SQL Remote™

Version 16.0

Februar 2013

Version 16.0
Februar 2013

Inhalt

Über diese Dokumentation

Diese Dokumentation beschreibt das SQL Remote-Replikationssystem für mobile Datenverarbeitung, das die gemeinsame Datennutzung von einer konsolidierten SQL Anywhere-Datenbank und vielen entfernten SQL Anywhere-Datenbanken über eine indirekte Verbindung wie etwa E-Mail oder Datenübertragung ermöglicht.

SQL Remote-Systeme

SQL Remote ist eine nachrichtenbasierte Technologie für die Zwei-Weg-Replikation von Datenbanktransaktionen zwischen einer konsolidierten Datenbank und einer großen Anzahl von entfernten Datenbanken. Die Verwaltung und die Anforderungen an die Ressourcen am entfernten Standort sind minimal, sodass SQL Remote für mobile Geräte besonders gut geeignet ist.

SQL Remote stellt folgende Funktionen bereit:

- **Unterstützung für mehrere Subskribenten** SQL Remote ermöglicht es gelegentlich verbundenen Benutzern, Daten zwischen einer konsolidierten SQL Anywhere-Datenbank und einer großen Anzahl von entfernten SQL Anywhere-Datenbanken, üblicherweise vielen mobilen Datenbanken, zu replizieren.

- **Transaktionsbasierte Replikation** SQL Remote verwendet das Transaktionslog für die Replikation. Dadurch werden während einer Aktualisierung nur geänderte Daten repliziert. Dies stellt eine korrekte Transaktionsatomizität im Replikationssystem sicher und bewahrt die Konsistenz zwischen den Datenbanken, die an der Replikation teilnehmen.

- **Zentrale Verwaltung** SQL Remote wird zentral von der konsolidierten Datenbank aus verwaltet. Eine Firma kann einen umfangreichen Außendienst mit vielen einzelnen Datenbanken haben, ohne die jeweiligen entfernten Datenbanken einzeln pflegen zu müssen. Außerdem ist der SQL Remote-Betrieb für den Endanwender unsichtbar.

- **Sparsame Speichernutzung** Während der Ausführung verwendet SQL Remote den Speicher effizient und sparsam. Dies ermöglicht es Ihnen, SQL Remote auf vorhandenen entfernten Computern einzusetzen, ohne in neue Hardware investieren zu müssen. Eine Replikation ist an bzw. von Computern und Geräten mit beschränktem Speicherplatz möglich, da nur relevante Daten von der konsolidierten Datenbank an die entfernten Datenbanken repliziert werden.

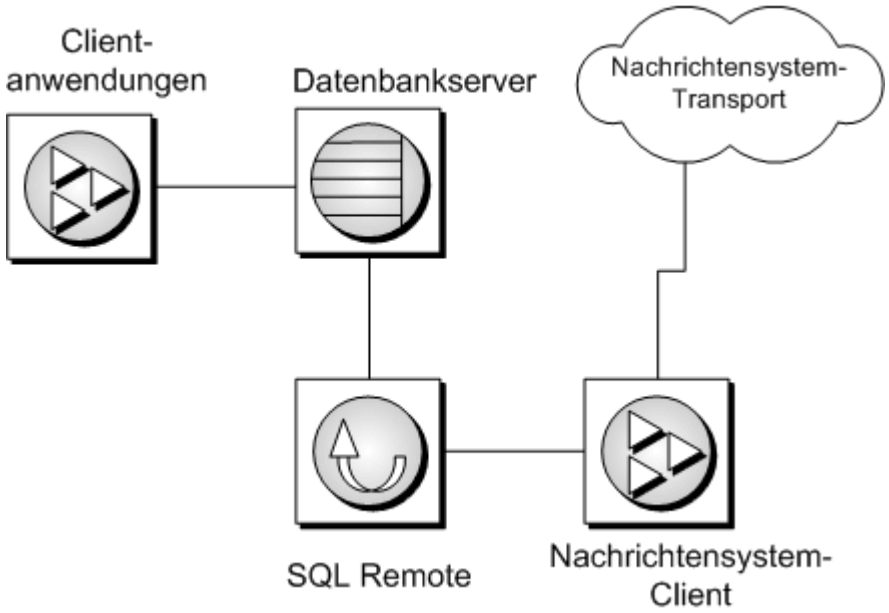
- **Unterstützung mehrerer Plattformen** SQL Remote wird auf einer Reihe von Betriebssystemen unterstützt und kann mit zahlreichen Nachrichtensystemen eingesetzt werden. SQL Anywhere-Datenbanken können von einer Datei oder einem Betriebssystem auf ein anderes kopiert werden.

Siehe auch

- http://www.sybase.com/detail?id=1061806
- „SQL Remote-Systeme erstellen“ auf Seite 9
- „SQL Remote-Systeme verwalten“ auf Seite 81
- „SQL Remote-Referenz“ auf Seite 203

Komponenten von SQL Remote

Folgende Komponenten sind für ein SQL Remote-System erforderlich:



- **Datenbankserver** Eine SQL Anywhere-Datenbank ist am konsolidierten Standort und an allen entfernten Standorten erforderlich.
- **SQL Remote** Um Replikationsnachrichten an Datenbanken zu senden bzw von Datenbanken zu empfangen, muss SQL Remote am konsolidierten Standort und an den entfernten Standorten installiert sein.

  Der SQL Remote-Nachrichtenagent stellt eine Verbindung zum Datenbankserver über eine Client-/Serververbindung her. Der SQL Remote-Nachrichtenagent kann auf demselben Computer wie der Datenbankserver oder auf einem anderen Computer laufen.
- **Nachrichtensystem-Clientsoftware** SQL Remote verwendet vorhandene Nachrichtensysteme, um Replikationsnachrichten zu übermitteln.

  Wenn Sie ein System mit gemeinsam genutzten Dateien oder ein FTP-Nachrichtensystem verwenden, ist das Nachrichtensystem in Ihrem Betriebssystem enthalten.

  Wenn Sie ein SMTP-E-Mail-System verwenden, muss am konsolidierten Standort und an allen entfernten Standorten ein E-Mail-Client installiert sein.
- **Clientanwendungen** Die Clientanwendung kann ODBC, Embedded SQL oder eine Reihe von anderen Programmierschnittstellen verwenden. Clientanwendungen brauchen nicht zu wissen, ob sie eine konsolidierte oder eine entfernte Datenbank verwenden. Aus der Perspektive der

Clientanwendung gibt es keinen Unterschied. Einzelheiten über die SQL Anywhere-Programmierschnittstellen finden Sie in der nachstehenden Liste:

- „SQL Anywhere .NET-Unterstützung“ [*SQL Anywhere Server - Programmierung*]
- „ODBC-Unterstützung“ [*SQL Anywhere Server - Programmierung*]
- „OLE DB- und ADO-Entwicklung“ [*SQL Anywhere Server - Programmierung*]
- „Embedded SQL“ [*SQL Anywhere Server - Programmierung*]
- „JDBC-Unterstützung“ [*SQL Anywhere Server - Programmierung*]
- „Sybase Open Client-Unterstützung“ [*SQL Anywhere Server - Programmierung*]
- „SQL Anywhere-C-API-Unterstützung“ [*SQL Anywhere Server - Programmierung*]
- „Perl/DBI-Unterstützung“ [*SQL Anywhere Server - Programmierung*]
- „SQL Anywhere-PHP-Unterstützung“ [*SQL Anywhere Server - Programmierung*]
- „Python-Unterstützung“ [*SQL Anywhere Server - Programmierung*]
- „SQL Anywhere-Ruby-API-Unterstützung“ [*SQL Anywhere Server - Programmierung*]

Typische SQL Remote-Einrichtungen

SQL Remote ist für Replikationssysteme mit folgenden Anforderungen ausgelegt:

- **Große Anzahl von entfernten Datenbanken** SQL Remote kann Tausende entfernter Datenbanken in einer einzelnen Installation unterstützen, weil die Nachrichten von vielen entfernten Datenbanken gleichzeitig vorbereitet werden können.

- **Fallweise verbunden** SQL Remote unterstützt Datenbanken, die nur fallweise oder indirekt mit dem Netzwerk verbunden sind. SQL Remote ist nicht für den sofortigen Abgleich von Daten an allen Standorten ausgelegt. Es kann beispielsweise ein SMTP-E-Mail-System verwendet werden, um die Replikation durchzuführen.

- **Niedrige bis hohe Latenzzeit** Eine hohe Latenzzeit bedeutet, dass eine große Zeitspanne zwischen der Eingabe von Daten in eine Datenbank und der Synchronisation mit jeder Datenbank im System vorliegt. Mit SQL Remote werden Replikationsnachrichten in Intervallen von Sekunden, Minuten, Stunden oder Tagen versendet.

- **Niedriges bis mittleres Verkehrsaufkommen** Da Replikationsnachrichten nur fallweise versendet werden, kann eine große Transaktionsmenge in den einzelnen entfernten Datenbanken zu einem großen Volumen von Nachrichten führen. SQL Remote ist am besten für Systeme mit relativ geringem Volumen von replizierten Daten pro entfernter Datenbank geeignet. In der konsolidierte Datenbank kann SQL Remote gleichzeitig Nachrichten für mehrere Datenbanken vorbereiten.

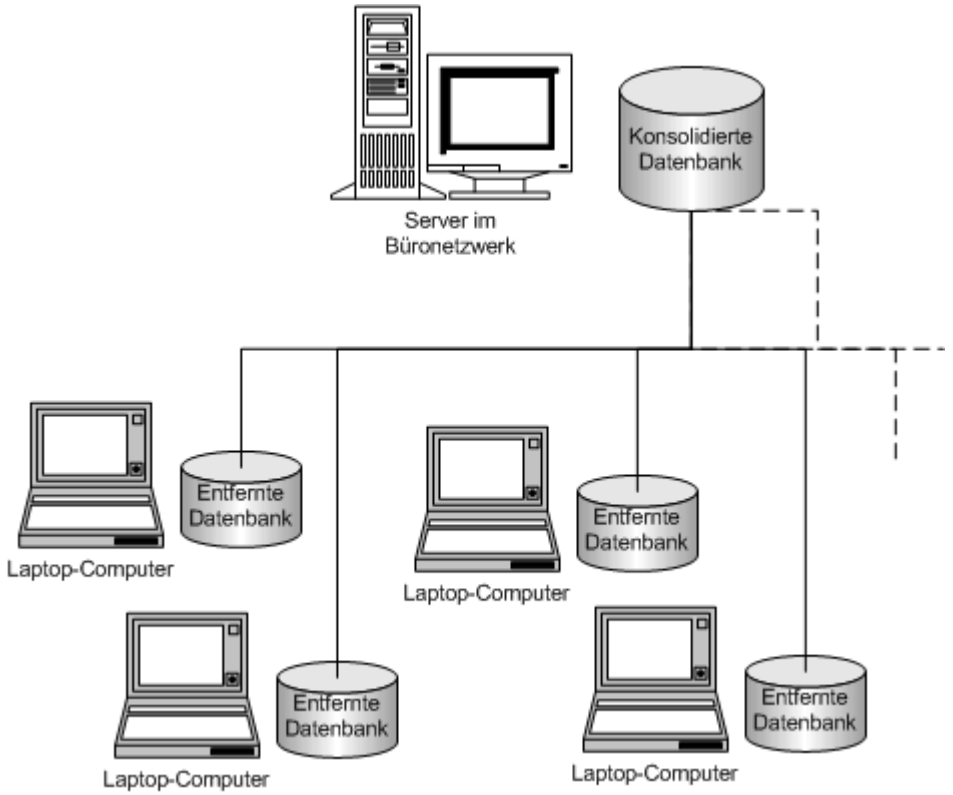
- **Homogene Datenbanken** Jede SQL Anywhere-Datenbank im System muss mit einem ähnlichen Schema aufgebaut sein.

Siehe auch

- „Synchronisationstechnologie im Vergleich“ [*SQL Anywhere 16 - Einführung*]
- „Hinweise zur Synchronisationstechnologie“ [*SQL Anywhere 16 - Einführung*]

Replikation von Server-Datenbank zu entfernter Datenbank im Außendienst

Im folgenden Beispiel stellt SQL Remote eine Zwei-Weg-Replikation zwischen einer konsolidierten Datenbank in einem Büronetzwerk und den persönlichen Datenbanken auf Laptop-Computern von Mitarbeitern zur Verfügung. Ein SMTP-E-Mailsystem wird für die Nachrichtenübermittlung verwendet.



Um die konsolidierte Datenbank zu verwalten, führt der Büronetzwerkserver einen SQL Anywhere-Datenbankserver aus. SQL Remote stellt eine Verbindung zur konsolidierten Datenbank auf dieselbe Art wie jede andere Clientanwendung her.

Die Laptop-Computer der Handelsvertreters enthalten einen SQL Anywhere-Personal Server, eine entfernte SQL Anywhere-Datenbank und SQL Remote.

Wenn ein Handelsvertreter unterwegs ist, kann er eine Verbindung zum Internet herstellen, um SQL Remote auszuführen, wobei die folgenden Funktionen durchgeführt werden:

- Sammeln von Publikationsaktualisierungen von der konsolidierten Datenbank auf dem Büronetzwerkserver.
- Übermitteln von lokalen Aktualisierungen, wie z.B. neue Bestellungen, an die konsolidierte Datenbank auf dem Büronetzwerkserver.

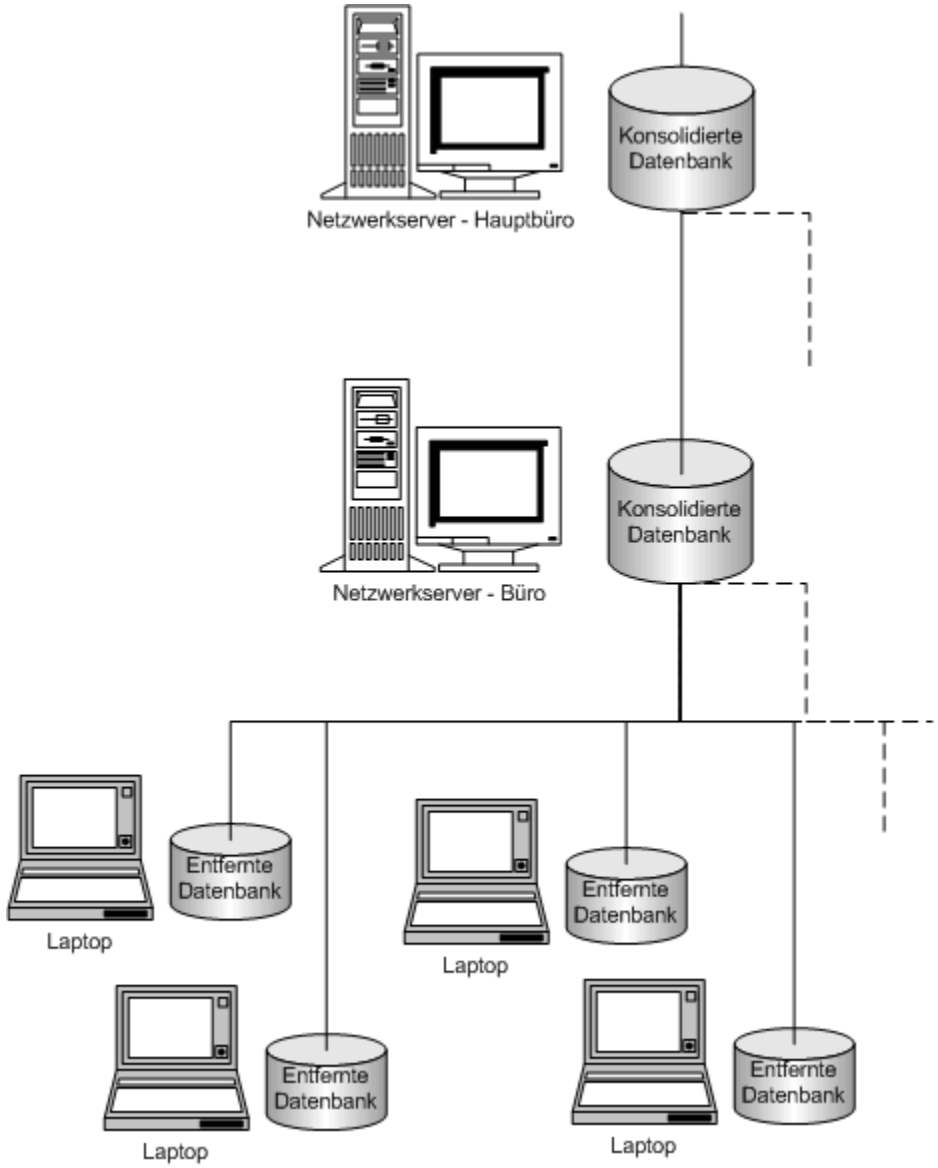
Die Publikationsaktualisierungen von der Büronetzwerk-Datenbank können neue Sonderangebote für Produkte, die der Handelsvertreter betreut, oder neue Preisgestaltungs- und Bestandsinformationen enthalten. Diese Aktualisierungen werden automatisch von SQL Remote auf dem Laptop eingelesen und in der entfernten Datenbank des Handelsvertreters übernommen, ohne dass zusätzliche Aktionen des Handelsvertreters notwendig wären.

Die vom Handelsvertreter eingegebenen neuen Bestellungen werden ebenfalls automatisch übermittelt und in der Büronetzwerk-Datenbank übernommen, ohne dass eine weitere Aktion des Handelsvertreters erforderlich ist.

Replikation von Server- zu Server-Datenbank zwischen Büros

In diesem Beispiel stellt SQL Remote eine Zwei-Weg-Replikation zwischen den Datenbankservern in Verkaufbüros bzw. Außenstellen und dem zentralen Hauptbüro zur Verfügung. Die einzige Arbeit, die in den Verkaufsbüros erforderlich ist, besteht in der anfänglichen Installation und der fortlaufenden Pflege des Datenbankservers.

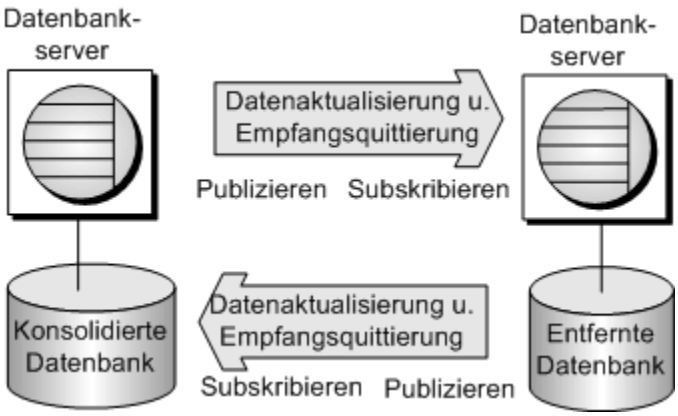
In die Hierarchie von SQL Remote können zusätzliche Schichten einbezogen werden. Zum Beispiel können Server in den Verkaufsbüros als konsolidierte Datenbank fungieren, die nur jene entfernten Subskribenten unterstützt, die für dieses Verkaufsbüro im Einsatz sind.



SQL Remote kann so konfiguriert werden, dass jedes Büro seine eigenen Datensätze erhält. Tabellen wie z.B. Mitarbeiter-Datensätze können in derselben Datenbank wie die replizierten Daten vertraulich bleiben.

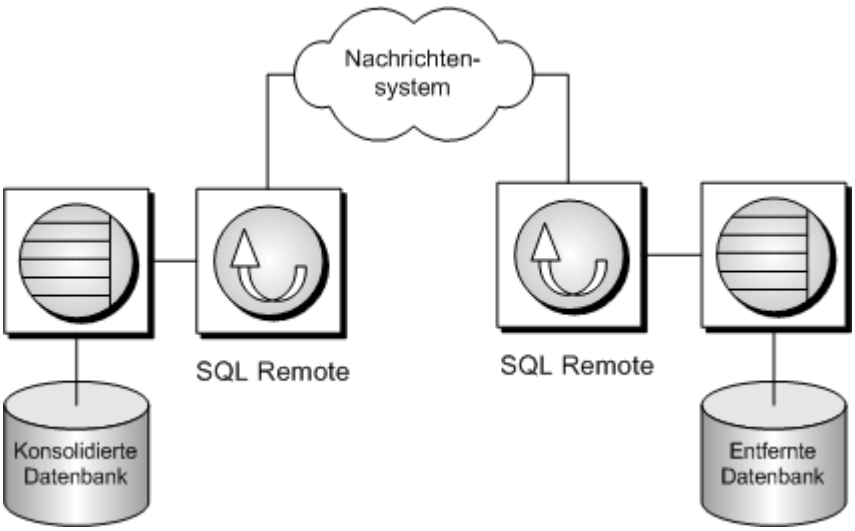
SQL Remote-Replikationsprozess

Bei SQL Remote werden Nachrichten immer in beide Richtungen gesendet. Die konsolidierte Datenbank sendet Nachrichten, die Publikationsaktualisierungen enthalten, an entfernte Datenbanken. Entfernte Datenbanken wiederum senden aktualisierte Daten und Empfangsbestätigungen an die konsolidierte Datenbank.



Wenn Benutzer der entfernten Datenbanken Daten ändern, werden ihre Änderungen an die konsolidierte Datenbank repliziert. Wenn diese Änderungen in der konsolidierten Datenbank angewendet werden, werden sie ein Teil der Publikation der konsolidierten Datenbank und in die Aktualisierungen aufgenommen, die an alle entfernten Datenbanken (außer an jene, von der die Aktualisierung stammte) gesendet werden. Auf diese Weise erfolgt die Replikation von der entfernten Datenbank zur entfernten Datenbank über die konsolidierte Datenbank.

Beispiel: Wenn Daten in einer Publikation einer konsolidierten Datenbank aktualisiert werden, sendet das Replikationssystem sie an die entfernten Datenbanken. Auch wenn die Daten in der entfernten Datenbank nie aktualisiert werden, werden Bestätigungsnachrichten an die konsolidierte Datenbank zurückgesendet, damit der Status der Replikation verfolgt werden kann.



Schritte des SQL Remote-Replikationsprozesses

1. Bei jeder konsolidierten und entfernten Datenbank, die an der Replikation teilnimmt, gibt es einen Nachrichtenagenten und ein Transaktionslog, das die Replikation verwaltet. Alle festgeschriebenen Änderungen werden aufgezeichnet und im Transaktionslog gespeichert.

2. Regelmäßig durchsucht der SQL Remote-Nachrichtenagent in der konsolidierten Datenbank das Transaktionslog und verpackt alle festgeschriebenen Transaktionen, die für die einzelnen Publikationen (Datenabschnitte) durchgeführt wurden, in Nachrichten. Der SQL Remote-Nachrichtenagent der konsolidierten Datenbank sendet anschließend die relevanten Änderungen an die entfernten Benutzer, die diese Publikationen subskribieren. Der SQL Remote-Nachrichtenagent sendet die Änderungen unter Verwendung eines Messaging-Systems. SQL Remote unterstützt SMTP-E-Mailsysteme, FTP, HTTP/S und FILE.

3. Der SQL Remote-Nachrichtenagent in der entfernten Datenbank akzeptiert die Nachrichten, die von der konsolidierten Datenbank gesendet wurden, und sendet eine Bestätigung an die konsolidierte Datenbank, dass die Nachrichten empfangen wurden. Der SQL Remote-Nachrichtenagent die wendet die Transaktionen anschließend in der entfernten Datenbank an.

4. Ein entfernter Benutzer kann jederzeit den SQL Remote-Nachrichtenagenten ausführen, um die in der entfernten Datenbank durchgeführten Transaktionen in Nachrichten zu verpacken und sie an die konsolidierte Datenbank zurückzusenden.

5. Der SQL Remote-Nachrichtenagent am konsolidierten Standort verarbeitet die Nachrichten von der entfernten Datenbank und wendet die Transaktionen in der konsolidierten Datenbank an.

Siehe auch

- „SQL Remote-Systeme erstellen“ auf Seite 9
- „SQL Remote-Systeme verwalten“ auf Seite 81

SQL Remote-Systeme erstellen

Verwenden Sie die konsolidierte Datenbank, um alle SQL Remote-Administrationsaufgaben durchzuführen. Um eine Verbindung mit Datenbanken unter Verwendung von SQL Remote herzustellen, müssen Sie über die SYS_RUN_REPLICATION_ROLE verfügen.

Das Folgende ist eine Zusammenfassung der Schritte, die Sie ausführen müssen, um ein SQL Remote-System zu erstellen.

1. Wählen Sie Ihre konsolidierte SQL Anywhere-Datenbank aus oder erstellen Sie eine neue SQL Anywhere-Datenbank. Die entfernten Datenbanken, die ebenfalls SQL Anywhere-Datenbanken sind, werden aus der konsolidierten Datenbank erstellt. Wenn Sie eine neue SQL Anywhere-Datenbank erstellen, beachten Sie, wie SQL Remote Primärschlüssel verwendet. (Es besteht die Möglichkeit, dass Primärschlüssel mehrfach vorhanden sind, wenn entfernte Datenbanken in die konsolidierte Datenbank repliziert werden.) Es empfiehlt sich, BIGINT mit GLOBAL AUTOINCREMENT als Datentyp der Primärschlüsselspalte zu verwenden.

   Bestimmen Sie, welche Daten repliziert werden sollen.

2. Um ein effizientes Replikationssystem zu erstellen, müssen Sie die zu verwendenden Tabellen festlegen sowie die Spalten aus diesen Tabellen und die Teilmenge der zu replizierenden Zeilen. Nehmen Sie nur Informationen auf, die wirklich benötigt werden.

3. Erstellen Sie Publikationen in der konsolidierten Datenbank.

   SQL Remote verwendet ein Publikations- und Subskriptionsmodell um sicherzustellen, dass die korrekten Informationen den vorgesehenen Benutzer erreichen. Ordnen Sie die Daten, die repliziert werden sollen, Publikationen in der konsolidierten Datenbank zu.

4. Erstellen Sie einen Benutzer als Publikationseigentümer in der konsolidierten Datenbank .

   Ein Publikationseigentümer ist ein Benutzer mit PUBLISH-Privileg.

5. Erstellen Sie die entfernten Benutzer in der konsolidierten Datenbank.

   Ein entfernter Benutzer wird verwendet, um eine entfernte Datenbank eindeutig zu kennzeichnen.

   Wenn Sie einen entfernten Benutzer erstellen, legen Sie den Nachrichtentyp fest, der bei der Übermittlung der Daten verwendet werden soll, sowie optional die Häufigkeit, mit der Daten gesendet werden sollen.

6. Subskribieren Sie Publikationen für die entfernten Benutzer, indem Sie Subskriptionen erstellen.

7. Legen Sie fest, wie die entfernten Benutzer die Daten verwenden können.

   Entfernte Benutzer können immer ihre Daten lesen. Sie können es ihnen auch ermöglichen, Daten zu aktualisieren, zu löschen und einzufügen.

8. Wählen Sie eine Methode aus, um Konflikte aufzulösen.

Konflikte können während einer Replikation auftreten, wenn Ihre entfernten Benutzer Daten aktualisieren, löschen oder einfügen. Sie müssen Methoden zur Auflösung von Konflikten implementieren.

9. Nehmen Sie das Deployment des SQL Remote-Systems vor.

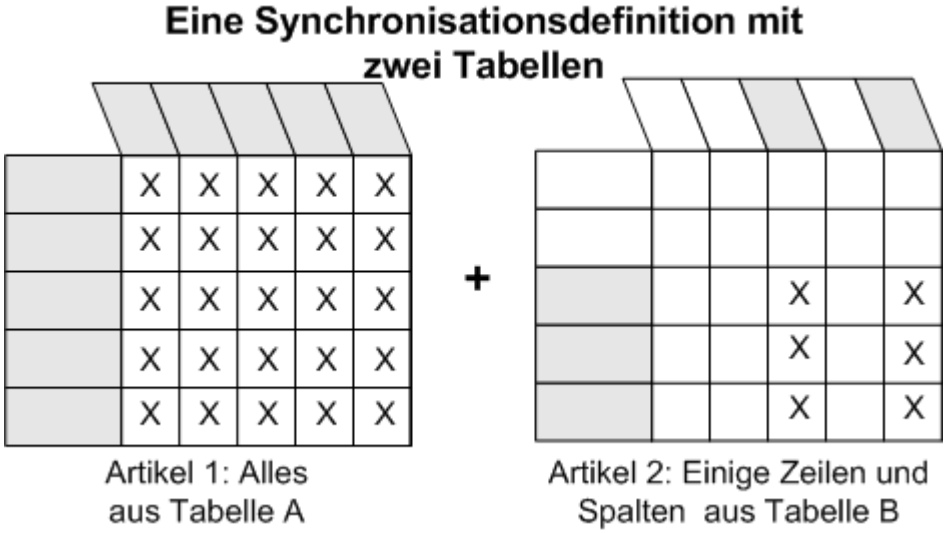
   Erstellen Sie die entfernten Datenbanken und installieren Sie die entsprechende Software.

Siehe auch

- „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Mehrfach vorhandene Primärschlüssel“ auf Seite 55
- „Publikationen und Artikel“ auf Seite 10
- „PUBLISH-Privileg“ auf Seite 21
- „REMOTE-Privileg“ auf Seite 24
- „Einen Nachrichtentyp erstellen“ auf Seite 115
- „Subskriptionen“ auf Seite 33
- „Transaktionsbasierte Replikation“ auf Seite 35
- „Standardlösung für Aktualisierungskonflikte“ auf Seite 45
- „SQL Remote-Systeme verwalten“ auf Seite 81

Publikationen und Artikel

Eine **Publikation** legt die Datenmenge fest, die repliziert werden soll. Eine Publikation kann Daten aus mehreren Datenbanktabellen enthalten. Ein **Artikel** bezieht sich auf eine Tabelle in einer Publikation. Jeder Artikel in einer Publikation kann aus der gesamten Tabelle oder einer Teilmenge der Zeilen und Spalten in der Tabelle bestehen.



Einschränkungen

Eine Publikation kann keine Ansichten und gespeicherten Prozeduren einbeziehen.

Publikationen und Artikel anzeigen (Sybase Central)

In Sybase Central werden Publikationen im Ordner **Publikationen** im linken Fensterausschnitt angezeigt. Artikel, die Sie für eine Publikation erstellen, erscheinen auf der Registerkarte **Artikel** im rechten Fensterausschnitt, wenn Sie eine Publikation auswählen.

Siehe auch

- „Prozedur-Replikation“ auf Seite 39
- „Trigger-Replikation“ auf Seite 39

Erstellen von Publikationen

Sie erstellen Publikationen basierend auf vorhandenen Tabellen in der konsolidierten Datenbank. Verwenden Sie die folgenden Verfahren, um Publikationen zu erstellen, die aus allen Spalten und Zeilen einer Tabelle bestehen.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur konsolidierten Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie im linken Fensterausschnitt auf den Ordner **Publikationen**.
3. Klicken Sie auf **Datei** » **Neu** » **Publikation**.
4. Geben Sie in das Feld **Wie lautet der Name der neuen Publikation?** einen Namen für die Publikation ein. Klicken Sie auf **Weiter**.
5. Wählen Sie im Fenster **Einen Publikationstyp wählen** den entsprechenden Publikationstyp. Klicken Sie auf **Weiter**.
6. Klicken Sie in der Liste **Tabellen angeben** auf eine oder mehrere Tabellen. Klicken Sie auf **Hinzufügen**.
7. Klicken Sie auf **Fertig stellen**.

Ergebnisse

Die Publikation wird wie angegeben erstellt.

Siehe auch

- „CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Nur einige Spalten einer Tabelle publizieren

Erstellen Sie eine Publikation, die alle Zeilen, aber nur einige der Spalten einer Tabelle enthält.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur konsolidierten Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Erweitern Sie im linken Fensterausschnitt den Ordner **Publikationen**.
3. Klicken Sie auf **Datei** » **Neu** » **Publikation**.
4. Geben Sie in das Feld **Wie lautet der Name der neuen Publikation?** einen Namen für die Publikation ein. Klicken Sie auf **Weiter**.
5. Wählen Sie im Fenster **Einen Publikationstyp wählen** den entsprechenden Publikationstyp. Klicken Sie auf **Weiter**.
6. Klicken Sie in der Liste **Tabellen angeben** auf eine oder mehrere Tabellen. Klicken Sie auf **Hinzufügen**.
7. Klicken Sie auf der Registerkarte **Verfügbare Spalten** auf das Symbol der Tabelle, um die Liste **Verfügbare Spalten** zu erweitern. Wählen Sie die Spalten aus, die Sie publizieren möchten, und klicken Sie auf **Hinzufügen**. Klicken Sie auf **Weiter**.
8. Wählen Sie im Fenster **Uploadprozeduren erstellen**, welche gespeicherten Prozeduren erstellt werden sollen.
9. Klicken Sie auf **Fertig stellen**.

Ergebnisse

Die Publikation wird wie angegeben erstellt.

Nächste Schritte

Erstellen Sie eine Subskription.

Siehe auch

- „Subskriptionen“ auf Seite 33
- „CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Fehler bei der referenziellen Integrität“ auf Seite 53

Nur einige Zeilen einer Tabelle publizieren

Um eine Publikation zu erstellen, die nur einige Zeilen einer Tabelle enthält, müssen Sie eine Suchbedingung schreiben, die nur auf die Zeilen zutrifft, die Sie publizieren wollen. Verwenden Sie eine der folgenden Klauseln in Ihrer Suchbedingung:

- **SUBSCRIBE BY-Klausel** Verwenden Sie die SUBSCRIBE BY-Klausel, wenn mehrere Subskribenten unterschiedliche Zeilen aus einer Tabelle erhalten.

  Die SUBSCRIBE BY-Klausel wird empfohlen, wenn Ihr SQL Remote-System eine große Anzahl von Subskriptionen erfordert. Im Gegensatz zur WHERE-Klausel erlaubt die SUBSCRIBE BY-Klausel viele Subskriptionen, die einer einzigen Publikation zugeordnet sind. Subskribenten erhalten Zeilen abhängig vom Wert eines angeführten Ausdrucks.

  Publikationen, die eine SUBSCRIBE BY-Klausel verwenden, sind kompakter, leichter verständlich und bieten eine bessere Performance als solche, die mehrere WHERE-Klauseln verwenden.

- **WHERE-Klausel** Verwenden Sie eine WHERE-Klausel, um eine Teilmenge von Zeilen in einen Artikel einzubeziehen. Alle Subskribenten für die Publikation, die diesen Artikel enthält, erhalten die Zeilen, die der WHERE-Klausel entsprechen.

  Alle nicht publizierten Zeilen müssen einen Standardwert haben. Ansonsten tritt ein Fehler auf, wenn die entfernte Datenbank versucht, neue Zeilen aus der konsolidierten Datenbank einzufügen.

  Sie können eine WHERE-Klausel in einem Artikel verwenden.

  Der Datenbankserver muss, direkt proportional zu der Anzahl der Publikationen, dem Transaktionslog Informationen hinzufügen und das Transaktionslog durchsuchen, um Nachrichten zu versenden. Im Gegensatz zur SUBSCRIBE BY-Klausel, erlaubt es die WHERE-Klausel nicht, dass viele Subskriptionen einer einzigen Publikation zugeordnet werden.

Beispiel

Sie benötigen eine Publikation, die es jedem Handelsvertreter ermöglicht, Folgendes durchzuführen:

- Seine Bestellungen zu subskribieren.
- Seine Bestellungen lokal zu aktualisieren.
- Seine Bestellungen an die konsolidierte Datenbank zu replizieren.

Wenn Sie die WHERE-Klausel verwenden, müssten Sie eine separate Publikation für jeden Handelsvertreter erstellen. Die folgende Publikation gilt für einen Handelsvertreter namens Sam Singer, und jeder andere Handelsvertreter würde eine ähnliche Publikation benötigen.

```
CREATE PUBLICATION PubOrdersSamSinger (
   TABLE SalesOrders
      WHERE Active = 1
);
```

Die folgende Anweisung subskribiert Sam Singer für die Publikation "PubsOrdersSamSinger".

```
CREATE SUBSCRIPTION
TO PubOrdersSamSinger
FOR Sam_Singer;
```

Wenn Sie die SUBSCRIBE BY-Klausel verwenden, benötigen Sie nur eine Publikation. Alle Handelsvertreter können die folgende Publikation verwenden:

```
CREATE PUBLICATION PubOrders (
   TABLE SalesOrders
   SUBSCRIBE BY SalesRepresentativeID
);
```

Die folgende Anweisung subskribiert Sam Singer für die Publikation "PubsOrders" anhand seiner ID, 8887.

```
CREATE SUBSCRIPTION
TO PubOrders ('8887')
FOR Sam_Singer;
```

Siehe auch

- „Publizieren von nur einigen Zeilen mit der SUBSCRIBE BY-Klausel“ auf Seite 14
- „Nur einige Zeilen mittels einer WHERE-Klausel publizieren“ auf Seite 15
- „CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Publizieren von nur einigen Zeilen mit der SUBSCRIBE BY-Klausel

Erstellen einer Publikation mithilfe der SUBSCRIBE BY-Klausel.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Hinweise zur Verwendung der SUBSCRIBE BY-Klausel und ihrer Alternative, der WHERE-Klausel, finden Sie unter „Nur einige Zeilen einer Tabelle publizieren“ auf Seite 13.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur konsolidierten Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.

2. Doppelklicken Sie im linken Fensterausschnitt auf **Publikationen**.
3. Klicken Sie auf **Datei** » **Neu** » **Publikation**.
4. Geben Sie in das Feld **Wie lautet der Name der neuen Publikation?** einen Namen für die Publikation ein. Klicken Sie auf **Weiter**.
5. Klicken Sie auf **Weiter**.
6. Wählen Sie in der Liste **Verfügbare Tabellen** eine Tabelle. Klicken Sie auf **Hinzufügen**. Klicken Sie auf **Weiter**.
7. Klicken Sie auf der Registerkarte **Verfügbare Spalten** auf das Symbol der Tabelle, um die Liste **Verfügbare Spalten** zu erweitern. Klicken Sie auf die Spalten, die Sie publizieren möchten, und klicken Sie auf **Hinzufügen** (halten Sie beim Klicken die STRG-Taste gedrückt, um mehrere Spalten auszuwählen). Klicken Sie auf **Weiter**.
8. Klicken Sie auf **Weiter**.
9. Auf der Seite **SUBSCRIBE BY-Einschränkungen angeben** führen Sie Folgendes durch:
   a. Klicken Sie auf eine Tabelle in der Liste **Artikel**.
   b. Klicken Sie auf **Spalte** und wählen Sie in der Dropdown-Liste eine Spalte aus.
10. Klicken Sie auf **Fertig stellen**.

Ergebnisse

Die Publikation wird wie angegeben erstellt.

Nächste Schritte

Einen Benutzer für die Publikation subskribieren.

Siehe auch

- „SQL Remote-Subskriptionen erstellen“ auf Seite 34
- „Nur einige Zeilen mittels einer WHERE-Klausel publizieren“ auf Seite 15
- „Disjunkte Datenpartitionen“ auf Seite 63
- „CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Nur einige Zeilen mittels einer WHERE-Klausel publizieren

Erstellen Sie eine Publikation, die eine WHERE-Klausel verwendet, um alle Spalten, aber nur einige Zeilen einer Tabelle einzubeziehen.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Benutzer können mehr als eine Publikation subskribieren und mehr als eine Subskription für eine einzelne Publikation haben.

Hinweise zur Verwendung der WHERE-Klausel und ihrer Alternative, der SUBSCRIBE BY-Klausel, finden Sie unter „Nur einige Zeilen einer Tabelle publizieren“ auf Seite 13.

Aufgabe

1. Verwenden Sie Sybase Central, um eine Verbindung zur konsolidierten Datenbank herzustellen.
2. Doppelklicken Sie auf den Ordner **Publikationen**.
3. Klicken Sie auf **Datei** » **Neu** » **Publikation**.
4. Geben Sie in das Feld **Wie lautet der Name der neuen Publikation?** einen Namen für die Publikation ein. Klicken Sie auf **Weiter**.
5. Klicken Sie auf **Weiter**.
6. Wählen Sie in der Liste **Verfügbare Tabellen** eine Tabelle. Klicken Sie auf **Hinzufügen**. Klicken Sie auf **Weiter**.
7. Doppelklicken Sie auf der Registerkarte **Verfügbare Spalten** auf das Symbol dieser Tabelle, um die Liste **Verfügbare Spalten** zu erweitern. Wählen Sie die Spalten aus, die Sie publizieren möchten, und klicken Sie auf **Hinzufügen**. Klicken Sie auf **Weiter**.
8. Auf der Seite **WHERE-Klauseln angeben** führen Sie Folgendes durch:
   a. Klicken Sie auf eine Tabelle in der Liste **Artikel**.
   b. Geben Sie eine WHERE-Klausel in das Feld **Der markierte Artikel hat folgende WHERE-Klausel** ein.
9. Klicken Sie auf **Fertig stellen**.

Ergebnisse

Die Publikation wird wie angegeben erstellt.

Nächste Schritte

Fügen Sie eine Subskription hinzu.

Siehe auch

- „Subskriptionen“ auf Seite 33
- WHERE-Klausel und Primärschlüssel auf Seite 52
- „CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Publikation ändern

Ändern Sie eine Publikation, indem Sie Artikel hinzufügen, ändern bzw. löschen oder indem Sie die Publikation umbenennen.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Vorsicht
Das Ändern von Publikationen in einem laufenden SQL Remote-System kann zu Replikationsfehlern und zu Datenverlusten im Replikationssystem führen. Siehe „Upgrades und Resynchronisation“ auf Seite 147.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf **Publikationen**.
3. Rechtsklicken Sie auf die Publikation, die Sie ändern möchten, und klicken Sie auf **Eigenschaften**, um die Publikation zu bearbeiten.

Ergebnisse

Die Publikation wird geändert.

