Vorgang
 - UltraLite, Ausführungspläne,481
 - Joins
 - UltraLite, FROM-Klausel-Syntax,450
 - K**
 - Kaskadierendes Aktualisieren
 - UltraLite-Einschränkungen,4
 - Kaskadierendes Löschen
 - UltraLite-Einschränkungen,4
 - Katalog
 - UltraLite-Systemtabellen,238
 - keep partial download, Synchronisationsparameter
 - UltraLite-Referenz ,97
 - Kennwort, Verbindungsparameter
 - UltraLite, Syntax,186
 - Kennwörter
 - Kennwort, UltraLite-Verbindungsparameter,186

- maximale Länge,445
- Salt-Wert für UltraLite-Verbindungsparameter,187
- UltraLite, Datenbanken,61
- UltraLite, neuer MobiLink-Kennwortparameter,98
- UltraLite, Synchronisationsparameter
 - password,101
- Zeichensatzkonvertierung,445
- KEY JOIN-Klausel
 - UltraLite, Syntax,450
- key, Verbindungsparameter
 - UltraLite, Syntax,179
- keyset, Vorgang
 - UltraLite, Ausführungspläne,481
- Kleinbuchstaben-Zeichenfolgen
 - UltraLite, LCASE-Funktion,363
 - UltraLite, LOWER-Funktion,369
- Kodierung
 - UltraLite, utf8_encoding verwenden,27
 - UltraLite, utf8_encoding-Erstellungsparameter,167
- Kollationen
 - UltraLite, collation-Erstellungsparameter,145
 - UltraLite, nicht unterstützt,34
 - UltraLite-Eigenschaft CollationName,191
- Kollationssequenzen
 - UltraLite, ändern,26
 - UltraLite, Info,26
 - UltraLite, LIKE-Suchbedingung,291
- Kommentare
 - UltraLite-Syntax,272
- Kompatibilität
 - UltraLite SQL,285
- Komprimierte Spalten
 - UltraLite, ALTER TABLE-Anweisung,427
- Konsolidierte Datenbanken
 - Auswahl in UltraLite,3
 - UltraLite-Beispiel,133
 - UltraLite-Kompatibilität,69
- Konstante
 - UltraLite, SQL-Syntax,278
- Konstante in Ausdrücken
 - UltraLite, Info,278
- Konvertieren
 - SQL Anywhere-Datenbanken in UltraLite-Datenbanken,33
 - UltraLite, zweideutige Datumsangaben,153
 - UltraLite-Datentypen,336
- Konvertierung
 - UltraLite, CAST,330
- Konvertierungen von Datentypen zwischen
 - UltraLite, CAST,330
- Konvertierungsfunktionen
 - UltraLite, alphabetische Liste,317
- Kopieren
 - UltraLite-Tabellenmethode,54
- Kopieren, Daten
 - UltraLite-Datenbanken,54
- Kosinus-Funktion
 - UltraLite, COS-Funktion,339
- Kotangens-Funktion
 - UltraLite, COT-Funktion,339
- Kürzen
 - UltraLite-Tabellen,463
 - UltraLiteJ-Tabellen,463
- L**
- Lade-Dienstprogramm für UltraLite Java Edition-Datenbank (uljload)
 - Syntax,253
- Laden
 - UltraLite, DROP TABLE-Anweisung,452
 - UltraLite, Masseneinfügungen,452
- Laden, Datenbanken
 - UltraLite-Datenbanken mit ulload,222
- Laufzeitbibliotheken
 - UltraLite,19
- Laufzeiten
 - (*Siehe auch* UltraLite, Laufzeit)
- LCASE-Funktion
 - UltraLite-Syntax,363
- Leere Datenbanken
 - UltraLite, auffüllen nach Ausführung von ulinit,222
- Leerstellen
 - UltraLite, Dateipfaddefinitionen,495
- LEFT OUTER JOIN-Klausel
 - UltraLite, Syntax,450
- LEFT-Funktion
 - UltraLite-Syntax,364
- LENGTH-Funktion
 - UltraLite-Syntax,365
- Lesen
 - UltraLite-Tabellenzeilen,44
- Lesen, UltraLite-Zugriffspläne
 - Info,481
- LESSER-Funktion

-
- UltraLite-Syntax,366
 - libulbase.lib
 - UltraLite C++-Entwicklung,117
 - libulrsa.lib
 - UltraLite C++-Entwicklung,117
 - libulrt.lib
 - UltraLite C++-Entwicklung,117
 - like-scan, Vorgang
 - UltraLite, Ausführungspläne,481
 - LIKE-Suchbedingung
 - UltraLite SQL,290
 - UltraLite, Berücksichtigung von Groß- und Kleinschreibung,291
 - UltraLite-Kollationen,291
 - UltraLite-Musterlänge,291
 - UltraLite-Suchbedingungen,283
 - LIST-Funktion
 - UltraLite-Syntax,366
 - Listen
 - UltraLite, LIST-Funktion, Syntax,366
 - Literale
 - UltraLite-Konstante,278
 - LOAD TABLE-Anweisung
 - UltraLite, Syntax,452
 - LOCATE-Funktion
 - UltraLite-Syntax,367
 - LOG-Funktion
 - UltraLite-Syntax,368
 - LOG10-Funktion
 - UltraLite-Syntax,369
 - Logik
 - UltraLite, für Synchronisationsplanung erfassen,74
 - Logische Operatoren
 - UltraLite, SQL-Syntax,286
 - lojoin, Vorgang
 - UltraLite, Ausführungspläne,481
 - LONG BINARY-Datentyp
 - UltraLite,312
 - LONG VARCHAR-Datentyp
 - UltraLite,297
 - Löschen
 - UltraLite , DROP USER-Anweisung,449
 - UltraLite SQL, CREATE INDEX-Anweisung,433
 - UltraLite SQL, DROP INDEX-Anweisung,446
 - UltraLite SQL, DROP PUBLICATION-Anweisung,447
 - UltraLite SQL, DROP TABLE-Anweisung,448
 - UltraLite, DROP SYNCHRONIZATION PROFILE-Anweisung,448
 - UltraLite, Indizes,60
 - UltraLite, Publikationen,87
 - UltraLite, Spalten,427
 - UltraLite, START SYNCHRONIZATION DELETE-Anweisung,460
 - UltraLite, Tabellenmethoden,53
 - UltraLite, TRUNCATE TABLE-Anweisung,463
 - UltraLite-Benutzer mit SQL-Anweisungen,65
 - UltraLite-Benutzer mit Sybase Central,67
 - UltraLite-Benutzer mit Verbindungsparametern,63
 - UltraLite-Datenbanken,485
 - UltraLite-Spalten,427
 - UltraLiteJ SQL, DROP INDEX-Anweisung,446
 - UltraLiteJ SQL, DROP TABLE-Anweisung,448
 - UltraLiteJ, DROP SYNCHRONIZATION PROFILE-Anweisung,448
 - UltraLiteJ, START SYNCHRONIZATION DELETE-Anweisung,460
 - UltraLiteJ, TRUNCATE TABLE-Anweisung,463
 - Löschen, Index
 - UltraLite, Info,60
 - Löschen, Publikationen
 - UltraLite-Clients,87
 - Löschen, UltraLite-Tabellen
 - Info,53
 - LOWER-Funktion
 - UltraLite-Syntax,369
 - LTRIM-Funktion
 - UltraLite-Syntax,370
- ## M
- Managementtools
 - UltraLite-Dienstprogramme,200
 - Massenimport von Daten
 - UltraLite, LOAD TABLE-Anweisung,452
 - Mathematische Ausdrücke
 - UltraLite, arithmetische Operatoren,293
 - MAX-Funktion
 - UltraLite-Syntax,371
 - max_hash_size, Eigenschaft
 - UltraLite, Beschreibung,191
 - max_hash_size, Erstellungsparameter
 - UltraLite, Beschreibung,151
 - Maximale Länge
 - UltraLite-Benutzer-ID,190

- Maximum
 - UltraLite, Datumsbereiche,296
 - UltraLite, physische Begrenzungen,9
 - Mehrere Datenbanken
 - UltraLite, maximale Anzahl,484
 - Mehrere Geräte
 - UltraLite-Verbindungsparameter,37
 - Mehrere Threads
 - UltraLite-Anwendungen,484
 - Mehrtabellen-Joins
 - UltraLite-Datenbanken,4
 - Metadaten
 - UltraLite, für Reservierungsgröße berücksichtigen,187
 - MIN-Funktion
 - UltraLite-Syntax,371
 - Minimum
 - UltraLite-Datumsbereiche,296
 - MINUTE-Funktion
 - UltraLite-Syntax,372
 - MINUTES-Funktion
 - UltraLite-Syntax,373
 - MIRROR_FILE, Verbindungsparameter
 - UltraLite-Syntax,184
 - ml_add_connection_script-Systemprozedur
 - hinzufügen,133
 - ml_add_table_script-Systemprozedur
 - hinzufügen,133
 - ML_GET_SERVER_NOTIFICATION
 - Syntax,374
 - ml_remote_id, Eigenschaft
 - UltraLite, Beschreibung,191
 - ml_remote_id, Option
 - UltraLite,199
 - ml_remote_id-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
 - ml_remote_id-Option
 - UltraLite-Eigenschaft, Konfiguration,213
 - mlasdesk.dll
 - Deployment von UltraLite-Anwendungen durchführen,128
 - mlasdev.dll
 - Deployment von UltraLite-Anwendungen durchführen,128
 - mlasinst, Dienstprogramm
 - UltraLite, Deployment mit DLL-Dateien,128
 - mlasinst-Dienstprogramm
 - Anwendungen für UltraLite-Clients registrieren,130
 - MobiLink
 - UltraLite, Eindeutigkeit der Benutzer-ID,199
 - UltraLite, SQL-Anweisungen,421
 - UltraLite-Clients,69
 - UltraLite, CREATE PUBLICATION-Anweisung,435
 - MobiLink, Synchronisation
 - timestamp_increment-Erstellungsparameter setzen,165
 - MobiLink-ActiveSync-Provider-Installationsprogramm (mlasinst)
 - Anwendungen für UltraLite-Clients registrieren,130
 - MobiLink-Clients
 - UltraLite-Fortschrittszähler,3
 - MobiLink-Dateiübertragung
 - UltraLite-Client, Übersicht,83
 - MobiLink-Projekt
 - UltraLite-Beispiel,133
 - MobiLink-Synchronisation
 - UltraLite-Clients,69
 - MOD-Funktion
 - UltraLite-Syntax,375
 - Modellierung
 - UltraLite-Datenbanken in MobiLink,23
 - MONEY, Datentyp
 - UltraLite-Entsprechung,316
 - MONTH-Funktion
 - UltraLite-Syntax,376
 - MONTHNAME-Funktion
 - UltraLite-Syntax,376
 - MONTHS-Funktion
 - UltraLite-Syntax,377
 - Musterlänge
 - UltraLite LIKE-Suchbedingung,291
 - Musterübereinstimmung
 - UltraLite LIKE-Suchbedingung,290
 - UltraLite PATINDEX-Funktion,381
 - UltraLite, Berücksichtigung von Groß- und Kleinschreibung,291
 - UltraLite-Kollationen,291
 - UltraLite-Musterlänge,291
- ## N
- Nächstes Jahrhundert, Hinweise zur Konvertierung