Siehe auch

- „ALTER PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Publikation löschen

Wenn Sie eine Publikation löschen, werden alle Subskriptionen für diese Publikation automatisch gelöscht.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Vorsicht
Das Löschen von Publikationen in einem laufenden SQL Remote-System kann zu Replikationsfehlern und zu Datenverlusten im Replikationssystem führen. Siehe „Upgrades und Resynchronisation“ auf Seite 147.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf **Publikationen**.
3. Rechtsklicken Sie auf die gewünschte Publikation und klicken Sie auf **Löschen**.

Ergebnisse

Die Publikation wird gelöscht und alle Subskriptionen für diese Publikation werden gelöscht.

Siehe auch

- „DROP PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Benutzerprivilegien

SQL Remote verwendet ein konsistentes System, um die Benutzer zu verwalten, die Privilegien für entfernte und konsolidierte Datenbanken haben.

Benutzer von Datenbanken, die in ein SQL Remote-Replikationssystem eingebunden sind, benötigen das MANAGE REPLICATION-Systemprivileg (Teil der SYS_REPLICATION_ADMIN_ROLE-Systemrolle), um die folgenden Privilegien zu erteilen oder zu entziehen:

- **PUBLISH** Jede Datenbank in einem SQL Remote-System publiziert Informationen. Daher muss jede Datenbank einen Publikationseigentümer haben. Um einen Publikationseigentümer zu erstellen, erteilen Sie einem Benutzer das PUBLISH-Privileg. Der Publikationseigentümer muss im gesamten SQL Remote-System eindeutig sein. Beim Versenden von Daten repräsentiert der Publikationseigentümer die Datenbank. Wenn eine Datenbank eine Nachricht versendet, wird der Publikationseigentümer-Benutzername in die Nachricht aufgenommen. Wenn eine Datenbank eine Nachricht erhält, kann sie die Datenbank, die die Nachricht gesendet hat, anhand des Publikationseigentümer-Namens in der Nachricht identifizieren.
- **REMOTE** Eine Datenbank, wie z.B. eine konsolidierte Datenbank, die Nachrichten an andere Datenbanken sendet, muss angeben, an welche Datenbanken sie Nachrichten sendet. Um diese

entfernten Datenbanken in der konsolidierten Datenbank anzugeben, erteilen Sie den Publikationseigentümern der entfernten Datenbanken das REMOTE-Privileg. Das REMOTE-Privileg identifiziert Datenbanken, die Nachrichten von der aktuellen Datenbank erhalten.

- **CONSOLIDATE** Jede entfernte Datenbank muss die konsolidierte Datenbank angeben, von der sie Nachrichten empfängt. Um eine konsolidierte Datenbank auf der entfernten Datenbank anzugeben, erteilen Sie dem Publikationseigentümer der konsolidierten Datenbank das CONSOLIDATE-Privileg. Eine entfernte Datenbank kann nur von einer einzigen konsolidierten Datenbank Nachrichten erhalten. Das CONSOLIDATE-Privileg identifiziert die Datenbank, die Nachrichten an diese entfernte Datenbank sendet.

Das Extraktionsdienstprogramm (dbxtract) setzt automatisch Privilegien.

Standardmäßig erteilen das Extraktionsdienstprogramm (dbxtract) und der **Assistent zum Extrahieren einer Datenbank** Benutzern in den entfernten Datenbanken die entsprechenden PUBLISH- und CONSOLIDATE-Privilegien.

Siehe auch

- „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214

Einschichtige Hierarchie

In einer einschichtigen Hierarchie gibt es eine konsolidierte Datenbank mit einer oder mehreren darunterliegenden entfernten Datenbanken. In einer solchen Hierarchie erteilt die konsolidierte Datenbank das REMOTE-Privileg an die Publikationseigentümer der entfernten Datenbanken. Jede entfernte Datenbank erteilt dem Publikationseigentümer der konsolidierten Datenbank das CONSOLIDATE-Privileg.

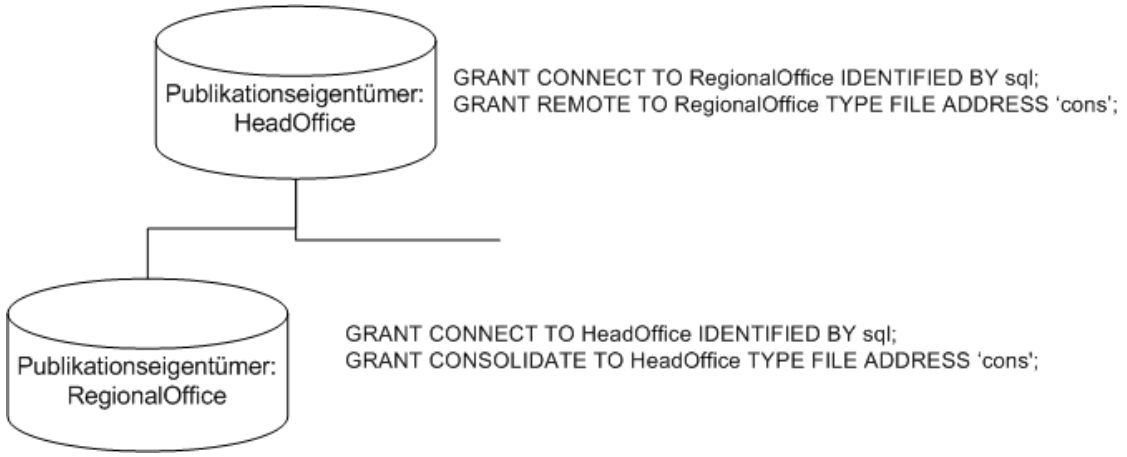
Beispiel: Eine konsolidierte Datenbank ist durch ihren Publikationseigentümer HeadOffice und eine entfernte Datenbank durch ihren Publikationseigentümer RegionalOffice gekennzeichnet.

In der konsolidierten Datenbank HeadOffice führen Sie Folgendes durch:

- Erstellen Sie einen Benutzer mit demselben Namen wie der Publikationseigentümer der entfernten Datenbank: RegionalOffice.
- Erteilen Sie das REMOTE-Privileg an RegionalOffice. Dies kennzeichnet RegionalOffice als Datenbank, die Nachrichten von HeadOffice empfängt.

In der entfernten Datenbank führen Sie Folgendes durch:

- Erstellen Sie einen Benutzer mit demselben Namen wie der Publikationseigentümer der konsolidierten Datenbank: HeadOffice.
- Erteilen Sie das CONSOLIDATE-Privileg an HeadOffice. Dies kennzeichnet HeadOffice als konsolidierte Datenbank für RegionalOffice, d.h. HeadOffice ist die Datenbank, die Nachrichten an RegionalOffice sendet.



Dbxtract setzt Privilegien automatisch.

Standardmäßig erteilen das Extraktionsdienstprogramm (dbxtract) und der **Assistent zum Extrahieren einer Datenbank** Benutzern in den entfernten Datenbanken die entsprechenden PUBLISH- und CONSOLIDATE-Privilegien.

Siehe auch

- „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214

Mehrschichtige Hierarchie

In einer mehrschichtigen Hierarchie wird allen entfernten Datenbanken unmittelbar unterhalb der aktuellen Datenbank das REMOTE-Privileg erteilt. Der Datenbank unmittelbar oberhalb der aktuellen Datenbank wird das CONSOLIDATE-Privileg erteilt.

Beispiel: Eine konsolidierte Datenbank wird durch ihren Publikationseigentümer HeadOffice gekennzeichnet und hat eine entfernte Datenbank namens RegionalOffice. Die RegionalOffice-Datenbank hat allerdings auch eine entfernte Datenbank namens Office.

In der konsolidierten Datenbank HeadOffice führen Sie Folgendes durch:

- Erstellen Sie einen Benutzer mit demselben Namen wie der Publikationseigentümer der entfernten Datenbank RegionalOffice.
- Erteilen Sie dem Benutzer RegionalOffice das REMOTE-Privileg. Dies kennzeichnet RegionalOffice als Datenbank, die Nachrichten von HeadOffice empfängt.

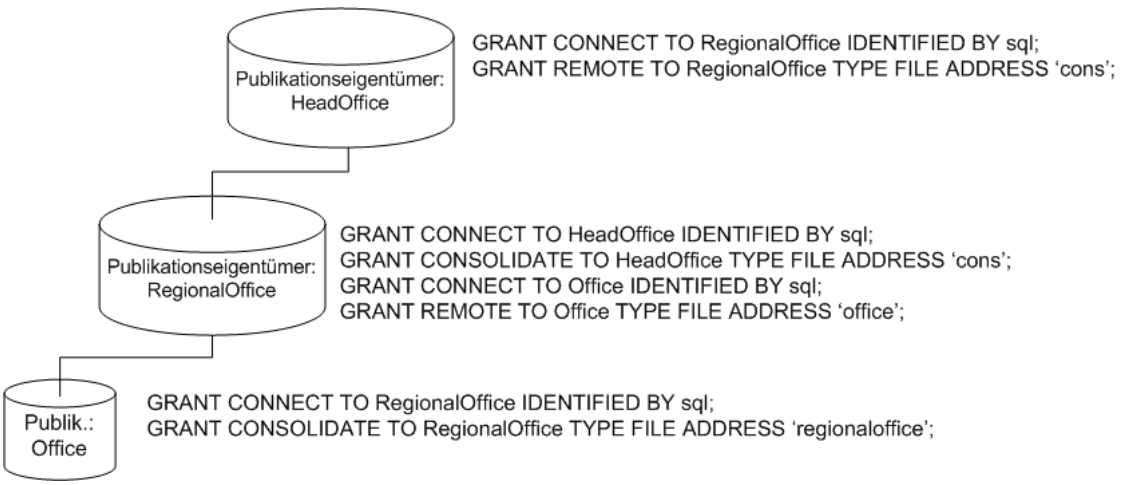
In der RegionalOffice-Datenbank führen Sie Folgendes durch:

- Erstellen Sie einen Benutzer mit demselben Namen wie der Publikationseigentümer der konsolidierten Datenbank HeadOffice.

- Erteilen Sie das CONSOLIDATE-Privileg an HeadOffice. Dies kennzeichnet HeadOffice als konsolidierte Datenbank für RegionalOffice, d.h. HeadOffice ist die Datenbank, die Nachrichten an RegionalOffice sendet.
- Erstellen Sie einen Benutzer mit demselben Namen wie die Datenbank unmittelbar unterhalb von RegionalOffice: Office.
- Erteilen Sie Office das REMOTE-Privileg. Dies kennzeichnet Office als Datenbank, die Nachrichten von RegionalOffice empfängt.

In der Office-Datenbank führen Sie Folgendes durch:

- Erstellen Sie einen Benutzer mit demselben Namen wie der Publikationseigentümer der konsolidierten Datenbank: RegionalOffice.
- Erteilen Sie dem RegionalOffice-Benutzer das CONSOLIDATE-Privileg. Dies kennzeichnet RegionalOffice als konsolidierte Datenbank für Office, d.h. dass RegionalOffice Nachrichten an Office sendet.



PUBLISH-Privileg

Jede Datenbank in einem SQL Remote-System erfordert einen Publikationseigentümer, der ein eindeutiger Benutzer mit PUBLISH-Privileg ist. Alle ausgehenden SQL Remote-Nachrichten, einschließlich Publikationsaktualisierungen und Empfangsbestätigungen, werden durch ihren Publikationseigentümer gekennzeichnet. Jede Datenbank in einem SQL Remote-System versendet Empfangsbestätigungen.

Das PUBLISH-Privileg besitzt keine weitere Berechtigung, als den Publikationseigentümer von versendeten Nachrichten zu kennzeichnen.

Wenn das PUBLISH-Privileg einer benutzererweiterten Rolle erteilt wird, wird es nicht an die anderen Mitglieder dieser Rolle vererbt.

Sie erstellen einen Publikationseigentümer, indem Sie einem Benutzer das PUBLISH-Privileg erteilen.

Erstellen eines Publikationseigentümers

Benutzer einrichten und ihnen das PUBLISH-Privileg erteilen.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Wenn eine entfernte Datenbank durch das Extraktionsdienstprogramm (dbxtract) oder den **Assistenten zum Extrahieren einer Datenbank** extrahiert wird, wird der entfernte Benutzer zum Publikationseigentümer der entfernten Datenbank, und ihm wird das PUBLISH-Privileg erteilt.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf den Datenbanknamen.
3. Doppelklicken Sie auf **Benutzer**.
4. Rechtsklicken Sie auf den Benutzernamen.
5. Klicken Sie auf **Auf Publikationseigentümer ändern**

Ergebnisse

Der Benutzer wird erstellt und erhält das PUBLISH-Privileg.

Siehe auch

- „GRANT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „GRANT PUBLISH-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „PUBLISH-Privileg entziehen“ auf Seite 22
- „Anzeigen des Publikationseigentümers“ auf Seite 23

PUBLISH-Privileg entziehen

Entziehen Sie einem Benutzer das PUBLISH-Privileg.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Vorsicht
Das Ändern eines Publikationseigentümers bei einer entfernten Datenbank oder bei der konsolidierten Datenbank kann schwerwiegende Probleme für Subskriptionen bewirken, die die Datenbank betreffen, einschließlich Datenverlust. Siehe „In einem laufenden System zu vermeidende Änderungen“ auf Seite 147.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf den Datenbanknamen.
3. Doppelklicken Sie auf **Benutzer**.
4. Rechtsklicken Sie auf den Benutzernamen.
5. Klicken Sie auf **Publikationseigentümerschaft entziehen**

Ergebnisse

Der Benutzer hat nun kein PUBLISH-Privileg mehr.

Siehe auch

- „PUBLISH-Privileg“ auf Seite 21
- „REVOKE PUBLISH-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Anzeigen des Publikationseigentümers“ auf Seite 23

Anzeigen des Publikationseigentümers

Benutzer identifizieren, die Publikationseigentümer sind.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

- Doppelklicken Sie auf **Benutzer**.

  Der Publikationseigentümer ist der Benutzer, dessen **Typ Publikationseigentümer** ist.

Ergebnisse

Sie können die Benutzer sehen, die Publikationseigentümer sind.

Siehe auch

- „PUBLISH-Privileg“ auf Seite 21
- „PUBLISH-Privileg entziehen“ auf Seite 22
- „db_publisher-Option“ [*SQL Anywhere Server - Datenbankadministration*]

REMOTE-Privileg

Das Erteilen von REMOTE-Privilegien wird auch als Hinzufügen eines entfernten Benutzers in die Datenbank bezeichnet. Publikationseigentümern von Datenbanken unmittelbar unterhalb der aktuellen Datenbank in einer SQL Remote-Hierarchie wird das REMOTE-Privileg durch die aktuelle Datenbank erteilt.

Wenn Sie einem Benutzer ein REMOTE-Privileg erteilen, müssen Sie die folgenden Einstellungen konfigurieren:

- **Nachrichtensystem** Sie können erst dann einen neuen entfernten Benutzer erstellen, wenn zumindest ein Nachrichtensystem in der Datenbank festgelegt wurde.
- **Sendefrequenz** Wenn Sie SQL-Anweisungen verwenden, um ein REMOTE-Privileg zu erteilen, ist das Festlegen einer Sendefrequenz optional.

Einem Benutzer das REMOTE-Privileg zu erteilen, bewirkt Folgendes:

- Kennzeichnet den Benutzer als entfernten Benutzer.
- Gibt einen Nachrichtentyp an, der beim Austausch von Nachrichten mit diesem entfernten Benutzer verwendet wird.
- Stellt eine Adresse bereit, an die Nachrichten zu senden sind.
- Gibt an, wie oft Nachrichten an den entfernten Benutzer zu senden sind.

Der Publikationseigentümer einer Datenbank kann nicht REMOTE- und CONSOLIDATE-Privilegien in derselben Datenbank haben. Dies würde den Publikationseigentümer sowohl als Sender als auch als Empfänger von ausgehenden Nachrichten kennzeichnen.

REMOTE-Privileg an Gruppen erteilen

Obwohl Sie das REMOTE-Privileg einer benutzererweiterten Rolle erteilen können, wird das REMOTE-Privileg *nicht* an die Berechtigungsempfänger der Rolle weitervererbt. Sie müssen das REMOTE-Privileg jedem Berechtigungsempfänger einer benutzererweiterten Rolle ausdrücklich erteilen.

Weitere Hinweise finden Sie auch unter

- „SQL Remote-Nachrichtensysteme“ auf Seite 114
- „Sendefrequenz“ auf Seite 93
- „GRANT REMOTE-Anweisung [SQLRemote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

REMOTE-Privilegien erteilen

Einen entfernten Benutzer hinzufügen oder einen vorhandenen Benutzer in einen entfernten Benutzer umwandeln.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf den Datenbanknamen.
3. Doppelklicken Sie auf **Benutzer**.
4. Rechtsklicken Sie auf den Benutzernamen.
5.

Option	Aktion
Fügen Sie einen neuen entfernten Benutzer hinzu.	a. Klicken Sie auf **Datei** » **Neu** » **Benutzer**. b. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen von entfernten Benutzern**. c. Rechtsklicken Sie auf den Benutzer und klicken Sie auf **Auf entfernten Benutzer ändern**.
Vorhandenen Benutzer in einen entfernten Benutzer umwandeln	a. Rechtsklicken Sie auf einen Benutzer und klicken Sie auf **Auf entfernten Benutzer ändern**. b. Im Fenster klicken Sie auf den Nachrichtentyp, geben eine Adresse ein, legen die Sendefrequenz fest und klicken auf **OK**.

Ergebnisse

Der entfernte Benutzer wird erstellt.

Siehe auch

- „GRANT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „GRANT REMOTE-Anweisung [SQLRemote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REMOTE-Privileg entziehen“ auf Seite 25
- „CREATE USER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

REMOTE-Privileg entziehen

Entfernen Sie einen Benutzer oder eine Rolle aus dem SQL Remote-System, wandeln Sie den Benutzer oder die Rolle in einen normalen Benutzer/eine normale Rolle um und heben Sie die Subskription des Benutzers oder der Rolle für alle Publikationen auf.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

So entziehen Sie einem Benutzer oder einer Gruppe das REMOTE-Privileg:

- Der Benutzer wird aus dem SQL Remote-System entfernt.
- Der Benutzer bzw. die Gruppe wird zu einem Benutzer oder einer Gruppe mit normalen Berechtigungen.
- Die Subskriptionen des Benutzers bzw. der Gruppe für alle Publikationen werden beendet.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf den Datenbanknamen.
3. Doppelklicken Sie auf **Benutzer**.
4. Rechtsklicken Sie auf den Benutzernamen.
5. Klicken Sie auf **Status als entfernter Benutzer entziehen**.

Ergebnisse

Der Benutzer bzw. die Gruppe wird auf einen normalen Benutzer/eine normale Gruppe zurückgesetzt, aus dem SQL Remote-System entfernt und die Subskriptionen für alle Publikationen werden beendet.

Siehe auch

- „REMOTE-Privilegien erteilen“ auf Seite 25
- „REVOKE REMOTE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

CONSOLIDATE-Privileg

Datenbanken unmittelbar oberhalb der aktuellen Datenbank in einer SQL Remote-Hierarchie wird das CONSOLIDATE-Privileg durch die aktuelle Datenbank erteilt. In jeder entfernten Datenbank muss der konsolidierten Datenbank das CONSOLIDATE-Privileg erteilt werden.

Das CONSOLIDATE-Privileg muss auch von schreibgeschützten entfernten Datenbanken an die konsolidierte Datenbank erteilt werden, da Empfangsbestätigungen von den entfernten Datenbanken an die konsolidierte Datenbank gesendet werden.

Wenn Sie einem Benutzer ein CONSOLIDATE-Privileg erteilen, müssen Sie die folgenden Einstellungen konfigurieren:

- **Nachrichtensystem** Sie können erst dann einen neuen konsolidierten Benutzer erstellen, wenn zumindest ein Nachrichtensystem in der Datenbank festgelegt wurde.
- **Sendefrequenz** Wenn Sie SQL-Anweisungen verwenden, um ein CONSOLIDATE-Privileg zu erteilen, ist das Festlegen einer Sendefrequenz optional.

CONSOLIDATE-Privileg erteilen:

- Ein Benutzer wird als konsolidierter Benutzer gekennzeichnet.
- Ein Nachrichtentyp wird festgelegt, der beim Austausch von Nachrichten mit diesem konsolidierten Benutzer verwendet wird.
- Stellt eine Adresse bereit, an die Nachrichten zu senden sind.
- Es wird festgelegt, wie oft Nachrichten an den konsolidierten Benutzer zu senden sind.

Der Publikationseigentümer einer Datenbank kann nicht REMOTE- und CONSOLIDATE-Privilegien in derselben Datenbank haben. Dies würde den Publikationseigentümer sowohl als Sender als auch als Empfänger von ausgehenden Nachrichten kennzeichnen.

Extraktionsdienstprogramm (dbxtract)

Wenn eine entfernte Datenbank durch das Extraktionsdienstprogramm (dbxtract) oder den **Assistenten zum Extrahieren einer Datenbank** extrahiert wird, wird die GRANT CONSOLIDATE-Anweisung automatisch in der entfernten Datenbank ausgeführt.

Siehe auch

- „SQL Remote-Nachrichtensysteme“ auf Seite 114
- „Sendefrequenz“ auf Seite 93

CONSOLIDATE-Privileg erteilen

CONSOLIDATE-Privileg an einen Benutzer erteilen

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Es wird empfohlen, dass Sie dem Publikationseigentümer der konsolidierten Datenbank das CONSOLIDATE-Privileg erteilen.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.

2. Doppelklicken Sie auf den Datenbanknamen.
3. Doppelklicken Sie auf **Benutzer**.
4. Rechtsklicken Sie auf den Benutzernamen.
5. Klicken Sie auf **Auf konsolidierten Benutzer ändern**.
6. Konfigurieren Sie die Einstellungen **Nachrichtentyp**, **Adresse** und **Sendefrequenz**.
7. Klicken Sie auf **OK**, um das Fenster **Auf konsolidierten Benutzer ändern** zu schließen.

Ergebnisse

Der Benutzer hat nun das CONSOLIDATE-Privileg.

Siehe auch

- „GRANT-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „GRANT CONSOLIDATE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CONSOLIDATE-Privileg entziehen“ auf Seite 28

CONSOLIDATE-Privileg entziehen

Einem Benutzer der Datenbank das CONSOLIDATE-Privileg entziehen.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Wenn Sie einem Benutzer das CONSOLIDATE-Privileg entziehen, führt SQL Anywhere Folgendes durch:

- Der Benutzer wird aus dem SQL Remote-System entfernt.
- Der Benutzer bzw. die Gruppe wird zu einem Benutzer oder einer Gruppe mit normalen Berechtigungen.
- Die Subskriptionen des Benutzers bzw. der Gruppe für alle Publikationen werden beendet.

Aufgabe

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

3. Doppelklicken Sie auf **Benutzer**.
4. Rechtsklicken Sie auf den konsolidierten Benutzer bzw. die konsolidierte Gruppe und klicken Sie auf **Status als konsolidierter Benutzer entziehen**.

Ergebnisse

Der Benutzer bzw. die Gruppe wird auf einen normalen Benutzer/eine normale Gruppe zurückgesetzt, aus dem SQL Remote-System entfernt und die Subskriptionen für alle Publikationen werden beendet.

Siehe auch

- „CONSOLIDATE-Privileg erteilen“ auf Seite 27
- „REVOKE CONSOLIDATE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

SYS_RUN_REPLICATION_ROLE-Systemrolle

Benutzer mit der SYS_RUN_REPLICATION_ROLE-Systemrolle haben nur dann vollständige Administratorprivilegien in der Datenbank, wenn sie eine Verbindung von SQL Remote aus herstellen. SQL Remote-Benutzer haben vollständigen Zugriff auf die Datenbank und können Änderungen durchführen, die in den Nachrichten angegeben sind. Nur Benutzer mit diesen Privilegien können SQL Remote ausführen.

Die SYS_RUN_REPLICATION_ROLE-Systemrolle hat folgende Eigenschaften:

- **Keine spezifischen Privilegien außer bei einer SQL Remote-Verbindung** Ein Benutzer, der die SYS_RUN_REPLICATION_ROLE-Systemrolle erteilt bekommen hat, kann die von dieser Rolle geerbten Privilegien nicht für eine Verbindung ausüben, außer für SQL Remote oder dbmlsync. Daher besteht sogar dann, wenn der Benutzername und das Kennwort für einen Benutzer mit der SYS_RUN_REPLICATION_ROLE-Systemrolle weithin verteilt sind, kein Sicherheitsrisiko. Solange dem Benutzernamen keine Privilegien außer CONNECT für die Datenbank zugeordnet sind, kann niemand diesen Benutzernamen verwenden, um auf Daten in der Datenbank zuzugreifen.
- **Vollständige Datenbank-Administrationsprivilegien von SQL Remote** Bei einer Verbindung aus SQL Remote hat ein Benutzer mit der SYS_RUN_REPLICATION_ROLE-Systemrolle vollständige Datenbank-Administrationsprivilegien für die Datenbank.

Wann die SYS_RUN_REPLICATION_ROLE-Systemrolle zu erteilen ist

Wenn Sie Benutzer in der konsolidierten Datenbank erstellen, wird empfohlen, dass Sie die SYS_RUN_REPLICATION_ROLE-Systemrolle dem Publikationseigentümer der konsolidierten Datenbank sowie jedem entfernten Benutzer erteilen. Wenn eine entfernte Datenbank durch das Extraktionsdienstprogramm (dbxtract) oder den **Assistenten zum Extrahieren einer Datenbank** extrahiert wird, wird der entfernte Benutzer zum Publikationseigentümer der entfernten Datenbank, und ihm wird das PUBLISH-Privileg und die SYS_RUN_REPLICATION_ROLE-Systemrolle erteilt.

Diese Empfehlung vereinfacht die Verwaltung der Benutzer. Jeder entfernte Benutzer benötigt nur einen Benutzernamen, um sich mit der Datenbank zu verbinden, unabhängig davon, ob er die Verbindung aus SQL Remote (was dem Benutzer die SYS_RUN_REPLICATION_ROLE-Systemrolle erteilt) oder aus

einer anderen Clientanwendung herstellt (wobei die SYS_RUN_REPLICATION_ROLE-Systemrolle dem Benutzer keine weiteren Privilegien erteilt).

Hinweis
GRANT REMOTE DBA wird nicht mehr empfohlen. Verwenden Sie stattdessen GRANT ROLE SYS_RUN_REPLICATION_ROLE. Siehe „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*].

Siehe auch

- „SYS_RUN_REPLICATION_ROLE-Systemrolle erteilen“ auf Seite 30
- „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

SYS_RUN_REPLICATION_ROLE-Systemrolle erteilen

Erteilen Sie die SYS_RUN_REPLICATION_ROLE-Systemrolle einem Benutzer, wodurch er vollen Zugriff auf die Datenbank erhält und Änderungen durchführen kann, die in den Nachrichten angegeben werden.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Nur Benutzer mit der SYS_RUN_REPLICATION_ROLE-Systemrolle können SQL Remote ausführen.

In Sybase Central ist die Vererbung von Systemprivilegien standardmäßig deaktiviert. Die Systemprivilegien stehen nur der Rolle zur Verfügung, nicht aber den Rollenempfängern.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf den Datenbanknamen.
3. Doppelklicken Sie auf **Benutzer**.
4. Doppelklicken Sie auf den Benutzernamen.
5. Klicken Sie mit der rechten Maustaste auf den Fensterausschnitt (nicht den Benutzernamen) und wählen Sie **Neu** » **Erteilte Rollen**.
6. Wählen Sie **SYS_RUN_REPLICATION_ROLE** aus und klicken Sie auf **OK**.

Ergebnisse

Der Benutzer verfügt nun über die SYS_RUN_REPLICATION_ROLE-Systemrolle in der Datenbank, wenn er sich von SQL Remote aus verbindet.

Weitere Hinweise finden Sie auch unter

- „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYS_RUN_REPLICATION_ROLE-Systemrolle entziehen“ auf Seite 31
- „Änderungen des Vererbungsverhaltens für einige Berechtigungen, die zu Rollen geworden sind“ [*SQL Anywhere Server - Datenbankadministration*]

SYS_RUN_REPLICATION_ROLE-Systemrolle entziehen

Die SYS_RUN_REPLICATION_ROLE-Systemrolle einem Benutzer entziehen

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf den Datenbanknamen.
3. Doppelklicken Sie auf **Benutzer**.
4. Doppelklicken Sie auf den Benutzernamen.
5. Rechtsklicken Sie auf der Registerkarte **Rollen** auf **SYS_RUN_REPLICATION_ROLE** und klicken Sie auf **Löschen**.

Ergebnisse

Der Benutzer verfügt nicht mehr über die SYS_RUN_REPLICATION_ROLE-Systemrolle.

Weitere Hinweise finden Sie auch unter

- „SYS_RUN_REPLICATION_ROLE-Systemrolle“ auf Seite 29
- „SYS_RUN_REPLICATION_ROLE-Systemrolle erteilen“ auf Seite 30
- „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „GRANT ROLE SYS_REPLICATION_ADMIN_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REVOKE ROLE SYS_REPLICATION_ADMIN_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REMOTE-Privilegien erteilen“ auf Seite 25

SYS_REPLICATION_ADMIN_ROLE-Systemrolle

Benutzer mit der SYS_REPLICATION_ADMIN_ROLE-Systemrolle können administrative Aufgaben in Bezug auf die Replikation durchführen, wie beispielsweise Replikationsrollen erteilen, Publikationen verwalten, Subskriptionen verwalten, Synchronisationsbenutzer und -profile verwalten, Nachrichtentypen verwalten und replikationsbezogene Optionen festlegen.

SYS_RUN_REPLICATION_ROLE werden die folgenden Systemprivilegien erteilt:

- SELECT ANY TABLE
- SET ANY USER DEFINED OPTION
- SET ANY SYSTEM OPTION
- BACKUP DATABASE
- SET ANY SYSTEM OPTION

Siehe auch

- „SYS_RUN_REPLICATION_ROLE-Systemrolle erteilen“ auf Seite 30
- „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

SYS_REPLICATION_ADMIN_ROLE-Systemrolle erteilen

SYS_REPLICATION_ADMIN_ROLE-Systemrolle einem Benutzer erteilen, sodass er administrative Aufgaben in Bezug auf die Replikation durchführen kann

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf den Datenbanknamen.
3. Doppelklicken Sie auf **Benutzer**.
4. Doppelklicken Sie auf den Benutzernamen.
5. Klicken Sie mit der rechten Maustaste auf den Fensterausschnitt (nicht den Benutzernamen) und wählen Sie **Neu** » **Erteilte Rollen**.
6. Wählen Sie **SYS_REPLICATION_ADMIN_ROLE** aus und klicken Sie auf **OK**.

Ergebnisse

Der Benutzer verfügt nun über die SYS_REPLICATION_ADMIN_ROLE-Systemrolle in der Datenbank, wenn er die Verbindung von SQL Remote aus hergestellt hat.

Weitere Hinweise finden Sie auch unter

- „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

SYS_REPLICATION_ADMIN_ROLE-Systemrolle entziehen

SYS_REPLICATION_ADMIN_ROLE-Systemrolle einem Benutzer entziehen

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf den Datenbanknamen.
3. Doppelklicken Sie auf **Benutzer**.
4. Doppelklicken Sie auf den Benutzernamen.
5. Rechtsklicken Sie auf der Registerkarte **Rollen** auf **SYS_REPLICATION_ADMIN_ROLE** und klicken Sie auf **Löschen**.

Ergebnisse

Der Benutzer verfügt nicht mehr über die SYS_REPLICATION_ADMIN_ROLE-Systemrolle.

Weitere Hinweise finden Sie auch unter

- „SYS_RUN_REPLICATION_ROLE-Systemrolle“ auf Seite 29
- „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REVOKE ROLE SYS_REPLICATION_ADMIN_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REMOTE-Privilegien erteilen“ auf Seite 25

Subskriptionen

Sie subskribieren einen Benutzer für eine Publikation, indem Sie eine **Subskription** erstellen. Jede Datenbank, die Informationen in einer Publikation gemeinsam nutzt, muss eine Subskription für die Publikation besitzen. In regelmäßigen Abständen werden die Änderungen, die in den einzelnen Publikationen einer Datenbank vorgenommen werden, an alle Subskribenten dieser Publikation repliziert. Diese Replikationen werden als **Publikationsaktualisierungen** bezeichnet

Um einen Benutzer für eine Publikation zu subskribieren, benötigen Sie die folgenden Informationen:

- **Publikationsname** Der Name der Publikation, die für den Benutzer subskribiert ist.
- **Subskriptionswert** Der Subskriptionswert wird nur angewendet, wenn Ihre Publikation eine SUBSCRIBE BY-Klausel enthält. Der Subskriptionswert ist der Wert, der mit der SUBSCRIBE BY-Klausel der Publikation verglichen wird. Wenn zum Beispiel eine Publikation den Namen einer Spalte mit einer Mitarbeiterkennung als SUBSCRIBE BY-Klausel verwendet, muss der Wert der Mitarbeiterkennung des subskribierenden Benutzers bereitgestellt werden, wenn die Subskription erstellt wird. Der Subskriptionswert ist stets eine Zeichenfolge.

  Dieser Wert wird nur benötigt, wenn die Publikation eine SUBSCRIBE BY-Klausel hat.
- **Subskribenten-ID** Der Benutzer, der für die Publikation subskribiert wird. Wenn Sie eine Subskription für einen entfernten Benutzer erstellen, muss dem entfernten Benutzer in der konsolidierten Datenbank das REMOTE-Privileg erteilt worden sein. Wenn Sie eine Subskription für einen konsolidierten Benutzer erstellen, muss diesem Benutzer in der entfernten Datenbank das CONSOLIDATE-Privileg erteilt worden sein. Standardmäßig erteilen das Extraktionsdienstprogramm (dbxtract) und der **Assistent zum Extrahieren einer Datenbank** Benutzern in den entfernten Datenbanken die entsprechenden PUBLISH- und CONSOLIDATE-Privilegien.

Siehe auch

- „CREATE SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „GRANT REMOTE-Anweisung [SQLRemote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Publizieren von nur einigen Zeilen mit der SUBSCRIBE BY-Klausel“ auf Seite 14
- „Resynchronisation von Subskriptionen“ auf Seite 151
- „Starten von Subskriptionen“ auf Seite 155
- „Stoppen von Subskriptionen“ auf Seite 156

SQL Remote-Subskriptionen erstellen

Subskribieren Sie einen Benutzer für eine Publikation, indem Sie eine Subskription erstellen.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben und die Publikation muss bereits erstellt sein.

Kontext und Bemerkungen

Jede Datenbank, die Informationen in einer Publikation gemeinsam nutzt, muss eine Subskription für die Publikation besitzen. In regelmäßigen Abständen werden die Änderungen, die in den einzelnen Publikationen einer Datenbank vorgenommen werden, an alle Subskribenten dieser Publikation repliziert. Diese Replikationen werden als Publikationsaktualisierungen bezeichnet

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.

2. Doppelklicken Sie auf den Datenbanknamen.
3. Doppelklicken Sie auf **SQL Remote-Subskriptionen**.
4. Klicken Sie auf **Datei** » **Neu** » **SQL Remote-Subskription**.
5. Befolgen Sie die Anweisungen des **Assistenten zum Erstellen von SQL Remote-Subskriptionen**.

   Die Einzelheiten der Subskription hängen davon ab, ob die Publikation einen Subskriptionsausdruck verwendet.

Ergebnisse

Der Benutzer wird für die Publikation subskribiert.

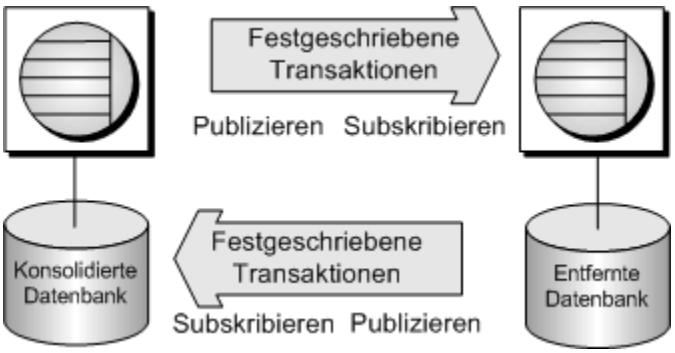
Transaktionsbasierte Replikation

SQL Remote repliziert Folgendes:

- **Festgeschriebene Änderungen** Änderungen, die in Datenbanken durchgeführt wurden, wie sie in ihren Transaktionslogs aufgezeichnet sind.
- **Änderungen an Daten, die zu Publikationen gehören** SQL Remote durchsucht das Transaktionslog nach festgeschriebenen Änderungen an Zeilen, die zu Publikationen gehören, verpackt die SQL-Anweisungen in Nachrichten und sendet sie an die subskribierten Datenbanken.

Von der konsolidierten Datenbank aus werden alle festgeschriebenen Transaktionen im Transaktionslog, die zu Publikationen gehören, regelmäßig an die entfernten Datenbanken gesendet.

Von den entfernen Datenbanken aus werden alle festgeschriebenen Transaktionen im Transaktionslog, die zu Publikationen gehören, regelmäßig an die konsolidierte Datenbank gesendet.



Der Datenbankserver verarbeitet Publikationen

Der SQL Anywhere-Datenbankserver ist die Komponente, die die Publikationen auswertet und die Informationen in das Transaktionslog schreibt. Je mehr Publikationen Sie haben, desto mehr Arbeitsleistung muss der Datenbankserver erbringen.

SQL Anywhere berücksichtigt den Subskriptionsausdruck für jede Aktualisierung, die in einer Tabelle durchgeführt wird, die Teil einer Publikation ist. Der Wert des Ausdrucks wird sowohl vor als auch nach der Aktualisierung dem Transaktionslog hinzugefügt. Bei einer Tabelle, die zu mehr als einer Publikation gehört, wird der Subskriptionsausdruck bei jeder Publikation vor und nach der Aktualisierung berücksichtigt.

Die zusätzlichen Informationen im Transaktionslog können sich auf die Performance in folgenden Fällen auswirken:

- **Kostenträchtige Ausdrücke** Wenn die Berücksichtigung eines Subskriptionsausdrucks kostenträchtig ist, kann sich das auf die Performance auswirken.
- **Viele Publikationen** Wenn eine Tabelle zu mehreren Publikationen gehört, müssen viele Ausdrücke berücksichtigt werden. Im Gegensatz dazu ist die Anzahl der *Subskriptionen* für den Datenbankserver ohne Bedeutung.
- **Vielwertige Ausdrücke** Manche Ausdrücke sind vielwertig, was zu zusätzlichen Informationen im Transaktionslog führen kann. Dies kann sich auf die Performance auswirken.

Subskriptionen werden von SQL Remote verarbeitet

SQL Remote ist die Komponente, die die Replikation von Anweisungen durchführt.

Während der Sendephase ordnet der SQL Remote-Nachrichtenagent die aktuellen Subskriptionen den Publikationsinformationen im Transaktionslog zu und generiert die entsprechenden Nachrichten für jeden entfernten Benutzer.

Siehe auch

- SQL Remote-Nachrichtenagent (dbremote) auf Seite 97
- „Das Transaktionslog“ [*SQL Anywhere Server - Datenbankadministration*]

Replikation der INSERT- und DELETE-Anweisung

Für SQL Remote sind die INSERT- und DELETE-Anweisungen die am einfachsten zu replizierenden Anweisungen.

- Wenn eine INSERT-Anweisung in einer Datenbank ausgeführt wird, wird sie als INSERT-Anweisung an die subskribierten Datenbanken im SQL Remote-System gesendet.
- Wenn eine DELETE-Anweisung in einer Datenbank ausgeführt wird, wird sie als DELETE-Anweisung an die subskribierten Datenbanken im SQL Remote-System gesendet.

Konsolidierte Datenbank

SQL Remote kopiert jede INSERT- oder DELETE-Anweisung im Transaktionslog der konsolidierten Datenbank und sendet sie an jede entfernte Datenbank, die die Zeile subskribiert, die eingefügt oder gelöscht wird. Wenn die Subskription nur eine Teilmenge der Spalten in der Tabelle erfasst, enthalten die INSERT-Anweisungen, die an die entfernten Datenbanken geschickt werden, nur diese Spalten.

Entfernte Datenbanken

SQL Remote kopiert jede INSERT- oder DELETE-Anweisung im Transaktionslog einer entfernten Datenbank und sendet sie an die konsolidierte Datenbank, die die Zeile subskribiert, die eingefügt oder gelöscht wird. Die konsolidierte Datenbank wendet anschließend die Anweisung an, was bedeutet, dass sie in ihr Transaktionslog geschrieben wird. Wenn das Transaktionslog der konsolidierten Datenbank von SQL Remote abgearbeitet wird, werden die Änderungen schließlich an die anderen entfernten Standorte verteilt. SQL Remote stellt sicher, dass Anweisungen nicht an den entfernten Benutzer gesendet werden, der sie ursprünglich ausgeführt hat.

Siehe auch

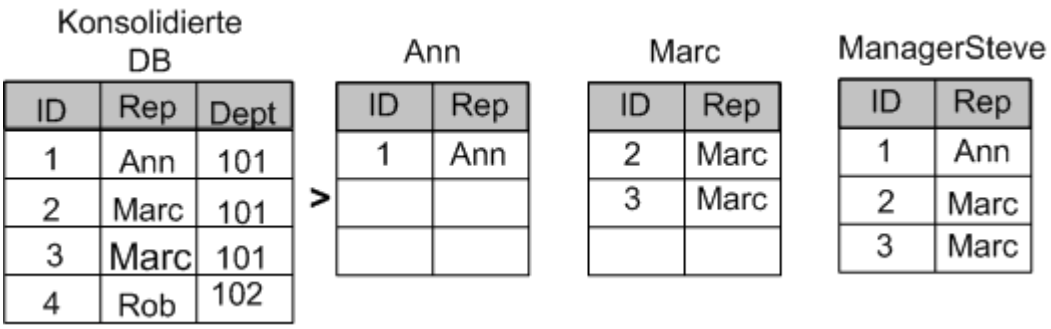
- „Mehrfach vorhandene Primärschlüssel“ auf Seite 55
- „Zeile nicht gefunden“ auf Seite 52

Replikation der UPDATE-Anweisung

UPDATE-Anweisungen werden möglicherweise nicht genau so repliziert, wie sie in die Datenbank eingegeben werden. Die folgenden Szenarien beschreiben, wie eine UPDATE-Anweisung repliziert wird:

- Wenn eine UPDATE-Anweisung bewirkt, dass eine Zeile einer gegebenen Subskription eines entfernten Benutzers aktualisiert wird, erfolgt ihre Sendung an diesen Benutzer als UPDATE-Anweisung.
- Wenn eine UPDATE-Anweisung bewirkt, dass eine Zeile aus einer gegebenen Subskription eines entfernten Benutzers entfernt wird, erfolgt ihre Sendung an diesen Benutzer als DELETE-Anweisung.
- Wenn eine UPDATE-Anweisung bewirkt, dass eine Zeile einer gegebenen Subskription eines entfernten Benutzers hinzugefügt wird, erfolgt ihre Sendung an diesen Benutzer als INSERT-Anweisung.

Um zu veranschaulichen, wie eine UPDATE-Anweisung repliziert werden kann, verwendet das folgende Beispiel eine konsolidierte Datenbank und drei entfernte Datenbanken für die Benutzer Ann, Marc und ManagerSteve.



In der konsolidierten Datenbank gibt es eine Publikation namens cons, die mit der folgenden Anweisung erstellt wird:

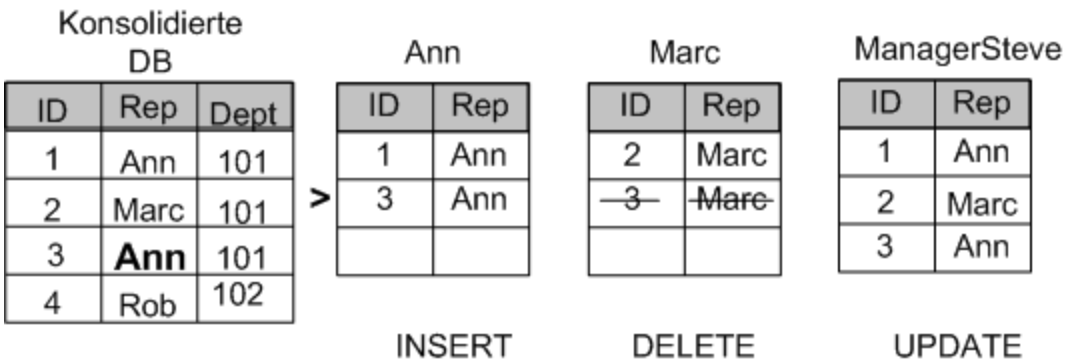
```
CREATE PUBLICATION "cons"."p1" (
 TABLE "DBA"."customers" ( "ID", "Rep") SUBSCRIBE BY repid
);
```

Ann und Marc subskribieren die cons-Publikation anhand ihrer jeweiligen Rep-Spaltenwerte. ManagerSteve subskribiert die cons-Publikation anhand der Rep-Spaltenwerte von Ann und Marc. Die folgenden Anweisungen subskribieren die drei Benutzer für die cons-Publikation:

```
CREATE SUBSCRIPTION
  TO "cons"."p1"( 'Ann' )
  FOR "Ann";
CREATE SUBSCRIPTION
  TO "cons"."p1"( 'Marc' )
  FOR "Marc";
CREATE SUBSCRIPTION
  TO "cons"."p1"( 'Ann' )
  FOR "ManagerSteve";
CREATE SUBSCRIPTION
  TO "cons"."p1"( 'Marc' )
  FOR "ManagerSteve";
```

In der konsolidierten Datenbank wird eine UPDATE-Anweisung, die den Rep-Wert einer Zeile von Marc zu Ann ändert, folgendermaßen repliziert:

- Marc als DELETE-Anweisung..
- Ann als INSERT-Anweisung.
- ManagerSteve als UPDATE-Anweisung.



Diese Neuzuweisung von Zeilen wird manchmal als **territoriale Neuaufteilung** bezeichnet, da sie eine übliche Eigenschaft bei Anwendungen ist, die den Verkauf automatisieren und bei denen Kunden dem Verkaufspersonal regelmäßig neu zugeordnet werden.

Siehe auch

- „Aktualisierungskonflikte“ auf Seite 44
- „UPDATE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Prozedur-Replikation

SQL Remote repliziert Prozeduren, indem die Aktionen der Prozedur repliziert werden. Der Prozeduraufruf wird nicht repliziert. Statt dessen werden die einzelnen Aktionen (INSERT-, UPDATE- und DELETE-Anweisungen) der Prozedur repliziert.

Trigger-Replikation

Üblicherweise sind bei entfernten Datenbanken dieselben Trigger wie bei der konsolidierten Datenbank definiert.

Standardmäßig repliziert SQL Remote die Aktionen nicht, die von Triggern durchgeführt werden. Wenn eine Aktion, die einen Trigger in der konsolidierten Datenbank auslöst, in der entfernten Datenbank repliziert wird, erfolgt eine automatische Auslösung des mehrfach vorhandenen Triggers in der entfernten Datenbank. Dadurch werden Privilegienprobleme verhindert und sichergestellt, dass keine Aktion zweimal eintritt. Es gibt einige Ausnahmen zu dieser Regel.

- **RESOLVE UPDATE-Trigger replizieren** Die von der Konfliktlösung oder den RESOLVE UPDATE-Trigger durchgeführten Aktionen werden von einer konsolidierten Datenbank an alle entfernten Datenbanken *repliziert*, einschließlich jene entfernte Datenbank, welche die Nachricht versendet hat, die den Konflikt erstellt hat.

- **Replikation von BEFORE-Triggern** Die Aktionen eines BEFORE-Triggers, der eine Zeile, die aktualisiert wird, ändert, werden vor den UPDATE-Anweisungsaktionen *repliziert*.

  So würde zum Beispiel ein BEFORE UPDATE-Trigger, der eine Zählungsspalte in der Zeile erhöht, um die Anzahl der Aktualisierungen einer Zeile zu protokollieren, eine doppelte Zählung bewirken, wenn er repliziert wird, da der BEFORE UPDATE-Trigger in der entfernten Datenbank ausgelöst wird, wenn die UPDATE-Anweisung repliziert wird.

  Ein BEFORE UPDATE-Trigger, der eine Spalte auf den Zeitpunkt der letzten Aktualisierung setzt, erhält auch den Zeitpunkt, an dem die UPDATE-Anweisung repliziert wird.

  Um dieses Problem zu vermeiden, müssen Sie sicherstellen, dass in der Subskribenten-Datenbank der BEFORE UPDATE-Trigger nicht vorhanden ist beziehungsweise die replizierte Aktion nicht ausführt.

Eine Option, um Triggeraktionen zu replizieren

Um beim Versenden von Nachrichten alle Triggeraktionen zu replizieren, verwenden Sie die Option -t für den SQL Remote-Nachrichtenagenten (dbremote).

Wenn Sie die Option -t verwenden, achten Sie darauf, dass die Triggeraktionen in entfernten Datenbanken nicht zweimal durchgeführt werden: einmal wenn die replizierten Triggeraktionen angewendet werden und einmal wenn der Trigger in der entfernten Datenbank ausgelöst wird.

Um sicherzustellen, dass Triggeraktionen nicht zweimal durchgeführt werden, verwenden Sie eine der folgenden Optionen:

- Binden Sie den Hauptteil des Triggers in eine Anweisung IF CURRENT REMOTE USER IS NULL ... END IF ein.
- Setzen Sie die SQL Anywhere-Option fire_triggers für den SQL Remote-Benutzernamen auf "Off".

Triggerfehler vermeiden

Wenn eine Publikation nur eine Teilmenge einer Datenbank umfasst, kann sich ein Trigger in der konsolidierten Datenbank auf Tabellen oder Zeilen beziehen, die in der konsolidierten Datenbank vorhanden sind, nicht aber in den entfernten Datenbanken. Wenn solch ein Trigger in der entfernten Datenbank ausgelöst wird, treten Fehler auf. Um diese Fehler zu vermeiden, verwenden Sie eine IF-Anweisung, um die Triggeraktionen bedingt zu machen, sowie:

- Machen Sie die Aktionen des Triggers vom Wert von CURRENT PUBLISHER abhängig.
- Machen Sie die Aktionen des Triggers davon abhängig, dass die object_id-Funktion nicht NULL zurückgibt. Die object_id-Funktion nimmt eine Tabelle oder ein anderes Objekt als Argument und gibt die ID-Nummer dieses Objekts zurück, oder NULL, wenn das Objekt nicht existiert.
- Machen Sie Aktionen des Triggers von einer SELECT-Anweisung abhängig, die ermittelt, ob die Zeilen vorhanden sind.

Extraktionsdienstprogramm (dbxtract)

Standardmäßig extrahieren das Extraktionsdienstprogramm (dbxtract) und der **Assistent zum Extrahieren einer Datenbank** die Triggerdefinitionen.

Siehe auch

- „Standardlösung für Aktualisierungskonflikte“ auf Seite 45
- Den CURRENT REMOTE USER-Spezialwert verwenden auf Seite 49
- „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203
- „fire_triggers-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „CURRENT PUBLISHER-Spezialwert“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Systemfunktionen“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Datendefinitionsanweisungen

Datendefinitionsanweisungen (CREATE, ALTER und DROP) werden von SQL Remote nicht repliziert, außer sie werden im Passthrough-Modus ausgeführt.

Siehe auch

- „SQL Remote-Passthrough-Modus“ auf Seite 148

Datentypen

SQL Remote führt keine Zeichensatzkonvertierungen durch.

Kompatible Sortierreihenfolge und Zeichensätze verwenden

Der Zeichensatz und die Kollation, die von der konsolidierten SQL Anywhere-Datenbank verwendet werden, müssen mit denjenigen in der entfernten Datenbank übereinstimmen. Weitere Hinweise zu unterstützten Zeichensätzen finden Sie unter „Internationale Sprachen und Zeichensätze“ [*SQL Anywhere Server - Datenbankadministration*].

BLOBs

BLOBs umfassen die LONG VARCHAR-, LONG BINARY-, TEXT- und IMAGE-Datentypen.

Wenn SQL Remote eine INSERT- oder UPDATE-Anweisung repliziert, wird eine Variable anstatt des BLOB-Werts verwendet. Das bedeutet, dass der BLOB in Teile aufgegliedert und in Abschnitten repliziert wird. In der Empfängerdatenbank werden die Teile mithilfe einer SQL-Variablen wiederhergestellt und verkettet. Der Wert der Variablen wird durch eine Sequenz von Anweisungen im folgenden Format aufgebaut:

```
SET var = var || 'more_stuff';
```

Die Variable vermindert die Größe der SQL-Anweisungen, die große Werte betreffen, damit sie in eine einzelne Nachricht passen.

Die SET-Anweisungen sind getrennte SQL-Anweisungen, wodurch der BLOB tatsächlich auf mehrere SQL Remote-Nachrichten aufgeteilt wird.

Replikation von BLOBs kontrollieren

Die Option blob_threshold von SQL Anywhere ermöglicht eine weitergehende Kontrolle der Replikation von großen Werten. Jeder Wert, der länger als die Option blob_threshold ist, wird repliziert, als ob er ein BLOB-Wert wäre.

Die Option verify_threshold zum Minimieren der Nachrichtengröße verwenden

Die Datenbankoption verify_threshold kann verhindern, dass große Werte überprüft werden (in der VERIFY-Klausel von einer replizierten UPDATE-Anweisung). Der Standardwert für die Option beträgt 1000. Wenn der Datentyp einer Spalte größer als der Schwellenwert ist, werden alte Werte in der Spalte beim Replizieren einer UPDATE-Anweisung nicht überprüft. Dies hält die Größe von SQL Remote-Nachrichten niedrig, hat aber den Nachteil, dass kollidierende Aktualisierungen langer Werte nicht erkannt werden.

Verwenden Sie die folgende Methode, um Konflikte zu erkennen, wenn die verify_threshold-Option eingesetzt wird, um die Größe von Nachrichten zu verringern:

1. Konfigurieren Sie Ihre Datenbanken so, dass, sobald ein BLOB aktualisiert wird, auch eine last_modified-Spalte in der gleichen Tabelle aktualisiert wird.
2. Konfigurieren Sie Ihre Publikationen dahingehend, dass die Spalte last_modified mit der BLOB-Spalte repliziert wird.
3. Wenn die BLOB-Spalte und die last_modified-Spalte repliziert werden, können die Werte in der last_modified-Spalte überprüft werden. Wenn es einen Konflikt mit der last_modified-Spalte gibt, dann gibt es auch einen Konflikt mit der BLOB-Spalte.

Eine Arbeitstabelle zur Vermeidung von redundanten Aktualisierungen verwenden

Wiederholte Aktualisierungen eines BLOBs sollten in einer Arbeitstabelle durchgeführt werden, wobei dann die endgültige Version in der replizierten Tabelle übernommen wird. Wenn zum Beispiel ein Dokument zwanzig Mal während eines Tages aktualisiert und SQL Remote einmal am Ende des Tages ausgeführt wird, werden alle zwanzig Aktualisierungen repliziert. Wenn das Dokument 200 kB groß ist, werden 4 MB Nachrichten gesendet.

Es wird empfohlen, dass Sie eine *document_in_progress*-Tabelle verwenden. Wenn der Benutzer mit der Revision des Dokuments fertig ist, verschiebt die Anwendung das Dokument von der *document_in_progress*-Tabelle zur replizierten Tabelle. Das Ergebnis ist eine einzige Aktualisierung (200 kB Nachrichten).

Siehe auch

- „SET-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „blob_threshold-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]
- „verify_threshold-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]

Datumsangaben und Uhrzeiten

Wenn Datums- oder Uhrzeitspalten repliziert werden, verwendet SQL Remote die Einstellung der Datenbankoptionen sr_date_format, sr_time_format und sr_timestamp_format, um das Datum zu formatieren.

Die folgende Einstellung teilt beispielsweise SQL Remote mit, die Datumsangabe 2. Mai 1998 als 1998-05-02 zu senden.

```
SET OPTION sr_date_format = 'yyyy-mm-dd';
```

Beim Replizieren von Datumsangaben und Uhrzeiten beachten Sie Folgendes:

- Die Uhrzeit-, Datums- und Zeitstempelformate müssen innerhalb des SQL Remote-Systems konsistent sein.
- Die Reihenfolge für Jahr, Monat und Tag, die für das Datums- und Zeitstempelformat verwendet wird, muss der Einstellung der date_order-Datenbankoption entsprechen.
- Die date_order-Option für die Dauer der einzelnen Verbindungen kann geändert werden.

Siehe auch

- „sr_date_format-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]
- „sr_time_format-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]
- „sr_timestamp_format-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]
- „date_order-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Replikationskonflikte und -fehler

SQL Remote ermöglicht es Datenbanken, in mehreren Datenbanken aktualisiert zu werden. Es ist eine umsichtige Planung nötig, um Replikationsfehler zu vermeiden, vor allem, wenn die Datenbank eine komplizierte Struktur aufweist.

Replikationskonflikte

Replikationskonflikte unterscheiden sich von Fehlern. Richtig gehandhabt stellen Konflikte in SQL Remote kein Problem dar.

Konflikte können in vielen Systemen auftreten. SQL Remote ermöglicht das entsprechende Lösen von Konflikten als Teil der regulären Vorgänge in einem SQL Remote-System, wobei Trigger und Prozeduren verwendet werden.

Replikationsfehler

Replikationsfehler werden in folgende Kategorien unterteilt:

- **Zeile nicht gefunden** Ein Benutzer löscht eine Zeile (mit einem gegebenen Primärschlüsselwert). Ein zweiter Benutzer aktualisiert oder löscht dieselbe Zeile an einem anderen Standort. In diesem Fall schlägt die zweite Anweisung fehl, weil die Zeile nicht gefunden wird.

- **Fehler bei der referenziellen Integrität** Wenn eine Spalte, die einen Fremdschlüssel enthält, in eine Publikation aufgenommen wird, der zugehörige Primärschlüssel hingegen nicht, schlagen INSERT-Anweisungen fehl, die den Fremdschlüssel referenzieren.

  Überdies können Fehler der referenziellen Integrität auftreten, wenn eine Primärtabelle eine SUBSCRIBE BY-Klausel enthält, die zugeordnete Fremdtabelle aber nicht. Zeilen aus der Fremdtabelle werden möglicherweise repliziert, aber die Zeilen aus der Primärtabelle werden nicht unbedingt in die Publikation aufgenommen.

- **Mehrfach vorhandene Primärschlüssel** Zwei Benutzer fügen eine Zeile unter Verwendung derselben Primärschlüsselwerte ein oder ein Benutzer aktualisiert einen Primärschlüssel und ein zweiter Benutzer fügt einen Primärschlüssel des neuen Wertes ein. Der zweite Vorgang scheitert beim Versuch, eine angegebene Datenbank im Replikationssystem zu erreichen, weil er einen mehrfach vorhandenen Primärschlüssel erzeugen würde.

Zustellungsfehler

Hinweise zu Zustellungsfehlern und der damit zusammenhängenden Fehlerbehebung finden Sie unter „System der garantierten Nachrichtenzustellung“ auf Seite 108.

Siehe auch

- „Standardlösung für Aktualisierungskonflikte“ auf Seite 45
- „Berichterstellung und Behandlung von Replikationsfehlern“ auf Seite 141
- „Aktualisierungskonflikte“ auf Seite 44
- „Zeile nicht gefunden“ auf Seite 52
- „Fehler bei der referenziellen Integrität“ auf Seite 53
- „Mehrfach vorhandene Primärschlüssel“ auf Seite 55

Aktualisierungskonflikte

Aktualisierungskonflikte können dort nicht vorkommen, wo Daten nur gemeinsam gelesen werden, oder wenn jede Zeile (gekennzeichnet durch ihren Primärschlüssel) nur in einer Datenbank aktualisiert wird. Aktualisierungsfehler können nur auftreten, wenn Daten in mehr als einer Datenbank aktualisiert werden.

Um UPDATE-Anweisungen zu replizieren, gibt SQL Remote eine separate UPDATE-Anweisung für jede Zeile aus. Diese einzeiligen Anweisungen können aus einem der folgenden Gründe fehlschlagen:

- **Die zu aktualisierende Zeile unterscheidet sich in einer oder mehreren Spalten** Wenn einer der erwarteten Werte von einem anderen Benutzer geändert wurde, tritt ein **Aktualisierungskonflikt** auf.

  In entfernten Datenbanken findet die Aktualisierung unabhängig von den Werten in der Zeile statt.

  In der konsolidierten Datenbank lässt SQL Remote Vorgänge zur **Konfliktlösung** zu. Beispiel: Wenn ein Konflikt erkannt wird, kann die konsolidierte Datenbank Folgendes durchführen:

  - Die Standard-Konfliktlösung verwenden.
  - Eine benutzerdefinierte Konfliktlösung verwenden, die die VERIFY-Klausel verwendet.
  - Eine benutzerdefinierte Konfliktlösung verwenden, die Trigger verwendet.

  Hinweis
  UPDATE-Anweisungskonflikte gelten *nicht* für Primärschlüssel-Aktualisierungen. Sie dürfen Primärschlüssel in einem SQL Remote-System nicht aktualisieren. Primärschlüssel-Konflikte im System müssen durch die richtige Planung vermieden werden.

- **Die zu aktualisierende Zeile existiert nicht** Jede Zeile ist durch ihre Primärschlüsselwerte gekennzeichnet. Wenn die Zeile gelöscht oder ein Primärschlüssel von einem anderen Benutzer geändert wurde, kann die Zeile, die aktualisiert werden soll, nicht gefunden werden.

  In entfernten Datenbanken findet die Aktualisierung nicht statt.

  In der konsolidierten Datenbank findet die Aktualisierung nicht statt.

- **Eine Tabelle ohne Primärschlüssel oder Eindeutigkeits-Integritätsregel bezieht sich auf alle Spalten in der WHERE-Klausel von replizierten Aktualisierungen** Wenn zwei entfernte Datenbanken separate Aktualisierungen in derselben Zeile durchführen und die Änderungen in die konsolidierte Datenbank replizieren, werden die Änderungen, die zuerst in der konsolidierten Datenbank eintreffen, angewendet, während Änderungen von der zweiten Datenbank nicht angewendet werden.

  Dies führt dazu, dass Datenbanken inkonsistent werden. Alle replizierten Tabellen sollten einen Primärschlüssel oder eine Eindeutigkeits-Integritätsregel aufweisen. Die Spalten in der Integritätsregel sollten nie aktualisiert werden.

Siehe auch

- „Standardlösung für Aktualisierungskonflikte“ auf Seite 45
- „Benutzerdefinierte Konfliktlösung mittels einer VERIFY-Klausel“ auf Seite 46
- „Benutzerdefinierte Konfliktlösung mit Trigger“ auf Seite 48

Standardlösung für Aktualisierungskonflikte

Aktualisierungskonflikte können nur auftreten, wenn Daten an mehr als einem Standort aktualisiert werden. Zum Beispiel:

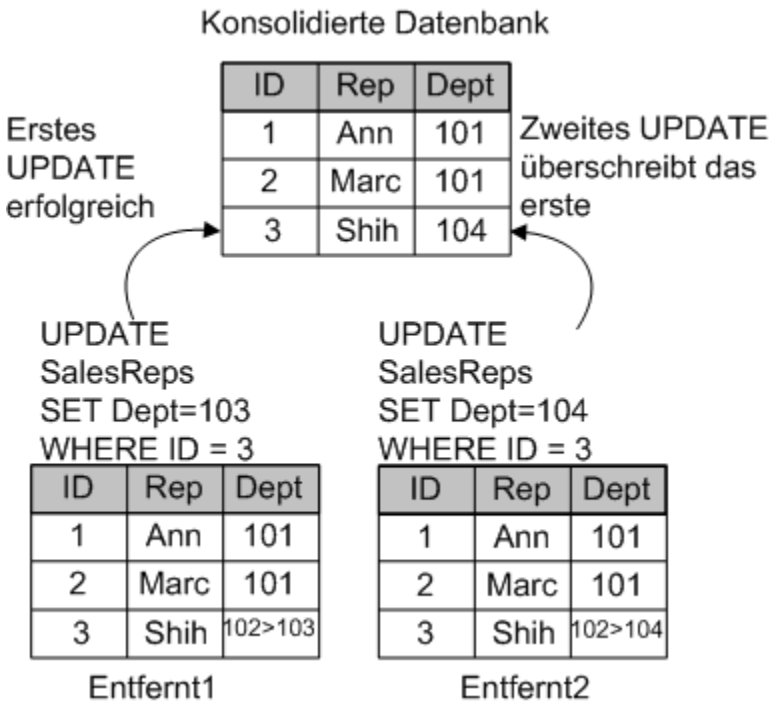
1. Benutzer 1 aktualisiert eine Zeile am entfernten Standort 1.
2. Benutzer 2 aktualisiert dieselbe Zeile am entfernten Standort 2.
3. Die Aktualisierung von Benutzer 1 wird an die konsolidierte Datenbank gesendet und angewendet.
4. Die Aktualisierung von Benutzer 2 wird an die konsolidierte Datenbank gesendet.

   Die **Standard**-Methode zur Lösung dieses Typs von Aktualisierungskonflikten:

   a. Der aktuellere Vorgang (in diesem Beispiel der von Benutzer 2) ist erfolgreich. Er wird der Wert in der konsolidierten Datenbank und ist der Wert, der an alle Datenbanken repliziert wird, die diese Zeile subskribiert haben.

   b. Alle anderen Aktualisierungen (in diesem Beispiel die von Benutzer 1) gehen verloren.

   c. Über den Konflikt wird kein Bericht erstellt.



Siehe auch

- „Zeile nicht gefunden“ auf Seite 52

Benutzerdefinierte Konfliktlösung mittels einer VERIFY-Klausel

SQL Remote generiert UPDATE-Anweisungen in den Nachrichten, die die VERIFY-Klausel verwenden. Eine UPDATE-Anweisung ändert den Wert einer oder mehrerer Zeilen von einem vorhandenen Wert auf einen neuen Wert. UPDATE-Anweisungen, die eine VERIFY-Klausel einbeziehen, enthalten auch den vorhandenen Wert der Zeile.

Bei der Anwendung einer UPDATE-Anweisung vergleicht die konsolidierte Datenbank ihren vorhandenen Wert der Zeile mit dem Wert, den die entfernte Datenbank als den vorhandenen Wert der Zeile erwartet. Ein Aktualisierungskonflikt wird vom Datenbankserver erkannt, wenn die Werte in der VERIFY-Klausel nicht mit den Zeilen in der Datenbank übereinstimmen.

Beispiel: Ein Aktualisierungskonflikt tritt auf, wenn die folgende Abfolge von Ereignissen stattfindet:

1. Benutzer 1 aktualisiert eine Zeile am entfernten Standort 1.
2. Benutzer 2 aktualisiert dieselbe Zeile am entfernten Standort 2.
3. Die Aktualisierung von Benutzer 1 wird an die konsolidierte Datenbank gesendet und angewendet.

4. Die Aktualisierung von Benutzer 2 wird an die konsolidierte Datenbank gesendet.

   Da die UPDATE-Anweisung eine VERIFY-Klausel enthält, kann die konsolidierte Datenbank Konflikte erkennen. In der konsolidierten Datenbank vergleicht SQL Remote den Wert in seiner Zeile mit dem alten Wert, den Benutzer 2 gesendet hat. Da diese Werte nicht übereinstimmen, gibt es einen Aktualisierungskonflikt.

   Wenn ein Aktualisierungskonflikt erkannt wird, führt die konsolidierte Datenbank Folgendes durch:

   a. Sie löst etwaige Konfliktlösungstrigger aus, die für den Vorgang festgelegt sind.

      Sie legen **Konfliktlösungstrigger** fest, um Aktualisierungskonflikte zu behandeln. Konfliktlösungstrigger werden nur dann in einer konsolidierten Datenbank aktiviert, wenn Nachrichten von einem entfernten Benutzer übernommen werden.

   b. Sie führt die UPDATE-Anweisungen aus.

   c. Sie sendet die Aktionen des Konfliktlösungstriggers und die UPDATE-Anweisung an alle entfernten Datenbanken, auch an den Absender der Nachricht, die den Konflikt ausgelöst hat.

      Üblicherweise repliziert SQL Remote keine Triggeraktionen: Es wird vorausgesetzt, dass der Trigger in der entfernten Datenbank vorhanden ist. Konfliktlösungstrigger werden nur in konsolidierten Datenbanken aktiviert, und daher werden ihre Aktionen in entfernte Datenbanken repliziert.

5. Die entfernten Datenbanken empfangen die UPDATE-Anweisungen von der konsolidierten Datenbank.

   In entfernten Datenbanken werden RESOLVE UPDATE-Trigger nicht ausgelöst, wenn eine Nachricht von einer konsolidierten Datenbank einen Aktualisierungskonflikt enthält.

6. In der entfernten Datenbank werden die UPDATE-Anweisungen abgearbeitet.

   Am Ende des Prozesses sind die Daten innerhalb des Systems konsistent.

Siehe auch

- „Benutzerdefinierte Konfliktlösung mit Trigger“ auf Seite 48

UPDATE-Anweisungen mit einer VERIFY-Klausel

Eine UPDATE-Anweisung mit einer VERIFY-Klausel hat das folgende Format:

UPDATE *table-list*
SET *column-name* = *expression*, ...
[**VERIFY** (*column-name*, ...)
  VALUES (*expression*, ...)]
[**WHERE** *search-condition*]
[**ORDER BY** *expression* [**ASC** | **DESC**], ...]

Die VERIFY-Klausel kann nur verwendet werden, wenn der *table-list*-Parameter aus einer einzigen Tabelle besteht. Sie vergleicht die Werte von angegebenen Spalten mit einer Reihe von erwarteten

Werten, die in der Publikationseigentümer-Datenbank vorhanden waren, als die UPDATE-Anweisung übernommen wurde. Wenn die VERIFY-Klausel angegeben ist, kann immer nur eine Tabelle gleichzeitig aktualisiert werden.

Die VERIFY-Klausel ist nur bei einzeiligen Aktualisierungen sinnvoll. Eine mehrzeilige UPDATE-Anweisung in einer Datenbank wird allerdings als Reihe von einzeiligen UPDATE-Anweisungen von SQL Remote repliziert, daher kommen dadurch keine Integritätsregeln bei Clientanwendungen zur Geltung.

Siehe auch

- „UPDATE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Die verify_all_columns-Option

Standardmäßig ist die Datenbankoption verify_all_columns auf Off. Wenn sie auf Off gesetzt ist, werden nur die Spalten überprüft, die aktualisiert werden. Wenn die Option verify_all_columns auf On gesetzt ist, gilt Folgendes:

- Alle Spalten werden bei replizierten UPDATE-Anweisungen überprüft.
- Ein RESOLVE UPDATE-Trigger wird ausgelöst, wenn Spalten unterschiedlich sind.
- Die Größe der Nachrichten nimmt zu, weil mehr Informationen bei jeder UPDATE-Anweisung gesendet werden.

Sie können die Option verify_all_columns entweder für die PUBLIC-Rolle oder nur für den einzelnen Benutzer einstellen, der in der SQL Remote-Verbindungszeichenfolge enthalten ist.

Das Extraktionsdienstprogramm (dbxtract)

Wenn die Option verify_all_columns in der konsolidierten Datenbank eingestellt ist, bevor die entfernten Datenbanken extrahiert werden, wird die Option verify_all_columns in den entfernten Datenbanken durch das Extraktionsdienstprogramm (dbxtract) und den **Assistenten zum Extrahieren einer Datenbank** gesetzt.

Siehe auch

- „verify_all_columns-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]

Benutzerdefinierte Konfliktlösung mit Trigger

Die benutzerdefinierte Konfliktlösung kann unterschiedliche Formen annehmen. In einigen Anwendungen kann die Konfliktlösung Folgendes bewirken:

- Die Datumsangaben der ursprünglichen Transaktionen werden verglichen.
- Berechnungen auf den Ergebnissen von zwei oder mehr Aktualisierungen werden durchgeführt.
- Der Konflikt wird in einer Tabelle aufgezeichnet.

Die benutzerdefinierte Konfliktlösung erfordert, dass Sie RESOLVE UPDATE-Trigger schreiben.

Den RESOLVE UPDATE-Konfliktlösungstrigger verwenden

RESOLVE UPDATE-Trigger lösen aus, bevor die jeweilige Zeile aktualisiert wird. Die Syntax für einen RESOLVE UPDATE-Trigger lautet:

```
CREATE TRIGGER trigger-name
RESOLVE UPDATE
OF column-name ON table-name
[ REFERENCING [ OLD AS old-val ]
   [ NEW AS new-val ]
     [ REMOTE AS remote-val ] ]
FOR EACH ROW
BEGIN
   …
END
```

Die REFERENCING-Klausel ermöglicht den Zugriff auf die Werte in der Zeile der zu aktualisierenden Tabelle (OLD), auf die Werte, die mit der Zeile aktualisiert werden (NEW), und auf die Zeilen, die entsprechend der VERIFY-Klausel (REMOTE) vorhanden sein sollten. Nur Spalten in der VERIFY-Klausel können in der REMOTE AS-Klausel referenziert werden. Andere Spalten geben einen Fehler zurück.

Den CURRENT REMOTE USER-Spezialwert verwenden

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

Siehe auch

- „Lösung von Datumskonflikten“ auf Seite 49
- „Lösung von Inventory-Konflikten“ auf Seite 50
- „CREATE TRIGGER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „UPDATE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Trigger-Replikation“ auf Seite 39
- -t, Option, SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote) auf Seite 211

Lösung von Datumskonflikten

Um zu veranschaulichen, wie Sie Datumskonflikte lösen können, nehmen wir eine Tabelle in einem Kontaktverwaltungssystem mit einer Spalte an, die die aktuelleren Kontakte mit den einzelnen Kunden enthält.

Ein Vertreter spricht mit einem Kunden an einem Freitag, sendet jedoch seine Änderungen an die konsolidierte Datenbank erst am Montag. Inzwischen trifft ein zweiter Vertreter den Kunden am Samstag und aktualisiert die Änderungen am gleichen Abend.

Es gibt keinen Konflikt, wenn die Aktualisierung vom Samstag an die konsolidierte Datenbank repliziert wird, aber wenn die Aktualisierung am Montag eintrifft, findet sie die Zeile bereits geändert vor.

Standardmäßig würde die Montag-Aktualisierung übernommen werden, wodurch die Spalte die falsche Information enthält, dass der letzte Kontakt am Freitag stattgefunden hat. Aktualisierungskonflikte in dieser Spalte müssen jedoch gelöst werden, indem das aktuellere Datum in die Zeile eingefügt wird.

Die Lösung implementieren

Der folgende RESOLVE UPDATE-Trigger wählt den aktuelleren der beiden neuen Werte aus und gibt ihn in die Datenbank ein.

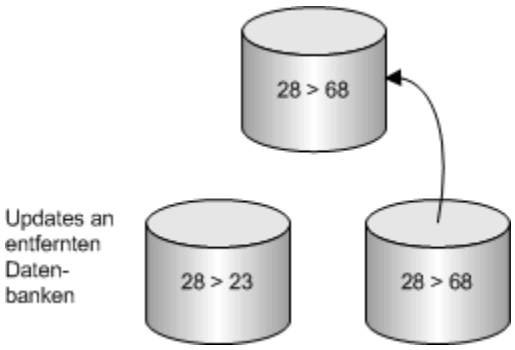
```
CREATE TRIGGER contact_date RESOLVE UPDATE
 ON Contacts
 REFERENCING OLD AS old_name
  NEW AS new_name
 FOR EACH ROW
 BEGIN
   IF new_name.contact_date <
      old_name.contact_date THEN
          SET new_name.contact_date
           = old_name.contact_date
   END IF
 END;
```

Wenn der Wert, der aktualisiert wird, höher (später) als der Wert ist, der ihn ersetzen würde, wird der neue Wert so eingestellt, dass der Eintrag unverändert bleibt.

Lösung von Inventory-Konflikten

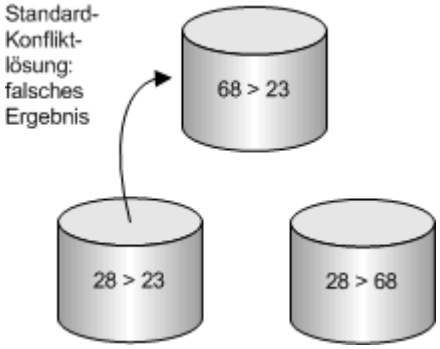
Nehmen wir als Beispiel das Lagerhaussystem eines Produzenten für Sportartikel. Es gibt eine Tabelle mit Produktinformationen, in der eine Quantity-Spalte die Anzahl der einzelnen Produkte auf Lager enthält. Eine Aktualisierung in dieser Spalte würde typischerweise die Menge an Lagergütern verringern oder sie im Falle einer frischen Lieferung erhöhen.

Ein Handelsvertreter, der mit einer entfernten Datenbank arbeitet, gibt eine Bestellung ein, wodurch der Bestand an bestimmten T-Shirts von 28 auf 23, also um fünf, verringert wird. Bevor diese Aktualisierung an die konsolidierte Datenbank repliziert wird, erhält ein anderer Handelsvertreter 40 T-Shirts retourniert. Dieser Handelsvertreter gibt diese Rückgabe in seine entfernte Datenbank ein und repliziert die Änderungen an die konsolidierte Datenbank im Lagerhaus, wodurch 40 der Quantity-Spalte hinzugefügt werden, was 68 ergibt.

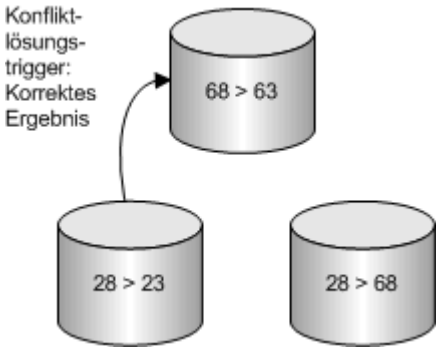


Der Lagerhaus-Eintrag wird der Datenbank hinzugefügt. Die Quantity-Spalte zeigt, dass es nun 68 T-Shirts auf Lager gibt. Wenn die Aktualisierung des ersten Handelsvertreters eintrifft, entsteht ein Konflikt: SQL Anywhere erkennt, dass die Aktualisierung von 28 auf 23 lautet, der aktuelle Wert der Spalte jedoch 68 beträgt.

Standardmäßig setzt sich die aktuellere Aktualisierung durch, sodass die Inventargröße fälschlicherweise auf 23 gesetzt wird.



In diesem Fall sollte der Konflikt durch das Zusammenzählen der Änderungen in der Inventarspalte gelöst werden, um das Endergebnis zu berechnen, damit der endgültige Wert von 63 in die Datenbank eingetragen wird.



Die Lösung implementieren

Ein geeigneter RESOLVE UPDATE-Trigger in dieser Situation würde die Veränderungen aus den zwei Aktualisierungen addieren. Zum Beispiel:

```
CREATE TRIGGER resolve_quantity
 RESOLVE UPDATE OF Quantity
 ON "DBA".Products
 REFERENCING OLD AS old_name
  NEW AS new_name
  REMOTE AS remote_name
 FOR EACH ROW
 BEGIN
   SET new_name.Quantity = new_name.Quantity
                         + old_name.Quantity
                         - remote_name.Quantity
 END;
```

Dieser Trigger addiert die Differenz zwischen dem alten Wert in der konsolidierten Datenbank (68) und dem alten Wert in der entfernten Datenbank, als die ursprüngliche UPDATE-Anweisung ausgeführt wurde (28), zu einem neuen Wert, der versendet wird, bevor die UPDATE-Anweisung implementiert wird. Daher wird new_name.Quantity 63 (= 23 + 68 - 28), und dieser Wert wird in die Quantity-Spalte eingegeben.

Die Konsistenz wird in der entfernten Datenbank folgendermaßen gewahrt:

1. Die ursprüngliche entfernte UPDATE-Anweisung änderte den Wert von 28 auf 23.
2. Der Eintrag des Lagerhauses wird in die entfernte Datenbank repliziert, was allerdings scheitert, da der alte Wert nicht der ist, der erwartet wurde.
3. Die Änderungen durch den RESOLVE UPDATE-Trigger werden in die entfernte Datenbank repliziert.

Zeile nicht gefunden

Ein Benutzer löscht eine Zeile (mit einem gegebenen Primärschlüsselwert). Ein zweiter Benutzer aktualisiert oder löscht dieselbe Zeile an einem anderen Standort. In diesem Fall schlägt die zweite Anweisung fehl, weil die Zeile nicht gefunden wird.

Um UPDATE- und DELETE-Anweisungen korrekt zu replizieren, müssen Sie alle Primärschlüsselspalten in den Artikel einbeziehen.

Wenn eine UPDATE- oder eine DELETE-Anweisung repliziert wird, verwendet SQL Remote die Primärschlüsselspalten, um die Zeilen, die aktualisiert oder gelöscht werden, eindeutig zu identifizieren. Alle Tabellen, die repliziert werden, müssen einen deklarierten Primärschlüssel oder eine Eindeutigkeits-Integritätsregel aufweisen. Ein eindeutiger Index reicht nicht aus.

WHERE-Klausel und Primärschlüssel

Die Primärschlüsselspalten werden in der WHERE-Klausel von replizierten UPDATE- und DELETE-Anweisungen verwendet. Wenn eine Tabelle keinen Primärschlüssel besitzt, bezieht sich die WHERE-Klausel auf alle Spalten in der Tabelle.

Siehe auch

- „Berichterstellung und Behandlung von Replikationsfehlern“ auf Seite 141
- „Standardlösung für Aktualisierungskonflikte“ auf Seite 45
- „Replikation der INSERT- und DELETE-Anweisung“ auf Seite 36

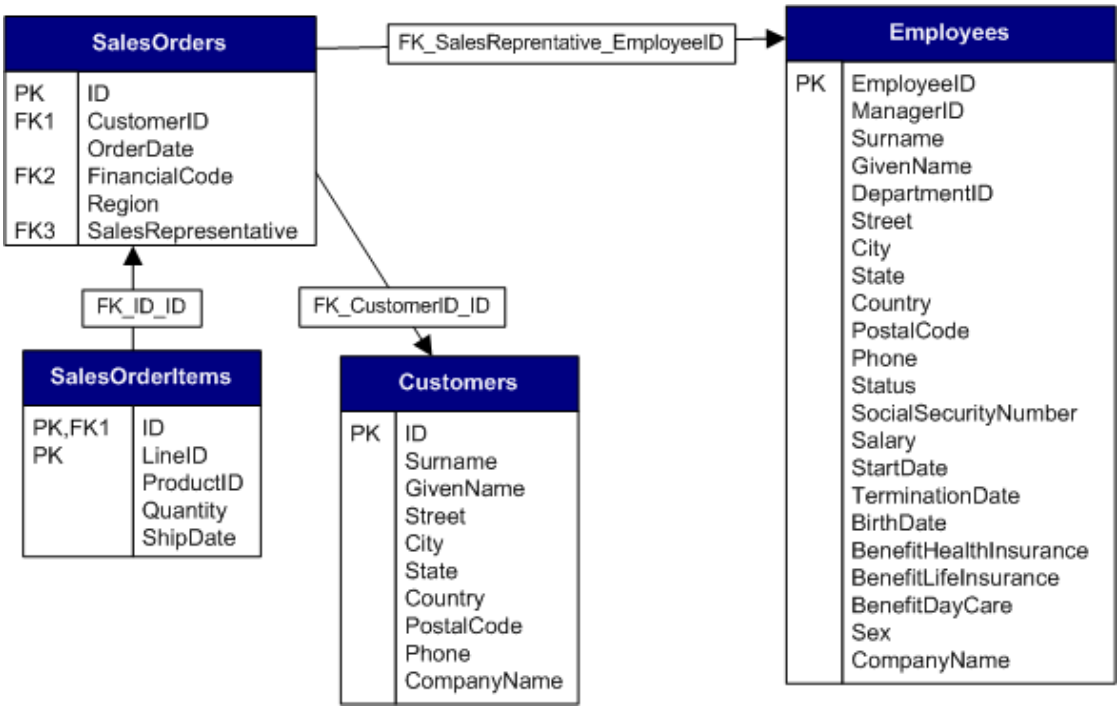
Fehler bei der referenziellen Integrität

Die Tabellen in einer relationalen Datenbank sind häufig durch Fremdschlüssel-Referenzen miteinander verknüpft. Das führt dazu, dass referenzielle Integritätsregeln sicherstellen, dass die Datenbank konsistent bleibt.