- UltraLite, Info,153
- name, Eigenschaft
 - UltraLite, Beschreibung,191
- Namen
 - UltraLite-Spaltennamen,278
- NATURAL JOIN-Klausel
 - UltraLite, Syntax,450
- nearest_century, Eigenschaft
 - UltraLite, Beschreibung,191
- nearest_century, Erstellungsparameter
 - UltraLite, Beschreibung,153
- nearest_century-Datenbankoption
 - UltraLite Java Edition-Beschreibung,249
- nearest_century-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
- Netzwerkprotokolle
 - UltraLite, Sync Result-Synchronisationsparameter,109
 - UltraLite-Synchronisation mit HTTP,107
 - UltraLite-Synchronisation mit HTTPS,107
 - UltraLite-Synchronisation mit TCP/IP,107
 - von UltraLite unterstützt,3
- Neu startbare Downloads
 - keep partial download in UltraLite,97
 - partial download retained in UltraLite,100
 - resume partial download in UltraLite,104
- Neue Publikation erstellen, Assistent
 - UltraLite, Publikationen erstellen,78
- new password
 - UltraLite-Synchronisationsparameter,98
- NEWID-Funktion
 - UltraLite-Syntax,379
- NewMobiLinkPwd, Synchronisationsparameter
 - UltraLite, Referenz,98
- Nicht eindeutige Indizes
 - UltraLite, Indexerstellung,59
- Nicht festgeschriebene Anweisungen lesen
 - UltraLite-Isolationsstufen,43
- Nicht festgeschriebene Transaktionen
 - UltraLite, Isolationsstufe,44
 - UltraLite, Übersicht,486
- Nicht wiederholbare Lesevorgänge
 - UltraLite, Info,44
- Nicht-eindeutige Indizes
 - UltraLite, Merkmale,57
- nosync, Tabellen
 - nicht synchronisierte Tabellen in UltraLite ,76
- NOT

- UltraLite, logische Operatoren,286
- UltraLite-Bit-Operator,294
- NOW-Funktion
 - UltraLite-Syntax,379
- NT_FILE, Verbindungsparameter
 - UltraLite, Syntax ,185
- NULL
 - UltraLite SQL,273
 - UltraLite, ISNULL-Funktion,363
 - zurückgegeben von Funktionen, wenn ein NULL-Argument angegeben wird,316
- NULLIF-Funktion
 - UltraLite mit CASE-Ausdrücken verwenden,280
 - UltraLite, Info,380
- number of authentication, Parameter
 - UltraLite-Synchronisationsparameter,99
- NUMERIC-Datentyp
 - UltraLite,303
- Nummerische Funktionen
 - UltraLite, alphabetische Liste,320
- Nummerische Gesamtstellenzahl
 - UltraLite, precision-Erstellungsparameter,157

O

- obfuscate, Datenbank-Erstellungsparameter
 - UltraLite, Beschreibung,154
- obfuscate, Datenbankeigenschaft
 - UltraLite, Verwendung,29
- Observer, Synchronisationsparameter
 - UltraLite, Beschreibung,99
- Öffentliche Zertifikat
 - UltraLite, Anwendungszugriff auf,229
- ON-Formulierung
 - UltraLite-Suchbedingungen,283
- Operatoren
 - UltraLite, arithmetische Operatoren,293
 - UltraLite, logische Operatoren,286
 - UltraLite, Operatoren-Vorrang,295
 - UltraLite, SQL-Syntax,293
 - UltraLite-Bit-Operatoren,294
 - UltraLite-Vergleichsoperatoren,285
 - UltraLite-Zeichenfolgenoperatoren,294
- Operatorvorrang
 - UltraLite SQL-Syntax,295
- Optimierer
 - (Siehe auch Abfrageoptimierer)
 - UltraLite, außer Kraft setzen,477

- UltraLite, Ausführungsplan-Zugriffsoptionen,477
- UltraLite, Auswirkungen,477
- UltraLite, Planinterpretation,481
- UltraLite, Planvorgänge,481
- UltraLite, Verwendung,477
- Optimierung
 - UltraLite,488
 - UltraLite SQL,477
 - UltraLite, Abfragen,458
 - UltraLite, Ausführungsplan-Zugriffsoptionen,477
 - UltraLite, Checkpoints,487
 - UltraLite, Indizes,474
- Optimierung der Performance
 - UltraLite, Index-Hashing,472
 - UltraLite, max_hash_size,151
- Optimierung, UltraLite
 - Abfrageperformance,469
- Option
 - UltraLite-Dienstprogramm zum Validieren von Datenbanken (ulvalid),237
- Optionen
 - UltraLite durchsuchen,40
 - UltraLite Java Edition,246
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202
 - UltraLite-Dienstprogramm für den SQL Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - UltraLite-Dienstprogramm zum Löschen der Datenbank (ulerase),212
 - UltraLite-Informationsdienstprogramm (ulinfo),213
 - UltraLite-Synchronisationsdienstprogramm (ulsync),227
 - UltraLite-Synchronisationsprofile,230
- Optionen (UltraLite)
 - cache_allocation,195
 - commit_flush_count,196
 - commit_flush_timeout,197
 - DB_PROPERTY-Funktion,351
 - global_database_id,198
 - isolation_level,199
 - ml_remote_id,199

- SET OPTION-Anweisung,459
- OR
 - UltraLite, logische Operatoren,286
 - UltraLite-Bit-Operatoren,294
- Oracle, konsolidierte Datenbanken
 - UltraLite-Probleme mit,69
- ORDER BY-Klausel
 - UltraLite, Fehlerbehandlung mit,499
 - UltraLite, SELECT-Anweisung,458
- Overhead
 - UltraLite, für Reservierungsgröße berücksichtigen,187

P

- Packen, Zeilen
 - UltraLite, Auswirkungen,156
- page_size, Eigenschaft
 - UltraLite, Beschreibung,191
- page_size, Erstellungsparameter
 - UltraLite, Beschreibung,155
- page_size-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
- Parallelität
 - UltraLite-Probleme,484
 - UltraLite-Synchronisation,484
- Parameter
 - UltraLite, Interactive SQL (dbisql), Dienstprogramm,202
 - UltraLite, SQL-Eingabe,282
 - UltraLite, Verbindungen, Liste,168
 - UltraLite, Verbindungsüberblick,35
 - UltraLite-Datenbankerstellung,141
 - UltraLite-Dienstprogramm für den SQL Präprozessor (sqlpp),207
 - UltraLite-Dienstprogramm zum Entladen von Datenbanken (ulunload),233
 - UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
 - UltraLite-Dienstprogramm zum Laden der XML-Datei in die Datenbank (ulload),223
 - UltraLite-Dienstprogramm zum Löschen der Datenbank (ulerase),212
 - UltraLite-Dienstprogramm zum Validieren von Datenbanken (ulvalid),237
 - UltraLite-Informationsdienstprogramm (ulinfo),213

-
- UltraLite-Synchronisationsdienstprogramm (ulsync),227
 - Parameter des Synchronisationsdatenstroms
 - UltraLite, stream type ,107
 - partial download retained, Synchronisationsparameter
 - UltraLite, Referenz,100
 - Partitionieren
 - UltraLite-Zeilen publizieren,87
 - Partitionsgrößen
 - ausgeschöpft in UltraLite,71
 - UltraLite-Standard auswählen,71
 - UltraLite-Standards aufheben,73
 - password
 - UltraLite, Synchronisationsparameter,101
 - PATINDEX-Funktion
 - UltraLite-Syntax,381
 - PC, erstellen
 - UltraLite-Info,21
 - Performance
 - Festschreibungen von Checkpoints trennen,487
 - reine Download-Synchronisation, UltraLite,96
 - reine Upload-Synchronisation, UltraLite,110
 - UltraLite, Abfrageoptimierung,458
 - UltraLite, Abfrageoptimierung durch Index-Hashing,472
 - UltraLite, Abfrageoptimierungsverfahren,469
 - UltraLite, Ausführungsplan anzeigen,480
 - UltraLite, CACHE_MAX_SIZE-Verbindungsparameter,170
 - UltraLite, CACHE_MIN_SIZE-Verbindungsparameter,171
 - UltraLite, CACHE_SIZE-Verbindungsparameter,172
 - UltraLite, FOR READ ONLY-Klausel festlegen,477
 - UltraLite, Index für Anwendung verwenden,33
 - UltraLite, Index für große Tabellen verwenden,57
 - UltraLite, Index verwenden,58
 - UltraLite, Index zur Verbesserung der Abfrageperformance verwenden,434
 - UltraLite, Index-Hash,151
 - UltraLite, optimale Hash-Größe auswählen,474
 - UltraLite, Seitengrößen,155
 - UltraLite, Speicherausfall verhindern,187
 - Pfade
 - UltraLite, Verbindungsparameter,37
 - UltraLite-Engine,495
 - Phantomzeilen
 - UltraLite,44
 - Physische Einschränkungen
 - UltraLite,9
 - PI-Funktion
 - UltraLite-Syntax,383
 - ping
 - UltraLite, Synchronisationsparameter,102
 - Pläne
 - UltraLite, Cursor,355
 - UltraLite, Planinterpretation,481
 - UltraLite, Planvorgänge,481
 - UltraLite-Abfragen verwenden,477
 - UltraLite-Abfragen, außer Kraft setzen,477
 - UltraLite-Abfragen, lesen,481
 - UltraLite-Syntax,355
 - UltraLite-Textpläne,480
 - UltraLite-Vorgänge,481
 - Planung
 - UltraLite-Synchronisationsplanung, Überblick,74
 - Plattformen
 - UltraLite, Dateispeicherung,37
 - UltraLite, mehrere Verbindungsparameter,37
 - Platzhalter
 - UltraLite PATINDEX-Funktion,381
 - UltraLite, SQL-Eingabeparameter,282
 - Platzhalterzeichen
 - UltraLite LIKE-Suchbedingung,290
 - Plug-Ins
 - UltraLite, Fehlerbehandlung,495
 - Pools
 - ungenutzte Global IDs in UltraLite,71
 - Portnummer
 - UltraLite, ULSynchronize-Argumente,108
 - POWER-Funktion
 - UltraLite-Syntax,384
 - Prädikate
 - UltraLite IN,289
 - UltraLite LIKE-Suchbedingung,290
 - UltraLite SQL,283
 - UltraLite, ALL,287
 - UltraLite, ANY,288
 - UltraLite, BETWEEN,288
 - UltraLite, EXISTS,289
 - UltraLite-Vergleichsoperatoren,285
 - Präfix
 - UltraLite für Windows Mobile-Datenbanken ,38
 - Praktische Einführungen
 - UltraLite CustDB,133
-