Wenn Sie nur einen Teil einer Datenbank replizieren, müssen Sie sicherstellen, dass die replizierte Datenbank weiterhin referenzielle Integrität hat.

Sie müssen versuchen, Fehler bei nicht-replizierten referenzierten Tabellen zu vermeiden. Ihre entfernten Datenbanken sollten keine Fremdschlüssel enthalten, die auf nicht-replizierte Tabellen zeigen.

Beispiel: In einer konsolidierten Datenbank hat die SalesOrders -Tabelle einen Fremdschlüssel zur Employees-Tabelle. SalesOrders.SalesRepresentative ist der Fremdschlüssel, der den Primärschlüssel Employees.EmployeeID referenziert.



Eine Publikation, PubSales, wird erstellt, die die Employees-Tabelle ausschließt, aber die gesamte SalesOrder-Tabelle umfasst.

```
CREATE PUBLICATION PubSales (
   TABLE Customers,
   TABLE SalesOrders,
   TABLE SalesOrderItems,
 );
```

Ein entfernter Benutzer, Rep1, subskribiert die PubSales-Publikation. Anschließend extrahieren Sie Rep1 aus der konsolidierten Datenbank und versuchen, eine Datenbank für Rep1 zu erstellen. Die Datenbankerstellung schlägt allerdings fehl, weil Rep1 die Employees-Tabelle fehlt. Um dieses Problem zu vermeiden, können Sie Folgendes durchführen:

- **Entfernen Sie die Fremdschlüssel-Referenz** Um Fremdschlüssel-Referenzen auszuschließen, geben Sie die Option -xf an, wenn Sie das Extraktionsdienstprogramm (dbxtract) verwenden.

  Wenn Sie allerdings die Fremdschlüssel-Referenz aus der entfernten Datenbank entfernen, gibt es keine Integritätsregel in der entfernten Datenbank, um zu verhindern, dass ein ungültiger Wert in die SalesRepresentative-Spalte der SalesOrders-Tabelle eingefügt wird.

  Wenn ein ungültiger Wert in die SalesRepresentative-Spalte in der entfernten Datenbank eingefügt wird, schlägt die replizierte INSERT-Anweisung in der konsolidierten Datenbank fehl.

- **Beziehen Sie die fehlende Tabelle in die Publikation ein** Nehmen Sie die Employees-Tabelle (oder zumindest ihren Primärschlüssel) in die Publikation auf. Beispiel:

  ```
  CREATE PUBLICATION PubSales (
     TABLE Customers,
     TABLE SalesOrders,
     TABLE SalesOrderItems,
     TABLE Products,
     TABLE Employees
  );
  ```

Siehe auch

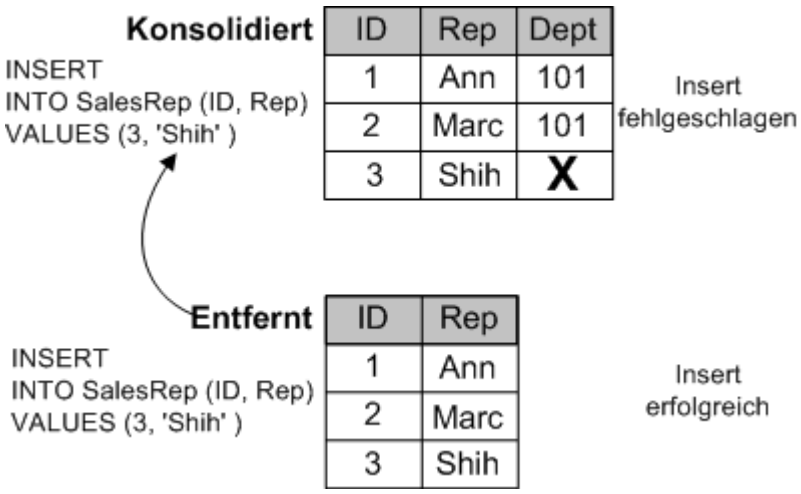
- „Berichterstellung und Behandlung von Replikationsfehlern“ auf Seite 141
- „Entitätsintegrität und referenzielle Integrität“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Einfügefehler

Wenn INSERT-Anweisungen von einer entfernten Datenbank in einer konsolidierten Datenbank repliziert werden, können Sie nur die folgenden Spalten von der Publikation ausschließen:

- Spalten, die NULL zulassen
- Spalten, die Standardwerte haben

Wenn Sie Spalten ausschließen, die nicht einer dieser Anforderungen genügen, werden INSERT-Anweisungen, die in der entfernten Datenbank durchgeführt werden, fehlschlagen, sobald sie in die konsolidierte Datenbank repliziert werden.



Hinweis
Eine Ausnahme zu diesem Beispiel ist, wenn ein BEFORE-Trigger verwendet wird, um die Spalten aufrechtzuerhalten, die nicht in der INSERT-Anweisung enthalten sind.

Siehe auch

- „Replikationskonflikte und -fehler“ auf Seite 43

Mehrfach vorhandene Primärschlüssel

Wenn alle Benutzer mit derselben Datenbank verbunden sind, kann problemlos sichergestellt werden, dass jede INSERT-Anweisung einen eindeutigen Primärschlüssel verwendet. Wenn ein Benutzer versucht, einen Primärschlüssel erneut zu verwenden, schlägt die INSERT-Anweisung fehl.

Die Situation ist in einem Replikationssystem anders, weil die Benutzer mit verschiedenen Datenbanken verbunden sind. Ein Problem kann entstehen, wenn zwei Benutzer, eine Zeile mit demselben Primärschlüssel einfügen. Beide Anweisungen sind erfolgreich, weil der Primärschlüsselwert in der jeweiligen Datenbank eindeutig ist.

Wenn diese beiden Benutzer allerdings ihre Datenbanken mit der konsolidierten Datenbank replizieren, gibt es ein Problem. Die erste Datenbank, die mit der konsolidierten Datenbank repliziert, ist erfolgreich. Die zweite Einfügung scheitert jedoch.

Primärschlüsselwerte müssen eindeutig sein

Um Primärschlüssel-Fehler zu vermeiden, müssen Sie sicherstellen, dass, wenn eine Datenbank eine Zeile einfügt, ihr Primärschlüssel garantiert eindeutig bei allen Datenbanken im System ist. Es gibt mehrere Methoden, um dieses Ziel zu erreichen, insbesondere:

1. Die GLOBAL AUTOINCREMENT-Funktion von SQL Anywhere-Datenbanken für Standardwerte verwenden.

2. Einen Primärschlüsselpool verwenden, um eine Liste von ungenutzten, eindeutigen Primärschlüsselwerten an jedem Standort bereitzuhalten.

Diese Methoden können entweder separat oder zusammen verwendet werden, um mehrfach vorhandene Primärschlüssel zu vermeiden.

Siehe auch

- „Berichterstellung und Behandlung von Replikationsfehlern“ auf Seite 141
- „GLOBAL AUTOINCREMENT-Spalten“ auf Seite 56
- „Primärschlüsselpools“ auf Seite 57

GLOBAL AUTOINCREMENT-Spalten

Verwenden Sie den GLOBAL AUTOINCREMENT-Standardwert, um jeder entfernten Datenbank eine eindeutige globale Datenbank-Identifizierungsnummer zuzuordnen.

Wenn Sie den GLOBAL AUTOINCREMENT-Standardwert für eine Spalte angeben, wird die Domäne für diese Spalte aufgeteilt. Jede Teilmenge enthält dieselbe Anzahl von Werten. Wenn Sie zum Beispiel die Partitionsgröße einer Ganzzahlspalte in einer Datenbank auf 1000 festlegen, erstreckt sich eine Partition von 1001 bis 2000, die folgende von 2001 bis 3000 usw.

SQL Anywhere liefert Standardwerte in einer Datenbank nur von der Partition, die eindeutig durch diese Datenbanknummer gekennzeichnet ist. Beispiel: Wenn Sie einer entfernten Datenbank die Identifizierungsnummer 10 zugeordnet haben, würden die Standardwerte in dieser Datenbank im Bereich 10001-11000 ausgewählt werden. Eine andere entfernte Datenbank, der die Identifizierungsnummer 11 zugewiesen ist, würde einen Standardwert für dieselbe Spalte im Bereich 11001-12000 liefern.

Siehe auch

- „Der Standardwert GLOBAL AUTOINCREMENT“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

DEFAULT GLOBAL AUTOINCREMENT-Deklaration

Sie können Standardwerte in Ihrer Datenbank einstellen, indem Sie die Spalteneigenschaften in Sybase Central auswählen oder die Klausel DEFAULT GLOBAL AUTOINCREMENT in eine CREATE TABLE- oder ALTER TABLE-Anweisung aufnehmen.

Partitionsgröße

Wahlweise kann die Partitionsgröße in Klammern unmittelbar nach dem AUTOINCREMENT-Schlüsselwort angegeben werden. Die Partitionsgröße kann jede nicht-negative Ganzzahl sein, wobei dieser Wert normalerweise so eingeteilt wird, dass seine Größe nicht überschritten werden kann.

Bei Spalten des Typs INT oder UNSIGNED INT ist die Standard-Partitionsgröße 2^{16} = 65536; bei Spalten anderen Typs ist die Standard-Partitionsgröße 2^{32} = 4294967296. Da diese Standardwerte möglicherweise unpassend sind, besonders wenn Ihre Spalten nicht vom Typ INT oder BIGINT sind, wird empfohlen, dass Sie die Partitionsgröße explizit angeben.

Beispiel

Beispiel: Die folgende SQL-Anweisung erstellt eine Tabelle mit zwei Spalten: Eine Ganzzahl, die eine Kunden-Identifizierungsnummer angibt, und eine Zeichenfolge, die den Namen des Kunden enthält. Die Identifizierungsnummer-Spalte ID verwendet den GLOBAL AUTOINCREMENT-Standardwert und hat eine Partitionsgröße von 5000.

```
CREATE TABLE Customers (
   ID   INT  DEFAULT GLOBAL AUTOINCREMENT (5000),
   name VARCHAR(128) NOT NULL,
   PRIMARY KEY (ID)
);
```

Siehe auch

- „CREATE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Primärschlüsselpools

Ein **Primärschlüsselpool** ist eine Tabelle, die eine Menge von Primärschlüsselwerten für jede Datenbank im SQL Remote-System enthält. Eine Primärschlüsselpool-Mastertabelle wird erstellt und in der konsolidierten Datenbank gespeichert. Entfernte Benutzer subskribieren die Primärschlüsselpool-Tabelle der konsolidierten Datenbank, um einen eigenen Satz von Primärschlüsselwerten zu beziehen. Wenn ein entfernter Benutzer eine neue Zeile in eine Tabelle einfügt, verwendet er eine gespeicherte Prozedur, um einen gültigen Primärschlüssel aus dem Pool auszuwählen. Der Pool wird aufrechterhalten, indem regelmäßig eine Prozedur in der konsolidierten Datenbank ausgeführt wird, die den Pool wieder auffüllt.

Die Primärschlüsselpool-Methode erfordert die folgenden Komponenten:

- **Primärschlüsselpool-Tabelle** In der konsolidierten Datenbank benötigen Sie eine Tabelle, um gültige Primärschlüsselwerte für jede Datenbank im System aufzunehmen.
- **Nachfüllprozedur** In der konsolidierten Datenbank benötigen Sie eine gespeicherte Prozedur, um die Schlüsselpooltabelle anzufüllen.
- **Primärschlüsselpools gemeinsam nutzen** Jede entfernte Datenbank im System muss ihre eigene Menge von gültigen Werten aus der Primärschlüsselpool-Tabelle der konsolidierten Datenbank subskribieren.
- **Dateneingabeprozeduren** In den entfernten Datenbanken werden neue Zeilen mit einer gespeicherten Prozedur eingegeben, die den nächsten gültigen Primärschlüsselwert aus dem Pool auswählt und dann diesen Wert aus dem Primärschlüsselpool löscht.

Siehe auch

- „Erstellen einer Primärschlüsselpool-Tabelle“ auf Seite 58
- „Primärschlüsselpool replizieren“ auf Seite 59
- „Schlüsselpool füllen und nachfüllen“ auf Seite 59
- „Verwenden der Primärschlüssel aus dem Schlüsselpool“ auf Seite 61

Erstellen einer Primärschlüsselpool-Tabelle

Erstellen einer Primärschlüsselpool-Tabelle.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. In der konsolidierten Datenbank führen Sie die folgende Anweisung aus, um eine Primärschlüsselpool-Tabelle zu erstellen:

```
CREATE TABLE KeyPool (
   table_name VARCHAR(128) NOT NULL,
   value INTEGER NOT NULL,
   location CHAR(12) NOT NULL,
   PRIMARY KEY (table_name, value),
);
```

Spalte	Beschreibung
table_name	Enthält die Namen von Tabellen, für die Primärschlüsselpools verwaltet werden müssen. Beispiel: Wenn neue Handelsvertreter nur in der konsolidierten Datenbank hinzugefügt werden, benötigt nur die Customers-Tabelle einen Primärschlüsselpool und diese Spalte wäre redundant.
value	Enthält eine Liste von Primärschlüsselwerten. Jeder Wert ist für jede in table_name aufgelistete Tabelle eindeutig.
location	Ein Bezeichner für den Empfänger. In manchen Systemen kann dies derselbe wie der rep_key-Wert in der SalesReps-Tabelle sein. In anderen Systemen gibt es andere Benutzer als Handelsvertreter, und hier sollten die beiden Bezeichner unterschiedlich sein.

2. Um die Performance zu verbessern, führen Sie die folgende Anweisung aus, um einen Index für die Primärschlüssel-Tabelle zu erstellen:

```
CREATE INDEX KeyPoolLocation
ON KeyPool (table_name, location, value);
```

Ergebnisse

Der Primärschlüsselpool wird erstellt.

Siehe auch

- „CREATE TABLE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE INDEX-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Primärschlüsselpool replizieren

Erstellen Sie eine getrennte Publikation für den Primärschlüsselpool und subskribieren Sie Benutzer dafür.

Voraussetzungen

Die Primärschlüsselpool-Tabelle muss bereits erstellt sein.

Kontext und Bemerkungen

Sie können den Primärschlüsselpool entweder in eine vorhandene Publikation einbeziehen oder als getrennte Publikation verteilen.

Den Primärschlüsselpool replizieren

1. In der konsolidierten Datenbank erstellen Sie eine Publikation für die Primärschlüsselpool-Daten.

   ```
   CREATE PUBLICATION KeyPoolData (
      TABLE KeyPool SUBSCRIBE BY location
   );
   ```

2. Erstellen Sie Subskriptionen von der "KeyPoolData"-Publikation für jede entfernte Datenbank.

   ```
   CREATE SUBSCRIPTION
     TO KeyPoolData( 'Sam_Singer' )
     FOR user1;
   CREATE SUBSCRIPTION
     TO KeyPoolData( 'user2' )
     FOR user2;
   ...
   ```

   Das Subskriptionsargument ist der Bezeichner für den Empfänger (location).

Ergebnisse

Der Primärschlüsselpool wird repliziert.

Siehe auch

- „CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Schlüsselpool füllen und nachfüllen

Jedes Mal, wenn ein entfernter Benutzer einen neuen Kunden hinzufügt, wird sein Pool an verfügbaren Primärschlüsseln um einen vermindert. Sie müssen regelmäßig den Inhalt der Primärschlüsselpool-Tabelle in der konsolidierten Datenbank auffüllen und die neuen Primärschlüssel an die entfernten Datenbanken replizieren.

Voraussetzungen

Die Primärschlüsselpool-Tabelle muss bereits erstellt sein.

Aufgabe

1. In der konsolidierten Datenbank erstellen Sie eine Prozedur, um den Primärschlüsselpool zu füllen.

   Hinweis
   Sie können keinen Trigger verwenden, um den Primärschlüsselpool anzufüllen, da Triggeraktionen nicht repliziert werden.

   Zum Beispiel:

   ```
   CREATE PROCEDURE ReplenishPool()
   BEGIN
      FOR EachTable AS TableCursor
      CURSOR FOR
         SELECT table_name
         AS CurrTable, max(value) as MaxValue
         FROM KeyPool
         GROUP BY table_name
      DO
         FOR EachRep AS RepCursor
         CURSOR FOR
            SELECT location
            AS CurrRep, COUNT(*) AS NumValues
            FROM KeyPool
            WHERE table_name = CurrTable
            GROUP BY location
         DO
            // make sure there are 100 values.
            // Fit the top-up value to your
            // requirements
            WHILE NumValues < 100 LOOP
               SET MaxValue = MaxValue + 1;
               SET NumValues = NumValues + 1;
               INSERT INTO KeyPool
               (table_name, location, value)
               VALUES
               (CurrTable, CurrRep, MaxValue);
            END LOOP;
         END FOR;
      END FOR;
   END;
   ```

2. Fügen Sie einen ersten Primärschlüsselwert in den Primärschlüsselpool für jeden Benutzer ein.

   Die ReplenishPool-Prozedur verlangt, dass es zumindest einen Primärschlüssel für jeden Subskribenten geben muss, damit sie den Höchstwert bestimmen und eins hinzufügen kann, um die nächste Menge zu erzeugen.

   Um den Pool anfänglich zu füllen, fügen Sie einen einzelnen Wert für jeden Benutzer ein und rufen dann ReplenishPool auf, um den Rest aufzufüllen. Das folgende Beispiel zeigt dies für drei entfernte Benutzer und einen einzigen konsolidierten Benutzer namens Office:

   ```
   INSERT INTO KeyPool VALUES( 'Customers', 40, 'user1' );
   INSERT INTO KeyPool VALUES( 'Customers', 41, 'user2' );
   ```

```
INSERT INTO KeyPool VALUES( 'Customers', 42, 'user3' );
INSERT INTO KeyPool VALUES( 'Customers', 43, 'Office');
CALL ReplenishPool();
```

Die ReplenishPool-Prozedur füllt den Pool für jeden Benutzer mit bis zu 100 Werten an. Der benötigte Wert hängt davon ab, wie oft Benutzer Zeilen in die Tabellen der Datenbank einfügen.

3. Führen Sie ReplenishPool regelmäßig aus.

   Die ReplenishPool-Prozedur muss in der konsolidierten Datenbank regelmäßig ausgeführt werden, um den Pool von Primärschlüsselwerten in der KeyPool-Tabelle wieder aufzufüllen.

Ergebnisse

Die Primärschlüsselpool-Tabelle der konsolidierten Datenbank wird wieder aufgefüllt und die neuen Primärschlüssel werden in die entfernten Datenbanken repliziert.

Verwenden der Primärschlüssel aus dem Schlüsselpool

Wenn entfernte Benutzer neue Kunden hinzufügen, kommt der Primärschlüssel aus dem Pool an verfügbaren Primärschlüsseln des entfernten Benutzers.

Voraussetzungen

Die Primärschlüsselpool-Tabelle muss bereits erstellt sein.

Aufgabe

Wenn ein Handelsvertreter einen neuen Kunden der Customers-Tabelle hinzufügt, wird der einzufügende Primärschlüsselwert mittels einer gespeicherten Prozedur ermittelt. Dieses Beispiel verwendet eine gespeicherte Prozedur, die den Primärschlüsselwert liefert. Auch das Einfügen erfolgt mithilfe einer gespeicherten Prozedur.

1. Erstellen Sie eine Prozedur zur Ausführung auf den entfernten Datenbanken, um einen Primärschlüssel von der Primärschlüsselpool-Tabelle zu erhalten.

   Beispiel: Die NewKey-Prozedur liefert einen Ganzzahlwert vom Primärschlüsselpool und löscht den Wert aus dem Pool.

```
CREATE PROCEDURE NewKey(
      IN @table_name VARCHAR(40),
      OUT @value INTEGER )
BEGIN
   DECLARE NumValues INTEGER;

   SELECT COUNT(*), MIN(value)
   INTO NumValues, @value
        FROM KeyPool
        WHERE table_name = @table_name
        AND location = CURRENT PUBLISHER;
   IF NumValues > 1 THEN
      DELETE FROM KeyPool
      WHERE table_name = @table_name
      AND value = @value;
```

```
   ELSE
   // Never take the last value, because
   // ReplenishPool will not work.
   // The key pool should be kept large enough
   // that this never happens.
      SET @value = NULL;
   END IF;
END;
```

Die NewKey-Prozedur nützt den Umstand, dass der "Sales Representative"-Bezeichner der aktuelle Publikationseigentümer (CURRENT PUBLISHER) der entfernten Datenbank ist.

2. Erstellen Sie eine Prozedur, die in den entfernten Datenbanken ausgeführt wird, um eine neue Zeile in eine subskribierte Tabelle einzufügen.

   Beispiel: Die NewCustomers-Prozedur fügt einen neuen Kunden in die Tabelle ein, wobei der von NewKey gelieferte Wert verwendet wird, um den Primärschlüssel zu ermitteln.

```
CREATE PROCEDURE NewCustomers(
      IN customer_name CHAR( 40 ) )
BEGIN
   DECLARE new_cust_key INTEGER ;
   CALL NewKey( 'Customers', new_cust_key );
   INSERT
   INTO Customers (
      cust_key,
      name,
      location
      )
   VALUES (
      'Customers ' ||
      CONVERT (CHAR(3), new_cust_key),
      customer_name,
      CURRENT PUBLISHER
      );
END
```

   Sie können diese Prozedur erweitern, indem Sie den new_cust_key-Wert, der von NewKey geliefert wird, testen, um zu überprüfen, ob er NULL ist, und ein Einfügen zu verhindern, falls dies zutrifft.

Ergebnisse

Der Primärschlüssel wird festgelegt.

Zeilenverteilung unter entfernten Datenbanken

Jede entfernte Datenbank verfügt unter Umständen über eine jeweils andere Teilmenge der Daten, die in der konsolidierten Datenbank gespeichert sind. Sie können Ihre Publikationen und Subskriptionen so erstellen, dass Daten auf entfernte Datenbanken verteilt werden.

Die Verteilung kann disjunkt sein oder Überlappungen enthalten. Wenn zum Beispiel jeder Mitarbeiter seinen eigenen Kundenkreis hat und es keine gemeinsamen Kunden gibt, ist die Verteilung **disjunkt**. Wenn es gemeinsame Kunden gibt, die in mehreren entfernten Datenbanken vorhanden sind, enthält die Verteilung **Überlappungen**.

Manchmal müssen die Zeilen einer Tabelle partitioniert werden, auch wenn der Subskriptionsausdruck nicht in der Tabelle vorkommt.

Manchmal, wenn es eine Viele-zu-Viele-Beziehung in der Datenbank gibt, müssen die Tabellen partitioniert werden.

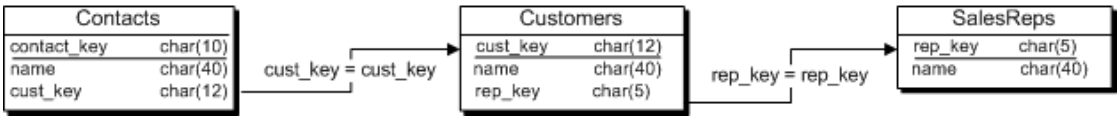
Siehe auch

- „Disjunkte Datenpartitionen“ auf Seite 63
- „Partitionen mit Überlappung“ auf Seite 68

Disjunkte Datenpartitionen

Eine Datenpartitionierung ist disjunkt, wenn die entfernten Datenbanken keine Daten gemeinsam nutzen. Beispiel: Jeder Handelsvertreter hat seinen eigenen Kundenstamm und teilt keine Kunden mit anderen Handelsvertretern.

Im folgenden Beispiel speichern drei Tabellen Informationen über die Interaktionen zwischen Handelsvertretern und Kunden: Customers, Contacts und SalesReps. Jeder Handelsvertreter bedient mehrere Kunden. Bei manchen Kunden gibt es einen einzigen Kontakt, während andere Kunden mehrere Kontakte haben.



Beschreibung der Contacts-, Customers- und SalesReps-Tabellen

Die folgende Tabelle beschreibt die Customers-, Contacts- und SalesReps-Datenbanktabellen. Weitere Hinweise zu diesen Tabellen finden Sie unter „Disjunkte Datenpartitionen“ auf Seite 63.

Tabelle	Beschreibung	Tabellendefinition
Contacts	Alle einzelnen Kontakte, die mit der Firma Geschäfte machen. Jeder Kontakt gehört einem einzigen Kunden. Die Contacts-Tabelle enthält folgende Spalten: • **contact_key** Ein Bezeichner für jeden Kontakt. Dies ist der Primärschlüssel. • **name** Der Name jedes Kontakts • **cust_key** Ein Bezeichner für den Kunden, dem der Kontakt gehört. Dies ist ein Fremdschlüssel zur Customers-Tabelle.	`CREATE TABLE Contacts ( contact_key CHAR(12) NOT NULL, name CHAR(40) NOT NULL, cust_key CHAR(12) NOT NULL, FOREIGN KEY REFERENCES Customers, PRIMARY KEY (contact_key) );`

Tabelle	Beschreibung	Tabellendefinition
Customers	Alle Kunden, die mit der Firma Geschäfte machen. Die Customers-Tabelle enthält die folgenden Spalten: • **cust_key** Ein Bezeichner für jeden Kunden. Dies ist der Primärschlüssel. • **name** Der Name jedes Kunden • **rep_key** Ein Bezeichner für den Handelsvertreter in einer Geschäftsbeziehung. Dies ist ein Fremdschlüssel zur Tabelle "SalesReps".	`CREATE TABLE Customers ( cust_key CHAR(12) NOT NULL, name CHAR(40) NOT NULL, rep_key CHAR(12) NOT NULL, FOREIGN KEY REFERENCES SalesReps, PRIMARY KEY (cust_key) );`
SalesReps	Alle Handelsvertreter, die für die Firma arbeiten. Die SalesReps-Tabelle enthält die folgenden Spalten: • **rep_key** Ein Bezeichner für jeden Handelsvertreter. Dies ist der Primärschlüssel. • **name** Der Name jedes Handelsvertreters	`CREATE TABLE SalesReps ( rep_key CHAR(12) NOT NULL, name CHAR(40) NOT NULL, PRIMARY KEY (rep_key) );`

Ein Handelsvertreter muss eine Publikation subskribieren, die die folgenden Informationen bereitstellt:

- **Eine Liste aller Handelsvertreter, die für die Firma arbeiten** Die folgende Anweisung erstellt eine Publikation, die die gesamte SalesRep-Tabelle publiziert:

  ```
  CREATE PUBLICATION SalesRepData (
     Table SalesReps ...)
  );
  ```

- **Eine Liste der ihm zugeordneten Kunden** Diese Informationen sind in der Customers-Tabelle verfügbar. Die folgende Anweisung erstellt eine Publikation, die die Customers-Tabelle publiziert, die die Zeilen enthält, die mit dem Wert der rep_key-Spalte in der Customers-Tabelle übereinstimmen:

  ```
  CREATE PUBLICATION SalesRepData (
     TABLE Customers SUBSCRIBE BY rep_key ...
  );
  ```

- **Eine Liste der Kontaktinformationen für seine zugeordneten Kunden** Diese Informationen sind in der Contacts-Tabelle verfügbar. Die Contacts-Tabelle muss unter den Handelsvertretern aufgeteilt werden, aber es gibt keine Referenz zum rep_key-Wert in der SalesRep-Tabelle. Um dieses Problem zu lösen, verwenden Sie eine Unterabfrage im Contacts-Artikel, die die rep_key-Spalte der Customers-Tabelle referenziert.

  Die folgende Anweisung erstellt eine Publikation, die die Contacts-Tabelle mit den Zeilen enthält, die eine Referenz auf die rep_key-Spalte in der Customers-Tabelle enthalten:

  ```
  CREATE PUBLICATION SalesRepData ( ...
     TABLE Contacts
  ```

```
        SUBSCRIBE BY (SELECT rep_key
           FROM Customers
           WHERE Contacts.cust_key = Customers.cust_key )
   );
```

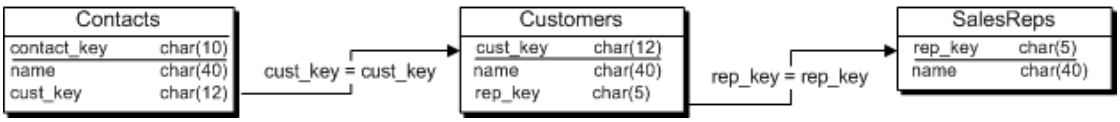
Eine Zeile in der Customers-Tabelle hat den cust_key-Wert in der aktuellen Zeile der Contacts-Tabelle, und die WHERE-Klausel in der SUBSCRIBE BY-Anweisung stellt sicher, dass die Unterabfrage nur einen einzigen Wert zurückgibt.

Dir folgende Anweisung erstellt die vollständige Publikation:

```
CREATE PUBLICATION SalesRepData (
   TABLE SalesReps,
   TABLE Customers
      SUBSCRIBE BY rep_key,
   TABLE Contacts
      SUBSCRIBE BY (SELECT rep_key
         FROM Customers
         WHERE Contacts.cust_key = Customers.cust_key )
);
```

BEFORE UPDATE-Trigger

Im folgenden Beispiel speichern drei Tabellen Informationen über die Interaktionen zwischen Handelsvertretern und Kunden: Customers, Contacts und SalesReps. Jeder Handelsvertreter bedient mehrere Kunden. Bei manchen Kunden gibt es einen einzigen Kontakt, während andere Kunden mehrere Kontakte haben.



Eine detaillierte Beschreibung der Tabellen finden Sie unter Beschreibung der Contacts-, Customers- und SalesReps-Tabellen auf Seite 63.

Ein Handelsvertreter subskribiert eine Publikation, die eine Kopie der SalesRep-Tabelle, eine Kopie der Customers-Tabelle mit den ihm zugeordneten Kunden und eine Kopie der Contacts-Tabelle mit den Details der Kontakte bereitstellt, die sich auf seine Kunden beziehen. Beispiel: Jeder Handelsvertreter subskribiert die folgende Publikation:

```
CREATE PUBLICATION SalesRepData (
   TABLE SalesReps,
   TABLE Customers
      SUBSCRIBE BY rep_key,
   TABLE Contacts
      SUBSCRIBE BY (SELECT rep_key
         FROM Customers
         WHERE Contacts.cust_key = Customers.cust_key )
);
```

Eine detaillierte Beschreibung dieser Publikation finden Sie unter „Disjunkte Datenpartitionen“ auf Seite 63.

Referenzielle Integrität bewahren

Diese Neuzuweisung von Zeilen wird manchmal als **territoriale Neuaufteilung** bezeichnet, da sie eine übliche Eigenschaft bei Anwendungen ist, die den Verkauf automatisieren und bei denen Kunden dem Verkaufspersonal regelmäßig neu zugeordnet werden.

Wenn ein Kunde einem neuen Handelsvertreter zugewiesen wird, erfolgt in der konsolidierten Datenbank eine Aktualisierung des rep_key-Werts in der Customers-Tabelle.

Die folgende Anweisung weist einem Kunden, cust1, einen anderen Handelsvertreter, rep2, zu..

```
UPDATE Customers
SET rep_key = 'rep2'
WHERE cust_key = 'cust1';
```

Diese Aktualisierung wird folgendermaßen repliziert:

- Als DELETE-Anweisung in der Customers-Tabelle in der entfernten Datenbank des alten Handelsvertreters.
- Als INSERT-Anweisung in der Customers-Tabelle in der entfernten Datenbank des neuen Handelsvertreters.

Die Contacts-Tabelle wird **nicht** geändert. Es gibt keine Einträge im Transaktionslog der konsolidierten Datenbank zur Contacts-Tabelle. Das führt dazu, dass SQL Remote in den entfernten Datenbanken nicht in der Lage ist, die cust_key-Zeilen der Contacts-Tabelle neu zuzuweisen. Das kann das folgende Problem mit der referenziellen Integrität bewirken: Die Contacts-Tabelle in der entfernten Datenbank des alten Handelsvertreters enthält einen cust_key-Wert, für den es keine Kunden in der Tabelle mehr gibt.

Eine Lösung ist die Verwendung eines BEFORE UPDATE-Triggers. Ein BEFORE UPDATE-Trigger führt keine Änderungen in den Datenbanktabellen durch, aber erstellt einen Eintrag im Transaktionslog der konsolidierten Datenbank.

Dieser BEFORE UPDATE-Trigger muss folgendermaßen ausgelöst werden:

- Bevor die UPDATE-Anweisung ausgeführt wird, damit der BEFORE-Wert der Zeile ausgewertet und dem Transaktionslog hinzugefügt wird.
- FOR EACH ROW anstatt für jede Anweisung. Die vom Trigger gelieferten Informationen müssen der neue Subskriptionsausdruck sein.

Beispiel: Die folgende Anweisung erstellt einen BEFORE UPDATE-Trigger.

```
CREATE TRIGGER "UpdateCustomer" BEFORE UPDATE OF "rep_key"
// only fire the trigger when rep_key is modified, not any other column
ORDER 1 ON "Cons"."Customers"
/* REFERENCING OLD AS old_name NEW AS new_name */
REFERENCING NEW AS NewRow
   OLD AS OldRow
FOR EACH ROW
BEGIN
// determine the new subscription expression
   // for the Customers table
     UPDATE Contacts
   PUBLICATION SalesRepData
```

```
   OLD SUBSCRIBE BY ( OldRow.rep_key )
   NEW SUBSCRIBE BY ( NewRow.rep_key )
   WHERE cust_key = NewRow.cust_key;
END
;
```

SQL Remote verwendet die im Transaktionslog aufgezeichneten Informationen, um zu ermitteln, welche Subskribenten welche Zeilen erhalten.

Das Transaktionslog der konsolidierten Datenbank enthält zwei Einträge, nachdem diese Anweisung ausgeführt wurde:

- INSERT- und DELETE-Anweisungen für die Contacts-Tabelle, die vom BEFORE UPDATE-Trigger generiert wurden.

```
--BEGIN TRIGGER-1029-0000461705
--BEGIN TRANSACTION-1029-0000461708
BEGIN TRANSACTION
go
--UPDATE PUBLICATION-1029-0000461711 Cons.Contacts
--PUBLICATION-1029-0000461711-0002-NEW_SUBSCRIBE_BY-rep2
--PUBLICATION-1029-0000461711-0002-OLD_SUBSCRIBE_BY-rep1
--NEW-1029-0000461711
--INSERT INTO Cons.Contacts(contact_key,name,cust_key)
--VALUES ('5','Joe','cust1')
go
--OLD-1029-0000461711
--DELETE FROM Cons.Contacts
-- WHERE contact_key='5'
go
--END TRIGGER-1029-0000461743
```

- Die ursprüngliche UPDATE-Anweisung, die ausgeführt wurde, sowie INSERT- und DELETE-Anweisungen für jene Benutzer, die eine Zeile gewonnen bzw. verloren haben.

```
--PUBLICATION-1029-0000461746-0002-NEW_SUBSCRIBE_BY-rep2
--PUBLICATION-1029-0000461746-0002-OLD_SUBSCRIBE_BY-rep1
--NEW-1029-0000461746
--INSERT INTO Cons.Customers(cust_key,name,rep_key)
--VALUES ('cust1','company1','rep2')
go
--OLD-1029-0000461746
--DELETE FROM Cons.Customers
-- WHERE cust_key='cust1'
go
--UPDATE-1029-0000461746
UPDATE Cons.Customers
   SET rep_key='rep2'
VERIFY (rep_key)
VALUES ('1')
 WHERE cust_key='cust1'
go
--COMMIT-1029-0000461785
COMMIT WORK
```

SQL Remote durchsucht das Transaktionslog nach den BEFORE- und AFTER-Parametern. Basierend auf diesen Informationen kann ermittelt werden, welche entfernten Benutzer eine INSERT-, UPDATE- oder DELETE-Anweisung erhalten.

- Wenn ein Benutzer in der BEFORE-Liste und nicht in der AFTER-Liste steht, wird eine DELETE-Anweisung in der Contacts-Tabelle gesendet.
- Wenn ein Benutzer in der AFTER-Liste und nicht in der BEFORE-Liste steht, wird eine INSERT-Anweisung in der Contacts-Tabelle gesendet.
- Wenn ein Benutzer sowohl in der BEFORE- als auch der AFTER-Liste steht, wird nichts in der Contacts-Tabelle durchgeführt, aber die UPDATE-Anweisung in der Customers-Tabelle wird gesendet.

Wenn die BEFORE- und AFTER-Listen identisch sind, besitzt der entfernte Benutzer die Zeile bereits und es wird eine UPDATE-Anweisung gesendet.

Hinweise zu Triggern

Im folgenden Beispiel müssen Sie einen BEFORE UPDATE-Trigger verwenden. In anderen Fällen sind BEFORE DELETE und BEFORE INSERT erforderlich.

```
UPDATE table-name
PUBLICATION pub-name
   SUBSCRIBE BY sub-expression
WHERE search-condition;
```

In diesem Beispiel verwenden Sie einen BEFORE-Trigger.

```
UPDATE table-name
PUBLICATION publication-name
   OLD SUBSCRIBE BY old-subscription-expression
   NEW SUBSCRIBE BY new-subscription-expression
WHERE search-condition;
```

Die UPDATE-Anweisung listet die betroffene Publikation und Tabelle auf. Die WHERE-Klausel in der Anweisung beschreibt die betroffenen Zeilen. Die UPDATE-Anweisung ändert nicht die Daten in der Tabelle; sie führt Einträge im Transaktionslog durch.

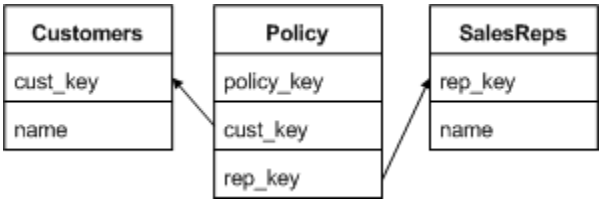
In diesem Beispiel gibt der Subskriptionsausdruck einen einzelnen Wert zurück. Unterabfragen, die mehrere Werte ausgeben, können allerdings ebenfalls verwendet werden. Der Wert des Subskriptionsausdrucks muss der Wert nach der Aktualisierung sein.

In diesem Beispiel ist der einzige Subskribent der Zeile der neue Handelsvertreter. Ein Beispiel für eine Zeile, die bestehende und neue Subskribenten hat, finden Sie unter „Partitionen mit Überlappung“ auf Seite 68.

Partitionen mit Überlappung

Eine Datenpartitionierung überlappt, wenn die entfernten Datenbanken Daten gemeinsam nutzen. Beispiel: Handelsvertreter haben teilweise dieselben Kunden.

Nehmen wir an, dass drei Tabellen Informationen über die Beziehungen zwischen Handelsvertretern und Kunden speichern: Customers, Policy und SalesReps. Jeder Handelsvertreter betreut mehrere Kunden, und einige Kunden machen mit mehr als einem Handelsvertreter Geschäfte. Die Policy-Tabelle hat Fremdschlüssel zur Customers-Tabelle und zur SalesReps-Tabelle. Es gibt eine Viele-zu-Viele-Beziehung zwischen Customers und SalesReps.



Beschreibung der Customers-, Policy- und SalesReps-Tabellen

Die folgende Tabelle beschreibt die Customers-, Policy- und SalesReps-Datenbanktabellen, wie unter „Partitionen mit Überlappung“ auf Seite 68 behandelt.

Tabelle	Beschreibung

Customers	Alle Kunden, die mit der Firma Geschäfte machen. Die Customers-Tabelle hat die folgenden Spalten:
	• **cust_key** Eine Primärschlüsselspalte, die einen Bezeichner für jeden Kunden enthält.
	• **name** Eine Spalte, die den Namen jedes Kunden enthält.
	Die folgenden Anweisungen erstellen diese Tabelle:
	`CREATE TABLE Customers (`
	`  cust_key CHAR(12) NOT NULL,`
	`  name CHAR(40) NOT NULL,`
	`  PRIMARY KEY (cust_key)`
	`);`

Tabelle	Beschreibung

Policy	Eine Tabelle mit drei Spalten, die die Viele-zu-Viele-Beziehung zwischen Kunden und Handelsvertretern verkörpert. Die Tabelle "Policy" umfasst folgende Spalten:
	• **policy_key** Eine Primärschlüsselspalte, die einen Bezeichner für die Geschäftsbeziehung enthält
	• **cust_key** Eine Spalte mit einem Fremdschlüssel für den Kundenvertreter in einer Geschäftsbeziehung
	• **rep_key** Eine Spalte mit einem Fremdschlüssel für den Handelsvertreter in einer Geschäftsbeziehung
	Die folgenden Anweisungen erstellen diese Tabelle:
	`CREATE TABLE Policy (`
	`policy_key CHAR(12) NOT NULL,`
	`cust_key CHAR(12) NOT NULL,`
	`rep_key CHAR(12) NOT NULL,`
	`FOREIGN KEY ( cust_key )`
	`REFERENCES Customers ( cust_key ),`
	`FOREIGN KEY ( rep_key )`
	`REFERENCES SalesReps (rep_key ),`
	`PRIMARY KEY ( policy_key )`
	`);`

SalesReps	Alle Handelsvertreter, die für die Firma arbeiten. Die SalesReps-Tabelle hat die folgenden Spalten:
	• **rep_key** Ein Bezeichner für jeden Handelsvertreter. Dies ist der Primärschlüssel.
	• **name** Der Name jedes Handelsvertreters
	Die folgenden Anweisungen erstellen diese Tabelle:
	`CREATE TABLE SalesReps (`
	`rep_key CHAR(12) NOT NULL,`
	`name CHAR(40) NOT NULL,`
	`PRIMARY KEY (rep_key)`
	`);`

Daten verteilen

Die Viele-zu-Viele-Beziehung zwischen Kunden und Handelsvertretern stellt eine Herausforderung für die korrekte gemeinsame Nutzung der Informationen dar.

Handelsvertreter müssen eine Publikation subskribieren, die die folgernden Informationen bereitstellt:

- **Die gesamte SalesReps-Tabelle** Es gibt keine Qualifizierer für diesen Artikel, daher wird die gesamte SalesReps-Tabelle in die Publikation aufgenommen.

```
...
   TABLE SalesReps,
...
```

- **Diejenigen Zeilen aus der Policy-Tabelle, die die Geschäftsbeziehungen des Handelsvertreters enthalten, der die Daten subskribiert** Dieser Artikel verwendet einen SUBSCRIBE BY-Subskriptionsausdruck, um eine Spalte zu bestimmen, die die Daten unter den Handelsvertretern aufteilt:

  ```
  ...
     TABLE Policy
     SUBSCRIBE BY rep_key,
  ...
  ```

  Der Subskriptionsausdruck stellt sicher, dass jeder Handelsvertreter nur jene Zeilen der Tabelle erhält, bei denen der Wert der rep_key-Spalte dem Wert entspricht, der in der Subskription angeführt ist.

  Die Partitionierung der Policy-Tabelle ist **disjunkt**: Es gibt keine Zeilen, die von mehr als einem Subskribenten genutzt werden.

- **Diejenigen Zeilen aus der Customers-Tabelle, die Kunden auflisten, die Geschäftsbeziehungen mit dem Handelsvertreter unterhalten, der die Daten subskribiert** Die Customers-Tabelle hat keine Referenz zum Handelsvertreter-Wert, der in den Subskriptionen verwendet wird, um die Daten aufzuteilen. Dieses Problem kann gelöst werden, indem eine Unterabfrage in der Publikation verwendet wird.

  Jede Zeile in der Customers-Tabelle kann sich auf viele Zeilen in der SalesReps-Tabelle beziehen und von vielen Handelsvertreter-Datenbanken gemeinsam genutzt werden. Das bedeutet, dass es überlappende Subskriptionen gibt.

  Ein Subskriptionsausdruck mit einer Unterabfrage wird verwendet, um die Partition zu definieren. Der Artikel wird folgendermaßen festgelegt:

  ```
  ...
     TABLE Customers SUBSCRIBE BY (
        SELECT rep_key
        FROM Policy
        WHERE Policy.cust_key =
           Customers.cust_key
     ),
  ...
  ```

  Die Partitionierung der Customers-Tabelle ist **nicht disjunkt**: Es gibt Zeilen, die von mehr als einem Subskribenten genutzt werden.

Dir folgende Anweisung erstellt die vollständige Publikation:

```
CREATE PUBLICATION SalesRepData (
   TABLE SalesReps,
   TABLE Policy SUBSCRIBE BY rep_key,
   TABLE Customers SUBSCRIBE BY (
      SELECT rep_key FROM Policy
      WHERE Policy.cust_key =
         Customers.cust_key
   )
);
```


Mehrwertige Unterabfragen in Publikationen

Die Unterabfrage im Customers-Artikel gibt eine einzelne Spalte (rep_key) in der Ergebnismenge zurück, kann aber mehrere Zeilen ausgeben, die allen jenen Handelsvertretern entsprechen, die mit diesem bestimmten Kunden Geschäfte machen. Wenn ein Subskriptionsausdruck mehrere Werte hat, wird die Zeile an alle Subskribenten repliziert, deren Subskription einem der Werte entspricht. Diese Fähigkeit, mehrwertige Subskriptionsausdrücke zu haben, ermöglicht eine überlappende Partitionierung einer Tabelle.

Siehe auch

- „CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Wahrung der referenziellen Integrität, wenn Zeilen Subskribenten neu zugeordnet werden

Um eine Geschäftsbeziehung zwischen einem Kunden und einem Handelsvertreter zu beenden, wird eine Zeile in der Policy-Tabelle gelöscht. In diesem Beispiel wird die Änderung an der Policy-Tabelle korrekt an den alten Handelsvertreter repliziert. Es wurden allerdings keine Änderungen in der Customers-Tabelle durchgeführt, und daher werden auch keine Änderungen in der Customers-Tabelle an den alten Handelsvertreter repliziert.

Ohne Trigger kann das zu inkorrekten Daten in der Customers-Tabelle eines Subskribenten führen. Die gleiche Art von Problem taucht auf, wenn eine neue Zeile der Policy-Tabelle hinzugefügt wird.

Mit Triggern das Problem lösen

Die Lösung liegt darin, BEFORE-Trigger zu schreiben die ausgelöst werden, sobald Änderungen in der Policy-Tabelle vorgenommen werden. Die speziellen Trigger führen keine Änderungen an den Datenbanktabellen durch, sondern setzen einen Eintrag in das Transaktionslog, das SQL Remote zum Aufrechterhalten der Daten in den Datenbanken der Subskribenten verwendet.

Ein BEFORE INSERT-Trigger

Beispiel: Die folgenden Anweisungen erstellen einen BEFORE INSERT-Trigger, der Einfügungen in der Policy-Tabelle protokolliert und sicherstellt, dass die entfernten Datenbanken die korrekten Daten enthalten.

```
CREATE TRIGGER InsPolicy
BEFORE INSERT ON Policy
REFERENCING NEW AS NewRow
FOR EACH ROW
BEGIN
   UPDATE Customers
   PUBLICATION SalesRepData
   SUBSCRIBE BY (
      SELECT rep_key
      FROM Policy
      WHERE cust_key = NewRow.cust_key
      UNION ALL
      SELECT NewRow.rep_key
   )
```

```
   WHERE cust_key = NewRow.cust_key;
END;
```

Ein BEFORE DELETE-Trigger

Die folgenden Anweisungen erstellen einen BEFORE DELETE-Trigger, der Löschungen in der Policy-Tabelle protokolliert:

```
CREATE TRIGGER DelPolicy
BEFORE DELETE ON Policy
REFERENCING OLD AS OldRow
FOR EACH ROW
BEGIN
   UPDATE Customers
   PUBLICATION SalesRepData
   SUBSCRIBE BY (
      SELECT rep_key
      FROM Policy
      WHERE cust_key = OldRow.cust_key
      AND Policy_key <> OldRow.Policy_key
   )
   WHERE cust_key = OldRow.cust_key;
END;
```

Die SUBSCRIBE BY-Klausel der UPDATE PUBLICATION-Anweisung enthält eine Unterabfrage, und diese Unterabfrage kann mehrwertig sein.

Mehrwertige Unterabfragen

Die Unterabfrage in der SUBSCRIBE-Klausel von UPDATE PUBLICATION ist ein UNION-Ausdruck und kann mehrwertig sein:

```
...
SELECT rep_key
FROM Policy
WHERE cust_key = NewRow.cust_key
UNION ALL
SELECT NewRow.rep_key
...
```

- Der erste Teil von UNION ist die Menge von vorhandenen Handelsvertretern, die den Kunden betreuen. Er wird der "Policy"-Tabelle entnommen.

  Die Ergebnismenge der Subskriptionsabfrage muss aus all den Handelsvertretern bestehen, die die Zeile empfangen, und nicht nur aus den neuen Handelsvertretern.

- Der zweite Teil des UNION-Ausdrucks ist der der INSERT-Anweisung entnommene rep_key-Wert für den neuen Handelsvertreter, der den Kunden betreut.

Die Unterabfrage im BEFORE DELETE-Trigger ist mehrwertig:

```
...
SELECT rep_key
FROM Policy
WHERE cust_key = OldRow.cust_key
AND rep_key <> OldRow.rep_key
...
```

- Die Unterabfrage nimmt rep_key-Werte aus der Policy-Tabelle. Die Werte enthalten die Primärschlüssel-Werte von allen Handelsvertretern, die den Kunden betreuen, der übertragen wird

(WHERE **cust_key** = **OldRow.cust_key**), mit Ausnahme dessen, der gelöscht wird (AND **rep_key** <> **OldRow.rep_key**).

Die Ergebnismenge der Subskriptionsabfrage muss aus all jenen Werten bestehen, die Handelsvertretern entsprechen, welche die Zeile erhalten, nachdem die Löschung erfolgte.

Hinweise

- Daten in der Customers-Tabelle werden nicht einem bestimmten Subskribenten zugeordnet (zum Beispiel durch einen Primärschlüssel-Wert) und werden von mehr als einem Subskribenten gemeinsam genutzt. Dadurch wird es möglich, Daten an mehr als einem entfernten Standort zwischen den einzelnen Replikatnachrichten zu aktualisieren, was zu Replikationskonflikten führen kann. Sie können dieses Problem entweder durch Privilegien lösen (indem Sie z.B. nur bestimmten Benutzern das Aktualisieren der Customers-Tabelle erlauben) oder indem Sie der Datenbank RESOLVE UPDATE-Trigger hinzufügen, die diese Konflikte programmiertechnisch lösen.

- Aktualisierungen in der Policy-Tabelle wurden hier nicht beschrieben. Sie sollten entweder vermieden werden oder Sie müssen einen BEFORE UPDATE-Trigger erstellen, der die Funktionen der BEFORE INSERT- und BEFORE DELETE-Trigger kombiniert, wie im Beispiel gezeigt.

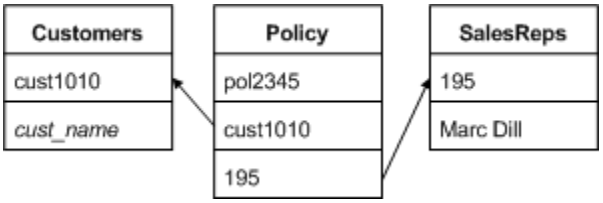
Die subscribe_by_remote-Option in Viele-zu-Viele-Beziehungen

Wenn die Option subscribe_by_remote auf On gesetzt wurde, wird bei Vorgängen von entfernten Datenbanken mit einem SUBSCRIBE BY-Wert von NULL oder einer leeren Zeichenfolge davon ausgegangen, dass der entfernte Benutzer die Zeile subskribiert hat. Standardmäßig ist die Option "subscribe_by_remote" auf On gesetzt.

Die Option "subscribe_by_remote" löst ein Problem, das sonst in manchen Publikationen auftreten würde. Die folgende Publikation verwendet eine Unterabfrage für den Subskriptionsausdruck der Customers-Tabelle, da Kunden zu mehreren Handelsvertretern gehören können:

```
CREATE PUBLICATION SalesRepData (
   TABLE SalesReps,
   TABLE Policy SUBSCRIBE BY rep_key,
   TABLE Customers SUBSCRIBE BY (
      SELECT rep_key FROM Policy
      WHERE Policy.cust_key =
         Customers.cust_key
   ),
);
```

Beispiel: Marc Dill ist ein Handelsvertreter, der gerade eine Police für einen neuen Kunden erstellt hat. Er fügt eine neue Zeile in die Customers-Tabelle und zusätzlich eine Zeile in die Policy-Tabelle ein, um sich selbst den neuen Kunden zuzuordnen.



In der konsolidierten Datenbank führt SQL Remote die Einfügung der Customers-Zeile durch und SQL Anywhere zeichnet den Subskriptionswert im Transaktionslog zum Zeitpunkt der Einfügung auf.

Später, wenn SQL Remote das Transaktionslog durchsucht, wird eine Liste der Subskribenten anhand des Subskriptionsausdrucks zusammengestellt, und Marc Dill befindet sich nicht auf der Liste, da die Zeile in der Policy-Tabelle, die ihm den Kunden zuordnet, noch nicht übernommen wurde. Wenn subscribe_by_remote auf Off gesetzt ist, tritt der Fall ein, dass der neue Kunde an Marc Dill als eine DELETE-Anweisung zurückgesendet wird.

Solange subscribe_by_remote auf On gesetzt ist, nimmt SQL Remote an, dass die Zeile dem Handelsvertreter gehört, der sie eingefügt hat. Die INSERT-Anweisung wird nicht an Marc Dill zurück repliziert, und das Replikationssystem bleibt intakt.

Wenn "subscribe_by_remote" auf Off gesetzt ist, müssen Sie sicherstellen, dass die "Policy"-Zeile vor der "Customers"-Zeile eingefügt wird, wobei eine Verletzung der referenziellen Integrität dadurch vermieden werden kann, dass die Überprüfung bis ans Ende der Transaktion aufgeschoben wird.

Siehe auch

- „subscribe_by_remote-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]

Eindeutige Identifizierungsnummern für entfernte Datenbanken

Sie müssen jeder entfernten Datenbank eine unterschiedliche Identifizierungsnummer zuordnen. Es gibt mehrere Möglichkeiten, die Kennnummern zu erstellen und zu verteilen. Eine Methode besteht darin, die Werte in eine Tabelle zu platzieren und die entsprechende Zeile beim Download in die einzelnen Datenbanken anhand einer anderen eindeutigen Eigenschaft zuzuordnen, wie zum Beispiel eines Benutzernamens.

Die Option global_database_id verwenden

Die öffentliche Option global_database_id muss in jeder Datenbank als eindeutige, nicht-negative Ganzzahl eingestellt sein. Der Bereich der Standardwerte für eine bestimmte Datenbank liegt zwischen $pn + 1$ bis $p(n + 1)$, wobei p die Partitionsgröße ist und n der Wert der öffentlichen Option global_database_id. Wenn zum Beispiel die Partitionsgröße 1000 und global_database_id auf 3 gesetzt sind, erstreckt sich der Bereich von 3001 bis 4000.

Wenn global_database_id auf eine nicht-negative Ganzzahl eingestellt ist, wählt SQL Anywhere Standardwerte aus, indem die folgenden Regeln angewendet werden:

- Wenn die Spalte in der aktuellen Partition keine Werte enthält, ist der erste Standardwert *pn* + 1.
- Wenn die Spalte in der aktuellen Partition Werte enthält, die alle kleiner sind als *p*(*n* + 1), ist der nächste Standardwert um eins größer als der vorherige 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 die öffentliche Option global_database_id auf den Standardwert 2147483647 gesetzt ist, wird NULL in die Spalte eingefügt. Falls NULL nicht zulässig ist, bewirkt der Versuch einer Einfügung in die Zeile einen Fehler. Dies geschieht etwa, wenn die Spalte im Primärschlüssel der Tabelle enthalten ist.

Da die öffentliche Option global_database_id nicht auf negative Werte eingestellt werden kann, sind die ausgewählten Werte immer positiv. Die maximale Identifizierungsnummer wird nur durch den Spaltendatentyp und die Partitionsgröße beschränkt.

Null-Standardwerte werden auch generiert, wenn der Vorrat von Werten innerhalb der Partition aufgebraucht ist. In diesem Beispiel sollte der Datenbank ein neuer global_database_id-Wert zugewiesen werden, damit Standardwerte aus einer anderen Partition gewählt werden können. Der Versuch, NULL einzufügen, bewirkt einen Fehler, wenn die Spalte NULL nicht zulässt. Um festzustellen, ob der Vorrat von ungenutzten Werten zu Ende geht, und um diese Situation zu beheben, erstellen Sie ein Ereignis vom Typ GlobalAutoincrement.

Sollten die Werte in einer bestimmten Partition zu Ende gehen, können Sie dieser Datenbank eine neue Datenbank-ID zuordnen. Sie können neue Datenbank-Identifizierungsnummern auf verschiedene Arten zuordnen. Eine Möglichkeit ist, einen Pool von nicht benutzten Datenbank-ID-Werten einzurichten. Dieser Pool wird auf dieselbe Weise wie ein Pool von Primärschlüsseln verwaltet.

Sie können einen Event-Handler einstellen, um den Datenbankadministrator automatisch zu benachrichtigen (oder um eine andere Aktion auszuführen), wenn die Partition beinahe aufgebraucht ist.

Siehe auch

- „global_database_id-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Primärschlüsselpools“ auf Seite 57
- „DEFAULT GLOBAL AUTOINCREMENT-Deklaration“ auf Seite 56
- „Triggerbedingungen für Ereignisse“ [*SQL Anywhere Server - Datenbankadministration*]

Den Wert global_database_id festlegen

Legen Sie eine Identifizierungsnummer für entfernte Datenbanken fest.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Die Identifizierungsnummer muss eine nicht-negative Ganzzahl sein.

Aufgabe

- Legen Sie den Wert der Option global_database_id fest.

  Beispiel: Die folgende Anweisung setzt die Datenbank-Identifizierungsnummer auf 20.

  ```
  SET OPTION PUBLIC.global_database_id = 20;
  ```

  Wenn die Größe für eine bestimmte Spalte 5000 beträgt, werden die Standardwerte für diese Datenbank aus dem Bereich 100001-105000 ausgewählt.

Ergebnisse

Die global_database_id-Nummer ist eingestellt.

Siehe auch

- „global_database_id-Option“ [*SQL Anywhere Server - Datenbankadministration*]

Datenbank-Identifizierungsnummern beim Extrahieren von Datenbanken festlegen

Wenn Sie das Extraktionsdienstprogramm (dbxtract) oder den **Assistenten zum Extrahieren einer Datenbank** verwenden, um Ihre entfernten Datenbanken zu erstellen, können Sie eine gespeicherte Prozedur schreiben, um das Festlegen von eindeutigen Datenbank-Identifizierungsnummern zu automatisieren.

Voraussetzungen

Hook-Prozeduren können von jedem Benutzer mit dem MANAGE REPLICATION-Systemprivileg erstellt werden. Um jedoch zu gewährleisten, dass ein Hook auf die #hook_dict-Tabelle zugreifen kann, die verwendet wird, um Daten an und aus Hooks zu übergeben, müssen Hooks eine der folgenden Anforderungen erfüllen:

- Eigentümer muss ein Benutzer sein, der die Systemprivilegien SELECT ANY TABLE und UPDATE ANY TABLE hat.
- Sie müssen durch die SQL SECURITY INVOKER-Klausel der CREATE PROCEDURE-Anweisung definiert sein.

Aufgabe

1. Erstellen Sie eine gespeicherte Prozedur namens **sp_hook_dbxtract_begin**.

   Beispiel: Um eine Datenbank für den entfernten Benutzer user2 mit einer user_id von 1001 zu extrahieren, führen Sie die folgenden Anweisungen aus:


```
SET OPTION "PUBLIC"."global_database_id" = '1';
CREATE TABLE extract_id (next_id INTEGER NOT NULL) ;
INSERT INTO extract_id VALUES( 1 );
CREATE PROCEDURE sp_hook_dbxtract_begin
AS
    DECLARE @next_id  INTEGER
    UPDATE extract_id SET next_id = next_id + 1000
    SELECT @next_id = (next_id )
    FROM extract_id
    COMMIT
    UPDATE #hook_dict
    SET VALUE = @next_id
    WHERE NAME = 'extracted_db_global_id';
```

Jede extrahierte oder wieder extrahierte Datenbank erhält einen unterschiedlichen global_database_id-Wert. Der Erste beginnt mit 1001, dann kommt 2001, und so weiter.

2. Führen Sie das Extraktionsdienstprogramm (dbxtract) mit der Option -v oder den **Assistenten zum Extrahieren einer Datenbank** aus, um Ihre entfernten Datenbanken zu extrahieren. Das Extraktionsdienstprogramm führt die folgenden Aufgaben aus:
   a. Es erstellt eine temporäre Tabelle namens #hook_dict mit dem folgenden Inhalt:

name	Wert
extracted_db_global_id	Zu extrahierende Benutzer-ID

   Wenn Sie eine sp_hook_dbxtract_begin-Prozedur schreiben, um den Spaltenwert der Zeile zu ändern, wird dieser Wert als die global_database_id-Option der extrahierten Datenbank verwendet und markiert den Anfang des Bereichs von Primärschlüsselwerten für GLOBAL DEFAULT AUTOINCREMENT-Werte.

   - Wenn Sie keine sp_hook_dbxtract_begin-Prozedur festlegen, hat die extrahierte Datenbank eine global_database_id von 101.
   - Wenn Sie eine sp_hook_dbxtract_begin-Prozedur festlegen, die keine Zeilen in der #hook_dict-Tabelle ändert, wird die global_database_id weiterhin auf 101 gesetzt.

   b. Ruft **sp_hook_dbxtract_begin** auf.
   c. Gibt die folgenden Informationen aus, um die Fehlersuche von Prozedureinstiegen zu unterstützen:
      - Die gefundenen Prozedureinstiege.
      - Der Inhalt von #hook_dict vor dem Aufruf des Prozedureinstiegs.
      - Der Inhalt von #hook_dict nach dem Aufruf des Prozedureinstiegs.

Ergebnisse

Die eindeutigen Datenbank-Identifizierungsnummern sind festgelegt.

Siehe auch

- Die Tabelle #hook_dict auf Seite 230
- „SQL Remote-Systemprozeduren“ auf Seite 229
- „global_database_id-Option“ [*SQL Anywhere Server - Datenbankadministration*]
- „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214

SQL Remote-Systeme verwalten

Deployment und Administration eines SQL Remote-Systems erfolgen aus der konsolidierten Datenbank. Um SQL Remote-Systeme zu verwalten, müssen Sie über die SYS_REPLICATION_ADMIN_ROLE-Systemrolle verfügen.

Deployment und Administration eines SQL Remote-Systems

1. Richten Sie die konsolidierte Datenbank ein.
2. Siehe „SQL Remote-Systeme erstellen“ auf Seite 9.
3. Überprüfen und testen Sie Ihr SQL Remote-System.

   Ein gründlicher Test Ihres SQL Remote-Systems sollte vor dem Deployment durchgeführt werden, vor allem, wenn zahlreiche entfernte Datenbanken verwendet werden.
4. Erstellen Sie entfernte Datenbanken und nehmen Sie das Deployment des Designs vor.

   Als DBA der konsolidierten Datenbank nehmen Sie das Deployment von SQL Remote vor, indem Sie Folgendes durchführen:

   a. Eine SQL Anywhere-Datenbank für jeden entfernten Benutzer anhand einer eigenen Erstkopie der Daten erstellen und ihre Subskriptionen starten. Siehe „Extraktion von entfernten Datenbanken“ auf Seite 82.

   b. Auf dem Computer aller entfernten Benutzer den SQL Anywhere-Datenbankserver, die entfernte Datenbank, SQL Remote und die Clientanwendung installieren. Siehe „Deployment von Anwendungen mit eingebetteten Datenbanken“ [*SQL Anywhere Server - Programmierung*] und „Deployment von SQL Remote“ [*SQL Anywhere Server - Programmierung*].
5. Führen Sie den SQL Remote-Nachrichtenagenten (dbremote) zum Austausch von Nachrichten aus.

   Um Nachrichten auszutauschen, müssen Sie Folgendes durchführen:

   a. Festlegen, ob der SQL Remote-Nachrichtenagent (dbremote) im kontinuierlichen Modus oder im Batchmodus auf der konsolidierten und den entfernten Datenbanken ausgeführt werden soll. Siehe „Modi des SQL Remote-Nachrichtenagenten (dbremote)“ auf Seite 91.

   b. Stellen Sie sicher, dass das System richtig konfiguriert ist und mit den korrekten Benutzernamen, den korrekten Verbindungszeichenfolgen für den SQL Remote-Nachrichtenagenten (dbremote), den korrekten Privilegien und so weiter versehen wurde. Siehe „SQL Remote-Nachrichtenagent (dbremote)“ auf Seite 90.
6. Nachrichten verwalten.

   Verwenden Sie das System der garantierten Nachrichtenzustellung, um die Nachrichten zu verwalten, die zwischen den vielen Datenbanken hin und her gesendet werden. Siehe „System der garantierten Nachrichtenzustellung“ auf Seite 108.

7. Performance steigern.

   Siehe „SQL Remote-Performance“ auf Seite 97.

8. Eine Sicherungs- und Wiederherstellungsstrategie implementieren .

   Sie müssen eine Sicherungs- und Wiederherstellungsstrategie für die konsolidierte Datenbank erstellen und implementieren. Siehe „SQL Remote-Systemsicherungen“ auf Seite 131.

9. Fehler behandeln.

   Siehe „Berichterstellung und Behandlung von Replikationsfehlern“ auf Seite 141.

10. Ein Upgrade der Software und der Datenbankschemata, falls erforderlich, durchführen.

    Siehe „Upgrades und Resynchronisation“ auf Seite 147.

Siehe auch

„GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Extraktion von entfernten Datenbanken

Um eine Datenbank für einen entfernten Benutzer zu erstellen, **extrahieren** Sie die entfernte Datenbank aus der konsolidierten Datenbank.

Sie können entweder den **Assistenten zum Extrahieren einer Datenbank** oder das Extraktionsdienstprogramm (dbxtract) verwenden, um eine entfernte Datenbank aus einer konsolidierten Datenbank für einen angegebenen entfernten Benutzer zu extrahieren. Beide Methoden ermöglichen es Ihnen, folgende Aufgaben durchzuführen:

- **Das Schema und die Daten direkt in eine neue oder vorhandene Datenbank automatisch extrahieren und neu laden.** Dies ist die empfohlene Methode, wenn Sie sich mit SQL Remote vertraut machen. Wenn Sie diese Methode verwenden, wird keine Zwischenkopie der Daten auf der Festplatte angelegt. Diese Methode bietet größere Sicherheit für Ihre Daten. Sie erfordert allerdings einen größeren Zeitaufwand für die Implementierung.

- **Das Schema und die Daten in Dateien extrahieren und sie anschließend in eine neue oder vorhandene Datenbank laden.** Beim Deployment von SQL Remote ist dies die bevorzugte Methode. Sie können die Schemadatei bearbeiten, um die Extraktion und die Erstellung Ihrer entfernten Datenbanken anzupassen.

  Eine Methode, um die Effizienz zu erhöhen, ist die Erstellung von mehr als einer entfernten Datenbank.

Siehe auch

- „Entfernte Datenbank automatisch extrahieren“ auf Seite 83
- „Extraktion von entfernten Datenbanken in eine Reload-Datei“ auf Seite 84
- „Mehrere entfernte Datenbanken erstellen“ auf Seite 88

Entfernte Datenbank automatisch extrahieren

Extrahieren Sie eine konsolidierte Datenbank und laden Sie das Schema und die Daten in eine neue Datenbank. Es wird keine Zwischenkopie der Daten auf der Festplatte abgelegt.

Voraussetzungen

Sie müssen die Privilegien EXECUTE ANY PROCEDURE und SELECT ANY TABLE haben. Das SELECT ANY TABLE-Privileg wird mit der SYS_REPLICATION_ADMIN_ROLE-Rolle bereitgestellt.

Kontext und Bemerkungen

Weitere Informationen zum Extrahieren von entfernten Datenbanken in eine Reload-Datei finden Sie unter „Extraktion von entfernten Datenbanken“ auf Seite 82.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur konsolidierten Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Klicken Sie auf **Extras** >> **SQL Anywhere 16** » **Datenbank extrahieren**.
3. Wählen Sie die konsolidierte Datenbank, mit der eine Verbindung hergestellt werden soll, und extrahieren Sie.
4. Wenn Sie dazu aufgefordert werden, klicken Sie auf **Extrahieren und in eine neue Datenbank laden**.

   Wenn Sie dazu aufgefordert werden, klicken Sie auf **Struktur und Daten extrahieren**.
5. Befolgen Sie die Anweisungen im Assistenten und akzeptieren Sie die Standardwerte.

Ergebnisse

Die neue entfernte Datenbank wird mit dem entsprechenden Schema und den entsprechenden entfernten Benutzern, Publikationen, Subskriptionen und Triggern erstellt. Standardmäßig werden die Daten aus der konsolidierten Datenbank in die entfernte Datenbank extrahiert und die Subskriptionen gestartet. Der Assistent startet allerdings nicht den SQL Remote-Nachrichtenagenten, daher werden keine Nachrichten ausgetauscht.

Siehe auch

- „SQL Remote-Nachrichtenagent (dbremote)“ auf Seite 90
- „Extraktion von entfernten Datenbanken in eine Reload-Datei“ auf Seite 84
- „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214

Extraktion von entfernten Datenbanken in eine Reload-Datei

Weitere Hinweise zum automatischen Extrahieren von entfernten SQL Anywhere-Datenbanken finden Sie unter „Extraktion von entfernten Datenbanken“ auf Seite 82.

In den meisten Deploymentszenarien müssen Sie die Extraktion und die Erstellung von entfernten Datenbanken anpassen. Sie können eine benutzerdefinierte Extraktion erstellen, indem Sie die Datenbank in eine Skriptdatei und eine Reihe von Textdateien extrahieren. Anschließend können Sie diese Dateien falls erforderlich bearbeiten.

Wenn Sie die Datenbank in Dateien extrahieren, legen Sie fest, ob Sie Folgendes erstellen wollen:

- **Eine SQL-Skriptdatei namens *reload.sql*, die die für den Aufbau des Schemas der entfernten Datenbank erforderlichen Anweisungen enthält** Weitere Hinweise finden Sie bei der Befehlszeilenoption -n unter „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214.

  Führen Sie zum Beispiel folgenden Befehl aus:

  ```
  dbxtract -c "UID=DBA;PWD=sql;DBF=c:\cons\cons.db" -n "c:
  \remote1\reload.sql" UserName
  ```

- **Eine Reihe von Datendateien, die jeweils den Inhalt einer Datenbanktabelle enthalten** Eine Reihe von Datendateien, die jeweils den Inhalt einer Datenbanktabelle enthalten. Ein neues Verzeichnis namens *extract* wird erstellt, um die Datendateien aufzunehmen. Sie verwenden diese Dateien, um Daten in eine vorhandene entfernte Datenbank zu laden. Weitere Hinweise finden Sie bei der Befehlszeilenoption -d unter „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214.

  Führen Sie zum Beispiel folgenden Befehl aus:

  ```
  dbxtract -c "UID=DBA;PWD=sql;DBF=c:\cons\cons.db" -d "c:\remote1" UserName
  ```

- **Sowohl die *reload.sql*-Datei als auch die Datendateien** Ein neues Verzeichnis namens *extract* wird erstellt, um die Datendateien aufzunehmen. Die *reload.sql*-Datei enthält Anweisungen für das Laden der Datendateien. Führen Sie zum Beispiel folgenden Befehl aus:

  ```
  dbxtract -c "UID=DBA;PWD=sql;DBF=c:\cons\cons.db" "c:\remote1\reload.sql"
  UserName
  ```

Die Datei *reload.sql*

Die Datei *reload.sql* enthält die SQL-Anweisungen für den Aufbau des Datenbankschemas, dies beinhaltet die Befehle zum Erstellen von Folgendem:

- Publikationseigentümer sowie entfernte und konsolidierte Benutzer
- Publikationen und Subskriptionen
- Nachrichtentypen
- Tabellen
- Ansichten
- Trigger
- Prozeduren

Hinweis
Sie müssen möglicherweise die Datei *reload.sql* bearbeiten, wenn Sie entfernte Datenbanken erstellen. Das Extraktionsdienstprogramm (dbxtract) unterstützt Sie bei der Vorbereitung entfernter Datenbanken, doch kann es nicht als Blackbox-Lösung für alle Situationen eingesetzt werden.

Siehe auch

- „Entfernte Datenbank automatisch extrahieren“ auf Seite 83
- „Bearbeiten der Datei reload.sql“ auf Seite 86
- „SQL Remote-Nachrichtenagent (dbremote)“ auf Seite 90

Entfernte Datenbank mittels der Datei *reload.sql* (Befehlszeile) erstellen

Erstellen Sie eine entfernte Datenbank, indem Sie eine vorhandene Datenbank als Modell verwenden. Außerdem werden ein Schema, entfernte Benutzer, Publikationen, Subskriptionen und Trigger generiert.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. Verwenden Sie das Extraktionsdienstprogramm (dbxtract), um das Datenbankschema und die Daten in Dateien zu extrahieren. Führen Sie zum Beispiel folgenden Befehl aus:

   ```
   dbxtract -c "UID=DBA;PWD=sql;DBF=c:\cons\cons.db" "c:\remote1\reload.sql"
   UserName
   ```

   Standardmäßig werden Subskriptionen für den angegebenen entfernten Benutzer automatisch gestartet.

2. Bearbeiten Sie die *reload.sql*-Datei, falls erforderlich.

3. Erstellen Sie eine leere SQL Anywhere-Datenbank.

   Führen Sie zum Beispiel folgenden Befehl aus:

```
dbinit -dba DBA,sql c:\remote1\rem1.db
```

4. Verbinden Sie sich aus Interactive SQL mit der Datenbank und führen Sie die Skriptdatei *reload.sql* aus.

   Führen Sie beispielsweise die folgende Anweisung aus:

   ```
   READ remote1\reload.sql
   ```

   Die neue entfernte Datenbank *rem1.db* wird mit dem entsprechenden Schema und den entsprechenden entfernten Benutzern, Publikationen, Subskriptionen und Triggern erstellt. Das Extraktionsdienstprogramm (dbxtract) startet allerdings nicht den SQL Remote-Nachrichtenagenten, daher werden keine Nachrichten ausgetauscht.

Ergebnisse

Die entfernte Datenbank wird erstellt.

Siehe auch

- „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214
- „Dienstprogramm Initialisierung (dbinit)“ [*SQL Anywhere Server - Datenbankadministration*]

Bearbeiten der Datei *reload.sql*

Bearbeiten Sie die Skriptdatei *reload.sql* nach Bedarf, wenn Sie entfernte Datenbanken erstellen. Sie müssen beispielsweise die *reload.sql*-Datei in den folgenden Fällen bearbeiten:

Nicht-replizierte Tabellen der entfernten Datenbank hinzufügen

Entfernte Datenbanken können Tabellen haben, die in ihrer konsolidierten Datenbank nicht vorhanden sind, solange diese Tabellen nicht an der Replikation teilnehmen. Das Extraktionsdienstprogramm (dbxtract) und der **Assistent zum Extrahieren einer Datenbank** können nicht-replizierte Tabellen aus einer konsolidierten Datenbank nicht extrahieren.

Nach dem Extrahieren der Datenbank sollten Sie die *reload.sql* bearbeiten, um solche Tabellen hinzuzufügen.

Prozeduren, Trigger und Ansichten extrahieren

Standardmäßig extrahiert das Extraktionsdienstprogramm (dbxtract) und der **Assistent zum Extrahieren einer Datenbank** alle gespeicherten Prozeduren, Trigger und Ansichten aus der Datenbank. Während einige dieser Ansichten und Prozeduren am entfernten Standort erforderlich sind, werden andere möglicherweise nicht benötigt. Beispiel: Eine Prozedur könnte Teile einer Datenbank referenzieren, die nicht am entfernten Standort vorhanden sind.

Nach dem Extrahieren der Datenbank sollten Sie *reload.sql* bearbeiten, um nicht benötigte Prozeduren, Trigger und Ansichten zu entfernen.

Das Extraktionsdienstprogramm (dbxtract) in vielstufigen Systemen verwenden

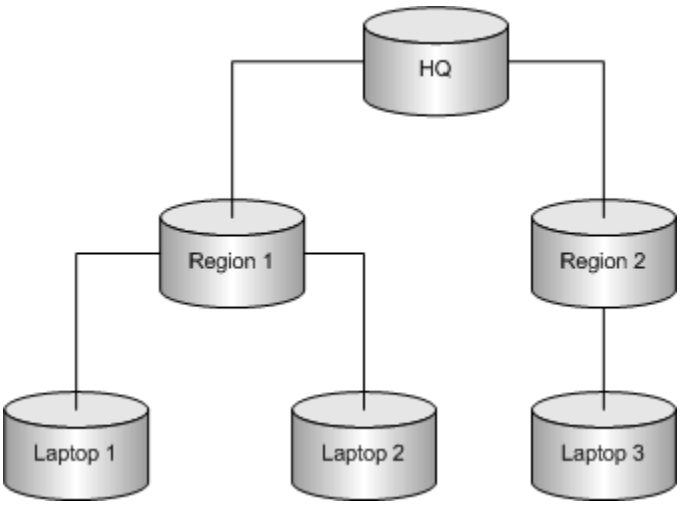
Siehe „Datenbankextraktion für ein vielstufiges Hierarchiesystem“ auf Seite 87.

Siehe auch

- „Mehrere entfernte Datenbanken erstellen“ auf Seite 88
- „Extraktion von entfernten Datenbanken in eine Reload-Datei“ auf Seite 84

Datenbankextraktion für ein vielstufiges Hierarchiesystem

Um die Rolle des Extraktionsdienstprogramms (dbxtract) und des **Assistenten zum Extrahieren einer Datenbank** in vielstufigen Hierarchien zu veranschaulichen, untersuchen wir ein dreistufiges SQL Remote-System. Dieses System wird im folgenden Diagramm dargestellt.



So erstellen Sie entfernte Datenbanken für ein Drei-Schichten-System:

1. Verwenden Sie das Extraktionsdienstprogramm (dbxtract) bei der obersten Schicht, der konsolidierten Datenbank HQ, um die Datenbanken der zweiten Schicht, Region 1 und Region 2 zu erstellen.

2. Verwenden Sie das Extraktionsdienstprogramm (dbxtract) bei den Datenbanken der zweiten Schicht, Region 1 und Region 2, um die Datenbanken der dritten Schicht für die Benutzer Laptop 1, Laptop 2 und Laptop 3 zu erstellen. Die Datenbanken der zweiten Schicht sind entfernte Datenbanken für die Datenbank der obersten Schicht, HQ, und konsolidierte Datenbanken für die Datenbanken der dritten Schicht, Laptop 1, Laptop 2 und Laptop 3.

Datenbanken in einem vielstufigen Hierarchiesystem erneut extrahieren

Wenn Sie das *Schema* für die Datenbank der zweiten Schicht aus der konsolidierten Datenbank der obersten Schicht erneut extrahieren müssen, löscht das Extraktionsdienstprogramm (dbxtract) die entfernten Benutzer (Laptop 1, Laptop 2 und Laptop 3) zusammen mit ihren Subskriptionen und Privilegien. Dies führt dazu, dass Sie diese Benutzer der dritten Schicht und ihre Subskriptionen manuell wieder erstellen müssen.

Wenn Sie nur die *Daten* für die Datenbanken der zweiten Schicht aus der konsolidierten Datenbank der obersten Schicht erneut extrahieren müssen, wirkt sich das Extraktionsdienstprogramm (dbxtract) nicht

auf die entfernten Benutzer aus. Weitere Hinweise finden Sie bei der Befehlszeilenoption -d unter „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214.

Voll qualifizierte Publikationsdefinitionen

Voll qualifizierte Publikationsdefinitionen enthalten WHERE- und SUBSCRIBE BY-Klauseln. Normalerweise müssen Sie voll qualifizierte Publikationsdefinitionen für eine entfernte Datenbank nicht extrahieren, da die entfernte Datenbank üblicherweise alle Zeilen in die konsolidierte Datenbank zurück repliziert.

Siehe auch

- „Mehrere entfernte Datenbanken erstellen“ auf Seite 88
- „Extraktion von entfernten Datenbanken in eine Reload-Datei“ auf Seite 84

Mehrere entfernte Datenbanken erstellen

Verwenden Sie diese Prozedur, um die Erstellung von mehreren Datenbanken gleichzeitig zu rationalisieren.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Erstellen Sie eine Kopie der konsolidierten Datenbank und starten Sie die Subskriptionen für die entfernten Benutzer von der konsolidierten Datenbank aus. Zum Beispiel:
   a. Fahren Sie die konsolidierte Datenbank und den SQL Remote-Nachrichtenagenten (falls er ausgeführt wird) herunter.
   b. Starten Sie eine lokale Kopie der konsolidierten Datenbank mit dbeng16 und einem anderen Servernamen, um zu gewährleisten, dass keine anderen Prozesse eine Verbindung mit der lokalen Kopie herstellen.
   c. Starten Sie die Subskriptionen.

      Die Subskriptionen müssen zum selben Zeitpunkt gestartet werden, zu dem die Kopie der konsolidierten Datenbank angefertigt wird. Vorgänge, die zwischen dem Kopieren der Datenbank und dem Start der Subskriptionen stattfinden, können verlorengehen und zu Fehlern in entfernten Datenbanken führen. Das Starten der Subskriptionen in der konsolidierten Datenbank ermöglicht es, dass Nachrichten verpackt und an Subskribenten gesendet werden, auch wenn die Subskribenten-Datenbanken noch nicht vorhanden sind.

      Um mehrere Subskriptionen in einer einzigen Transaktion zu starten, verwenden Sie die REMOTE RESET-Anweisung.
   d. Fahren Sie die konsolidierte Datenbank sofort herunter.
   e. Kopieren Sie die konsolidierte Datenbank.

      Standardmäßig werden das Extraktionsdienstprogramm (dbxtract) und der **Assistent zum Extrahieren einer Datenbank** auf der Isolationsstufe 3 ausgeführt. Diese Isolationsstufe

gewährleistet, dass die Daten in der extrahierten Datenbank mit den Daten auf dem Datenbankserver konsistent sind, sie kann allerdings verhindern, dass andere Benutzer die Datenbank verwenden. Es wird empfohlen, dass Sie Ihre entfernte Datenbank unter Verwendung einer Kopie der konsolidierten Datenbank extrahieren.

f. Starten Sie die konsolidierte Datenbank erneut und starten Sie den SQL Remote-Nachrichtenagenten auf der konsolidierten Datenbank erneut, falls er ausgeführt wurde.

2. Extrahieren Sie das Schema der entfernten Datenbank von der Kopie der konsolidierten Datenbank. Da die Datenbank eine Kopie ist, gibt es keine Sperren- oder Parallelitätsprobleme, bei einer großen Anzahl von entfernten Datenbanken kann dieser Prozess jedoch länger dauern.