- precision, Eigenschaft
 - UltraLite, Beschreibung,191
 - precision, Erstellungsparameter
 - UltraLite, Beschreibung,157
 - precision-Datenbankoption
 - UltraLite Java Edition-Beschreibung,249
 - precision-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
 - Primärschlüssel
 - UltraLite, eindeutige Werte generieren,379
 - UltraLite, eindeutige Werte mit UUIDs generieren,379
 - UltraLite, Fehlerbehandlung, Datenimporte,498
 - UltraLite, Indizierung,33
 - UltraLite, Integritätsregeln,439
 - UltraLite, Merkmale,57
 - UltraLite, Spaltenfolge,439
 - UltraLite, Tabellen,51
 - UltraLite, Tabellenfolge,80
 - UltraLiteJ, Integritätsregeln,439
 - UltraLiteJ, Spaltenfolge,439
 - UltraLiteUUIDs und GUIDs,379
 - Primärschlüsselindizes
 - UltraLite, Verwendung umgehen,479
 - Protokollieren
 - UltraLite-Synchronisation,3
 - Protokollierung
 - UltraLite, interner Mechanismus,47
 - Provider
 - UltraLite, ActiveSync-Dateien,128
 - Providerdateien
 - UltraLite, Deployment von ActiveSync,128
 - Prozeduren
 - UltraLite-Einschränkungen,4
 - Prozesse beenden
 - UltraLite, Fehlerbehandlung von Upgrades,495
 - Prozessintegrierte Laufzeitbibliothek
 - UltraLite-Info,19
 - Prüfsummen
 - UltraLite-Eigenschaft checksum_level,191
 - UltraLite-Erstellungsparameter checksum_level,144
 - publications
 - UltraLite, Synchronisationsparameter,103
 - Publikationen
 - publication, Synchronisationsparameter in UltraLite,103
 - Synchronisation in UltraLite,103
 - sysarticles, UltraLite Java Edition-Systemtabelle,263
 - syspublications, UltraLite Java Edition-Systemtabelle,268
 - UltraLite Java Edition, Tabellenliste in Schema,263
 - UltraLite Java Edition-Datenbankschema, Beschreibung,268
 - UltraLite planen mit,74
 - UltraLite SQL, CREATE INDEX-Anweisung,433
 - UltraLite SQL, DROP INDEX-Anweisung,446
 - UltraLite SQL, DROP PUBLICATION-Anweisung,447
 - UltraLite SQL, DROP TABLE-Anweisung,448
 - UltraLite, ALTER PUBLICATION-Anweisung,424
 - UltraLite, arbeiten mit,85
 - UltraLite, CREATE PUBLICATION-Anweisung,435
 - UltraLite, Kopiermethode,55
 - UltraLite, löschen ,87
 - UltraLite, sysarticle-Systemtabelle,239
 - UltraLite, syspublication-Systemtabelle,242
 - UltraLite, Tabellen publizieren,86
 - UltraLite, Überblick über die Veröffentlichung,78
 - UltraLite, WHERE-Klausel verwenden,87
 - UltraLite-Beschränkung,214
 - UltraLite-Zeilen publizieren,87
 - UltraLiteJ SQL, DROP INDEX-Anweisung,446
 - UltraLiteJ SQL, DROP TABLE-Anweisung,448
 - Publizieren
 - UltraLite, ganze Tabelle,78
 - UltraLite-Tabellen,86
 - UltraLite-Zeilen,87
 - Publizieren, ganze Tabellen
 - UltraLite,86
- ## Q
- Quadratwurzelfunktion
 - UltraLite, SQRT-Funktion,398
 - QUARTER-Funktion
 - UltraLite-Syntax,384
 - QUOTES-Klausel
 - UltraLite, LOAD TABLE-Anweisung,454
- ## R
- RADIANS-Funktion

- UltraLite-Syntax,385
- RAND-Funktion
 - Syntax,385
 - UltraLiteJ-Syntax,385
- Räumliche Daten
 - Einführung (UltraLite),313
 - Info (UltraLite),313
 - Liste der unterstützten Funktionen (UltraLite),314
- Räumliche Datentypen
 - ST_Geometry (UltraLite),314
- Räumliche Funktionen
 - ST_ AsBinary, UltraLite,398
 - ST_ AsExtText, UltraLite,399
 - ST_ AsText, UltraLite,399
 - ST_ Distance, UltraLite,400
 - ST_ IntersectsRect, UltraLite,401
 - ST_ PointFromText, UltraLite,403
 - ST_ PointFromWKB, UltraLite,403
 - ST_ SRID, UltraLite,404
 - ST_ X, UltraLite,404
 - ST_ Y, UltraLite,405
 - ST_Equals, UltraLite,400
 - ST_Point, UltraLite,402
 - ST_PointFromExtText, UltraLite,402
- Readme-Dateien
 - UltraLite, CustDB-Anwendungen,17
- REAL-Datentyp
 - UltraLite,304
- Referenzdatenbanken
 - für UltraLite erstellen,33
 - Info,33
 - Optionen für UltraLite,33
- Referenzielle Integrität
 - UltraLite, Datenbanken,4
 - UltraLite, Indizes,58
 - UltraLite, Tabellenfolge,80
- Registrieren
 - MobiLink UltraLite-Anwendungen mit ActiveSync,130
- Reihenfolge der Vorgänge
 - UltraLite, SQL-Operator-Vorrang,295
- Reine Download-Synchronisation
 - UltraLite, Übersicht über die Definition,81
 - UltraLite-Synchronisationsparameter,96
- Reine Upload-Synchronisation
 - UltraLite-Datenbanken,110
 - upload_only, Synchronisationsparameter in UltraLite,110
- reload.sql
 - UltraLite, laden,452
- REMAINDER-Funktion
 - UltraLite-Syntax,387
- REPEAT-Funktion
 - UltraLite-Syntax,387
- REPLACE-Funktion
 - UltraLite-Syntax,388
- REPLICATE-Funktion
 - UltraLite-Syntax,389
- RESERVE_SIZE, Verbindungsparameter
 - UltraLite, Syntax,187
- Reservierte Wörter
 - UltraLite SQL,271
- resume partial download, Synchronisationsparameter
 - UltraLite-Referenz,104
- RIGHT OUTER JOIN-Klausel
 - UltraLite, Syntax,450
- RIGHT-Funktion
 - UltraLite-Syntax,390
- ROLLBACK-Anweisung
 - UltraLite, Syntax,457
 - UltraLiteJ, Syntax,457
- Rollennamen
 - UltraLite-Fremdschlüssel,439
 - UltraLite-Rollennamen,439
 - UltraLiteJ, Fremdschlüssel,439
 - UltraLiteJ, Rollennamen,439
- ROUND-Funktion
 - UltraLite-Syntax,390
- rowlimit, Vorgang
 - UltraLite, Ausführungspläne,481
- RSA, Verschlüsselungsalgorithmus
 - UltraLite, TLS-aktivierte Synchronisation,117
- RTRIM-Funktion
 - UltraLite-Syntax,391
- Rückgabecodes
 - Interactive SQL (dbisql), Dienstprogramm für UltraLite,206
- Rückgängig machen
 - UltraLite-Transaktionen,457
 - UltraLiteJ-Transaktionen,457
- Runden
 - UltraLite, Dezimalstellen,158
 - UltraLite, scale-Erstellungsparameter ,158

S

- Salt-Wert für Kennwort
 - UltraLite, Syntax,187
- scale, Eigenschaft
 - UltraLite, Beschreibung,191
- scale, Erstellungsparameter
 - UltraLite, Beschreibung,158
- scale-Datenbankoption
 - UltraLite Java Edition-Beschreibung,250
- scale-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
- scan, Vorgang
 - UltraLite, Ausführungspläne,481
- Scans
 - UltraLite, Datenbankseiten in Abfragen,479
 - UltraLite, Indizes in Abfragen,471
- Schema
 - UltraLite-Änderungen, Sicherheitsvorkehrungen,49
 - UltraLite-Tabellenkatalog,49
- Schemas
 - UltraLite, SQL ALTER DATABASE SCHEMA FROM FILE, Syntax,423
- Schemata
 - Arbeit mit UltraLite,49
 - UltraLite-Systemtabellen,238
- Schlüssel
 - UltraLite, Index-Hash,151
 - UltraLite, Indexerstellung,59
 - UltraLite, primär,51
- Schlüsselwörter
 - UltraLite SQL,271
- Schrägstrich-Stern
 - UltraLite, Kommentarindikator,272
- Schreibgeschützte Tabellen
 - UltraLite, synchronisieren,81
 - UltraLite-Datenbanken,81
- scjview
 - vor Installation beenden,495
- SECOND-Funktion
 - UltraLite-Syntax,392
- SECONDS, Sekunden
 - UltraLite-Syntax,392
- Seiten
 - UltraLite, Hinweise zur Größe,155
- Seitengrößen
 - UltraLite Java Edition-Eigenschaft page_size,245
 - UltraLite-Eigenschaft page_size,191
- SELECT-Anweisung
 - UltraLite, Ergebnismengen kopieren,55
 - UltraLite, Fehlerbehandlung von Ergebnismengen,499
 - UltraLite, Syntax,457
 - UltraLiteJ, Syntax,457
- send_download_ack, Synchronisationsparameter
 - UltraLite-Referenz,104
- Serialisierte Transaktionen
 - UltraLite-Verarbeitung,486
- SET OPTION-Anweisung
 - UltraLite, Syntax,459
- setObserver, Methode
 - UltraLite, Beispiel,99
- setScriptVersion, Methode
 - UltraLite, Beispiel,113
- setStream, Methode
 - UltraLite, Beispiel,107
- setStreamParms, Methode
 - UltraLite, Beispiel,108
- setUserData, Methode
 - UltraLite, Beispiel,111
- SHORT_PLAN-Funktion
 - UltraLite-Syntax,394
- Sicherheit
 - UltraLite,29
 - UltraLite, AES_FIPS-Datenbankdeployment,19
 - UltraLite, FIPS-Verschlüsselung,150
 - UltraLite, TLS-aktivierte Synchronisation,117
 - UltraLite, Überblick,29
- Sicherung
 - UltraLite-Info,47
- Sicherung durchführen (*Siehe* Sicherungen)
- Sicherungen
 - UltraLite, interner Mechanismus,47
 - UltraLite-Datenbanken unter Windows Mobile,38
 - UltraLite-Transaktionsübersicht ,486
- SIGN-Funktion
 - UltraLite-Syntax,395
- SIMILAR-Funktion
 - UltraLite-Syntax,395
- SIN-Funktion
 - UltraLite-Syntax,396
- SKIP-Klausel
 - UltraLite, LOAD TABLE-Anweisung,455
- Skriptgesteuerte Uploads
 - UltraLite, CREATE PUBLICATION-Syntax,435
- Skriptversionen