   Beim Extrahieren des Schemas der entfernten Datenbank wählen Sie die folgenden Optionen aus:

   a. Extrahieren Sie nur das Schema für die entfernte Datenbank.

      Standardmäßig extrahieren das Extraktionsdienstprogramm (dbxtract) und der **Assistent zum Extrahieren einer Datenbank** jeweils eine Datenbank samt Schema und Daten für jeden Benutzer. In den meisten Deploymentszenarien allerdings verwenden die entfernten Datenbanken dasselbe Schema, aber unterschiedliche Daten. Die Verwendung des Extraktionsdienstprogramms (dbxtract) oder des **Assistenten zum Extrahieren einer Datenbank**, um sowohl das Schema als auch die Daten für jeden Benutzer zu extrahieren, führt dazu, dass wiederholt dasselbe Schema extrahiert wird. Weitere Hinweise finden Sie bei der Befehlszeilenoption -n unter „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214.

   b. Die Daten nach Primärschlüssel sortieren.

      Standardmäßig sind die Daten in jeder Tabelle nach dem Primärschlüssel aufgelistet. Das Laden von Daten in die entfernte Datenbank geht schneller, wenn die Daten nach Primärschlüssel sortiert sind. Weitere Hinweise finden Sie bei der Befehlszeilenoption -u unter „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214.

3. Erstellen Sie eine leere entfernte Datenbank unter Verwendung der *reload.sql*-Datei. Kopieren Sie diese Datenbankdatei, um die erforderliche Anzahl von entfernten Datenbanken zu erstellen.

4. Bei jeder entfernten Datenbank legen Sie die SQL Remote-Definitionen fest, die für den jeweiligen entfernten Benutzer spezifisch sind.

5. Für jeden entfernten Benutzer extrahieren Sie nur die entsprechenden Daten aus der konsolidierten Datenbank. Weitere Hinweise finden Sie bei der Befehlszeilenoption -d unter „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214.

6. Laden Sie die Daten für jeden entfernten Benutzer in die entsprechende entfernte Datenbank.

   Während die jeweilige entfernte Datenbank erstellt wird, stimmen ihre Daten mit der echten konsolidierten Datenbank nicht mehr überein.

   Wenn Sie den SQL Remote-Nachrichtenagenten (dbremote) ausführen, kann jeder Benutzer jedoch Nachrichten empfangen und anwenden, die von der echten konsolidierten Datenbank gesendet wurden, um sich auf den letzten Stand zu bringen.

Ergebnisse

Die entfernten Datenbanken werden erstellt.

Siehe auch

- „Datenbankextraktion für ein vielstufiges Hierarchiesystem“ auf Seite 87
- „Extraktion von entfernten Datenbanken in eine Reload-Datei“ auf Seite 84
- „Bearbeiten der Datei reload.sql“ auf Seite 86
- „Starten von Subskriptionen“ auf Seite 155
- „SQL Remote-Nachrichtenagent (dbremote)“ auf Seite 90
- „START SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REMOTE RESET-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Benutzerprivilegien“ auf Seite 18
- „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

SQL Remote-Nachrichtenagent (dbremote)

Der SQL Remote-Nachrichtenagent (dbremote) ist eine Schlüsselkomponente in der SQL Remote-Replikation. Er muss in jeder Datenbank im System installiert und ausgeführt werden. Der SQL Remote-Nachrichtenagent (dbremote) verwaltet das Senden und das Empfangen von Nachrichten.

Um dbremote auszuführen, müssen Sie über die SYS_RUN_REPLICATION_ROLE-Systemrolle verfügen.

Er führt die folgenden Funktionen durch:

- **Aufgaben des SQL Remote-Nachrichtenagenten (dbremote) beim Versenden von Nachrichten**
  - Er durchsucht das Transaktionslog in jeder Publikationseigentümer-Datenbank und konvertiert die Transaktionslog-Einträge in Nachrichten für Subskribenten.
  - Er sendet die Nachrichten an Subskribenten.
  - Wenn der SQL Remote-Nachrichtenagent (dbremote) eine Anforderung zum erneuten Versenden von Nachrichten empfängt, sendet er die Nachrichten erneut an die Datenbank, von der die Anforderung gemacht stammt.
  - Er hält die Nachrichteninformationen in den Systemtabellen aufrecht und verwaltet das System der garantierten Nachrichtenzustellung.
- **Aufgaben des SQL Remote-Nachrichtenagenten (dbremote) beim Empfangen von Nachrichten**
  - Er verarbeitet die eintreffenden Nachrichten und übernimmt sie in der korrekten Reihenfolge in die Datenbank.

- Er fordert das erneute Senden von fehlenden Nachrichten an.
- Er hält die Nachrichteninformationen in den Systemtabellen aufrecht und verwaltet das System der garantierten Nachrichtenzustellung.

Verbindungen

Der SQL Remote-Nachrichtenagent (dbremote) verwendet mehrere Verbindungen zum Datenbankserver. Dies sind:

- **Eine globale Verbindung** Diese Verbindung ist die ganze Zeit aktiv, in der der SQL Remote-Nachrichtenagent (dbremote) ausgeführt wird.
- **Eine Verbindung für den Transaktionslog-Scan** Diese Verbindung ist nur während der Scan-Phase aktiv.
- **Eine Verbindung zum Ausführen von Befehlen vom Transaktionslog-Scan-Thread** Diese Verbindung ist nur während der Scan-Phase aktiv.
- **Eine Verbindung für die Verarbeitung von Anforderungen zur Subskriptionssynchronisierung** Diese Verbindung ist nur während der Sende-Phase aktiv.
- **Eine Verbindung für jeden Worker-Thread** Diese Verbindungen sind nur während der Empfangsphase aktiv.

Siehe auch

- „Aufgaben zum Senden von Nachrichten“ auf Seite 104
- „Aufgaben zum Empfangen von Nachrichten“ auf Seite 98

Modi des SQL Remote-Nachrichtenagenten (dbremote)

Der SQL Remote-Nachrichtenagent (dbremote) kann in zwei Modi ausgeführt werden:

- **Kontinuierlicher Modus** Im kontinuierlichen Modus versendet der SQL Remote-Nachrichtenagent (dbremote) regelmäßig Nachrichten zu einem Zeitpunkt, der durch die Sendefrequenz-Eigenschaften des jeweiligen entfernten Benutzers festgelegt wird. Wenn er keine Nachrichten versendet, empfängt er Nachrichten, sobald sie eintreffen.

  Der kontinuierliche Modus ist bei konsolidierten Datenbanken nützlich, in denen Nachrichten jederzeit eintreffen und gesendet werden können, weil damit die Systemlast gleichmäßiger verteilt und die zeitnahe Replikation sichergestellt wird.

- **Batchmodus** Im Batchmodus empfängt und verarbeitet der SQL Remote-Nachrichtenagent (dbremote) eintreffende Nachrichten, durchsucht einmal das Transaktionslog und erstellt und versendet die ausgehenden Nachrichten, um dann zu stoppen.

  Der Batchmodus ist bei gelegentlich verbundenen entfernte Datenbanken nützlich, bei denen Nachrichten mit der konsolidierten Datenbank nur dann ausgetauscht werden können, wenn die Verbindung besteht. Das trifft zum Beispiel zu, wenn sich die entfernte Datenbank über eine Wählleitung mit dem Hauptnetzwerk verbindet.

Anforderungen für den SQL Remote-Nachrichtenagenten (dbremote)

SQL Remote ist sehr flexibel. Innerhalb eines Systems können Sie den SQL Remote-Nachrichtenagenten (dbremote) in beiden Modi, auf mehreren Geräten und auf mehreren Betriebssystemen ausführen. SQL Remote hat allerdings die folgenden Anforderungen:

- **SYS_RUN_REPLICATION_ROLE-Systemrolle** Der SQL Remote-Nachrichtenagent (dbremote) muss von einem Benutzer mit der SYS_RUN_REPLICATION_ROLE-Systemrolle ausgeführt werden.

- **Die maximale Nachrichtenlänge muss bei allen SQL Remote-Nachrichtenagenten (dbremote) im System gleich sein** Diese Länge kann durch Betriebssystem-Speicherzuweisungsgrenzen eingeschränkt sein. Empfangene Nachrichten, die länger als der Maximalwert sind, werden als beschädigte Nachrichten gelöscht. Der Standardwert ist 50000 Byte. Diese Länge lässt sich mit der Befehlszeilenoption -l des SQL Remote-Nachrichtenagenten (dbremote) konfigurieren.

Siehe auch

- „SQL Remote-Nachrichtenagenten (dbremote) im kontinuierlichen Modus ausführen“ auf Seite 92
- „Ausführen des SQL Remote-Nachrichtenagenten (dbremote) im Batchmodus“ auf Seite 95
- „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203

SQL Remote-Nachrichtenagenten (dbremote) im kontinuierlichen Modus ausführen

Im kontinuierlichen Modus sendet der SQL Remote-Nachrichtenagent (dbremote) Nachrichten zu den Zeiten, die in der SEND AT- oder SEND EVERY-Frequenz in den Eigenschaften des jeweiligen entfernten Benutzers angegeben sind.

Voraussetzungen

Sie müssen über die SYS_RUN_REPLICATION_ROLE-Systemrolle verfügen.

Um den SQL Remote-Nachrichtenagenten (dbremote) im kontinuierlichen Modus ausführen zu können, z.B. in der konsolidierten Datenbank, müssen Sie sicherstellen, dass für jeden REMOTE-Benutzer eine Sendefrequenz festgelegt ist.

Die maximale Nachrichtenlänge, wie sie von der Befehlszeilenoption -l angegeben wird, muss dieselbe bei allen Datenbanken im System sein.

Kontext und Bemerkungen

Üblicherweise wird die konsolidierte Datenbank im kontinuierlichen Modus ausgeführt. Im kontinuierlichen Modus sendet der SQL Remote-Nachrichtenagent (dbremote) Nachrichten zu Zeiten, die mit der SEND AT- oder SEND EVERY-Eigenschaft angegeben sind.

Aufgabe

1. Stellen Sie sicher, dass jeder REMOTE-Benutzer entweder eine SEND AT- oder eine SEND EVERY-Frequenz angegeben hat.
2. Starten Sie den SQL Remote-Nachrichtenagenten *ohne* die Befehlszeilenoption -b.

   Unter Windows heißt der SQL Remote-Nachrichtenagent (dbremote) *dbremote.exe*. Unter Unix lautet der Name dbremote. Unter Mac OS X können Sie auch SyncConsole verwenden, um den SQL Remote-Nachrichtenagenten (dbremote) zu starten.

   Der folgende Befehl führt beispielsweise dbremote im kontinuierlichen Modus für die Datenbankdatei *c:\mydata.db* aus und verbindet sich als Benutzer ManagerSteve mit dem Kennwort sql:

   ```
   dbremote -c "UID=ManagerSteve;PWD=sql;DBF=c:\mydata.db" -l 40000
   ```

Ergebnisse

Der SQL Remote-Nachrichtenagent (dbremote) ist jetzt so eingestellt, dass er im kontinuierlichen Modus ausgeführt wird.

Siehe auch

- „SQL Remote-Nachrichtenagent (dbremote)“ auf Seite 90
- „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203
- Anforderungen für den SQL Remote-Nachrichtenagenten (dbremote) auf Seite 92
- „Den SQL Remote-Nachrichtenagenten (dbremote) unter Mac OS X ausführen“ auf Seite 96
- „Den SQL Remote-Nachrichtenagenten (dbremote) unter Unix ausführen“ auf Seite 97
- „Sendefrequenz“ auf Seite 93

Den SQL Remote-Nachrichtenagenten (dbremote) als Dienst im kontinuierlichen Modus ausführen

Wenn Sie den SQL Remote-Nachrichtenagenten (dbremote) im kontinuierlichen Modus ausführen, können Sie festlegen, dass der SQL Remote-Nachrichtenagent (dbremote) ausgeführt wird, wann immer der Datenbankserver läuft. Dies können Sie gewährleisten, indem Sie den SQL Remote-Nachrichtenagenten (dbremote) als Windows-Dienst ausführen. Ein Dienst kann so konfiguriert werden, dass er auch weiter läuft, wenn sich der aktuelle Benutzer abmeldet, und startet, wenn das Betriebssystem gestartet wird.

Siehe auch

- „Dienstprogramm für Dienste (dbsvc) für Windows“ [*SQL Anywhere Server - Datenbankadministration*]

Sendefrequenz

Um den SQL Remote-Nachrichtenagenten (dbremote) im kontinuierlichen Modus auszuführen, beispielsweise in der konsolidierten Datenbank, müssen Sie sicherstellen, dass jeder REMOTE-Benutzer eine Sendefrequenz angegeben hat. Im kontinuierlichen Modus sendet der SQL Remote-Nachrichtenagent (dbremote) Nachrichten zu Zeiten, die mit der SEND AT- oder SEND EVERY-Eigenschaft angegeben sind.

Der SQL Remote-Nachrichtenagent (dbremote) unterstützt folgende Sendefrequenzwerte:

- **SEND EVERY** Gibt das Intervall zwischen dem Versenden von Nachrichten an.

  Wenn an einen Benutzer mit eingestelltem SEND EVERY-Parameter Nachrichten versendet werden, erfolgt der Versand gleichzeitig an alle anderen Benutzer mit derselben Frequenz. So werden zum Beispiel an alle entfernten Benutzer, die Aktualisierungen alle zwölf Stunden erhalten, die Aktualisierungen zum gleichen Zeitpunkt versendet, und nicht gestaffelt. Das vermindert die Häufigkeit, mit der das SQL Anywhere-Transaktionslog verarbeitet werden muss. Sie sollten daher die Frequenzen möglichst einheitlich ansetzen.

  Eine Sendefrequenz kann im Format ***HH*:*MM*:*SS*** in Stunden, Minuten und Sekunden angegeben werden.

- **SEND AT** Gibt die Uhrzeit an, zu der Nachrichten versendet werden.

  Aktualisierungen werden täglich zum angegebenen Zeitpunkt gesendet. Sie sollten so wenige unterschiedliche Zeitpunkte wie möglich verwenden, anstatt die Sendezeiten zu staffeln. Sie sollten Zeitpunkte auswählen, zu denen die Datenbank nicht ausgelastet ist.

- **Standardeinstellung (keine SEND-Klausel)** Wenn ein Benutzer keine SEND AT- oder SEND EVERY-Klausel angegeben hat, läuft der SQL Remote-Nachrichtenagent (dbremote) im Batchmodus, indem er Nachrichten bei jeder Ausführung versendet und anschließend stoppt.

Nachrichten zu häufig versenden

Wenn Sie häufig Nachrichten senden, ist anzunehmen, dass kleine Nachrichten gesendet werden. Wenn Nachrichten weniger häufig gesendet werden, können mehr Anweisungen in einer einzelnen Nachricht zusammengefasst werden. Wenn eine große Anzahl von kleinen Nachrichten Ihr Nachrichtensystem zu stark belasten würde, sollten Sie kurze Sendefrequenzen vermeiden.

Siehe auch

- „GRANT REMOTE-Anweisung [SQLRemote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Ausführen des SQL Remote-Nachrichtenagenten (dbremote) im Batchmodus“ auf Seite 95

Sendefrequenz für Nachrichten einstellen

Um den SQL Remote-Nachrichtenagenten (dbremote) im kontinuierlichen Modus auszuführen, beispielsweise in der konsolidierten Datenbank, müssen Sie sicherstellen, dass jeder REMOTE-Benutzer eine Sendefrequenz angegeben hat.

Voraussetzungen

Sie müssen über die SYS_RUN_REPLICATION_ROLE-Systemrolle verfügen.

Kontext und Bemerkungen

Im kontinuierlichen Modus sendet der SQL Remote-Nachrichtenagent (dbremote) Nachrichten zu Zeiten, die mit der SEND AT- oder SEND EVERY-Eigenschaft angegeben sind.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf **SQL Remote-Benutzer**.
3. Rechtsklicken Sie auf einen Benutzer und klicken Sie auf **Eigenschaften**.
4. Klicken Sie auf die Registerkarte **SQL Remote**.
5. Klicken Sie entweder auf **Sendeintervall** oder auf **Täglich senden um** und geben Sie eine Zeit an. Klicken Sie auf **OK**.

Ergebnisse

Die Frequenz wird eingestellt.

Siehe auch

- „SQL Remote-Nachrichtenagenten (dbremote) im kontinuierlichen Modus ausführen“ auf Seite 92
- „GRANT REMOTE-Anweisung [SQLRemote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Ausführen des SQL Remote-Nachrichtenagenten (dbremote) im Batchmodus

Sie können den SQL Remote-Nachrichtenagenten (dbremote) so einrichten, dass er eintreffende Nachrichten empfängt und verarbeitet, das Transaktionslog durchsucht, ausgehende Nachrichten erstellt und sendet und dann stoppt.

Voraussetzungen

Sie müssen über die SYS_RUN_REPLICATION_ROLE-Systemrolle verfügen. Die maximale Nachrichtenlänge, wie sie von der Befehlszeilenoption -l angegeben wird, muss dieselbe bei allen Datenbanken im System sein.

Aufgabe

1. Stellen Sie sicher, dass zumindest ein entfernter Benutzer weder die SEND AT- noch SEND EVERY-Option in seinen entfernten Eigenschaften eingestellt hat.

   Wenn *alle* Ihre entfernten Benutzer eine SEND AT- oder eine SEND EVERY-Klausel festgelegt haben und Sie Nachrichten senden bzw. empfangen und anschließend herunterfahren wollen, müssen Sie den SQL Remote-Nachrichtenagenten (dbremote) mit der Befehlszeilenoption -b starten.

2. Starten Sie den SQL Remote-Nachrichtenagenten (dbremote).

   Unter Windows heißt der SQL Remote-Nachrichtenagent (dbremote) *dbremote.exe*. Unter Unix lautet der Name dbremote. Unter Mac OS X können Sie auch **SyncConsole** verwenden, um den SQL Remote-Nachrichtenagenten (dbremote) zu starten.

Die folgende Anweisung führt beispielsweise dbremote im Batchmodus mit der Datenbankdatei *c:\mydata.db* aus und verbindet sich mit dem Benutzernamen ManagerSteve und dem Kennwort "sql":

```
dbremote -c "UID=ManagerSteve;PWD=sql;DBF=c:\mydata.db"
```

Der SQL Remote-Nachrichtenagent (dbremote) empfängt und verarbeitet eintreffende Nachrichten, durchsucht einmal das Transaktionslog und erstellt und versendet die ausgehenden Nachrichten, um dann zu stoppen.

Ergebnisse

Der SQL Remote-Nachrichtenagent (dbremote) wird gestartet.

Siehe auch

- „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203
- Anforderungen für den SQL Remote-Nachrichtenagenten (dbremote) auf Seite 92
- „SQL Remote-Nachrichtenagent (dbremote)“ auf Seite 90
- „Den SQL Remote-Nachrichtenagenten (dbremote) unter Mac OS X ausführen“ auf Seite 96
- „Den SQL Remote-Nachrichtenagenten (dbremote) unter Unix ausführen“ auf Seite 97

Den SQL Remote-Nachrichtenagenten (dbremote) unter Mac OS X ausführen

SyncConsole verwenden, um den SQL Remote-Nachrichtenagenten (dbremote) unter Mac OS X zu starten

Voraussetzungen

Sie müssen über die SYS_RUN_REPLICATION_ROLE-Systemrolle verfügen. Die Nachrichtenlänge, wie sie von der Befehlszeilenoption -l angegeben wird, muss dieselbe bei allen Datenbanken im System sein.

Kontext und Bemerkungen

Sie können auch den SQL Remote-Nachrichtenagenten unter Mac OS X mit dem dbremote-Dienstprogramm starten.

Aufgabe

1. Wechseln Sie im Finder zu */Anwendungen/SQLAnywhere16*.
2. Doppelklicken Sie auf **SyncConsole**.
3. Klicken Sie auf **Datei** » **Neu** » **SQL Remote**.

   Ein Fenster mit Clientoptionen wird angezeigt.
4. Geben Sie die Verbindungsdaten für dbremote an.

Beispiel: Der folgende Verbindungsparameter verwendet eine ODBC-Datenquelle für die SQL Anywhere-Beispieldatenbank:

```
DSN="SQL Anywhere 16 Demo"
```

Ergebnisse

Der SQL Remote-Nachrichtenagent (dbremote) wird gestartet.

Siehe auch

- Anforderungen für den SQL Remote-Nachrichtenagenten (dbremote) auf Seite 92
- „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203

Den SQL Remote-Nachrichtenagenten (dbremote) unter Unix ausführen

Auf Unix-Plattformen können Sie den SQL Remote-Nachrichtenagenten (dbremote) als Daemon ausführen, indem Sie die Befehlszeilenoption -ud angeben. Weitere Hinweise finden Sie bei der Befehlszeilenoption -ud unter „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203.

Sie müssen über die SYS_RUN_REPLICATION_ROLE-Systemrolle verfügen. Die maximale Nachrichtenlänge, wie sie von der Befehlszeilenoption -l angegeben wird, muss dieselbe bei allen Datenbanken im System sein. Siehe Anforderungen für den SQL Remote-Nachrichtenagenten (dbremote) auf Seite 92.

Eine vollständige Liste der dbremote-Befehlszeilenoptionen, die Sie angeben können, finden Sie unter „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203.

SQL Remote-Performance

Bei jedem Einfügen, Löschen oder Aktualisieren einer Zeile in einer Tabelle wird eine Nachricht für die Benutzer erstellt, die die Zeile subskribiert haben. Zusätzlich kann eine Aktualisierung bewirken, dass der Subskriptionsausdruck geändert und die Anweisung an verschiedene Subskribenten als Löschen, Aktualisieren oder Einfügen gesendet wird.

Die Aufgabe, zu bestimmen, wer was bekommt, teilen sich der Datenbankserver und der SQL Remote-Nachrichtenagent (dbremote).

Der Datenbankserver

Der Datenbankserver verarbeitet Publikationen.

SQL Remote-Nachrichtenagent (dbremote)

Der **SQL Remote-Nachrichtenagent (dbremote)** verarbeitet Subskriptionen.

Der SQL Remote-Nachrichtenagent (dbremote) liest die auszuwertenden Subskriptionsausdrücke oder Subskriptionsspalten-Einträge aus dem Transaktionslog und vergleicht die Vorher- und Nachher-Werte

mit dem Subskriptionswert für jeden Subskribenten der Publikation. Auf diese Weise kann der SQL Remote-Nachrichtenagent (dbremote) die richtigen Vorgänge an die einzelnen Subskribenten versenden.

Auch wenn eine große Anzahl von Subskribenten keine Auswirkung auf die Datenbankserver-Performance hat, kann sie sich auf die Performance des SQL Remote-Nachrichtenagenten (dbremote) auswirken. Das Vergleichen von Subskriptionswerten mit einer großen Anzahl von anderen Subskriptionswerten und das Versenden der Nachrichten können zeitraubend sein.

Siehe auch

- Der Datenbankserver verarbeitet Publikationen auf Seite 35

Aufgaben zum Empfangen von Nachrichten

Der SQL Remote-Nachrichtenagent (dbremote) führt die folgenden Aufgaben durch, wenn er Nachrichten empfängt:

- **Eintreffende Nachrichten abrufen** Um zu überprüfen, ob neue Nachrichten in der Datenbank eingetroffen sind, ruft der SQL Remote-Nachrichtenagent (dbremote) neue Nachrichten ab.
- **Die Nachrichten lesen** Wenn Nachrichten eintreffen, werden Sie vom SQL Remote-Nachrichtenagenten (dbremote) gelesen und im Cachespeicher abgelegt, bis sie übernommen werden können.

  Wenn eine Nachricht fehlt und der SQL Remote-Nachrichtenagent (dbremote) im kontinuierlichen Modus ausgeführt wird, wartet der SQL Remote-Nachrichtenagent (dbremote) darauf, dass die Nachricht in einem nachfolgenden Abruf eintrifft. Die Anzahl der Abrufe, die der SQL Remote-Nachrichtenagent (dbremote) wartet, wird als **Wartestatus** bezeichnet und mit der Befehlszeilenoption -rp angegeben.

  - Wenn die fehlende Nachricht eintrifft, bevor der Wartestatus des SQL Remote-Nachrichtenagenten (dbremote) abläuft, wird die fehlende Nachricht in der korrekten Sortierfolge dem Cache hinzugefügt.
  - Wenn die fehlende Nachricht nicht eintrifft und der Wartestatus des SQL Remote-Nachrichtenagenten (dbremote) abläuft, sendet der SQL Remote-Nachrichtenagent (dbremote) eine Anforderung, die Nachricht erneut von der Publikationseigentümer-Datenbank zu senden.

  Nachrichten werden weiterhin gelesen und dem Cache hinzugefügt, bis die Cachespeichernutzung überschritten wird. Wenn die Cachespeichernutzung, die mit der Befehlszeilenoption -m angegeben wird, überschritten wird, werden Nachrichten gelöscht.
- **Die Nachrichten anwenden** Der SQL Remote-Nachrichtenagent (dbremote) wendet die Nachrichten in der korrekten Reihenfolge in der Subskribenten-Datenbank an.
- **Auf die Bestätigung warten, dass die Nachrichten in den Subskribenten-Datenbanken angewendet werden** Sobald die Nachricht in der subskribierten Datenbank empfangen und übernommen wurde, wird eine Bestätigung zurück an den Publikationseigentümer gesendet. Wenn der SQL Remote-Nachrichtenagent (dbremote) des Publikationseigentümers die Bestätigung empfängt, protokolliert er die Bestätigung in einer Systemtabelle.

Siehe auch

- „Anpassungen des Abrufintervalls zum Überprüfen auf neue Nachrichten“ auf Seite 99
- „Anpassen des Durchsatzes durch Caching von erhaltenen Nachrichten“ auf Seite 100
- „Anpassungen an Anforderungen zum erneuten Versenden von Nachrichten“ auf Seite 101
- „Datenbank-Worker-Threads“ auf Seite 103
- „System der garantierten Nachrichtenzustellung“ auf Seite 108

Performance beim Nachrichtenempfang

Der größte Engpass für den Durchsatz der Replikation in einem SQL Remote-System ist im Allgemeinen der Empfang von Nachrichten von vielen entfernten Datenbanken und das Einfügen dieser Nachrichten in die Datenbank. Um diese Verzögerung beim Ausführen des SQL Remote-Nachrichtenagenten (dbremote) im kontinuierlichen Modus zu verringern, können Sie die folgenden Variablen anpassen:

- Wie häufig der SQL Remote-Nachrichtenagent (dbremote) überprüft, ob Nachrichten eingetroffen sind.
- Wie viel Speicher vom SQL Remote-Nachrichtenagenten (dbremote) verwendet wird, um die zu versendenden Nachrichten aufzubewahren.
- Wie lange der SQL Remote-Nachrichtenagent (dbremote) auf eine Nachricht außerhalb der Reihenfolge wartet, bevor er ein erneutes Senden der Nachricht anfordert.
- Wie viele Worker-Threads verwendet werden, um die empfangenen Nachrichten zu verarbeiten.

Siehe auch

- „Anpassungen des Abrufintervalls zum Überprüfen auf neue Nachrichten“ auf Seite 99
- „Anpassen des Durchsatzes durch Caching von erhaltenen Nachrichten“ auf Seite 100
- „Anpassungen an Anforderungen zum erneuten Versenden von Nachrichten“ auf Seite 101
- „Datenbank-Worker-Threads“ auf Seite 103

Anpassungen des Abrufintervalls zum Überprüfen auf neue Nachrichten

Um zu überprüfen, ob neue Nachrichten in der Datenbank eingetroffen sind, ruft der SQL Remote-Nachrichtenagent (dbremote) neue Nachrichten ab. Das Standard-Abrufintervall vom Ende eines Abrufs bis zum Start des nächsten beträgt 1 Minute. Sie können das Abrufintervall mittels der Befehlszeilenoption -rd konfigurieren, aber der Standardwert ist üblicherweise ausreichend.

Das Abrufintervall erhöhen

Sie können häufiger abrufen, indem Sie einen Wert in Sekunden verwenden. Beispiel: Der folgende Befehl führt einen Abruf alle dreißig Sekunden durch:

```
dbremote -c "DSN=SQL Anywhere 16 Demo" -rd 30s
```

Im Allgemeinen sollten Sie kurze Abfrageintervalle nur verwenden, wenn ein bestimmter Grund besteht, eine rasche Reaktion auf Nachrichten zu gewährleisten. Ein sehr kurzes Intervall kann sich auf den System-Gesamtdurchsatz aus folgenden Gründen negativ auswirken:

- Sie können Ressourcen bei Abrufen vergeuden, wenn sich keine Nachrichten in der Warteschlange befinden. Wenn Sie beispielsweise E-Mail verwenden, erhöht jeder Abruf des Mailservers die Arbeitslast Ihres Nachrichtensystems. Zu häufiges Abrufen kann negative Auswirkungen auf das Nachrichtensystem haben, ohne dass im Gegenzug Vorteile gewonnen werden.

- Sie können Ihr System mit Neusendeanforderungen überlasten. Wenn Sie das Abrufintervall anpassen, sollten Sie auch den Wartestatus des SQL Remote-Nachrichtenagenten (dbremote) anpassen. Der Wartestatus ist die Anzahl der Abrufe, die der SQL Remote-Nachrichtenagent (dbremote) auf das Eintreffen einer Nachricht außerhalb der Reihenfolge abwartet, bevor er eine erneute Versendung anfordert.

Das Abrufintervall verkürzen

Sie können weniger häufig abrufen, wie z.B. mit dem folgenden Befehl, durch den alle 5 Minuten abgerufen wird:

```
dbremote -c "DSN=SQL Anywhere 16 Demo" -rd 5
```

Ein größeres Abrufintervall kann einen besseren Gesamtdurchsatz der Nachrichten in Ihrem System bewirken, aber auch die Zeitspanne verlängern, die zum Anwenden der Nachrichten benötigt wird. Beispiel: Wenn die Abfrageperiode für ankommende Nachrichten im Vergleich zur Häufigkeit ankommender Nachrichten zu lang ist, kann dies dazu führen, dass Nachrichten in der Warteschlange stehen und auf die Verarbeitung warten.

Siehe auch

- „Performance beim Nachrichtenempfang“ auf Seite 99
- „Anpassen des Durchsatzes durch Caching von erhaltenen Nachrichten“ auf Seite 100
- „Anpassungen an Anforderungen zum erneuten Versenden von Nachrichten“ auf Seite 101
- „Datenbank-Worker-Threads“ auf Seite 103
- „Performance beim Nachrichtenversand“ auf Seite 105

Anpassen des Durchsatzes durch Caching von erhaltenen Nachrichten

Wenn Nachrichten eintreffen, liest der SQL Remote-Nachrichtenagent (dbremote) die Nachrichten und legt sie dann im Cachespeicher ab, bis sie angewendet werden. Das Caching der Nachrichten verhindert Folgendes:

- Erneutes Lesen von Nachrichten außerhalb der Reihenfolge vom Nachrichtensystem, weil dadurch die Performance bei umfangreicheren Systemen verringert werden kann. Das Caching ist bei Nachrichten wichtig, die über ein WAN gelesen werden (wie bei Remote Access Services oder POP3 mit Modem).

- Konflikte zwischen Datenbank-Worker-Threads, die Nachrichten lesen (eine Einzel-Thread-Aufgabe), weil sich der Nachrichteninhalt in einem Cache befindet.

Caching von Nachrichten

Nachrichten werden im Speicher abgelegt, bis sie vom SQL Remote-Nachrichtenagenten (dbremote) angewendet werden, wenn eine der folgenden Bedingungen eintritt:

- Die Transaktionen sind so groß, dass sie mehrteilige Nachrichten erfordern.
- Die Nachrichten treffen außerhalb der Reihenfolge ein.

Größe des Nachrichtencaches angeben

Verwenden Sie die Befehlszeilenoption -m des SQL Remote-Nachrichtenagenten (dbremote), um die Größe des Nachrichtencaches anzugeben. Die Befehlszeilenoption -m bestimmt die maximale Menge an Speicher, die vom SQL Remote-Nachrichtenagenten (dbremote) zum Aufnehmen von Nachrichten verwendet werden kann. Die zulässige Größe kann als *n* (in Byte), *n*kB oder *n*MB angegeben werden. Der Standardwert ist 2048 kB (2 MB). Wenn die angegebene Cachespeichernutzung überschritten wird, werden Nachrichten gelöscht.

Die Befehlszeilenoption -m ist nützlich, wenn Sie eine einzige konsolidierte Datenbank und eine große Anzahl von entfernten Datenbanken haben. Weitere Hinweise finden Sie bei der Befehlszeilenoption -m unter „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203.

Beispiel

Die folgende Befehlszeile startet einen SQL Remote-Nachrichtenagenten (dbremote), wobei 12 MB Speicher als Nachrichtencache verwendet werden:

```
dbremote -c "DSN=SQL Anywhere 16 Demo" -m 12M
```

Siehe auch

- „Performance beim Nachrichtenempfang“ auf Seite 99
- „Anpassungen des Abrufintervalls zum Überprüfen auf neue Nachrichten“ auf Seite 99
- „Anpassungen an Anforderungen zum erneuten Versenden von Nachrichten“ auf Seite 101
- „Datenbank-Worker-Threads“ auf Seite 103
- „Performance beim Nachrichtenversand“ auf Seite 105

Anpassungen an Anforderungen zum erneuten Versenden von Nachrichten

Wenn eine Nachricht in einer Sequenz fehlt, wartet der SQL Remote-Nachrichtenagent (dbremote) eine angegebene Anzahl von Abrufen ab, bevor er anfordert, dass die fehlende Nachricht erneut gesendet wird. Die Anzahl der Abrufe, die der SQL Remote-Nachrichtenagent (dbremote) wartet, wird als Wartestatus bezeichnet. Standardmäßig hat der SQL Remote-Nachrichtenagent (dbremote) einen Wartestatus von 1.

Wenn der SQL Remote-Nachrichtenagent (dbremote) einen Wartestatus von 1 hat und erwartet, dass er Nachricht 6 empfangen wird, aber Nachricht 7 empfängt, führt der SQL Remote-Nachrichtenagent (dbremote) keine Aktion durch. Statt dessen wartet der SQL Remote-Nachrichtenagent (dbremote) auf die Ergebnisse des nächsten Abrufs. Wenn nach dem nächsten Abruf Nachricht 6 weiterhin fehlt, gibt der SQL Remote-Nachrichtenagent (dbremote) eine Neusendeanforderung für Nachricht 6 aus.

Neusende-Wartestatus erhöhen

Angenommen, Sie haben ein sehr kleines Abrufintervall und ein Nachrichtensystem, das die Reihenfolge nicht aufrecht erhält, in der Nachrichten eintreffen. Es ist zu erwarten, dass Nachrichten außerhalb der Reihenfolge eintreffen, nachdem zwei oder drei Abrufe stattgefunden haben. In diesem Beispiel wird

empfohlen, dass Sie die Befehlszeilenoption -rp verwenden, um den Wartestatus des SQL Remote-Nachrichtenagenten (dbremote) zu erhöhen, damit nicht eine große Anzahl unnötiger Neusendeanforderungen gesendet wird. Die Befehlszeilenoption -rp wird oft mit der Option -rd verwendet, mit der das Abrufintervall eingestellt wird.

Beispiel

Es gibt zwei entfernte Benutzer namens Benutzer1 und Benutzer2, die beide den SQL Remote-Nachrichtenagenten (dbremote) mit einem Abrufintervall von 30 Sekunden und einem Wartestatus von 3 Abrufen ausführen. Sie verwenden beispielsweise den folgenden Befehl, um ihre SQL Remote-Nachrichtenagenten (dbremote) auszuführen:

```
dbremote -c "DSN=SQL Anywhere 16 Demo"  -rd 30s -rp 3
```

In der folgenden Sequenz von Vorgängen werden Nachrichten als *userX.n* markiert, wobei X der Benutzername und *n* die Nachrichtennummer ist. Beispiel: Benutzer1.5 ist die fünfte Nachricht von Benutzer1. Der SQL Remote-Nachrichtenagent (dbremote) erwartet, dass für beide Benutzer die Nachrichten bei Nummer 1 beginnen.

Zum Zeitpunkt 0 Sekunden:

1. Der SQL Remote-Nachrichtenagent (dbremote) liest Benutzer1.1, Benutzer2.4.
2. Der SQL Remote-Nachrichtenagent (dbremote) verarbeitet Benutzer1.1.
3. Der SQL Remote-Wartestatus des Nachrichtenagenten (dbremote) ist jetzt: Benutzer1: k.A., Benutzer2: 3, da eine Nachricht außerhalb der Reihenfolge von Benutzer 2 angekommen ist

Zum Zeitpunkt 30 Sekunden:

1. Der SQL Remote-Nachrichtenagent (dbremote) liest: keine neuen Nachrichten.
2. Der SQL Remote-Nachrichtenagent (dbremote) verarbeitet: nichts.
3. Der Wartestatus des SQL Remote-Nachrichtenagenten (dbremote) ist jetzt: Benutzer1: k.A., Benutzer2: 2.

Zum Zeitpunkt 60 Sekunden:

1. Der SQL Remote-Nachrichtenagent (dbremote) liest Benutzer1.3.
2. Der SQL Remote-Nachrichtenagent (dbremote) verarbeitet: keine neuen Nachrichten.
3. Der Wartestatus des SQL Remote-Nachrichtenagenten (dbremote) ist: Benutzer1: 3, Benutzer2: 1.

Zum Zeitpunkt 90 Sekunden:

1. Der SQL Remote-Nachrichtenagent (dbremote) liest: Benutzer1.4.
2. Der SQL Remote-Nachrichtenagent (dbremote) verarbeitet: nichts.
3. Der Wartestatus des SQL Remote-Nachrichtenagenten (dbremote) ist: Benutzer1: 3, Benutzer2: 0.

4. Der SQL Remote-Nachrichtenagent (dbremote) gibt eine Neusendeanforderung an den Benutzer2 aus.

Wenn ein Benutzer eine neue Nachricht erhält, setzt er den Wartestatus des SQL Remote-Nachrichtenagenten (dbremote) zurück, auch wenn die Nachricht nicht jene ist, die er erwartet hat.

Zum Zeitpunkt 120 Sekunden:

1. Der SQL Remote-Nachrichtenagent (dbremote) liest: Benutzer1.2 und Benutzer2.2.

2. Der SQL Remote-Nachrichtenagent (dbremote) verarbeitet Benutzer1.2, Benutzer1.3, Benutzer1.4 und Benutzer2.2.

3. Der Wartestatus des SQL Remote-Nachrichtenagenten (dbremote) ist Benutzer1: k.A., Benutzer2: k.A.

Siehe auch

- „Performance beim Nachrichtenempfang“ auf Seite 99
- „Anpassungen des Abrufintervalls zum Überprüfen auf neue Nachrichten“ auf Seite 99
- „Anpassen des Durchsatzes durch Caching von erhaltenen Nachrichten“ auf Seite 100
- „Datenbank-Worker-Threads“ auf Seite 103
- „Performance beim Nachrichtenversand“ auf Seite 105

Datenbank-Worker-Threads

Die folgenden Schritte beschreiben, wie der SQL Remote-Nachrichtenagent (dbremote) eintreffende Nachrichten verarbeitet:

1. Er liest die Nachrichten. Nachrichten werden ausgelesen und die Header-Daten werden überprüft (um die korrekte Reihenfolge der Anwendung zu ermitteln). Zum Auslesen der Nachrichten aus dem Nachrichtensystem wird nur ein Thread benutzt.

2. Er wendet die Nachrichten an. Gelesene Nachrichten werden an Datenbank-Worker-Threads zur Anwendung übergeben.

   In entfernten Datenbanken werden die Nachrichten üblicherweise seriell angewendet. In einem mehrschichtigen System kann eine entfernte Datenbank auch eine konsolidierte Datenbank für andere entfernte Datenbanken sein. Bei diesem Typ einer entfernten Datenbank werden die Nachrichten wie auf einer konsolidierten Datenbank angewendet.

   Auf der konsolidierten Datenbank werden die Nachrichten standardmäßig seriell angewendet. Sie können zusätzliche Datenbank-Worker-Threads verwenden, um von entfernten Benutzern eintreffende Nachrichten parallel anzuwenden. Weitere Hinweise finden Sie bei der Befehlszeilenoption -w unter „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203.

   Wenn Datenbank-Worker-Threads in einer konsolidierten Datenbank verwendet werden, gilt Folgendes:

- Nachrichten von verschiedenen entfernten Benutzern werden parallel übernommen.
- Nachrichten von einem einzelnen entfernten Benutzer werden seriell übernommen.

  Beispiel: Zehn Nachrichten von einem einzelnen entfernten Benutzer werden von einem einzelnen Worker-Thread in der korrekten Reihenfolge in die Datenbank übernommen.

Vorteile der Verwendung von Datenbank-Worker-Threads

Die Verwendung von Datenbank-Worker-Threads in der konsolidierten Datenbank kann den Durchsatz verbessern, indem es möglich wird, Nachrichten parallel statt seriell anzuwenden. Die Performancesteigerung ist signifikant, wenn der Datenbankserver auf einem System im Striping-Modus ausgeführt wird.

Nachteile der Verwendung von Datenbank-Worker-Threads

Die Verwendung von Datenbank-Worker-Threads in der konsolidierten Datenbank kann den Durchsatz vermindern, wenn die Worker-Threads viele Sperren zwischen Benutzern bewirken.

Ein Deadlock wird verarbeitet, indem die zurückgesetzte Transaktion zu einem späteren Zeitpunkt erneut angewendet wird.

Legt die Anzahl von Datenbank-Worker-Threads fest.