-
- setScriptVersion-Methode in UltraLite ,113
 - UltraLite, getScriptVersion,113
 - Version, Synchronisationsparameter in UltraLite ,113
 - SMALLINT-Datentyp
 - UltraLite,304
 - SMALLMONEY, Datentyp
 - UltraLite-Entsprechung,316
 - Sortierreihenfolgen
 - UltraLite-Kollationen,28
 - SOUNDEX-Funktion
 - UltraLite-Syntax,397
 - SPACE-Funktion
 - UltraLite-Syntax,397
 - Spalten
 - syscolumn, UltraLite Java Edition-Systemtabelle,264
 - UltraLite, Aliase,458
 - UltraLite, ALTER TABLE-Anweisung,427
 - UltraLite, Kopiermethode,55
 - UltraLite, Methoden hinzufügen,52
 - UltraLite, Verwendung ändern,52
 - UltraLite-Einschränkungen,9
 - Spalten mit dem Standardwert global autoincrement deklarieren
 - UltraLite-Clients in MobiLink-Systemen,74
 - Spaltenkomprimierung
 - UltraLite, SQL ALTER TABLE-Anweisung,427
 - Spaltennamen
 - UltraLite, SQL-Syntax,278
 - Spaltennamen in Ausdrücken
 - UltraLite, Info,278
 - Speicher
 - UltraLite-Begrenzungen,9
 - Speicherausfall
 - UltraLite, verhindern,187
 - Speicherbedarf
 - UltraLite-Datenbanken,4
 - Speichernutzung
 - UltraLite SQL, Performance,489
 - UltraLite-Datenbank-Speicherung,37
 - UltraLite-Indizes,471
 - UltraLite-Zeilenzustand,485
 - Speicherung
 - UltraLite, Reservierungsgröße,187
 - Sperre
 - UltraLite, Parallelität,486
 - Spezialwerte
 - UltraLite SQL,273
 - UltraLite, CURRENT DATE,274
 - UltraLite, CURRENT TIME,274
 - UltraLite, CURRENT TIMESTAMP,275
 - UltraLite, CURRENT UTC TIMESTAMP,276
 - UltraLite, SQLCODE,276
 - SQL
 - (*Siehe auch* UltraLite SQL)
 - Datentypen in UltraLite,296
 - UltraLite, reservierte Wörter,271
 - UltraLite, Schemaänderungen,49
 - UltraLite, Zahlen,273
 - UltraLite-Anweisungstypen,422
 - UltraLite-Ausdrücke,277
 - UltraLite-Bezeichner ,271
 - UltraLite-Operatoren,293
 - UltraLite-Schlüsselwörter ,271
 - UltraLite-Suchbedingungen,283
 - UltraLite-Variablen,296
 - UltraLite-Vergleichsoperatoren,285
 - UltraLite-Zeichenfolgen,272
 - SQL Anywhere-Datenbanken
 - Datenbankvergleich mit UltraLite,4
 - SQL Anywhere-Datenbanken konvertieren
 - Info,33
 - SQL Flagger
 - UltraLite, Verwendung,207
 - SQL Syntax
 - UltraLite-Funktionen,316
 - SQL-Anweisungen
 - alphabetische Liste der UltraLite-Anweisungen,421
 - UltraLite,421
 - SQL-API für räumliche Daten
 - ST_Geometry-Datentyp für UltraLite,314
 - SQL-Code
 - UltraLite, Schema-Upgrade-Fehler,127
 - SQL-Funktionen
 - Funktionstypen für UltraLite,316
 - NULL zurückgeben, wenn NULL-Argument angegeben wird,316
 - UltraLite, Datentypkonvertierung,317
 - UltraLite, Datum und Zeit,317
 - UltraLite, Einführung,316
 - UltraLite, numerische,320
 - UltraLite, System,322
 - UltraLite, verschiedene,319
 - UltraLite, Zeichenfolge,321

- UltraLite-Aggregat,317
- SQL-Präprozessor-Dienstprogramm (sqlpp)
 - UltraLite, Syntax ,207
- SQL-Syntax
 - UltraLite, CASE-Ausdruck,280
 - UltraLite, IF-Ausdrücke ,279
 - UltraLite, Spaltennamen,278
 - UltraLite, Spezialwerte,273
 - UltraLite, SQLCODE-Spezialwert,276
 - UltraLite-Eingabeparameter,282
 - UltraLite-Kommentare,272
 - UltraLite-Konstante,278
 - UltraLite-Prädikate,283
- SQLCODE
 - UltraLite, Prüfung der Parallelität,486
 - UltraLite-Spezialwert,276
- SQLCODE SQLE_LOCKED
 - UltraLite, Fehler bei Parallelität,486
- SQLE_CONVERSION_ERROR
 - UltraLite, Upgrade-Warnung,127
- SQLE_DATABASE_ERROR
 - UltraLite, Datenbeschädigung,496
- SQLE_DEVICE_ERROR
 - UltraLite, Datenbeschädigung,496
- SQLE_DOWNLOAD_CONFLICT, Fehler
 - UltraLite synchronisieren,81
- SQLE_MAX_ROW_SIZE_EXCEEDED
 - UltraLite-Fehler,50
- SQLE_MEMORY_ERROR
 - UltraLite, Datenbeschädigung,496
- SQLE_NOTFOUND
 - UltraLite, Fehler bei Parallelität,486
- SQLE_ROW_DROPPED_DURING_SCHEMA_UPGRADE
 - UltraLite, Upgrade-Warnung,125
- SQLE_UNABLE_TO_CONNECT_OR_START
 - Fehlerbehandlung,495
- sqlpp, Dienstprogramm
 - UltraLite, Syntax,207
- SQRT-Funktion
 - UltraLite-Syntax,398
- ST_AsBinary
 - Funktion, UltraLite,398
- ST_AsExtText
 - Funktion, UltraLite,399
- ST_AsText
 - Funktion, UltraLite,399
- ST_Distance
 - Funktion, UltraLite,400
- ST_Equals
 - Funktion, UltraLite,400
- ST_Geometry
 - Typ für UltraLite,314
- ST_IntersectsRect
 - Funktion, UltraLite,401
- ST_Point
 - Funktion, UltraLite,402
- ST_PointFromExtText
 - Funktion, UltraLite,402
- ST_PointFromText
 - Funktion, UltraLite,403
- ST_PointFromWKB
 - Funktion, UltraLite,403
- ST_SRID
 - Funktion, UltraLite,404
- ST_X
 - Funktion, UltraLite,404
- ST_Y
 - Funktion, UltraLite,405
- Standardwerte
 - UltraLite, Autoincrement,439
 - UltraLite, CURRENT DATE,274
 - UltraLite, CURRENT TIME,274
 - UltraLite, CURRENT TIMESTAMP,275
 - UltraLite, CURRENT UTC TIMESTAMP,276
 - UltraLite, SQLCODE,276
 - UltraLiteJ, Autoincrement,439
- Starke Verschlüsselung
 - UltraLite, Entwicklungsschritte,19
 - UltraLite, fips-Erstellungsparameter,150
 - UltraLite, Verwendung,29
- START SYNCHRONIZATION DELETE-Anweisung
 - UltraLite, Syntax,460
 - UltraLiteJ, Syntax,460
- START, Verbindungsparameter
 - UltraLite, Syntax,189
- START-Verbindungsparameter
 - UltraLite für uleng16 unter Windows Mobile,19
- Statusverwaltung
 - UltraLite, Überblick,483
- Steuern von Synchronisationen
 - UltraLite-Publikationen,78
- STOP SYNCHRONIZATION DELETE-Anweisung
 - UltraLite, Syntax,461
 - UltraLiteJ, Syntax,461
- STR-Funktion

- UltraLite-Syntax,405
- stream type
 - UltraLite, Synchronisationsparameter,107
- stream_error, Synchronisationsparameter
 - UltraLite, ul_stream_error-Struktur,105
 - UltraLite-Referenz,105
- stream_parms, Synchronisationsparameter
 - UltraLite-Referenz,108
- STRING-Funktion
 - UltraLite-Syntax,406
- STRIP-Klausel
 - UltraLite, LOAD TABLE-Anweisung,455
- STRTOUUID-Funktion
 - UltraLite-Syntax,407
- STUFF-Funktion
 - UltraLite-Syntax,407
- subquery, Vorgang
 - UltraLite, Ausführungspläne,481
- SUBSCRIBE BY-Klausel
 - UltraLite-Synchronisationseinschränkungen,78
- SUBSTR-Funktion
 - UltraLite-Syntax,408
- SUBSTRING-Funktion
 - UltraLite-Syntax,408
- Suchbedingungen
 - UltraLite LIKE,290
 - UltraLite SQL,283
 - UltraLite, ALL,287
 - UltraLite, ANY,288
 - UltraLite, BETWEEN,288
 - UltraLite, EXISTS,289
 - UltraLite, IN,289
- SUM-Funktion
 - UltraLite-Syntax,409
- SWITCHOFFSET-Funktion
 - Syntax,410
- Sybase Central
 - Fehlerbehandlung von UltraLite-Verbindungen,495
- sync result
 - UltraLite, Synchronisationsparameter,109
- SYNC_PROFILE_OPTION_VALUE-Funktion
 - Syntax,411
- Synchronisation
 - clientspezifische Daten in UltraLite,78
 - ignorierte Zeilen in UltraLite,97
 - schreibgeschützte Tabellen in UltraLite ,81
 - timestamp_increment-Erstellungsparameter setzen,165
 - überwachen in UltraLite,99
- ulsync, Dienstprogramm für UltraLite-Datenbanken,227
- UltraLite anhalten,99
- UltraLite SQLE_DOWNLOAD_CONFLICT, Fehler,81
- UltraLite, ALTER TABLE-Anweisung,427
- UltraLite, Einführung ,69
- UltraLite, Parameter download_only,96
- UltraLite, Parameter upload only,110
- UltraLite, praktische Einführung mit CustDB,133
- UltraLite, referenzielle Integrität,80
- UltraLite, Schemaänderungen,49
- UltraLite, Tabellen mit Publikationen ausschließen,78
- UltraLite-Anwendung implementieren ,81
- UltraLite-Aufgaben, Überblick,69
- UltraLite-Clients,69
- UltraLite-Datenbanken unter Windows Mobile,88
- UltraLite-Fortschrittszähler,3
- UltraLite-Fremdschlüssel,80
- UltraLite-Planung, Überblick,74
- UltraLite-Systemtabelle,244
- UltraLite-Systemtabelle syssyncresult,242
- UltraLiteJ, CREATE TABLE-Anweisung,439
- Zeichensätze in UltraLite,26
- Synchronisation in UltraLite planen
 - Info,74
- Synchronisationsdatenströme
 - getStream-Methode,107
 - stream, Synchronisationsparameter in UltraLite ,107
 - UltraLite einstellen ,107
 - UltraLite, setStream-Methode,107
 - UltraLite, setStreamParms-Methode,108
 - UltraLite, stream_error, Synchronisationsparameter,105
 - UltraLite, stream_parms, Synchronisationsparameter,108
 - UltraLite, ULHTTPStream,107
 - UltraLite, ULHTTPStream,107
- Synchronisationsmodelle
 - UltraLite-Datenbanken,23
- Synchronisationsparameter
 - Authentifizierungswert in UltraLite,95
 - keep partial download in UltraLite,97
 - Parallelität in UltraLite deaktivieren, Überblick,484

- partial download retained in UltraLite,100
- resume partial download in UltraLite,104
- UltraLite,90
- UltraLite erforderlich,90
- UltraLite, additionalparms,91
- UltraLite, download_only,96
- UltraLite, getScriptVersion ,113
- UltraLite, getStream-Methode ,107
- UltraLite, getUploadOK-Methode,110
- UltraLite, NewMobiLinkPwd,98
- UltraLite, Observer ,99
- UltraLite, Password,101
- UltraLite, Ping,102
- UltraLite, Publication,103
- UltraLite, send_download_ack ,104
- UltraLite, setObserver-Methode,99
- UltraLite, setStream-Methode ,107,113
- UltraLite, setStreamParms-Methode,108
- UltraLite, setUserData-Methode,111
- UltraLite, stream type ,107
- UltraLite, stream_error,105
- UltraLite, stream_parms,108
- UltraLite, Sync Result ,109
- UltraLite, upload_only,110
- UltraLite, user_data,111
- UltraLite, user_name,112
- UltraLite, version ,113
- Synchronisationsprofil, Optionen
 - Info, UltraLite-Clients,230
- Synchronisationsprofile
 - UltraLite, ALTER SYNCHRONIZATION PROFILE-Anweisung,425
 - UltraLite, CREATE SYNCHRONIZATION PROFILE-Anweisung,436,448
 - UltraLite, SYNCHRONIZE-Anweisung,462
 - UltraLiteJ, DROP SYNCHRONIZATION PROFILE-Anweisung,448
 - UltraLiteJ, SYNCHRONIZE-Anweisung,462
- Synchronisationsskripten
 - durchsuchen, UltraLite-Beispiel,133
- Synchronisationsslogik
 - durchsuchen, Sybase Central in UltraLite,133
- SYNCHRONIZE-Anweisung
 - UltraLite, Syntax,462
 - UltraLiteJ, Syntax,462
- Syntax
 - UltraLite, arithmetische Operatoren,293
 - UltraLite, CASE-Ausdruck,280
 - UltraLite, CURRENT DATE-Spezialwert,274
 - UltraLite, CURRENT TIMESTAMP-Spezialwert,275
 - UltraLite, CURRENT UTC TIMESTAMP-Spezialwert,276
 - UltraLite, IF-Ausdrücke,279
 - UltraLite, Konstanten,278
 - UltraLite, logische Operatoren,286
 - UltraLite, Spezialwerte,273
 - UltraLite, SQL CURRENT TIME-Spezialwert,274
 - UltraLite, SQL-Eingabeparameter,282
 - UltraLite, SQL-Kommentare,272
 - UltraLite, SQL-Operator-Vorrang,295
 - UltraLite, SQL-Operatoren,293
 - UltraLite, SQLCODE-Spezialwert,276
 - UltraLite-Bit-Operatoren,294
 - UltraLite-Funktionen,316
 - UltraLite-Prädikate,283
 - UltraLite-Spaltennamen,278
 - UltraLite-Suchbedingungen,283
 - UltraLite-Vergleichsoperatoren,285
 - UltraLite-Zeichenfolgenoperatoren,294
- SYS
 - UltraLite-Systemtabellen,238
- sysarticle, Systemtabelle [UltraLite]
 - Info,239
- sysarticles-Systemtabelle [UltraLite Java Edition-Datenbanken]
 - Info,263
- syscolumn, Systemtabelle [UltraLite]
 - Info,239
- syscolumn-Systemtabelle [UltraLite Java Edition-Datenbanken]
 - Info,264
- sysfkcol-Systemtabelle [UltraLite Java Edition-Datenbanken]
 - Info,266
- sysforeignkey-Systemtabelle [UltraLite Java Edition-Datenbanken]
 - Info,265
- sysindex, Systemtabelle [UltraLite]
 - Info,240
- sysindex-Systemtabelle [UltraLite Java Edition-Datenbanken]
 - Info,266
- sysindexcolumn-Systemtabelle [UltraLite Java Edition-Datenbanken]
 - Info,267

sysixcol, Systemtabelle [UltraLite]
 Info,241

syspublication, Systemtabelle [UltraLite]
 Info,242

syspublications-Systemtabelle UltraLite Java Edition-Datenbanken
 Info,268

sysyncresult, Systemtabelle [UltraLite]
 Info,242

systable, Systemtabelle [UltraLite]
 Info,244

systable-Systemtabelle [UltraLite Java Edition-Datenbanken]
 Info,268

system_error_code, Werte
 UltraLite, Fehler im
 Synchronisationsdatenstrom,105

Systemausfall
 UltraLite-Datenbanken, Übersicht ,486
 UltraLite-Wiederherstellung,47

Systemfunktionen
 UltraLite, Info,322
 UltraLite-Einschränkungen,4

Systemtabellen
 UltraLite Java Edition, sysarticles,263
 UltraLite Java Edition, syscolumn,264
 UltraLite Java Edition, sysfkcol,266
 UltraLite Java Edition, sysforeignkey,265
 UltraLite Java Edition, sysindex,266
 UltraLite Java Edition, sysindexcolumn,267
 UltraLite Java Edition, syspublications,268
 UltraLite Java Edition, systable,268
 UltraLite Java Edition, sysuldata,269
 UltraLite, Durchsuchungsmethoden,54
 UltraLite, Info,238
 UltraLite, sysarticle,239
 UltraLite, syscolumn,239
 UltraLite, sysindex,240
 UltraLite, sysixcol,241
 UltraLite, syspublication,242
 UltraLite, sysyncresult,242
 UltraLite, systable,244

sysuldata-Systemtabelle [UltraLite Java Edition-Datenbanken]
 Info,269

T

Tabellen

UltraLite, allsync,78

UltraLite, ALTER TABLE-Anweisung,427

UltraLite, arbeiten mit,50

UltraLite, Bearbeitungsmethoden,54

UltraLite, CREATE TABLE-Anweisung,438

UltraLite, Durchsuchungsmethoden,54

UltraLite, INSERT-Anweisung,452

UltraLite, Kopiermethode,55

UltraLite, Kopiermethoden,54

UltraLite, Löschmethoden,53

UltraLite, Massenimport von Daten,452

UltraLite, Methoden erstellen,51

UltraLite, nosync,76

UltraLite, nur Download,77

UltraLite, Reihenfolge,80

UltraLite, Synchronisationen mit Publikationen steuern,78

UltraLite, temporäre Tabelle verwenden,480

UltraLite, TRUNCATE TABLE-Anweisung,463

UltraLite-Einschränkungen,9

UltraLite-Größenbegrenzung,50

UltraLite-Publikationen,78

UltraLite-Zeilen publizieren,87

UltraLiteJ, CREATE TABLE-Anweisung,438

UltraLiteJ, INSERT-Anweisung,452

UltraLiteJ, TRUNCATE TABLE-Anweisung,463

Tabellen nur zum Download
 nicht synchronisierte Tabellen in UltraLite,77

Tabellen, Integritätsregeln
 hinzufügen, löschen oder ändern in UltraLite,427

Tabellen-Integritätsregeln
 UltraLiteJ, CREATE TABLE-Anweisung,439

Tabellenausdrücke
 UltraLite-Unterabfragen,281

Tabelleneigentümer
 UltraLite,271

Tabellengröße
 Anzahl der Zeilen,9
 UltraLite-Begrenzung,9

Tabellenintegritätsregeln
 UltraLite, CREATE TABLE-Anweisung,439

TableOrder
 UltraLite, Synchronisationsparameter,91

Tag der Woche
 UltraLite, DOW-Funktion,353

- TAN-Funktion
 - UltraLite-Syntax,412
- TCP/IP
 - (*Siehe auch* TCP/IP-Synchronisation)
- Teilen
 - UltraLite-Primärschlüssel,71
- Teilzeichenfolgen
 - UltraLite ersetzen,388
 - UltraLite Info,408
- temp, Vorgang
 - UltraLite, Ausführungspläne,481
- TEMP_DIR-Verbindungsparameter
 - UltraLite, Syntax,189
- Temporäre Dateien
 - UltraLite-Einschränkung,189
- Temporäre Tabellen
 - UltraLite, Info,478
 - UltraLite-Abfragen,480
 - UltraLite-Einschränkungen,9
 - UltraLite-Synchronisation,78
- Temporäre UltraLite-Dateien
 - Info,189
- TEXT, Datentyp
 - UltraLite-Entsprechung,316
- Textpläne
 - UltraLite, anzeigen,480
- THEN
 - UltraLite, IF-Ausdrücke,279
- Threads
 - UltraLite-Parallelität,484
- TIME-Datentyp
 - UltraLite ,307
- time_format, Eigenschaft
 - UltraLite, Beschreibung,191
- time_format, Erstellungsparameter
 - UltraLite, Beschreibung,160
- time_format-Datenbankoption
 - UltraLite Java Edition-Beschreibung,250
- time_format-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
- TIMESTAMP WITH TIME ZONE-Datentyp
 - UltraLite,309
- TIMESTAMP-Datentyp
 - UltraLite,308
 - UltraLite-Spalteneinschränkungen,4
- TIMESTAMP-Spezialwert
 - UltraLite, TIMESTAMP-Spalten,439
 - UltraLite-Spalteneinschränkungen,4
 - UltraLiteJ, TIMESTAMP-Spalten,439
- timestamp_format, Eigenschaft
 - UltraLite, Beschreibung,191
- timestamp_format, Erstellungsparameter
 - UltraLite, Beschreibung,162
- timestamp_format-Datenbankoption
 - UltraLite Java Edition-Beschreibung,251
- timestamp_format-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
- timestamp_increment, Eigenschaft
 - UltraLite, Beschreibung,191
- timestamp_increment, Erstellungsparameter
 - in der MobiLink-Synchronisation verwenden,165
 - UltraLite, Beschreibung,164
- timestamp_increment-Datenbankoption
 - UltraLite Java Edition-Beschreibung,251
- timestamp_increment-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
- timestamp_with_time_zone-Eigenschaft
 - UltraLite Java Edition-Beschreibung,245
 - UltraLite- Beschreibung,191
- timestamp_with_time_zone_format, Erstellungsparameter
 - UltraLite,166
- timestamp_with_time_zone_format-Datenbankoption
 - UltraLite Java Edition-Beschreibung,252
- TINYINT-Datentyp
 - UltraLite,305
- TODATETIMEOFFSET-Funktion
 - Syntax,412
- TODAY-Funktion
 - UltraLite-Syntax,413
- TOP-Klausel
 - SELECT-Anweisung, für UltraLite,457
 - UltraLiteJ, SELECT-Anweisung,457
- Transaktionen
 - UltraLite, Bereinigung,196
 - UltraLite, COMMIT-Anweisung,432
 - UltraLite, Isolationsstufe,44
 - UltraLite, Parallelität,484
 - UltraLite, zurücksetzen,457
 - UltraLite-Checkpoints,432
 - UltraLite-Datenbanken,485
 - UltraLite-Schemaänderungen, Auswirkungen,49
 - UltraLiteJ, zurücksetzen,457
- Transaktionslog
 - UltraLite, interner Mechanismus,47
- Transaktionsverarbeitung