In der konsolidierten Datenbank verwenden Sie die Befehlszeilenoption -w, um die Anzahl der Datenbank-Worker-Threads festzulegen. Beispiel: Der folgende Befehl legt die Anzahl von Worker-Threads auf 5 fest:

```
dbremote -c "DSN=SQL Anywhere 16 Demo"  -w 5
```

Siehe auch

- „SQL Remote-Nachrichtenagenten (dbremote) im kontinuierlichen Modus ausführen“ auf Seite 92
- „Performance beim Nachrichtenempfang“ auf Seite 99
- „Anpassungen des Abrufintervalls zum Überprüfen auf neue Nachrichten“ auf Seite 99
- „Anpassen des Durchsatzes durch Caching von erhaltenen Nachrichten“ auf Seite 100
- „Anpassungen an Anforderungen zum erneuten Versenden von Nachrichten“ auf Seite 101
- „Performance beim Nachrichtenversand“ auf Seite 105

Aufgaben zum Senden von Nachrichten

Der SQL Remote-Nachrichtenagent (dbremote) führt die folgenden Aufgaben durch, um Nachrichten zu senden:

- **Publikationseigentümer-Transaktionslog durchsuchen** Der SQL Remote-Nachrichtenagent (dbremote) durchsucht das Transaktionslog der Publikationseigentümer-Datenbank und konvertiert die Transaktionslog-Einträge in Nachrichten für Subskribenten. Die maximale Nachrichtenlänge, wie sie von der Befehlszeilenoption -l angegeben wird, muss dieselbe bei allen Datenbanken im System sein.

  Bei umfangreichen Transaktionen erstellt der SQL Remote-Nachrichtenagent (dbremote) mehrteilige Nachrichten. Diese Nachrichten enthalten jeweils eine Sequenznummer, die ihre Position in der

Transaktion protokolliert. Der SQL Remote-Nachrichtenagent (dbremote) in der Subskribenten-Datenbank verwendet diese Sequenznummer, um sicherzustellen, dass die Nachrichten in der korrekten Reihenfolge angewendet werden und keine Nachricht verloren geht.

- **Nachrichten an die entfernten Datenbanken senden** Der SQL Remote-Nachrichtenagent (dbremote) versendet Nachrichten zu einem Zeitpunkt, der durch die Sendefrequenz-Eigenschaften des jeweiligen entfernten Benutzers festgelegt wird.

  Der SQL Remote-Nachrichtenagent (dbremote) sendet Nachrichten früher, wenn sein Cachespeicher den eingestellten Wert überschreitet. Der SQL Remote-Nachrichtenagent (dbremote) speichert seine Nachrichten im Cachespeicher. Wenn der verwendete Cachespeicher den angegebenen Wert überschreitet, werden die Nachrichten gesendet.

- **Neusendeanforderungen von entfernten Datenbanken verarbeiten** Wenn ein Benutzer anfordert, dass eine Nachricht erneut gesendet wird, unterbricht der SQL Remote-Nachrichtenagent (dbremote) in der Publikationseigentümer-Datenbank den regulären Sendeprozess, um die Neusendeanforderung zu verarbeiten.

  Sie steuern die Dringlichkeit, mit der diese Neusendeanforderungen verarbeitet werden, mit der Befehlszeilenoption -ru.

- **Bestätigungen an die Publikationseigentümer-Datenbank senden** Sobald eine Nachricht in der subskribierten Datenbank empfangen und übernommen wurde, wird eine Bestätigung zurück an den Publikationseigentümer gesendet.

Siehe auch

- „Anpassungen der Sendeverzögerung“ auf Seite 106
- „Anpassen des Durchsatzes durch Caching von gesendeten Nachrichten“ auf Seite 106
- „Verarbeitungsgeschwindigkeit für Neusendeanforderungen“ auf Seite 107
- „System der garantierten Nachrichtenzustellung“ auf Seite 108

Performance beim Nachrichtenversand

Das Haupt-Performanceproblem beim Versenden von Nachrichten ist die Durchlaufzeit zwischen der Eingabe der Daten an einem Standort und ihrem Erscheinen an den anderen Standorten. Um diese Verzögerung beim Versenden von Nachrichten mit dem SQL Remote-Nachrichtenagenten (dbremote) zu verringern, können Sie die folgenden Variablen anpassen:

- Wie häufig Nachrichten an entfernte Datenbanken gesendet werden.
- Die Größe der Nachrichten.
- Wie schnell Neusendeanforderungen abgearbeitet werden.

Siehe auch

- „Anpassungen der Sendeverzögerung“ auf Seite 106
- „Performance beim Nachrichtenempfang“ auf Seite 99
- „Verarbeitungsgeschwindigkeit für Neusendeanforderungen“ auf Seite 107

Anpassungen der Sendeverzögerung

Um zu versendende Nachrichten zu erstellen, ruft der SQL Remote-Nachrichtenagent (dbremote) neue Daten aus dem Transaktionslog ab. Die Sendeverzögerung gibt an, wie lange zwischen Abrufen abgewartet wird, um weitere Transaktionslogdaten zu senden. Das Standard-Abrufintervall vom Ende eines Abrufs bis zum Start des nächsten beträgt 1 Minute. Sie können die Sendeverzögerung mittels der Befehlszeilenoption -sd konfigurieren, aber der Standardwert ist üblicherweise ausreichend. Die Sendeverzögerung sollte kleiner oder gleich der Sendefrequenz des entfernten Benutzers sein.

Die Sendeverzögerung vermindern

Sie können häufiger abrufen, indem Sie einen Wert in Sekunden verwenden. Beispiel: Der folgende Befehl führt einen Abruf alle dreißig Sekunden durch:

```
dbremote -c "DSN=SQL Anywhere 16 Demo"  -sd 30s ...
```

Die Sendeverzögerung erhöhen

Sie können weniger häufig abrufen, wie z.B. mit der folgenden Befehlszeile, durch die alle 60 Minuten abgerufen wird:

```
dbremote -c "DSN=SQL Anywhere 16 Demo"  -sd 60
```

In der Regel bedeuten größere Sendeintervalle, dass der SQL Remote-Nachrichtenagent (dbremote) den größten Teil der Nachrichtenerstellung vor dem Versenden der Nachrichten durchführt. Kürzere Intervalle werden im Allgemeinen bevorzugt, weil sie die Nachrichtenerstellung gleichmäßiger verteilen

Siehe auch

- „Anpassen des Durchsatzes durch Caching von gesendeten Nachrichten“ auf Seite 106
- „Verarbeitungsgeschwindigkeit für Neusendeanforderungen“ auf Seite 107
- „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203

Anpassen des Durchsatzes durch Caching von gesendeten Nachrichten

Der SQL Remote-Nachrichtenagent (dbremote) legt die zu versenden Nachrichten in einem konfigurierbaren Bereich des Cachespeichers ab.

Wenn alle entfernten Datenbanken unterschiedliche Teilmengen der Vorgänge erhalten, die repliziert werden, wird simultan je eine separate Nachricht für jede entfernte Datenbank aufgebaut. Für eine Gruppe von entfernten Benutzern, die dieselben Vorgänge erhalten, wird nur eine Nachricht aufgebaut. Nachrichten werden gesendet, wenn Folgendes eintritt:

- Die Sendefrequenz ist erreicht.
- Wenn der verwendete Cachespeicher den Wert -m überschreitet.
- Wenn die Größe der Nachricht ihre maximale Größe erreicht (wie durch die Befehlszeilenoption -l angegeben).

Größe des Nachrichtencaches angeben

Die Größe des Nachrichtencaches wird in der Befehlszeile des SQL Remote-Nachrichtenagenten (dbremote) mit der Befehlszeilenoption -m angegeben.

Die Befehlszeilenoption -m bestimmt die maximale Menge an Speicher, die vom SQL Remote-Nachrichtenagenten (dbremote) zum Aufbau von Nachrichten verwendet werden kann. Die zulässige Größe kann als *n* (in Byte), *n*kB oder *n*MB angegeben werden. Der Standardwert ist 2048 kB (2 MB).

Die Befehlszeilenoption -m ist nützlich, wenn Sie eine einzige konsolidierte Datenbank und eine große Anzahl von entfernten Datenbanken haben. Weitere Hinweise finden Sie bei der Befehlszeilenoption -m unter „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203.

Beispiel

Die folgende Befehlszeile startet einen SQL Remote-Nachrichtenagenten (dbremote), wobei 12 MB Speicher als Nachrichtencache verwendet werden:

```
dbremote -c "DSN=SQL Anywhere 16 Demo" -m 12M
```

Siehe auch

- „Anpassungen der Sendeverzögerung“ auf Seite 106
- „Verarbeitungsgeschwindigkeit für Neusendeanforderungen“ auf Seite 107

Verarbeitungsgeschwindigkeit für Neusendeanforderungen

Da das erneute Senden einer Nachricht den regulären Nachrichtensendeprozess unterbricht, verzögert der SQL Remote-Nachrichtenagent (dbremote) die Verarbeitung von Neusendeanforderungen. Standardmäßig wartet der SQL Remote-Nachrichtenagent (dbremote) eine Zeitspanne ab, die die Hälfte der Sendefrequenz des entfernten Benutzers darstellt, der das Neusenden angefordert hat.

Um eine Nachricht erneut zu senden, führt der SQL Remote-Nachrichtenagent (dbremote) die folgenden Aufgaben durch:

- Er stoppt das Durchsuchen des Transaktionslogs und den Aufbau von neuen Nachrichten.
- Er löscht die aktuellen Nachrichten, die im Cache abgelegt sind und auf ihre Versendung warten. Alle Arbeit, die der SQL Remote-Nachrichtenagent (dbremote) in das Lesen des Transaktionslogs und in den Aufbau dieser Nachrichten investiert hat, geht verloren.
- Er liest das Transaktionslog erneut ab dem Ausgangspunkt, der in der Neusendeanforderung angefordert wurde. Der SQL Remote-Nachrichtenagent (dbremote) baut die Nachrichten auf und legt sie im Cache ab.

- Er wartet, bis die nächste Sendefrequenz aktuell wird, und versendet dann die Nachrichten.

Sie müssen einen Ausgleich zwischen der Dringlichkeit des Versendens von Neusendeanforderungen für gesendete Nachrichten und der Priorität der Verarbeitung von regulären Nachrichten finden.

Die Befehlszeilenoption -ru steuert die Dringlichkeit der Neusendeanforderungen. Um die Verarbeitung von Neusendeanforderungen zu verschieben, bis weitere eingetroffen sind, setzen Sie diese Option auf eine längere Zeitspanne. Beispiel: Der folgende Befehl wartet eine Stunde, bevor die Neusendeanforderung verarbeitet wird:

```
dbremote -c "DSN=SQL Anywhere 16 Demo" -ru 1h
```

Siehe auch

- „Anpassungen der Sendeverzögerung“ auf Seite 106
- „Anpassen des Durchsatzes durch Caching von gesendeten Nachrichten“ auf Seite 106
- „Anpassungen an Anforderungen zum erneuten Versenden von Nachrichten“ auf Seite 101
- „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203

System der garantierten Nachrichtenzustellung

Das System der garantierten Nachrichtenzustellung stellt Folgendes sicher:

- Alle replizierten Vorgänge werden in der korrekten Reihenfolge übernommen.
- Keine replizierten Vorgänge werden übersehen.
- Kein replizierter Vorgang wird zweimal angewendet.

Das System der garantierten Nachrichtenzustellung verwendet die folgenden Informationen:

- **Die Statusinformationen, die in der SYSREMOTEUSER-Systemtabelle verwaltet werden** Diese Tabelle enthält eine Zeile für jeden Subskribenten mit Statusinformationen über Nachrichten, die an diesen Subskribenten versendet und von ihm empfangen werden. Zum Beispiel:
  - In der konsolidierten Datenbank enthält die SYSREMOTEUSER-Systemtabelle eine Zeile für jeden entfernten Benutzer.
  - In den jeweiligen entfernten Datenbanken enthält die SYSREMOTEUSER-Systemtabelle eine einzelne Zeile mit Informationen über die konsolidierte Datenbank.

  Die SYSREMOTEUSER-Systemtabelle wird durch den SQL Remote-Nachrichtenagenten (dbremote) verwaltet.

  In der Subskribenten-Datenbank sendet der SQL Remote-Nachrichtenagent (dbremote) eine Bestätigung an die Publikationseigentümer-Datenbank, um sicherzustellen, dass die SYSREMOTEUSER-Systemtabelle an jedem Ende der Subskription korrekt verwaltet ist.
- **Die Informationen im Header der Nachrichten** Der SQL Remote-Nachrichtenagent (dbremote) liest die Header-Daten in den Nachrichten und verwendet diese Informationen, um die

SYSREMOTEUSER-Systemtabelle zu aktualisieren. Eine Nachricht enthält die folgenden Daten in ihrem Header:

- **Den resend_count-Wert** Ein Zähler, der die Häufigkeit protokolliert, mit der die Empfängerdatenbank Nachrichten verloren hat.

  Im folgenden Beispiel ist der resend_count-Wert 1.

  ```
  Current message's header: (1-0000942712-0001119170-0)
  ```

- **Das Transaktionslog-Offset der letzten COMMIT-Anweisung in der vorherigen Nachricht.** Im folgenden Beispiel ist das Transaktionslog-Offset der letzten Festschreibung in der vorherigen Nachricht 0000942712.

  ```
  Previous message's header:(0-0000923357-0000942712-0)
  Current message's header: (0-0000942712-0001119170-0)
  ```

- **Transaktionslog-Offset der letzten COMMIT-Anweisung in der aktuellen Nachricht** Im folgenden Beispiel ist die letzte Festschreibung in der aktuellen Nachricht 0001119170:

  ```
  Current message's header: (0-0000942712-0001119170-0)
  ```

  Wenn eine Transaktion mehrere Nachrichten umfasst, können beide Transaktionslog-Offsets identisch sein, bis die letzte Nachricht ein COMMIT enthält.

  Im folgenden Beispiel tritt das COMMIT erst bei der vierten Nachricht auf:

  ```
  (0-0000942712-0000942712-0)
  (0-0000942712-0000942712-1)
  (0-0000942712-0000942712-2)
  (0-0000942712-0001119170-3)
  ```

- **Eine Sequenznummer** Wenn eine Transaktion mehrere Nachrichten umfasst, wird diese Sequenznummer verwendet, um die Nachrichten korrekt zu sortieren.

  Eine Sequenznummer von null kann folgendes angeben:

  - Die Nachricht ist nicht Teil einer mehrteiligen Nachricht, wenn die Transaktionslog-Offsets verschieden sind.

    Im folgenden Beispiel sind die Nachrichten nicht Teil einer mehrteiligen Nachricht:

    ```
    (0-0000923200-0000923357-0)
    (0-0000923357-0000942712-0)
    ```

  - Die Nachricht ist die erste einer mehrteiligen Nachricht, wenn die Transaktionslog-Offsets gleich sind.

    Im folgenden Beispiel ist die erste Nachricht ein Teil einer mehrteiligen Nachricht:

    ```
    (0-0000942712-0000942712-0)
    (0-0000942712-0000942712-1)
    (0-0000942712-0000942712-2)
    (0-0000942712-0001119170-3)
    ```

Siehe auch

- „Reihenfolge der Vorgänge“ auf Seite 110
- „Verlorene oder beschädigte Nachrichten“ auf Seite 111
- „Nachrichten werden nur einmal übernommen“ auf Seite 112

Reihenfolge der Vorgänge

Um sicherzustellen, dass die replizierten Anweisungen in der korrekten Reihenfolge übernommen werden, verwendet das System der garantierten Nachrichtenzustellung die Transaktionslog-Offsets der Publikationseigentümer- und Subskribenten-Datenbanken. Jedes COMMIT wird in einem Transaktionslog mit einem klar definierten Offset markiert. Die Reihenfolge von Transaktionen können Sie ermitteln, indem Sie ihre Transaktionslog-Offsetwerte vergleichen.

Jede Nachricht enthält die folgenden Transaktionslog-Offsets:

- Das Transaktionslog-Offset der letzten COMMIT-Anweisung in der vorherigen Nachricht Wenn eine Transaktion mehrere Nachrichten umfasst, werden die Nachrichten mithilfe der Sequenznummer innerhalb der Transaktion korrekt gereiht.
- Das Transaktionslog-Offset der letzten COMMIT-Anweisung in der Nachricht.

Nachrichtensortierung

Beim Versand werden Nachrichten anhand des Offsets der letzten COMMIT-Anweisung der vorherigen Nachricht aufgereiht. Wenn eine Transaktion mehrere Nachrichten umfasst, wird eine Sequenznummer innerhalb der Transaktion verwendet, um die Nachrichten korrekt zu reihen.

Nachrichten versenden

Die log_sent-Spalte in der SYSREMOTEUSER-Systemtabelle enthält das lokale Transaktionslog-Offset für die letzte Nachricht, die an den Subskribenten gesendet wurde.

Im Folgenden wird beschrieben, wie die SYSREMOTEUSER-Systemtabellen aktualisiert werden, wenn Nachrichten gesendet werden.

1. Wenn der SQL Remote-Nachrichtenagent (dbremote) des Publikationseigentümers eine Nachricht an einen Subskribenten sendet, setzt er auch den log_sent-Wert auf den Transaktionslog-Offsetwert der letzten COMMIT-Anweisung in der gesendeten Nachricht.

   Beispiel: Der Publikationseigentümer sendet die folgende Nachricht an Benutzer1.

   ```
   (0-0000923200-0000923357-0)
   ```

   In der SYSREMOTEUSER-Systemtabelle des Publikationseigentümers setzt der Publikationseigentümer den log_sent-Wert auf 0000923357 für Benutzer1.

2. Nach Empfang und Übernahme der Nachricht in der Subskribenten-Datenbank wird eine Bestätigung an den Publikationseigentümer gesendet. Die Bestätigung enthält das letzte Transaktionslog-Offset, das von der Subskribenten-Datenbank angewendet wurde.

Beispiel: Die Meldung bestätigt, dass Benutzer1 alle Transaktionen bis zum Transaktionslog-Offset 0000923357 angewendet hat.

3. Wenn der SQL Remote-Nachrichtenagent (dbremote) des Publikationseigentümers die Bestätigung erhält, setzt er die confirm_sent-Spalte auf den Wert des Bestätigungsoffsets für den Benutzer in der SYSREMOTEUSER-Systemtabelle.

   Beispiel: Der Publikationseigentümer setzt die confirm_sent-Spalte auf 0000923357 für Benutzer1 in der SYSREMOTEUSER-Systemtabelle des Publikationseigentümers.

Sowohl der log_sent- als auch der confirm_sent-Wert enthalten Transaktionslog-Offsets des Publikationseigentümer-Transaktionslogs. Der confirm_sent-Wert kann kein späteres Offset als der log_sent-Wert sein.

Nachrichten empfangen

Im Folgenden wird beschrieben, wie die SYSREMOTEUSER-Systemtabellen beim Empfang von Nachrichten aktualisiert werden.

1. Wenn der SQL Remote-Nachrichtenagent (dbremote) in einer Subskribenten-Datenbank eine Replikationsaktualisierung empfängt und anwendet, aktualisiert er die log_received-Spalte in der SYSREMOTEUSER-Systemtabelle mit dem Offset der letzten COMMIT-Anweisung in der Nachricht.

   Beispiel: Wenn der Subskribent die folgende Nachricht empfängt und anwendet, wird der log_received-Wert in der SYSREMOTEUSER-Systemtabelle auf 0000923357 gesetzt.

   ```
   (0-0000923200-0000923357-0)
   ```

   Die log_received-Spalte in einer beliebigen Subskribenten-Datenbank enthält daher ein Transaktionslog-Offset, das im Transaktionslog der Publikationseigentümer-Datenbank vorhanden ist.

2. Wenn die Vorgänge empfangen und angewendet werden, setzt der SQL Remote-Nachrichtenagent (dbremote) des Subskribenten den confirm_received-Wert in seiner SYSREMOTEUSER-Systemtabelle und sendet dann eine Bestätigung an die Publikationseigentümer-Datenbank.

Siehe auch

- „Reihenfolge der Vorgänge“ auf Seite 110
- „Verlorene oder beschädigte Nachrichten“ auf Seite 111
- „Nachrichten werden nur einmal übernommen“ auf Seite 112

Verlorene oder beschädigte Nachrichten

Die SYSREMOTEUSER-Systemtabelle enthält zwei Spalten, die das Neusenden von Nachrichten verwalten:

- **resend_count-Spalte** Ein Zähler, der die Häufigkeit protokolliert, mit der die Subskribenten-Datenbank Nachrichten verloren hat.

- **rereceive_count-Spalte** Ein Zähler, der protokolliert, wie häufig der SQL Remote-Nachrichtenagent (dbremote) ermittelt hat, dass Nachrichten von einem Publikationseigentümer-Benutzer verloren gegangen sind.

Wenn Nachrichten in der korrekten Reihenfolge in der Subskribenten-Datenbank empfangen werden, gilt Folgendes:

1. Der SQL Remote-Nachrichtenagent (dbremote) des Subskribenten wendet die Nachrichten in der korrekten Reihenfolge an und aktualisiert die SYSREMOTEUSER-Systemtabelle.
2. Der SQL Remote-Nachrichtenagent (dbremote) des Subskribenten sendet eine Bestätigungsmeldung an den Publikationseigentümer.
3. Wenn der Publikationseigentümer die Bestätigung empfängt, aktualisiert sein SQL Remote-Nachrichtenagent (dbremote) seine SYSREMOTEUSER-Systemtabelle.

Wenn Nachrichten nicht in der richtigen Reihenfolge empfangen werden, gilt Folgendes:

1. Der SQL Remote-Nachrichtenagent (dbremote) des Subskribenten sendet eine Neusendeaufforderung und inkrementiert den receive_count-Wert in seiner SYSREMOTEUSER-Systemtabelle.
2. Wenn der Publikationseigentümer die Neusendeaufforderung erhält, inkrementiert er den resend_count-Wert in seiner SYSREMOTEUSER-Systemtabelle für den Subskribenten.
3. In der SYSREMOTEUSER-Systemtabelle des Publikationseigentümers wird der log_sent-Wert auf den Wert in der confirm_sent-Spalte gesetzt. Das Zurücksetzen des log_sent-Werts bewirkt, dass Vorgänge erneut versendet werden.

Siehe auch

- „Reihenfolge der Vorgänge“ auf Seite 110
- „Nachrichten werden nur einmal übernommen“ auf Seite 112

Nachrichten werden nur einmal übernommen

Der SQL Remote-Nachrichtenagent (dbremote) des Subskribenten vergleicht den resend_count-Wert in einem Nachrichten-Header mit dem rereceive_count-Wert in seiner lokalen SYSREMOTEUSER-Systemtabelle. Wenn der resend_count-Wert kleiner als rereceive_count ist, wird die Nachricht nicht angewendet, sondern gelöscht. Dieses Verhalten gewährleistet, dass die Vorgänge nicht mehr als einmal übernommen werden.

Siehe auch

- „Reihenfolge der Vorgänge“ auf Seite 110
- „Verlorene oder beschädigte Nachrichten“ auf Seite 111

Nachrichtengröße

Der folgende Abschnitt behandelt das Nachrichten-Kodierungs- und Komprimierungsschema im SQL Remote-Nachrichtenagenten (dbremote).

Der SQL Remote-Nachrichtenagent stellt die folgenden Kodierungs- und Komprimierungsfunktionen bereit:

- **Kompatibilität** Das System kann so eingerichtet werden, dass es mit früheren Versionen von SQL Anywhere kompatibel ist.
- **Komprimierung** Sie können den Grad der Komprimierung für Ihre Nachrichten bestimmen.

  Die Nachrichtengröße wirkt sich auf die Effizienz aus, mit der Nachrichten durch ein System transportiert werden. Komprimierte Nachrichten können von einem Nachrichtensystem effizienter als nicht-komprimierte Nachrichten verarbeitet werden. Eine Komprimierung kann allerdings eine signifikante Zeitspanne erfordern.
- **Kodierung** SQL Remote kodiert Nachrichten, um sicherzustellen, dass sie den Durchgang durch die Nachrichtensysteme unbeschädigt überstehen. Das Kodierungsschema kann angepasst werden, um zusätzliche Funktionen bereitzustellen.

Siehe auch

- „SQL Remote-Upgrades“ [*SQL Anywhere 16 - Änderungen und Upgrades*]
- „compression-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]
- „Prävention der Nachrichtenbeschädigung mit Kodierung“ auf Seite 113
- Anforderungen für den SQL Remote-Nachrichtenagenten (dbremote) auf Seite 92

Prävention der Nachrichtenbeschädigung mit Kodierung

SQL Remote kodiert Nachrichten, um sicherzustellen, dass sie den Durchgang durch die Nachrichtensysteme unbeschädigt überstehen. Die Standard-Nachrichtenkodierung von SQL Remote führt Folgendes durch:

- Wenn das Nachrichtensystem binäre Nachrichtenformate verwenden kann, werden die Nachrichten nicht kodiert.
- Wenn das Nachrichtensystem, z.B. SMTP, textbasierte Nachrichtenformate erfordert, konvertiert eine Kodierungs-DLL (*dbencod16.dll*) die Nachrichten vor dem Senden in ein Textformat. Das Nachrichtenformat wird mit derselben DLL beim Empfänger dekodiert.

  Sie können das Kodierungsschema anpassen, um zusätzliche Funktionen bereitzustellen.
- Wenn die Datenbankoption compression auf -1 gesetzt ist, wird eine mit der Version 5 kompatible Kodierung für alle Nachrichtensysteme verwendet.

Siehe auch

- „Benutzerdefinierte Kodierungsschemata“ auf Seite 114
- „SQL Remote-Upgrades“ [*SQL Anywhere 16 - Änderungen und Upgrades*]

Benutzerdefinierte Kodierungsschemata

Um ein benutzerdefiniertes Kodierungsschema zu implementieren, können Sie eine angepasste Kodierungs-DLL erstellen. Mit dieser angepassten DLL können Sie spezielle Funktionen anwenden, die für ein bestimmtes Nachrichtensystem notwendig sind, oder Statistiken z.B. darüber erstellen, wie viele Nachrichten an den einzelnen Benutzer versendet werden.

Die Header-Datei *%SQLANY16%\SDK\Include\dbrmt.h* enthält eine Programmierschnittstelle, die Sie verwenden können, um ein benutzerdefiniertes Kodierungsschema aufzubauen.

Um Ihre benutzerdefinierte DLL zu verwenden, setzen Sie den Nachrichten-Steuerungsparameter encode_dll auf einen Wert, der den vollständigen Pfad zur benutzerdefinierten DLL wiedergibt. Beispiel:

```
SET REMOTE FTP OPTION "Public"."encode_dll" = 'c:\\sqlany16\\Bin32\
\custom.dll';
```

Hinweis
Kodierung und Dekodierung müssen kompatibel sein. Wenn Sie also eine benutzerdefinierte Kodierung implementieren, müssen Sie sicherstellen, dass die DLL beim Empfänger vorhanden und die DLL so eingerichtet ist, dass Ihre Nachrichten korrekt dekodiert werden.

Siehe auch

- „SET REMOTE OPTION-Anweisung [SQL Remote]“ auf Seite 238

SQL Remote-Nachrichtensysteme

Bei der SQL Remote-Replikation ist ein Nachrichtensystem ein Protokoll für den Austausch von Nachrichten zwischen der konsolidierten und einer entfernten Datenbank. SQL Remote tauscht Daten zwischen Datenbanken aus, indem ein oder mehrere darunterliegende Nachrichtensysteme verwendet werden. SQL Remote unterstützt die folgenden Nachrichtensysteme:

- **Gemeinsame Dateibenutzung** Ein einfaches System, das keine zusätzliche Software benötigt.
- **FTP** Internet File-Transfer-Protokoll.
- **HTTP** Hypertext-Transfer-Protokoll.
- **SMTP/POP** Internet E-Mail-Protokoll.

Sie wählen ein Nachrichtensystem aus, wenn Sie einem Benutzer ein REMOTE- oder CONSOLIDATE-Privileg erteilen.

Jedes Nachrichtensystem in einem SQL Remote-System besitzt Steuerungsparameter und andere Einstellungen, die eingerichtet werden müssen.

Nicht alle Nachrichtensysteme werden auf allen Betriebssystemen unterstützt.

Ein Nachrichtensystem einrichten

Bevor Sie ein Nachrichtensystem verwenden können, müssen Sie die Adresse des Publikationseigentümers einstellen.

Jede Nachrichtentypdefinition enthält den Namen des Nachrichtensystemtyps (**FILE**, **FTP**, **HTTP** oder **SMTP**) und die Adresse des Publikationseigentümers für diesen Nachrichtentyp.

Die in einer Nachrichtentypdefinition mitgelieferte Adresse ist eng mit der Publikationseigentümer-ID der Datenbank verknüpft.

Extraktionsdienstprogramm (dbxtract)

Die Adresse des Publikationseigentümers in einer konsolidierten Datenbank wird vom Extraktionsdienstprogramm (dbxtract) und vom **Assistenten zum Extrahieren einer Datenbank** als Rücksendeadresse bei der Erstellung von entfernten Datenbanken verwendet. Sie wird auch vom SQL Remote-Nachrichtenagenten (dbremote) verwendet, um herauszufinden, wo nach eintreffenden Nachrichten beim FILE-System gesucht werden soll.

Siehe auch

- „Das FILE-Nachrichtensystem“ auf Seite 119
- „Das FTP-Nachrichtensystem“ auf Seite 121
- „Das SMTP-Nachrichtensystem“ auf Seite 128
- „Das HTTP-Nachrichtensystem“ auf Seite 124
- „REMOTE-Privilegien erteilen“ auf Seite 25
- „CONSOLIDATE-Privileg erteilen“ auf Seite 27
- „Unterstützte Plattformen“ [*SQL Anywhere 16 - Einführung*]

Einen Nachrichtentyp erstellen

Fügen Sie Nachrichtentypen für SQL Remote-Benutzer hinzu.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf **SQL Remote-Benutzer**.
3. Klicken Sie auf die Registerkarte **Nachrichtentypen**.
4. Klicken Sie auf **Datei** » **Neu** » **Nachrichtentyp**.

5. Geben Sie einen Namen für den Nachrichtentyp ein. Der Name sollte einer Nachrichtentyp-DLL entsprechen, die bereits in Ihrem SQL Anywhere-Installationsverzeichnis installiert ist. Klicken Sie auf **Weiter**.

6. Geben Sie in das Feld **Wie lautet die Adresse des Publikationseigentümers?** eine Adresse des Publikationseigentümers ein. Klicken Sie auf **Fertig stellen**.

Ergebnisse

Der neue Nachrichtentyp wird erstellt.

Siehe auch

- „CREATE REMOTE [MESSAGE] TYPE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Einen Nachrichtentyp ändern“ auf Seite 117
- „Einen Nachrichtentyp löschen“ auf Seite 117

Remote-Nachrichtentypen unter Windows Mobile erstellen

Wenn Sie Windows Mobile-Dienste installiert haben, können Sie SQL Remote für die ActiveSync-Synchronisation aus Sybase Central einrichten.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Diese Option ändert Registrierungswerte auf Ihrem Windows Mobile-Gerät und legt das Verzeichnis für Nachrichten, die Sie über Ihre FILE-Nachrichtenverbindung empfangen, als Microsoft ActiveSync-Verzeichnis fest. SQL Remote liest Nachrichtenverbindungsparameter nur dann aus der Registrierung, wenn in der entfernten Datenbank keine Nachrichtenverbindungsparameter festgelegt sind. Wenn Ihr Windows Mobile-Gerät über die Docking-Station mit dem Desktop-PC verbunden wird, synchronisiert Microsoft ActiveSync die Dateien im Microsoft ActiveSync-Verzeichnis des PCs mit jenen im Windows Mobile ActiveSync-Verzeichnis.

Aufgabe

1. Verbinden Sie Ihr Windows Mobile-Gerät mit einem Desktopcomputer.

2. Verwenden Sie Sybase Central und klicken Sie unter **Extras** auf **SQL Anywhere 16** » **Windows Mobile-Nachrichtentypen bearbeiten**.

3. Ändern Sie, falls nötig, den Wert für den Parameter **Verzeichnis** auf das mit Ihrem Desktopcomputer synchronisierte Verzeichnis auf dem Windows Mobile-Gerät.

4. Klicken Sie auf **OK**, um Ihre Änderungen in der Registrierung des Windows Mobile-Geräts zu speichern.

Ergebnisse

Der entfernte Nachrichtentyp wird erstellt.

Siehe auch

- „CREATE REMOTE [MESSAGE] TYPE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Einen Nachrichtentyp ändern“ auf Seite 117
- „Einen Nachrichtentyp löschen“ auf Seite 117

Einen Nachrichtentyp ändern

Um die Adresse eines Publikationseigentümers zu ändern, ändern Sie seinen Nachrichtentyp. Den Namen eines vorhandenen Nachrichtentyps können Sie nicht ändern, sondern müssen ihn löschen und einen neuen Nachrichtentyp mit dem neuen Namen erstellen.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf **SQL Remote-Benutzer**.
3. Klicken Sie auf die Registerkarte **Nachrichtentypen**.
4. Rechtsklicken Sie auf den Nachrichtentyp, den Sie ändern möchten, und klicken Sie auf **Eigenschaften**.
5. Aktualisieren Sie die Eigenschaften des Nachrichtentyps und klicken Sie auf **OK**.

Ergebnisse

Die Meldungseigenschaften werden aktualisiert.

Siehe auch

- „ALTER REMOTE MESSAGE TYPE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Einen Nachrichtentyp erstellen“ auf Seite 115
- „Einen Nachrichtentyp löschen“ auf Seite 117

Einen Nachrichtentyp löschen

Das Löschen eines Nachrichtentyps entfernt die Adresse des Publikationseigentümers aus der Nachrichtendefinition.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Verwenden Sie Sybase Central, um eine Verbindung mit der Datenbank herzustellen.
2. Doppelklicken Sie auf **SQL Remote-Benutzer**.
3. Klicken Sie auf die Registerkarte **Nachrichtentypen**.
4. Rechtsklicken Sie auf den Nachrichtentyp, den Sie entfernen möchten, und klicken Sie auf **Löschen**.
5. Klicken Sie auf **Ja**.

Ergebnisse

Der Nachrichtentyp wird gelöscht.

Siehe auch

- „DROP REMOTE MESSAGE TYPE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Einen Nachrichtentyp erstellen“ auf Seite 115
- „Einen Nachrichtentyp ändern“ auf Seite 117

Festlegen von Steuerungsparametern für den entfernten Nachrichtentyp

Die Nachrichten-Steuerungsparameter werden in der Datenbank aufbewahrt. Führen Sie zum Setzen des Steuerungsparameters eine SET REMOTE OPTION-Anweisung aus. Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Beispiel: Die folgende Anweisung stellt den FTP-Host für die FTP-Verbindung des Benutzers myuser auf *ftp.mycompany.com* ein:

```
SET REMOTE FTP OPTION myuser.host = 'ftp.mycompany.com';
```

Um die Nachrichtenverbindungsparameter anzuzeigen (mit SQL), führen Sie eine Abfrage der SYSREMOTEOPTION-Systemansicht durch:

```
SELECT * from SYSREMOTEOPTION;
```

Auf der Festplatte gespeicherte Nachrichtenverbindungsparameter

Frühere Versionen von SQL Remote haben die Nachrichtenverbindungsparameter außerhalb der Datenbank gespeichert. Die externe Speicherung von Nachrichtenverbindungsparametern wird nicht empfohlen.

Die Steuerungsparameter der Nachrichtenverbindung werden an folgenden Standorten gespeichert:

- **Windows** In der Registrierung an der folgenden Position:

  ```
  \\HKEY_CURRENT_USER
     \Software
        \Sybase
           \SQL Remote
  ```

  Die Parameter für jede Nachrichtenverbindung werden in einem Schlüssel unter dem SQL Remote-Schlüssel gespeichert, zusammen mit dem Namen der Nachrichtenverbindung (z.B. 4, *SMTP*).

- **Unix** Die FILE-Systemverzeichnis-Einstellung wird in der SQLREMOTE-Umgebungsvariablen gespeichert.

Die SQL Remote-Umgebungsvariable speichert einen Pfad, der als Alternative zu einem der Steuerungsparameter für das FILE-Nachrichtensystem verwendet werden kann.

Wenn der SQL Remote-Nachrichtenagent (dbremote) eine Nachrichtenverbindung lädt, verwendet die Verbindung die Einstellung des aktuellen Publikationseigentümers, oder, wenn keine Einstellung angegeben ist, jene der Gruppen, zu denen der Publikationseigentümer gehört.

Wenn unter Windows der SQL Remote-Nachrichtenagent (dbremote), der das Speichern von Nachrichtenverbindungsparametern in der Datenbank unterstützt, das erste Mal ausgeführt wird, kopiert er die Verbindungsoptionen aus der Registrierung in die Datenbank.

Siehe auch

- „SET REMOTE OPTION-Anweisung [SQL Remote]“ auf Seite 238
- „SYSREMOTEOPTION-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Das FILE-Nachrichtensystem

SQL Remote können Sie auch einsetzen, wenn Sie kein E-Mail- oder FTP-System eingerichtet haben, indem Sie das FILE-Nachrichtensystem verwenden.

Adressen im FILE-Nachrichtensystem

Das FILE-Nachrichtensystem ist ein einfaches System der gemeinsamen Dateibenutzung. Eine FILE-Adresse für einen entfernten Benutzer ist ein Unterverzeichnis, in das alle seine Nachrichten geschrieben werden. Um die Nachrichten abzurufen, liest eine Anwendung die Nachrichten aus dem Verzeichnis, das die Dateien des Benutzers enthält. Rücksendenachrichten werden an die Adresse der konsolidierten Datenbank versendet (in ihr Verzeichnis geschrieben).

Wenn der SQL Remote-Nachrichtenagent (dbremote) als Dienst läuft, muss das Konto, unter dem er läuft, Lese- und Schreibberechtigungen für alle erforderlichen Verzeichnisse besitzen. Wenn die korrekten Berechtigungen nicht zugewiesen werden, kann der SQL Remote-Nachrichtenagent nicht auf Netzlaufwerke zugreifen.

Stammverzeichnisse für Adressen

Die FILE-Nachrichtensystemadressen sind üblicherweise Unterverzeichnisse eines gemeinsam genutzten Verzeichnisses, das allen SQL Remote-Benutzern zur Verfügung steht, entweder durch Modem oder auf einem lokalen Netzwerk (LAN). Jeder Benutzer sollte einen Registrierungseintrag, einen Eintrag in der

Initialisierungsdatei oder die SQLREMOTE-Umgebungsvariable aufweisen, die auf das gemeinsam genutzte Verzeichnis verweist.

Sie können das FILE-System auch verwenden, um die Nachrichten in Verzeichnisse auf dem konsolidierten Computer oder den entfernten Computern zu platzieren. Sie können ein einfaches Dateiübertragungsverfahren verwenden, um Dateien auszutauschen und die Replikation durchzuführen.

Steuerungsparameter für FILE-Nachrichten

Das FILE-Nachrichtensystem verwendet die folgenden Steuerungsparameter, die mithilfe der SET REMOTE OPTION-Anweisung gesetzt werden:

- **directory** Das Verzeichnis, in dem die Nachrichten gespeichert werden. Dieser Parameter ist eine Alternative zur SQLREMOTE-Umgebungsvariablen.
- **debug** Die Einstellung für diesen Parameter ist entweder YES oder NO. Der Standardwert ist NO. Bei der Einstellung YES werden alle FILE-Systemaufrufe der FILE-Verbindung im Ausgabelog angezeigt.
- **encode_dll** Wenn Sie ein benutzerdefiniertes Kodierungsschema verwenden, müssen Sie diesen Parameter auf den vollständigen Pfad der von Ihnen erstellten angepassten Kodierungs-DLL einstellen.
- **invalid_extensions** Eine durch Kommas getrennte Liste von Dateierweiterungen, die vom SQL Remote-Nachrichtenagenten (dbremote) beim Generieren von Dateien im Nachrichtensystem nicht verwendet werden sollen.
- **max_retries** Standardmäßig gilt: Wenn SQL Remote im kontinuierlichen Modus ausgeführt wird und beim Zugreifen auf das Nachrichtensystem ein Fehler auftritt, wird es nach Abschluss der Sende- und/oder Empfangsphase automatisch heruntergefahren. Verwenden Sie diesen Parameter, um festzulegen, wie oft SQL Remote die Sende- und/oder Empfangsphase wiederholen soll, bevor es beendet wird.
- **pause_after_failure** Dieser Parameter gilt, wenn der max_retries-Parameter auf einen anderen Wert als Null gesetzt ist, und SQL Remote im kontinuierlichen Modus ausgeführt wird. Wenn ein Fehler im Nachrichtensystem auftritt, definiert dieser Parameter die Anzahl der Sekunden, die SQL Remote wartet, bis die Sende- und/oder Empfangsphasen wiederholt werden.
- **unlink_delay** Die Sekunden, die abgewartet werden sollen, bevor eine Datei gelöscht wird, falls der vorherige Versuch, die Datei zu löschen, fehlgeschlagen ist. Wenn für unlink_delay kein Wert definiert ist, wird als Standardverhalten 1 Sekunde nach dem ersten fehlgeschlagenen Versuch, 2 Sekunden nach dem zweiten, 3 Sekunden nach dem dritten und 4 Sekunden nach dem vierten gewartet.

Windows Mobile und Microsoft ActiveSync

Der Nachrichtenagent (dbremote) sucht im Verzeichnis *C:\Eigene Dateien\Synchronisierte Dateien* nach der FILE-Verbindung. Auf dem Computer der konsolidierten Datenbank muss die SQLREMOTE-Umgebungsvariable oder der Verzeichnis-Nachrichtenverbindungsparameter für die FILE-Verbindung auf das folgende Verzeichnis festgelegt werden, wobei *userdi* und *Windows-mobile-device-name* auf die entsprechenden Werte gesetzt sind:

```
%SystemRoot%\Profiles\userid\Personal\Windows-mobile-device-name
\Synchronized Files
```

Bei diesem System synchronisiert Microsoft ActiveSync automatisch die Nachrichtendateien zwischen dem Computer der konsolidierten Datenbank und dem Windows Mobile-Gerät.

Um zu überprüfen, ob die FILE-Synchronisation aktiviert ist, gehen Sie zu **Geräte** » **Extras** » **ActiveSync-Optionen**.