-
- Checkpoints und Festschreibungen,487
 - UltraLite, COMMIT_FLUSH-Verbindungsparameter,175
 - UltraLite, commit_flush_count-Option,196
 - UltraLite, commit_flush_timeout-Option,197
 - Transaktionsverwaltung
 - UltraLite,483
 - Transfer-Dienstprogramm für UltraLite Java Edition-Datenbanken (uljdbtserv)
 - Syntax,258
 - Transportschichtsicherheit
 - (*Siehe auch* TLS)
 - Trennkommasten
 - UltraLite, LIST-Funktion, Syntax,366
 - Trigger
 - UltraLite-Einschränkungen,4
 - TRIM-Funktion
 - UltraLite-Syntax,413
 - TRUNCATE TABLE-Anweisung
 - UltraLite, Syntax,463
 - UltraLiteJ, Syntax,463
 - TRUNCATE-Funktion
 - UltraLite-Syntax,414
 - TRUNCNUM-Funktion
 - UltraLite-Syntax,414
 - U**
 - Überlaufterhler
 - AVG-Funktion,328
 - SUM-Funktion,410
 - Überwachung der Synchronisation
 - observer, Synchronisationsparameter in UltraLite,99
 - setObserver-Methode in UltraLite,99
 - UCASE-Funktion
 - UltraLite-Syntax,415
 - UDB
 - (*Siehe auch* UltraLite-Datenbanken)
 - UID, Verbindungsparameter
 - UltraLite, Syntax,190
 - ul_stream_error, Struktur
 - UltraLite, Beispiel,105
 - UL_SYNC_ALL, Makro
 - UltraLite, Publikationsliste,103
 - UL_SYNC_ALL_PUBS, Makro
 - UltraLite, Publikationsliste,103
 - uleng16, Dienstprogramm
 - Deployment unter Windows Mobile,19
 - uleng16-Dienstprogramm
 - Option -ud,210
 - prozessintegrierte Datenbankunterstützung,210
 - Syntax,210
 - ulerase, Dienstprogramm
 - Syntax,212
 - ULHTTPSSStream-Funktion [UL ESQ] UltraLite-Synchronisationsdatenstrom,107
 - ULHTTPStream-Funktion [UL ESQ] UltraLite-Synchronisationsdatenstrom,107
 - ulinfo-Dienstprogramm
 - Syntax,213
 - ulinit, Dienstprogramm
 - Behelfslösung für nicht unterstützte Kollationen,34
 - Syntax,214
 - verwenden,22
 - uljdbtserv, Dienstprogramm
 - Datenbank empfangen mit,261
 - starten,258
 - uljdbtserv-Dienstprogramm
 - Datenbank löschen,262
 - Datenbank übertragen mit,259
 - Datenbankinformationen anzeigen,263
 - Logdatei anzeigen,263
 - Syntax,258
 - uljinfo, Dienstprogramm
 - Syntax,252
 - uljload, Dienstprogramm
 - Syntax,253
 - uljunload, Dienstprogramm
 - Syntax,256
 - ulload, Dienstprogramm
 - Syntax,222
 - verwenden,23
 - ULSocketStream-Funktion
 - UltraLite-Synchronisationsdatenstrom,107
 - ULSQLCONNECT, Umgebungsvariable
 - Beschreibung,37
 - ulstop, Dienstprogramm
 - Syntax,211
 - ulsync, Dienstprogramm
 - Synchronisationsprofiloptionen,230
 - Syntax,227
 - UltraLite
 - (*Siehe auch* UltraLite Embedded SQL)
 - (*Siehe auch* UltraLite SQL)
 - (*Siehe auch* UltraLite, APIs)

- (*Siehe auch* UltraLite, Dienstprogramme)
- (*Siehe auch* UltraLite-Anwendungen)
- (*Siehe auch* UltraLite-Datenbanken)
- Anwendungen und Datenbanken installieren,117
- Architektur,1
- Beispieldatenbank CustDB,133
- Benutzer,61
- Datenbank-Erstellungsparameter,141
- Datenkonvertierung,317
- Deployment von Anwendungen und Datenbanken,117
- Deployment von UltraLite-Datenbanken,125
- Dienstprogramme ,200
- Engine starten,210
- Engine stoppen,211
- Engine und Laufzeitdatenbank,19
- Fehlerbehandlung,495
- Fehlercodes,201
- Hinweise zu Windows Mobile,18
- Info,1
- nicht unterstützte Kollationen,34
- Parallelität,484
- räumliche Funktion, ST_ AsBinary,398
- räumliche Funktion, ST_ Distance,400
- räumliche Funktion, ST_ Point,402
- räumliche Funktion, ST_ PointFromExtText,402
- räumliche Funktion, ST_ PointFromText,403
- räumliche Funktion, ST_ PointFromWKB,403
- räumliche Funktion, ST_ SRID,404
- räumliche Funktion, ST_ AsExtText,399
- räumliche Funktion, ST_ AsText,399
- räumliche Funktion, ST_ Equals,400
- räumliche Funktion, ST_ IntersectsRect,401
- räumliche Funktion, ST_ X,404
- räumliche Funktion, ST_ Y,405
- SQL-Anweisungen,421
- SQL-Funktionen, Aggregat,317
- SQL-Funktionen, Datentypkonvertierung,317
- SQL-Funktionen, Typen von,316
- SQL_MAX_ROW_SIZE_EXCEEDED-Fehler,50
- Systemfunktionen,322
- Tabelleneigentümer,271
- timestamp_with_time_zone_format, Erstellungsparameter,166
- Umgebungsvariablen,35
- unterstützte Netzwerkprotokolle,3
- UTF8BIN-Kollation für UNICODE-Zeichen,28
- UltraLite C/C++
- Engine- und Laufzeit-Unterstützung,19
- UltraLite Datenbanken
 - Indizes, Hash-Methode,151
- UltraLite Embedded SQL
 - (*Siehe auch* UltraLite Embedded SQL, Bibliotheksfunktionen)
 - Präprozessor,207
 - Zeichenfolgen,207
 - Zeilennummern,207
- UltraLite Java Edition
 - Info,1
- UltraLite Java Edition-Datenbankeigenschaften
 - Info,245
- UltraLite Java Edition-Datenbanken
 - (*Siehe auch* UltraLite Java Edition)
 - (*Siehe auch* UltraLiteJ)
 - Dienstprogramme,252
 - Eigenschaften speichern,269
 - Einschränkungen,9
 - Fremdschlüsselbeschreibung,265
 - Fremdschlüsselspaltenbeschreibung,266
 - Indizes speichern,266
 - Publikationen speichern,268
 - Publikationsbeschreibung,263
 - Spalten,264
 - Tabellenbeschreibung,268
 - Tabellenspaltenbeschreibung,267
 - uljdbtserv-Dienstprogramm,258
 - uljinfo-Dienstprogramm,252
 - uljload-Dienstprogramm,253
 - uljunload-Dienstprogramm,256
- UltraLite Java Edition-Datenbankoptionen
 - Info,246
- UltraLite Java Edition-Dienstprogramm für Datenbankinformationen (uljinfo)
 - Syntax,252
- UltraLite Java Edition-Dienstprogramm für die Datenbankübertragung (uljdbtserv)
 - Datenbank empfangen,261
 - Datenbank löschen,262
 - Datenbank übertragen,259
 - Datenbankinformationen anzeigen,263
 - Logdatei anzeigen,263
 - starten,258
- UltraLite Java Edition-Managementsystem
 - (*Siehe auch* UltraLite Java Edition-Datenbanken)
 - (*Siehe auch* UltraLiteJ)
- UltraLite SQL

-
- (*Siehe auch* UltraLite Embedded SQL)
 - Anweisungsperformance optimieren,489
 - Fehlerbehandlung von Abfragen,499
 - kommagetrennte Listen,366
 - Operatoren,293
 - SQL-Funktionen, Datum und Uhrzeit,317
 - SQL-Funktionen, numerische,320
 - SQL-Funktionen, verschiedene,319
 - SQL-Funktionen, Zeichenfolge,321
 - UltraLite SQL-Anweisungen
 - ALTER DATABASE SCHEMA FROM FILE, Anweisungssyntax,423
 - UltraLite, Administrationstools
 - Fehlerbehandlung,498
 - UltraLite, Anwendungen
 - (*Siehe auch* UltraLite C/C++-API)
 - (*Siehe auch* UltraLite für AppForge-API)
 - (*Siehe auch* UltraLite.NET-API)
 - Benchmarking,488
 - Fehler -764,499
 - Fehlerbehandlung,
 - SQLE_UNABLE_TO_CONNECT_OR_START, 495
 - öffentlicher Zertifikatzugriff,229
 - Synchronisation ,69
 - TLS-aktivierte Synchronisation,117
 - UltraLite, APIs (*Siehe* UltraLite C/C++-API) (*Siehe* UltraLite für AppForge-API) (*Siehe* UltraLite.NET-API)
 - UltraLite, Benutzer-IDs
 - Info,61
 - UltraLite, Datenbankeigenschaften
 - Info,191
 - UltraLite, Datenbanken
 - Benchmarktests,488
 - Deployment auf Geräten,117
 - Deployment für Datenverschlüsselung,19
 - Deployment für
 - Synchronisationsverschlüsselung,117
 - Erstellungsparameter,25
 - Fehlerbehandlung von Verbindungen,495
 - Größe verringern,497
 - mit fips-Erstellungsparameter verschlüsseln,150
 - Objekte während Schema-Upgrade umbenennen,125
 - Optionseinstellungen anzeigen,40
 - page_size-Erstellungsparameter,155
 - sichern unter Windows Mobile,38
 - Synchronisationsprofiloptionen,230
 - UltraLite-Schema-Upgrades,423
 - Upgrade von Schema auf Geräten,125
 - UltraLite, Engine
 - Fehlerbehandlung,495
 - Fehlerbehandlung, Fehler -764,499
 - Fehlercodes,201
 - UltraLite, Erstellungsparameter
 - Info,141
 - UltraLite, Kennwörter
 - Info,61
 - UltraLite, Laden der XML-Datei in die Datenbank
 - Syntax,222
 - UltraLite, Laufzeit
 - Parallelität,484
 - UltraLite, Optimierer
 - Ausführungsplan-Zugriffsoptionen,477
 - UltraLite, Plug-Ins
 - Fehlerbehandlung,495
 - UltraLite, Publikationen
 - während Schema-Upgrade umbenennen,125
 - UltraLite, Schema
 - Upgrade auf Gerät durchführen,125
 - UltraLite, Spalten
 - während Schema-Upgrade umbenennen,125
 - UltraLite, SQL
 - Ausdrücke,277
 - Ausführungspläne für,477
 - Bezeichner,271
 - Datentypen,296
 - Datumsangaben,277
 - indexbasierte Optimierungen,471
 - Kommentare,272
 - NULL,273
 - Schlüsselwörter,271
 - seitenbasierte Optimierungen,479
 - Spezialwerte,273
 - Variable,296
 - Zahlen,273
 - Zeichenfolgen,272
 - Zeitangaben,277
 - UltraLite, SQL-Anweisungen
 - ALTER PUBLICATION-Anweisungssyntax,424
 - ALTER SYNCHRONIZATION PROFILE-Anweisungssyntax,425
 - ALTER TABLE-Anweisungssyntax,427
 - ALTER USER-Anweisungssyntax,431
 - CHECKPOINT-Anweisungssyntax,432

- COMMIT-Anweisungssyntax,432
- CREATE INDEX-Anweisungssyntax,433
- CREATE PUBLICATION-Anweisungssyntax,435
- CREATE SYNCHRONIZATION PROFILE-Anweisungssyntax,436
- CREATE TABLE-Anweisungssyntax,438,444
- DELETE-Anweisungssyntax,445
- DROP INDEX-Anweisungssyntax,446
- FROM-Klausel,450
- INSERT-Anweisung,452
- Kategorien,422
- LOAD TABLE-Anweisungssyntax,452
- ROLLBACK-Anweisungssyntax,457
- SELECT-Anweisungssyntax,457
- SET OPTION-Anweisung, Syntax,459
- START SYNCHRONIZATION DELETE-Anweisungssyntax,460
- STOP SYNCHRONIZATION DELETE-Anweisungssyntax,461
- SYNCHRONIZE, Anweisungssyntax,462
- TRUNCATE TABLE-Syntax,463
- UltraLite, DROP TABLE, -Anweisungssyntax,448
- UltraLite, DROP TABLE-Anweisungssyntax,449
- UNION-Vorgangssyntax,465
- UNIQUE-Parameter,433
- UPDATE-Anweisungssyntax,466
- UltraLite, Synchronisation
 - Info,69
- UltraLite, Tabellen
 - vom Erstellungsprozess ausschließen,214
 - während Schema-Upgrade umbenennen,125
- UltraLite, temporäre Tabellen
 - verwalten,478
- UltraLite, Verbindungen
 - Fehlerbehandlung,495
- UltraLite, Verbindungsparameter
 - Info,35
- UltraLite-Anwendungen
 - CustDB-Anwendungen und Readme-Dateien,17
 - Dateien mit MobiLink übertragen,83
 - Definieren eines Speicherorts für die temporäre Datei beim Verbinden,189
 - Deployment auf Geräten,19
 - Deployment von ActiveSync-Providerdateien durchführen,128
 - Deployment von FIPS-aktivierten,151
 - Engine-Speicherort bei Verbindung definieren,189
 - mehrere Anforderungen verwalten,484
 - Parallelität,484
 - Windows Mobile-APIs,18
- UltraLite-Benchmark-Test
 - Info,488
- UltraLite-Benutzer-IDs
 - MobiLink, Eindeutigkeit,199
- UltraLite-C/C++
 - CustDB-Anwendung erstellen,17
- UltraLite-Clients
 - Info zu MobiLink,69
- UltraLite-Clientsynchronisation
 - Parameter und Optionen,90
- UltraLite-Datenbanken
 - Abrufen von Zeilen,44
 - Anforderungsübersicht,484
 - Artikel,239
 - Assistent zum Erstellen einer Datenbank,21
 - auffüllen nach Ausführung von ulinit,222
 - Benutzer verwalten,63,65,66,67
 - Benutzer-IDs und Kennwörter,61
 - Cachegröße anpassen,469
 - Checkpoints verwenden,487
 - Dateipfaddefinition,37
 - Daten- und Statusverwaltung,483
 - Datenbankintegrität,485
 - Datenbankvergleich mit SQL Anywhere,4
 - Deadlocks,486
 - Dienstprogramme ,200
 - Eigenschaften durchsuchen,39
 - eindeutige Schlüssel,59
 - Einführung in Synchronisation,484
 - Entity-Relationship-Diagramme,56
 - erstellen,21
 - erstellen aus XML,23
 - in MobiLink modellieren,23
 - Indizes erstellen,59,433
 - Indizes speichern,240
 - Indizes verwenden,56
 - Indizes, Typen,59
 - Indizierung von Primärschlüsseln,33
 - initialisieren von Sybase Central aus,21
 - Isolationsstufen,44
 - Kollationssequenzen,26
 - konvertieren,33
 - Liste der Verbindungsparameter,168
 - löschen,212
 - mehrere verwalten,484
 - Mehrtabellen-Joins,4

- mit SQL Anywhere-Referenzdatenbank erstellen,33
- Objekte, Kopiermethode,55
- Oracle als Referenzdatenbank,69
- Publikationen,242
- Publikationen löschen,87
- Publikationen, Info,85
- Schema,238
- Schemaänderungen,49
- Schemaübersicht,49
- Sicherheitsübersicht,29
- Spalten ändern,52
- Spalten hinzufügen,52
- Speichernutzung,485
- Synchronisationen zählen,3
- Synchronisationsdienstprogramm (ulsync), Syntax,227
- Systemausfall, Sicherung, Wiederherstellung,47
- Tabellen durchsuchen,54
- Tabellen erstellen,51
- Tabellen kopieren,54
- Tabellen löschen,53
- Tabellen publizieren,86
- Tabellenspalten,241
- temporäre Dateien,189
- temporäre Tabellen verwalten,478
- Threads-Übersicht,484
- Transaktionsübersicht,484
- Über die Eingabeaufforderung erstellen,22
- über die Eingabeaufforderung initialisieren,33
- UltraLite, Größenbegrenzung,9
- UltraLite-Eigenschaft file,191
- UltraLite-Eigenschaft name,191
- unterstützte Indextypen,57
- validieren,46
- Verbindungsparameter-Überblick,35
- Verbindungsübersicht,35,484
- Verwaltungsgrundlagen,483
- verwendete Zustandsbytes,485
- wiederherstellen,485
- Windows Mobile-Dateipfade,38
- Zeilen löschen,485
- Zeilen publizieren,87
- Zeilen, Größenbegrenzung,50
- Zeilenabruf,44
- Zeilenpackung,156
- Zeilenpackung, Einführung,50
- Zeilensperren,486
- Zeilenzustand verwalten,485
- zurücksetzen,485
- UltraLite-Datenbanksynchronisation (ulsync) Synchronisationsprofiloptionen,230
- UltraLite-Dienstprogramm zum Entladen von Datenbanken in XML Syntax,233
- UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit) Syntax,214
- UltraLite-Dienstprogramm zum Löschen der Datenbank Syntax,212
- UltraLite-Dienstprogramm zum Starten der Engine Syntax,210
- UltraLite-Dienstprogramm zum Stoppen der Engine (ulstop) Syntax,211
- UltraLite-Dienstprogramm zum Validieren von Datenbanken (ulvalid) Info,237
- UltraLite-Dienstprogramme Info,200
- UltraLite SQL-Präprozessor (sqlpp), Syntax,207
- UltraLite, XML in Datenbank laden (ulload), Syntax,222
- UltraLite- Dienstprogramm zum Stoppen der Engine (ulstop), Syntax,211
- UltraLite-Datenbank (ulunload), Syntax,233
- UltraLite-Datenbank löschen (ulerase), Syntax,212
- UltraLite-Datenbank validieren (ulvalid), Syntax,237
- UltraLite-Dienstprogramm zum Initialisieren einer Datenbank (ulinit),214
- UltraLite-Engine (uleng16) starten, Syntax,210
- UltraLite-Engine (uleng16), Syntax,210
- UltraLite-Informationen (ulinfo), Syntax,213
- UltraLite-Synchronisation (ulsync), Syntax,227
- UltraLite-Engine als Daemon ausführen,210
- Definieren eines Speicherorts für die temporäre Datei beim Verbinden,189
- Dienstprogramm zum Löschen der Datenbank,212
- Dienstprogramm zum Stoppen,211
- Info,19
- Parallelität,484
- Speicherort für Programmdatei bei Verbindung definieren,189
- Start-Dienstprogramm,210

- Windows Mobile-Deployment,19
- UltraLite-Informationsdienstprogramm (ulinfo)
 - Syntax,213
- UltraLite-Laufzeitbibliothek
 - Info,19
- UltraLite-Laufzeitbibliotheken
 - Linux,117
 - Windows,117
 - Windows Mobile,117
- UltraLite-SQL-Anweisungen
 - Info,421
- UltraLite-Synchronisation
 - entfernte IDs und Benutzer-IDs,199
- UltraLite-Synchronisationsdienstprogramm (ulsync)
 - Syntax,227
- UltraLite-Systemtabellen
 - Info,238
- UltraLite-Systemtabellen, Übersicht
 - Info,238
- UltraLite-Verbindungsparameter
 - Liste,168
- UltraLite.NET
 - CustDB-Anwendung erstellen,18
 - CustDB-Beispiel und Readme-Dateien,18
 - UltraLite-Engine- und Laufzeit-Unterstützung,19
- UltraLiteJ
 - (*Siehe auch* UltraLite Java Edition-Datenbanken)
 - (*Siehe auch* UltraLite Java Edition-
Managementsystem)
 - Verschlüsselung und Verschleierung,29
- UltraLiteJ, SQL-Anweisungen
 - DELETE-Anweisungssyntax,445
 - DROP INDEX-Anweisungssyntax,446
 - INSERT-Anweisung,452
 - ROLLBACK-Anweisungssyntax,457
 - START SYNCHRONIZATION DELETE-
Anweisungssyntax,460
 - STOP SYNCHRONIZATION DELETE-
Anweisungssyntax,461
 - SYNCHRONIZE-Anweisungssyntax,462
 - TRUNCATE TABLE-Syntax,463
 - UltraLite, DROP TABLE-Anweisungssyntax,448
- ulvalid-Dienstprogramm
 - Syntax,237
- Umbenennen
 - UltraLite-Datenbankobjekte während eines
Upgrade,125
 - UltraLite-Tabellen ,427
- Umgebungsvariable
 - ERRORLEVEL für UltraLite,206
 - UltraLite, ULSQLCONNECT,37
 - UltraLite, Verwendung,35
- Unbenannte Fremdschlüssel
 - UltraLite, Verwendung,439
 - UltraLiteJ, Verwendung,439
- UNICODE-Zeichen
 - UltraLite-Kollation,28
- UNION, Vorgang
 - UltraLite, Syntax,465
- union-all, Vorgang
 - UltraLite, Ausführungspläne,481
- UNION-Anweisung
 - UltraLite, Syntax,465
- Unions
 - UltraLite, mehrere SELECT-Anweisungen,465
- UNIQUE
 - UltraLite, CREATE INDEX-Parameter,433
- UNIQUEIDENTIFIER-Datentyp
 - UltraLite,312
- Universally Unique Identifiers (universell eindeutige
Bezeichner)
 - UltraLite SQL-Syntax für die NEWID-
Funktion,379
- Universell eindeutige Bezeichner
 - (*Siehe auch* UUIDs)
- Unterabfragen
 - UltraLite SQL,281
- UPDATE-Anweisung
 - UltraLite, Syntax,466
- Upgrade
 - UltraLite, Schemafehler-Callback,127
 - UltraLite, Schemavorgang,125
 - UltraLite, SQL ALTER DATABASE SCHEMA
FROM FILE, Syntax,423
- Upgrades, UltraLite
 - Fehlerbehandlung von Verbindungen,495
- UpgradeSchemaFromFile, Methode
 - UltraLite, Ersatz für Schema-Upgrade ,125
- upload ok
 - UltraLite, Synchronisationsparameter,110
- upload only
 - UltraLite, Synchronisationsparameter,110
- upload_ok, Parameter
 - UltraLite upload_ok ,110
- upload_ok, Synchronisationsparameter
 - UltraLite-Referenz,110

- upload_only, Synchronisationsparameter
 - UltraLite-Referenz,110
- UPPER-Funktion
 - UltraLite-Syntax,415
- user data
 - UltraLite, Synchronisationsparameter,111
- user name
 - UltraLite, Synchronisationsparameter,112
- user_data
 - UltraLite, Synchronisationsparameter,111
- user_name
 - UltraLite, Synchronisationsparameter,112
- utf8_encoding, Datenbankeigenschaft
 - UltraLite, Verwendung,27
- utf8_encoding, Erstellungsparameter
 - UltraLite, Beschreibung,167
- UTF8BIN-Kollation
 - UltraLite, Hinweise,28
- UUIDs
 - UltraLite SQL-Syntax für die NEWID-Funktion,379
 - UltraLite SQL-Syntax für die STRTOUUID-Funktion,407
 - UltraLite SQL-Syntax für die UUIDTOSTR-Funktion,416
- UUIDTOSTR-Funktion
 - UltraLite-Syntax,416

V

- Validieren
 - (*Siehe auch* Datenbanken validieren)
 - UltraLite-Datenbanken,46
 - UltraLite-Datenbanken mit ulvalid,237
 - UltraLite-Erstellungsparameter
 - checksum_level,144
- Validieren, Datenbankdienstprogramm (ulvalid)
 - UltraLite, Info,237
- Validieren, Datenbanken
 - UltraLite-Dienstprogramm zum Validieren von Datenbanken (ulvalid),237
- VARBINARY, Datentyp
 - UltraLite,313
- VARCHAR-Datentyp
 - UltraLite,297
- Variable
 - UltraLite SQL,296
- Verbindung erstellen

- UltraLite-Datenbank, Fehlerbehandlung,36
- Verbindung herstellen
 - MobiLink UltraLite, Stream Type-Synchronisationsparameter,107
- Verbindung mit einer UltraLite-Datenbank
 - Info,35
- Verbindungen
 - UltraLite connCount-Eigenschaft,191
 - UltraLite, Fehlerbehandlung,495
 - UltraLite, Parallelität,484
 - UltraLite, Überblick,35
- Verbindungsmethoden
 - UltraLite, Info,35
- Verbindungsparameter
 - alphabetische Liste (UltraLite),168
 - CE,182
 - DBN für UltraLite,180
 - iPhone,182
 - Mac,181
 - NT,181
 - UltraLite,35
 - UltraLite file_name,177
 - UltraLite, Auswahl,37
 - UltraLite, CACHE_MAX_SIZE,170
 - UltraLite, CACHE_MIN_SIZE,171
 - UltraLite, CACHE_SIZE,172
 - UltraLite, CE_FILE,174
 - UltraLite, COMMIT_FLUSH,175
 - UltraLite, CON,177
 - UltraLite, DBF,177
 - UltraLite, DBKEY,179
 - UltraLite, desktop,181
 - UltraLite, device,182
 - UltraLite, Kennwort,186
 - UltraLite, key,179
 - UltraLite, MIRROR_FILE,184
 - UltraLite, NT_FILE,185
 - UltraLite, RESERVE_SIZE ,187
 - UltraLite, START ,189
 - UltraLite, TEMP_DIR ,189
 - UltraLite, Überblick,35
 - UltraLite, UID ,190
 - UltraLite, userid ,190
 - UltraLite, Verbindungsüberblick,35
 - UltraLite, Vorrang,36
- Verbindungszeichenfolgen
 - UltraLite einstellen ,35
 - UltraLite, Parameterüberblick,35

- UltraLite-Verbindungsparameter,168
 - Verfahren
 - UltraLite, Anwendungs- und Datenbank-Deployment,117
 - Vergleich
 - UltraLite- und SQL Anywhere-Datenbanken,4
 - Vergleich von UltraLite und SQL Anywhere Info,4
 - Vergleiche
 - UltraLite-Suchbedingungen,283
 - Vergleichsoperatoren
 - UltraLite SQL,285
 - UltraLite, Dynamic SQL-Syntax,285
 - Verknüpfen
 - UltraLite, Laufzeitbibliotheken,19
 - Verschleierung
 - UltraLite, Verwendung,29
 - UltraLiteJ-Entwicklung,29
 - Verschlüsselung
 - UltraLite, Certicom-Modul,29
 - UltraLite, Chiffrierschlüssel,179
 - UltraLite, Eigenschaft obfuscate verwenden,29
 - UltraLite, Entwicklungsschritte,19
 - UltraLite, fips-Eigenschaft, Verwendung,29
 - UltraLite, fips-Erstellungsparameter,150
 - UltraLite, fips-Verwendung,29
 - UltraLite, obfuscate-Erstellungsparameter,154
 - UltraLite, Schlüssel ändern,150
 - UltraLite, TLS-Synchronisationskonfiguration,117
 - UltraLite-Daten-Deployment, Hinweise,19
 - UltraLite-Eigenschaft encryption,191
 - UltraLite-Synchronisations-Deployment, Hinweise,117
 - UltraLiteJ-Entwicklung,29
 - Version
 - UltraLite, Synchronisationsparameter,113
 - version, Synchronisationsparameter
 - UltraLite, Referenz,113
 - Versionen
 - UltraLite, Fehlerbehandlung bei Dienstprogrammen,498
 - Vertrauenswürdige Zertifikate
 - UltraLite, Anwendungszugriff auf Verschlüsselungsinformationen,229
 - Verwalten, Datenbanken
 - UltraLite-Daten und -Status,483
 - Verwalten, temporäre Tabellen
 - UltraLite, Info,478
 - Verwalten, Transaktionen
 - UltraLite, erhöhter Durchsatz,487
 - Verwaltung
 - UltraLite auf Geräten,117
 - Verwaltungsdienstprogramme
 - UltraLite-Dienstprogramme,200
 - Verzögern, Festschreibungen
 - Performancesteigerungen,487
 - Virtuelles Dateisystem (*Siehe* VFS)
 - Visual Studio
 - UltraLite, CustDB-Anwendung erstellen,17
 - Vorbereitete Anweisungen
 - UltraLite, Eingabeparameter,282
 - Vorrang
 - UltraLite, SQL-Operator-Vorrang,295
- ## W
- WEEKS-Funktion
 - UltraLite-Syntax,417
 - Werte
 - UltraLite-Index-Hash,151
 - WHEN
 - UltraLite, CASE-Ausdruck,280
 - WHERE-Klausel
 - CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote],435
 - UltraLite, CREATE PUBLICATION-Anweisung,435
 - UltraLite, DELETE-Anweisung,446
 - UltraLite, SELECT-Anweisung,458
 - UltraLite, UPDATE-Anweisung,466
 - UltraLite-Publikation, verwenden,87
 - UltraLite-Suchbedingungen,283
 - UltraLite-Synchronisationseinschränkungen,78
 - Wiederherstellen
 - UltraLite, Einführung,485
 - UltraLite-Transaktionsübersicht ,486
 - Wiederherstellung
 - UltraLite,47
 - UltraLite-Info,47
 - UltraLite-Transaktionsübersicht ,486
 - Wiederherstellung nach Systemausfall
 - UltraLite, interner Mechanismus,47
 - Windows
 - (*Siehe auch* Windows ME)
 - (*Siehe auch* Windows Mobile 5)
 - (*Siehe auch* Windows Vista)

-
- (Siehe auch Windows XP/200x)
 - UltraLite-Laufzeitbibliotheken,117
 - UltraLite-Zeichensätze,27
 - Windows Mobile
 - Auswahl einer API,18
 - Fehlerbehandlung, Fehler -764,499
 - Hinweise zu UltraLite,18
 - UltraLite MobiLink-Clients,88
 - UltraLite uleng 16-Deployment,19
 - UltraLite, ActiveSync-Deployment,128
 - UltraLite, CustDB-Anwendung mit .NET erstellen,18
 - UltraLite, FIPS-Aktivierung,151
 - UltraLite-Dateipfadpräfix,38
 - UltraLite-Engine, Entwicklung,19
 - UltraLite-Engine- und Laufzeit-Unterstützung,19
 - UltraLite-Laufzeitbibliotheken,117
 - UltraLite-Zeichensätze,27
 - WITH CHECKPOINT-Klausel
 - UltraLite, LOAD TABLE-Anweisung,455
 - Wörter
 - UltraLite, reservierte Wörter,271
 - UltraLite-Schlüsselwörter,271
 - X**
 - XML
 - in Datenbank laden,222
 - Quelle für UltraLite-Datenbanken ,23
 - UltraLite-Datenbanken entladen,227
 - XML, Datentyp
 - UltraLite-Entsprechung,316
 - Y**
 - YEAR-Funktion
 - UltraLite-Syntax,419
 - YEARS-Funktion
 - UltraLite-Syntax,419
 - YMD-Funktion
 - UltraLite-Syntax,421
 - Z**
 - Zahlen
 - UltraLite SQL,273
 - Zeichenfolgelänge
 - UltraLite, LENGTH-Funktion,365
 - Zeichenfolgen
 - UltraLite Embedded SQL,207
 - UltraLite ersetzen,388
 - UltraLite nachgestellte Leerzeichen entfernen ,391
 - UltraLite nearest_century, Konvertierung in Datumsangabe ,153
 - UltraLite SQL,272
 - UltraLite, Groß- und Kleinschreibung,272
 - UltraLite, maximale Größe,9
 - UltraLite, SQL-Funktionen,321
 - Zeichenfolgen konvertieren
 - UltraLite, Info,321
 - Zeichenfolgen verketteten
 - UltraLite-Zeichenfolgenoperatoren,294
 - Zeichenfolgenfunktionen
 - UltraLite, alphabetische Liste,321
 - Zeichenfolgenlitterale
 - UltraLite-Konstante,278
 - Zeichenfolgenoperatoren
 - UltraLite, Dynamic SQL-Syntax,294
 - Zeichenfolgeposition
 - UltraLite, LOCATE-Funktion,367
 - Zeichenfunktionen
 - UltraLite, alphabetische Liste,321
 - Zeichensätze
 - UltraLite unter Windows,27
 - UltraLite unter Windows Mobile,27
 - UltraLite, collation-Erstellungsparameter,145
 - UltraLite-Datenbanken,27
 - UltraLite-Eigenschaft char_set,191
 - UltraLite-Zeichenfolgen,272
 - Zeichensatzkonvertierung
 - Kennwörter,445
 - Zeilen
 - UltraLite, abrufen,44
 - UltraLite, alle Zeilen aus einer Tabelle löschen,463
 - UltraLite, INSERT-Anweisung,452
 - UltraLite, Masseneinfügung,452
 - UltraLite, publizieren,87
 - UltraLite, Sperren,486
 - UltraLiteJ, alle Zeilen aus einer Tabelle löschen,463
 - UltraLiteJ, INSERT-Anweisung,452
 - Zeilenlänge
 - UltraLite, sqlpp-Dienstprogramm, Ausgabe,207
 - Zeilenpackung
 - UltraLite, Auswirkungen,156
 - UltraLite, beobachten,187
 - UltraLite, Info,50
 - Zeitangaben
-

- UltraLite, Konvertierungsfunktionen,317
- Zeitangaben, Hinweise
 - UltraLite, Info,160
- Zeitfunktionen
 - UltraLite, alphabetische Liste,317
- Zeitstempel
 - UltraLite, timestamp_format-Erstellungsparameter,162
 - UltraLite, timestamp_increment-Erstellungsparameter,164
- Zentrale Administration von entfernten Datenbanken
 - UltraLite-Datenbanken,23
- Zertifikate
 - UltraLite, Anwendungszugriff auf Verschlüsselungsinformationen,229
- Ziffern
 - UltraLite, maximale Anzahl,157
- Zufallszahlen
 - RAND-Funktion,385
 - UltraLiteJ RAND,385
- Zurücksetzen
 - UltraLite-Datenbanken,485
 - UltraLite-Transaktionen,457
 - UltraLite-Transaktionsübersicht ,486
 - UltraLiteJ-Transaktionen,457
- Zusammenstellen, Parameter in
- Verbindungszeichenfolgen
 - UltraLite, Info,35
- Zustandsbyte
 - UltraLite-Datenbanken,485
- Zweideutige Zeichenfolgen in Datumsangaben konvertieren
 - UltraLite,153
- Zyklen
 - UltraLite, Probleme mit Fremdschlüsseln,80
 - UltraLite-Fremdschlüssel,80