Siehe auch

- „Das FTP-Nachrichtensystem“ auf Seite 121
- „Das SMTP-Nachrichtensystem“ auf Seite 128
- „Das HTTP-Nachrichtensystem“ auf Seite 124
- „CREATE REMOTE [MESSAGE] TYPE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SET REMOTE OPTION-Anweisung [SQL Remote]“ auf Seite 238
- „SQLREMOTE-Umgebungsvariable“ [*SQL Anywhere Server - Datenbankadministration*]
- „Nachrichtengröße“ auf Seite 113

Das FTP-Nachrichtensystem

Im FTP-Nachrichtensystem werden Nachrichten in Verzeichnissen unter einem Stammverzeichnis auf einem FTP-Host gespeichert. Der FTP-Host und das Stammverzeichnis werden durch Nachrichtensystem-Steuerungsparameter bestimmt, die in der Registrierung oder Initialisierungsdatei verzeichnet sind und bei denen die Adresse der einzelnen Benutzer das Unterverzeichnis ist, in dem ihre Nachrichten aufbewahrt werden.

Steuerungsparameter für FTP-Nachrichten

Das FTP-Nachrichtensystem verwendet die folgenden Steuerungsparameter, die mithilfe der SET REMOTE OPTION-Anweisung gesetzt werden:

- **host** Der Hostname des Computers, auf dem der FTP-Server läuft. Dieser Parameter kann ein Hostname (z.B. FTP.ianywhere.com) oder eine IP-Adresse (z.B. 192.138.151.66) sein.
- **user** Der Benutzername für den Zugriff auf den FTP-Host
- **password** Das Kennwort für den Zugriff auf den FTP-Host
- **root_directory** Das Stammverzeichnis auf der FTP-Host-Site, unter dem die Nachrichten gespeichert werden.
- **port** Die IP-Portnummer, die für die FTP-Verbindung verwendet wird. Dieser Parameter wird gewöhnlich nicht benötigt.
- **debug** Dieser Parameter ist entweder auf YES oder NO gesetzt. Der Standardwert ist NO. Bei der Einstellung YES werden Debug-Informationen im Ausgabelog angezeigt.
- **active_mode** Dieser Parameter steuert, wie SQL Remote die Client-/Server-Verbindung herstellt. Dieser Parameter ist entweder auf YES oder NO gesetzt. Der Standardwert ist NO (passiver Modus).

Der passive Modus ist der bevorzugte Übermittlungsmodus und der Standardwert für die FTP-Nachrichtenverbindung. Im passiven Modus werden alle Datenübertragungsverbindungen durch den Client initiiert, in diesem Fall durch die Nachrichtenverbindung. Im aktiven Modus initiiert der FTP-Server alle Datenverbindungen.

- **reconnect_retries** Gibt an, wie häufig die Verbindung versuchen soll, einen Socket zu öffnen, bevor der Versuch abgebrochen wird. Der Standardwert ist "4". Wenn Sie diesen Parameter setzen, sind nur erneute Verbindungen betroffen. Die ursprünglich vom FTP-Link erstellte Verbindung ist nicht betroffen.
- **reconnect_pause** Die Zeit in Sekunden, die zwischen den Verbindungsversuchen abgewartet wird. Der Standardwert ist 30 Sekunden. Wenn Sie diesen Parameter setzen, sind nur erneute Verbindungen betroffen. Die ursprünglich vom FTP-Link erstellte Verbindung ist nicht betroffen.
- **suppress_dialogs** Dieser Parameter ist auf ON oder OFF gesetzt. Wenn dieser Parameter auf ON gesetzt ist, wird das Fenster **Verbinden** nach gescheiterten Verbindungsversuchen zum FTP-Server nicht angezeigt. Stattdessen wird eine Fehlermeldung ausgegeben.
- **invalid_extensions** Eine durch Kommas getrennte Liste von Dateierweiterungen, die von dbremote beim Generieren von Dateien im Nachrichtensystem nicht verwendet werden sollen.
- **encode_dll** Wenn Sie ein benutzerdefiniertes Kodierungsschema implementiert haben, müssen Sie diesen Parameter auf den vollständigen Pfad der von Ihnen erstellten angepassten Kodierungs-DLL einstellen.
- **max_retries** Standardmäßig gilt: Wenn SQL Remote im kontinuierlichen Modus ausgeführt wird und beim Zugreifen auf das Nachrichtensystem ein Fehler auftritt, wird es nach Abschluss der Sende- und/oder Empfangsphase automatisch heruntergefahren. Verwenden Sie diesen Parameter, um festzulegen, wie oft SQL Remote die Sende- und/oder Empfangsphase wiederholen soll, bevor es beendet wird.
- **pause_after_failure** Dieser Parameter gilt, wenn der max_retries-Parameter auf einen anderen Wert als Null gesetzt ist, und SQL Remote im kontinuierlichen Modus ausgeführt wird. Wenn ein Fehler im Nachrichtensystem auftritt, definiert dieser Parameter die Anzahl der Sekunden, die SQL Remote wartet, bis die Sende- und/oder Empfangsphasen wiederholt werden.

Siehe auch

- „Unterstützte Plattformen“ [*SQL Anywhere 16 - Einführung*]
- „Nachrichtengröße“ auf Seite 113
- „Das FILE-Nachrichtensystem“ auf Seite 119
- „Das SMTP-Nachrichtensystem“ auf Seite 128
- „CREATE REMOTE [MESSAGE] TYPE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SET REMOTE OPTION-Anweisung [SQL Remote]“ auf Seite 238

Fehlerbehandlung bei FTP

Die meisten Probleme bei FTP-Nachrichtenverbindungen treten im Bereich des Netzwerksystems auf. Dieser Abschnitt enthält eine Liste von Tests, die Sie durchführen können, um Probleme zu beheben.

- **Den Steuerungsparameter für DEBUG-Nachrichten einstellen** Überprüfen Sie die Debug-Ausgaben, um zu ermitteln, ob Sie mit dem FTP-Server verbunden sind. Wenn Sie verbunden sind, sollten die Debug-Ausgaben angeben, welche FTP-Befehle fehlgeschlagen sind.

- **Ping zum FTP-Server** Wenn die FTP-Verbindung nicht in der Lage ist, den Client mit dem FTP-Server zu verbinden, überprüfen Sie Ihre System-Netzwerkkonfiguration. Führen Sie zum Beispiel folgenden Befehl aus:

  ```
  ping FTP-server-name
  ```

  Die IP-Adresse des FTP-Servers und die Ping-Zeit (Roundtrip) zum FTP-Server sollten zurückgegeben werden. Wenn Sie kein Ping zum FTP-Server senden können, haben Sie ein Netzwerk-Konfigurationsproblem und sollten sich an Ihren Netzwerkadministrator wenden.

- **Überprüfen, ob der passive Modus funktioniert** Wenn die FTP-Verbindung mit dem Server hergestellt ist, aber keine Datenverbindung öffnen kann, stellen Sie sicher, dass ein FTP-Client den passiven Modus verwenden kann, um Daten an den Server zu übermitteln.

  Der passive Modus ist der bevorzugte Übermittlungsmodus und der Standardwert für die FTP-Nachrichtenverbindung. Im passiven Modus werden alle Datenübertragungsverbindungen durch den Client initiiert, in diesem Fall durch die Nachrichtenverbindung. Im aktiven Modus initiiert der FTP-Server alle Datenverbindungen. Wenn sich Ihr FTP-Server hinter einer fehlerhaft eingestellten Firewall befindet, sind Sie möglicherweise nicht in der Lage, den Standardmodus der passiven Übertragung zu verwenden, weil die Firewall Socket-Verbindungen zum FTP-Server bei anderen Ports als dem FTP-Steuerungsport verhindert.

  Bei Verwendung eines FTP-Benutzerprogramms, mit dem Sie den Übertragungsmodus auf **aktiv** oder **passiv** einstellen können, stellen Sie den Übertragungsmodus auf passiv und versuchen einen Download oder Upload einer Datei. Wenn der Client, den Sie verwenden, nicht in der Lage ist, die Datei zu übertragen, ohne im aktiven Modus zu sein, dann sollten Sie Firewall und FTP-Server umkonfigurieren, um Übertragungen im passiven Modus zuzulassen, oder den active_mode-Nachrichten-Steuerungsparameter auf YES einstellen. Übertragungen im aktiven Modus funktionieren möglicherweise nicht in allen Netzwerkkonfigurationen. Beispiel: Wenn sich Ihr Client hinter einem IP-Masquerading-Gateway befindet, können ankommende Verbindungen je nach der verwendeten Gateway-Software möglicherweise scheitern.

- **Berechtigungen und Verzeichnisstrukturen überprüfen** Wenn die Verbindung zum FTP-Server funktioniert, aber keine Verzeichnisauflistungen eingelesen oder Dateien geändert werden können, überprüfen Sie Ihre Berechtigungen, um sicherzustellen, dass Sie die erforderlichen Berechtigungen haben und die benötigten Verzeichnisse vorhanden sind.

  Melden Sie sich mithilfe eines FTP-Programmes beim FTP-Server an. Wechseln Sie zu den Verzeichnissen, die im root_directory-Parameter gespeichert sind. Wenn die von Ihnen benötigten Verzeichnisse nicht angezeigt werden, ist der root_directory-Steuerungsparameter möglicherweise nicht korrekt eingestellt oder die Verzeichnisse existieren nicht.

  Sie testen Berechtigungen, indem Sie eine Datei in Ihrem Nachrichtenverzeichnis abrufen und den Upload einer Datei in das Verzeichnis der konsolidierten Datenbank durchführen. Wenn Fehler zurückgegeben werden, sind Ihre FTP-Serverberechtigungen nicht korrekt eingerichtet.

Das HTTP-Nachrichtensystem

Unter Verwendung des HTTP-Nachrichtensystems sendet SQL Remote Nachrichten über das Internet unter Verwendung des Hypertext Transfer Protocol (HTTP). Die Nachrichten werden in einem Textformat kodiert und über HTTP an die Zieldatenbank gesendet. Die Nachrichten werden mit einer als HTTP-Server fungierenden SQL Anywhere-Datenbank gesendet und empfangen.

Einrichten des HTTP-Nachrichtenservers

Sie verwenden den SQL Anywhere-Datenbankserver als den HTTP-Server, der SQL Remote-Nachrichten zu und von entfernten Datenbanken überträgt. Standardmäßig sind für eine neu initialisierte SQL Anywhere-Datenbank nicht die Webdienste festgelegt, mit denen sie als Nachrichtenserver fungieren könnte. Drei gespeicherte Systemprozeduren, sr_add_message_server, sr_drop_message_server und sr_update_message_server, sind in neu erstellten SQL Anywhere-Datenbanken festgelegt und ermöglichen es Ihnen, die erforderlichen Datenbankobjekte so zu definieren, dass die Datenbank als HTTP-Server zum Übertragen von SQL Remote-Nachrichten eingesetzt werden kann.

Hinweis
Die Datenbank muss mit SQL Anywhere 16.0 und mit einem Datenbankserver mit Build-Nummer 3336 oder höher initialisiert werden. Führen Sie eine Abfrage der SYS.SYSHISTORY-Systemtabelle aus, um zu ermitteln, welche Version und welcher Build des Datenbankservers zum Initialisieren der Datenbank verwendet wurden. Wenn die Datenbank mit 16.0 und einer Build-Nummer vor 3336 initialisiert wurde, aktualisieren Sie die Datenbank, indem Sie "ALTER DATABASE UPGRADE PROCEDURE ON" ausführen.

Sie müssen entscheiden, ob Sie einen separaten Datenbankserver ausführen oder die vorhandene konsolidierte Datenbank als Nachrichtenserver einsetzen möchten. Berücksichtigen Sie Folgendes, wenn Sie diese Entscheidung treffen:

1. Wenn eine entfernte Datenbank sich beim Nachrichtenserver authentifiziert, verwendet sie für die Authentifizierung den Publikationseigentümer der entfernten Datenbank und das angegebene Kennwort. Obwohl der Benutzer in der konsolidierten Datenbank vorhanden ist, wurde für ihn möglicherweise kein Kennwort festgelegt (d.h., der entfernte Benutzer hat möglicherweise keine CONNECT-Privilegien), was jedoch eine Anforderung des HTTP-Nachrichtensystems ist. Wenn das Erteilen von CONNECT-Privilegien an die entfernten Benutzer in der konsolidierten Datenbank ein Sicherheitsproblem darstellt, können Sie eine separate Datenbank als Nachrichtenserver einrichten.

2. Wenn die konsolidierte Datenbank eine hohe Last aufweist, kann das Hinzufügen der Nachrichtenserver-Funktion dazu führen, dass die Ressourcen auf dem Computer überlastet werden, wenn die konsolidierte Datenbank läuft.

Um die erforderlichen Datenbankobjekte einzurichten, damit die Datenbank als Nachrichtenserver fungieren kann, rufen Sie die gespeicherte Prozedur sr_add_message_server auf, mit der die SQL Remote-Definitionen in der Datenbank abgefragt werden. Siehe „sr_add_message_server-Systemprozedur“ auf Seite 228.

Wenn Sie den Nachrichtenserver als separate Datenbank erstellen, müssen Sie eine zweite Datenbank mit SQL Remote-Definitionen festlegen, die mit denen der konsolidierten Datenbank übereinstimmen. Verwenden Sie das Dienstprogramm dbunload, um eine Kopie der konsolidierten Datenbank zu erstellen,

und geben Sie die Option -n an, damit nur das Schema der konsolidierten Datenbank entladen wird und nicht die Daten:

```
dbunload -n -an -c "ENG=cons.DBN=cons;UID=DBA;PWD=sql"
```

Wenn Sie eine separate Datenbank als Nachrichtenserver verwenden und Änderungen an die SQL Remote-Definitionen in der konsolidierten Datenbank vorgenommen werden, müssen entsprechende Änderungen auch in der Nachrichtenserver-Datenbank vorgenommen werden.

Damit der Nachrichtenserver eingerichtet werden kann, muss der Datenbankserver Zugriff auf das Verzeichnis haben, in dem SQL Remote-Nachrichten gespeichert werden. Um festzulegen, in welchem Verzeichnis Nachrichten gespeichert werden, führen Sie den SET REMOTE OPTION-Befehl aus und setzen den HTTP-Nachrichtenparameter **root_directory** auf das Verzeichnis, in dem SQL Remote-Nachrichten gespeichert werden. Wählen Sie als nächstes den Datenbankbenutzer, der Eigentümer der zu erstellenden neuen Objekte sein soll, und stellen Sie sicher, dass der Benutzer eine Rolle ist. Führen Sie abschließend die gespeicherte Prozedur sr_add_message_server aus und übergeben Sie dabei den Namen des Benutzers, der Eigentümer der Objekte sein soll. Siehe „SET REMOTE OPTION-Anweisung [SQL Remote]“ auf Seite 238 und „sr_add_message_server-Systemprozedur“ auf Seite 228.

Bei Änderungen an der SQL Remote-Definition des Nachrichtenservers (wie z.B. Hinzufügen oder Entfernen von entfernten Benutzern) müssen Sie die gespeicherte Prozedur sr_update_message_system ausführen, um die Definition der erforderlichen Objekte für die Unterstützung des Nachrichtenservers zu aktualisieren. Der Nachrichtenserver ist für eine kurze Zeitspanne nicht für die Replikation verfügbar, während die gespeicherte Prozedur läuft und Objekte gelöscht und neu erstellt werden. Siehe „sr_update_message_server-Systemprozedur“ auf Seite 229.

Wenn Sie die Datenbank nicht mehr als Nachrichtenserver verwenden, können Sie die gespeicherte Prozedur sr_drop_message_system ausführen, um die Objekte zu entfernen, die zur Unterstützung des Nachrichtenservers erstellt wurden. Siehe „sr_drop_message_server-Systemprozedur“ auf Seite 229.

Starten des Nachrichtenserver-Datenbankservers

Nachdem die erforderlichen Objekte für die Unterstützung des Nachrichtenservers erstellt wurden, müssen Sie beim Starten des Nachrichtenserver-Datenbankservers die HTTP-Unterstützung (bzw. HTTPS-Unterstützung) für den Datenbankserver mithilfe der Option -xs aktivieren. Weitere Hinweise zur Verwendung der Option -xs finden Sie unter „Datenbankserveroption -xs “ [*SQL Anywhere Server - Datenbankadministration*].

Die folgenden serverseitigen HTTP Protokolloptionen sind für die Benutzer von Interesse, die die für den Nachrichtenserver benötigten Objekte festgelegt haben:

- **ServerPort | PORT** Gibt die Portnummer an, die der Datenbankserver auf HTTP- oder HTTPS-Anforderungen abhört, wenn die Standardports 80 und 443 auf dem Computer bereits verwendet werden. Siehe „ServerPort-Protokolloption (PORT)“ [*SQL Anywhere Server - Datenbankadministration*].

- **MaxRequestSize | MAXSIZE** Gibt die maximale Größe einer einzelnen HTTP-Anforderung an. Der Standardwert ist 100 kB. Wenn Sie Ihre SQL Remote-Nachrichtengröße (Option -l in der dbremote-Befehlszeile) auf einen Wert festgelegt haben, der größer ist als 100 kB, müssen Sie auch die Größe der größten HTTP-Anforderung erhöhen, die der Datenbankserver akzeptieren kann. Der

Standardwert für die SQL Remote-Nachrichtengröße ist 50 kB. Siehe „MaxRequestSize-Protokolloption (MAXSIZE)“ [*SQL Anywhere Server - Datenbankadministration*].

- **Identity (nur HTTPS)** Wenn Sie HTTPS verwenden, enthält die Identitätsdatei das öffentliche Zertifikat und seinen privaten Schlüssel. Bei nicht selbstsignierten Zertifikaten enthält die Identitätsdatei außerdem alle signierenden Zertifikate, u.a. auch das Verschlüsselungszertifikat. Das Kennwort für dieses Zertifikat muss mit dem Parameter **Identity_Password** festgelegt werden. Siehe „Identity-Protokolloption“ [*SQL Anywhere Server - Datenbankadministration*].

- **Identity_Password (nur HTTPS)** Wenn Sie Transportschichtsicherheit verwenden, gibt diese Option das Kennwort an, das dem Kennwort für das Verschlüsselungszertifikat entspricht, das durch die Identity-Protokolloption festgelegt wurde. Siehe „identity_password-Protokolloption“ [*SQL Anywhere Server - Datenbankadministration*].

HTTP-Adressen und Benutzer-IDs

Damit Sie SQL Remote und HTTP verwenden können, benötigt jede am System beteiligte Datenbank eine HTTP-Adresse sowie eine Benutzer-ID und ein Kennwort. Das sind unterschiedliche Bezeichner: Die HTTP-Adresse ist das Ziel der einzelnen Nachrichten. Die Benutzer-ID und das Kennwort sind der Name und das Kennwort, die ein Benutzer eingibt, wenn er sich beim E-Mail-Server authentifiziert.

Steuerungsparameter für HTTP-Nachrichten

Bevor der SQL Remote-Nachrichtenagent (dbremote) zum Nachrichtensystem eine Verbindung herstellt, um Nachrichten zu versenden oder zu empfangen, muss der Benutzer bereits einen Satz von Steuerungsparametern auf seinem Computer festgelegt haben, oder der Benutzer wird aufgefordert, die erforderlichen Informationen anzugeben. Diese Daten werden nur bei der ersten Verbindung benötigt. Sie werden gespeichert und bei späteren Verbindungen als Standardwerte verwendet.

Das HTTP-Nachrichtensystem verwendet die folgenden Steuerungsparameter, die mithilfe der SET REMOTE OPTION-Anweisung gesetzt werden:

- **certificate** Um eine sichere (HTTPS-)Anforderung durchführen zu können, muss der Client Zugriff auf das vom HTTPS-Server verwendete Zertifikat haben. Die benötigten Informationen werden in einer Zeichenfolge von durch Semikola getrennten Schlüssel/Werte-Paaren angegeben. Sie können mithilfe des Dateischlüssels den Dateinamen für das Zertifikat angeben. Sie können nicht sowohl einen Dateischlüssel als auch einen Zertifikatschlüssel angeben. Folgende Schlüssel sind verfügbar:

Schlüssel	Abkürzung	Beschreibung
file		Der Dateiname des Zertifikats
certificate	cert	Das Zertifikat selbst
company	co	Die im Zertifikat angegebene Firma
unit		Die im Zertifikat angegebene Firmeneinheit
name		Der im Zertifikat angegebene allgemeine Name

Zertifikate sind nur bei Anforderungen erforderlich, die entweder an den HTTPS-Server gerichtet sind oder von einem nicht-sicheren zu einem sicheren Server umgeleitet werden können. Nur PEM-formatierte Zertifikate werden unterstützt. **certificate=**'*file=filename*'

- **client_port** Kennzeichnet die Portnummer, auf der SQL Remote unter Verwendung von HTTP kommuniziert. Sie wird nur für Verbindungen über Firewalls bereitgestellt und empfohlen, die "ausgehende" TCP/IP-Verbindungen filtern. Sie können eine einzelne Portnummer, Bereiche von Portnummern oder eine Kombination aus beiden angeben. Die Angabe einer niedrigen Anzahl von Clientports kann dazu führen, dass SQL Remote nicht in der Lage ist, Nachrichten zu senden und zu empfangen, wenn das Betriebssystem die Ports nicht rechtzeitig freigibt, nachdem SQL Remote den Port bei einer früheren Ausführung geschlossen hat. **client_port=***nnnnn[-mmmmm]*'

- **debug** Wenn dieser Parameter auf YES eingestellt ist, werden alle HTTP-Befehle und -Antworten im Ausgabelog angezeigt. Diese Informationen können zur Fehlerbehandlung von HTTP-Problemen verwendet werden. Der Standardwert ist NO.

- **https** Geben Sie an, ob HTTPS (**https=yes**) oder HTTP (**https=no**) verwendet werden soll.

- **password** Das Kennwort für die Nachrichtenserver-Datenbank. Dient zum Authentifizieren bei HTTP-Servern und Gateways anderer Hersteller, die RFC 2617 Basic-Authentifizierung verwenden. **password=**'*password*'

- **proxy_host** Gibt den URI eines Proxyservers an. Wird verwendet, wenn SQL Remote über einen Proxyserver auf das Netzwerk zugreifen muss. Zeigt an, dass SQL Remote eine Verbindung zum Proxyserver herstellen soll, um die Anforderung durch ihn an den Nachrichtenserver zu senden.**proxy_host='** *http://proxy-server***[:***port-number***]'**

- **reconnect_retries** Gibt an, wie häufig die Verbindung versuchen soll, einen Socket zu öffnen, bevor der Versuch abgebrochen wird. Der Standardwert ist "4". Wenn Sie diesen Parameter setzen, sind nur erneute Verbindungen betroffen. Die ursprünglich vom FTP-Link erstellte Verbindung ist nicht betroffen.

- **reconnect_pause** Die Zeit in Sekunden, die zwischen den Verbindungsversuchen abgewartet wird. Der Standardwert ist 30 Sekunden. Wenn Sie diesen Parameter setzen, sind nur erneute Verbindungen betroffen. Die ursprünglich vom FTP-Link erstellte Verbindung ist nicht betroffen.

- **root_directory** Dieser HTTP-Steuerungsparameter wird ignoriert, wenn er auf Clientseite festgelegt wurde. Sie müssen diesen Steuerungsparameter im Nachrichtenserver festlegen, bevor Sie die gespeicherte Prozedur sr_add_message_server oder sr_update_message_server aufrufen. Geben Sie das für den Nachrichtenserver zugängliche Verzeichnis an, in dem die SQL Remote-Nachrichten gespeichert werden. Bei der Verwendung des HTTP-Nachrichtensystems kann die für einen entfernten Benutzer oder Publikationseigentümer angegebene Adresse nur ein einziges Unterverzeichnis enthalten und nicht mehrere Unterverzeichnisse. **root_directory=**'*c:\msgs*'

- **url** Geben Sie den Servernamen oder die IP-Adresse sowie optional die Portnummer für den verwendeten HTTP-Server an, durch Semikola getrennt. Wenn Anforderungen durch den Relay Server übergeben werden, können Sie optional eine URL-Erweiterung hinzufügen, um anzuzeigen, an welche Serverfarm die Anforderung übergeben werden soll. **url ='***server-name***[:***port-number***][***url-extension***]'**

- **user** Die Benutzer-ID für die Nachrichtenserver-Datenbank. Dient zum Authentifizieren bei HTTP-Servern und Gateways anderer Hersteller, die RFC 2617 Basic-Authentifizierung verwenden. **user=**'*userid*'

Siehe auch

- „Das FILE-Nachrichtensystem“ auf Seite 119
- „Das SMTP-Nachrichtensystem“ auf Seite 128
- „Das FTP-Nachrichtensystem“ auf Seite 121
- „CREATE REMOTE [MESSAGE] TYPE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Praktische Einführung: Einrichten eines Replikationssystems unter Verwendung des HTTP-Nachrichtensystems“ auf Seite 171
- „Praktische Einführung: Einrichten eines Replikationssystems unter Verwendung des HTTP-Nachrichtensystems mit der konsolidierten Datenbank als Nachrichtenserver“ auf Seite 181
- „Praktische Einführung: Einrichten eines Replikationssystems unter Verwendung des HTTP-Nachrichtensystems und mit der konsolidierten Datenbank als Nachrichtenserver über Relay Server“ auf Seite 191
- „SET REMOTE OPTION-Anweisung [SQL Remote]“ auf Seite 238

Das SMTP-Nachrichtensystem

Beim SMTP-System verwendet SQL Remote Internet-Mail, um Nachrichten zu versenden. Die Nachrichten werden in einem Textformat kodiert und in einer E-Mail an die Zieldatenbank versendet. Die Nachrichten werden mittels eines SMTP-Servers versendet und von einem POP3-Server abgerufen.

Eine Liste der Betriebssysteme, für die SMTP unterstützt wird, finden Sie unter „Unterstützte Plattformen“ [*SQL Anywhere 16 - Einführung*].

SMTP-Adressen und Benutzer-IDs

Um SQL Remote und ein SMTP-Nachrichtensystem verwenden zu können, benötigt jede am System beteiligte Datenbank eine SMTP-Adresse sowie eine POP3-Benutzer-ID und ein Kennwort. Das sind unterschiedliche Bezeichner: Die SMTP-Adresse ist das Ziel der einzelnen Nachrichten. Die POP3-Benutzer-ID und das POP3-Kennwort sind der Name und das Kennwort, die von einem Benutzer eingegeben werden, wenn er sich mit seinem E-Mail-Server verbindet.

Hinweis
Es wird empfohlen, dass Sie ein separates POP-E-Mail-Konto verwenden, um SQL Remote-Nachrichten zu versenden und zu empfangen. Siehe „Gemeinsame Nutzung von SMTP/POP-Adressen“ auf Seite 130.

Steuerungsparameter für SMTP-Nachrichten

Bevor der SQL Remote-Nachrichtenagent (dbremote) zum Nachrichtensystem eine Verbindung herstellt, um Nachrichten zu versenden oder zu empfangen, muss der Benutzer bereits einen Satz von Steuerungsparametern auf seinem Computer festgelegt haben, oder der Benutzer wird aufgefordert, die erforderlichen Informationen anzugeben. Diese Daten werden nur bei der ersten Verbindung benötigt. Sie werden gespeichert und als Standardwerte bei späteren Verbindungen verwendet.

Das SMTP-Nachrichtensystem verwendet die folgenden Steuerungsparameter, die mithilfe der SET REMOTE OPTION-Anweisung gesetzt werden:

- **local_host** Der Name des lokalen Computers. Dies ist auf Computern nützlich, bei denen SQL Remote nicht in der Lage ist, den lokalen Hostnamen zu bestimmen. Der lokale Hostname wird benötigt, um eine Sitzung mit einem SMTP-Server zu initialisieren. In den meisten Netzwerkumgebungen kann der Hostname automatisch ermittelt werden. In diesem Fall wird dieser Eintrag nicht gebraucht.

- **top_supported** SQL Remote verwendet einen POP3-Befehl namens TOP, um eintreffende Nachrichten zu verarbeiten. Der TOP-Befehl wird möglicherweise nicht von allen POP3-Servern unterstützt. Wenn Sie den Parameter top_supported auf NO setzen, verwendet SQL Remote den RETR-Befehl, der weniger effizient ist, aber mit allen POP-Servern funktioniert. Der Standardwert ist YES.

- **smtp_authenticate** Legt fest, ob die SMTP-Verbindung den Benutzer authentifiziert. Der Standardwert ist YES. Setzen Sie diesen Parameter auf NO, um die SMTP-Authentifizierung zu deaktivieren.

- **smtp_userid** Die Benutzer-ID für die SMTP-Authentifizierung. Standardmäßig übernimmt dieser Parameter denselben Wert wie der Parameter pop3_userid. Der Parameter smtp_userid muss nur gesetzt werden, wenn die Benutzer-ID von der des POP-Servers abweicht.

- **smtp_password** Das Kennwort für die SMTP-Authentifizierung. Standardmäßig übernimmt dieser Parameter denselben Wert wie der Parameter pop3_password. Der Parameter smtp_password muss nur gesetzt werden, wenn die Benutzer-ID von der des POP-Servers abweicht.

- **smtp_host** Der Name des Computers, auf dem der SMTP-Server läuft. Dies entspricht dem SMTP-Hostfeld im SMTP/POP3-Login-Fenster.

- **smtp_port** Der Nummer des Ports, den der SMTP-Server überwacht. Standardwert ist "25".

- **pop3_host** Der Name des Computers, auf dem der POP3-Server läuft. Üblicherweise ist dies derselbe Name wie der SMTP-Host:. Dies entspricht dem POP3-Hostfeld im SMTP/POP3-Login-Fenster.

- **pop3_userid** Die Benutzer-ID, die zum Abrufen der Mail verwendet wird. Die POP3-Benutzer-ID entspricht dem Benutzer-ID-Feld im SMTP/POP3-Login-Fenster. Sie erhalten eine Benutzer-ID bei Ihrem POP3-Host-Administrator.

- **pop3_password** Das Kennwort, das zum Abrufen der Mail verwendet wird. Dies entspricht dem Kennwortfeld im SMTP/POP3-Login-Fenster.

- **pop3_port** Die Nummer des Ports, auf dem der POP3-Server auf Verbindungen wartet. Der Standardwert ist 110.

- **debug** Wenn dieser Parameter auf YES eingestellt ist, werden alle Befehle und Antworten für SMTP und POP3 angezeigt. Diese Informationen können zur Fehlerbehandlung von SMTP/POP3-Problemen verwendet werden. Der Standardwert ist NO.

- **suppress_dialogs** Dieser Parameter ist auf ON oder OFF gesetzt. Wenn dieser Parameter auf OFF gesetzt ist, wird das Fenster **Verbinden** nach gescheiterten Verbindungsversuchen zum Mail-Server nicht angezeigt. Stattdessen wird eine Fehlermeldung ausgegeben.

- **encode_dll** Wenn Sie ein benutzerdefiniertes Kodierungsschema implementiert haben, müssen Sie dies auf den vollständigen Pfad der von Ihnen erstellten angepassten Kodierungs-DLL einstellen. Siehe „Nachrichtengröße“ auf Seite 113.

- **max_retries** Standardmäßig gilt: Wenn SQL Remote im kontinuierlichen Modus ausgeführt wird und ein Fehler tritt auf, wenn auf das Nachrichtensystem zugegriffen wird, wird es automatisch nach den Sende- und/oder Empfangsphasen heruntergefahren. Verwenden Sie diesen Parameter, um die Häufigkeit festzulegen, mit der Sie aus SQL Remote zu wiederholen Sie den Sende- und/oder von der Synchronisation, bevor sie beendet.

- **pause_after_failure** Dieser Parameter gilt, wenn der max_retries-Parameter auf einen anderen Wert als Null gesetzt ist, und SQL Remote im kontinuierlichen Modus ausgeführt wird. Wenn ein Fehler im Nachrichtensystem auftritt, definiert dieser Parameter die Anzahl der Sekunden, die SQL Remote wartet, bis die Sende- und/oder Empfangsphasen wiederholt werden.

Siehe auch

- „Das FILE-Nachrichtensystem“ auf Seite 119
- „Das FTP-Nachrichtensystem“ auf Seite 121
- „CREATE REMOTE [MESSAGE] TYPE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SET REMOTE OPTION-Anweisung [SQL Remote]“ auf Seite 238

Gemeinsame Nutzung von SMTP/POP-Adressen

Sie sollten ein separates E-Mail-Konto für SQL Remote-Nachrichten verwenden. Es wird nicht empfohlen, dass Sie SQL Remote-Nachrichten über das E-Mail-Konto versenden und empfangen, das Sie für persönliche oder geschäftliche E-Mail-Nachrichten verwenden.

Wenn Sie dasselbe E-Mail-Konto für SQL Remote-Nachrichten und reguläre E-Mail-Nachrichten verwenden, müssen Sie sicherstellen, dass Ihr E-Mail-Programm nicht alle Nachrichten vom Mail-Server, einschließlich SQL Remote-E-Mail- und persönliche Nachrichten, herunterlädt und löscht. Sie müssen das E-Mail-Programm so konfigurieren, dass es keine SQL Remote-Nachrichten ändert oder löscht, wenn es Ihre regulären E-Mail-Nachrichten herunterlädt. SQL Remote-Nachrichten enthalten den Betreff ---SQL Remote--.

Fehlerbehandlung für die SMTP-Verbindung

Wenn es Ihnen nicht gelingt, die SMTP-Verbindung herzustellen, versuchen Sie, sich mit dem SMTP/POP3-Server von demselben Computer aus zu verbinden, auf dem der SQL Remote-Nachrichtenagent (dbremote) läuft, indem Sie dasselbe Konto und Kennwort verwenden. Verwenden Sie ein Internet-E-Mail-Programm, das SMTP/POP3 unterstützt. Deaktivieren Sie dieses Programm, sobald die SMTP-Nachrichtenverbindung funktioniert.

Überprüfen, ob die E-Mail korrekt funktioniert

Wenn SQL Remote-Nachrichten nicht korrekt versendet und empfangen werden und Sie ein E-Mail-Nachrichtensystem verwenden, sollten Sie überprüfen, ob die E-Mail korrekt zwischen den beiden Computern funktioniert.

SQL Remote-Systemsicherungen

Die SQL Remote-Replikation hängt vom Zugriff auf die Vorgänge im Transaktionslog und vom Zugriff auf alte Transaktionslogs ab. Eine von Ihnen implementierte Sicherungsstrategie muss die Aufrechterhaltung der Transaktionslogs für SQL Remote umfassen.

Um Sicherungsaktivitäten durchzuführen, benötigen Sie das BACKUP DATABASE-Systemprivileg.

Der SQL Remote-Nachrichtenagent (dbremote) muss solange Zugriff auf das aktuelle Transaktionslog und auf alte Transaktionslogs haben, bis sie nicht mehr benötigt werden.

Eine konsolidierte Datenbank benötigt ihre Transaktionslogs nicht mehr, wenn alle entfernten Datenbanken empfangen und bestätigt haben, dass die in den Transaktionslogs enthaltenen Nachrichten erfolgreich angewendet wurden.

Eine entfernte Datenbank benötigt ihre Transaktionslogs nicht mehr, wenn die konsolidierte Datenbank empfangen und bestätigt, dass sie die in den Transaktionslogs enthaltenen Nachrichten erfolgreich angewendet hat.

Entfernte Datenbanken sichern

Bei Ihren entfernten Datenbanken müssen Sie festlegen, ob Sie Folgendes durchführen wollen:

- **Die Replikation in die konsolidierte Datenbank als Sicherungsmethode verwenden** Sicherungsprozeduren sind in den entfernten Datenbanken weniger wichtig als in der konsolidierten Datenbank. Sie könnten sich auf die Replikation in die konsolidierte Datenbank als Methode zur Datensicherung verlassen.

  Wenn Sie diese Methode wählen, *sollten* Sie eine Strategie entwickeln, um die Transaktionslogs der entfernten Datenbanken zu verwalten.

- **Sicherungsstrategie für die entfernte Datenbank erstellen** Wenn die Änderungen, die in den entfernten Datenbanken durchgeführt werden, wichtig sind, müssen Sie eine Sicherungsstrategie für die entfernten Datenbanken erstellen, die die Verwaltung der Transaktionslogs umfasst.

Konsolidierte Datenbanken sichern

Sie *müssen* eine Sicherungsstrategie für Ihre konsolidierte Datenbank haben, die die Verwaltung der Transaktionslogs umfasst.

Das Sicherungsdienstprogramm (dbbackup) und die Befehlszeilenoption -x des SQL Remote-Nachrichtenagenten (dbremote)

Sie sollten in einer Datenbank niemals den SQL Remote-Nachrichtenagenten (dbremote) mit der Befehlszeilenoption -x und das Sicherungsdienstprogramm (dbbackup) zusammen ausführen.

Die Befehlszeilenoption -x wird verwendet, um Transaktionslogs für die Replikation zu verwalten. Die Befehlszeilenoption -x stellt sicher, dass der SQL Remote-Nachrichtenagent Zugriff auf alte Transaktionslogs hat und diese löscht, wenn sie nicht mehr benötigt werden. Die Befehlszeilenoption -x sichert nicht das Transaktionslog.

Das Sicherungsdienstprogramm (dbbackup) wird verwendet, um das aktuelle Transaktionslog zu sichern. Wenn das Sicherungsdienstprogramm (dbbackup) mit den Befehlszeilenoptionen -r und -n ausgeführt wird, sichert es das aktuelle Transaktionslog in einem Sicherungsverzeichnis, benennt das aktuelle Transaktionslog um und startet es erneut. Das Sicherungsdienstprogramm (dbbackup) nimmt an, dass das aktuelle Transaktionslog dasselbe Transaktionslog ist, das nach der vorherigen Sicherung umbenannt und erneut gestartet wurde.

Wenn Sie versuchen, sowohl die Befehlszeilenoption -x des SQL Remote-Nachrichtenagenten als auch das Sicherungsdienstprogramm (dbbackup) auszuführen, beeinträchtigen sie sich gegenseitig. Sie können Transaktionslogs verlieren, wenn beide ausgeführt werden.

Führen Sie den SQL Remote-Nachrichtenagenten (dbremote) mit der Befehlszeilenoption -x nur auf einer entfernten Datenbank aus, die nicht gesichert wird.

Siehe auch

- „Transaktionslogs für entfernte Datenbanken verwalten“ auf Seite 132
- „Eine entfernte Datenbank löschen“ auf Seite 133
- „Schutz der konsolidierten Datenbank vor Datenträgerfehlern“ auf Seite 134

Transaktionslogs für entfernte Datenbanken verwalten

Führen Sie Transaktionslogs entfernter Datenbanken, wenn Sie sich auf die Replikation der konsolidierten Datenbank verlassen, um Ihre entfernten Datenbanken zu sichern. Das heißt, dass Sie *nicht* das Sicherungsdienstprogramm (dbbackup) in den entfernten Datenbanken und Transaktionslogs ausführen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Kontext und Bemerkungen

Vorsicht
Führen Sie den SQL Remote-Nachrichtenagenten (dbremote) mit der Befehlszeilenoption -x *nicht* in einer Datenbank aus, die gesichert wird.

Aufgabe

1. In der entfernten Datenbank führen Sie den SQL Remote-Nachrichtenagenten (dbremote) mit der Befehlszeilenoption -x aus und geben eine Größe für das Transaktionslog an. Diese Option bewirkt, dass der SQL Remote-Nachrichtenagent (dbremote) das Transaktionslog umbenennt und erneut startet, wenn das Transaktionslog die angegebene Größe überschreitet.

   Mit dem folgenden Befehl wird das Transaktionslog gelöscht, wenn es größer als 1 MB ist:

```
dbremote -x 1M  -c "UID=ManagerSteve;PWD=sql;DBF=c:\mydata.db"
```

2. In der entfernten Datenbank setzen Sie die Option delete_old_logs auf On. Ein Setzen der Option delete_old_logs bewirkt, dass die alten Transaktionslogdateien automatisch von dbremote gelöscht werden, wenn sie nicht mehr für die Replikation benötigt werden.

   Ein Transaktionslog wird nicht mehr benötigt, wenn alle Subskribenten bestätigt haben, dass sie alle Änderungen, die in der Transaktionslogdatei aufgezeichnet sind, empfangen und erfolgreich angewendet haben. Sie können die delete_old_logs-Option entweder für die PUBLIC-Rolle oder nur für den einzelnen Benutzer einstellen, der in der Verbindungszeichenfolge in dbremote enthalten ist.

   Die folgende Anweisung setzt die öffentliche delete_old_logs-Option so, dass Transaktionen, die vor mehr als 10 Tagen erstellt wurden, gelöscht werden:

   ```
   SET OPTION PUBLIC.delete_old_logs = '10 days';
   ```

Ergebnisse

Die Datenbank-Transaktionslogs werden gemäß der angegebenen Regeln gelöscht.

Siehe auch

- „delete_old_logs-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]

Eine entfernte Datenbank löschen

Verwenden Sie die folgende Prozedur, um entfernte Datenbanken zu sichern.

Voraussetzungen

Sie müssen das DATABASE BACKUP-Systemprivileg haben. Diese Prozedur umfasst eine Verwaltungsstrategie für die Verwendung der Transaktionslogs durch SQL Remote. Verwenden Sie *nicht* diese Prozedur, wenn Sie den SQL Remote-Nachrichtenagenten (dbremote) mit der Befehlszeilenoption -x ausführen.

Aufgabe

1. Erstellen Sie eine vollständige Sicherung der entfernten Datenbank.
   a. Stellen Sie eine Verbindung mit der Datenbank her.
   b. Führen Sie dbbackup mit den Befehlszeilenoptionen -r und -n aus.

      Nehmen Sie beispielsweise an, das Sicherungsverzeichnis ist *e:\archive*, die Datenbankdatei befindet sich im Verzeichnis *c:\live* und die dazugehörige Transaktionslogdatei im Verzeichnis *d:\live*:

      ```
      dbbackup -r -n -c "UID=DBA;PWD=sql;DBF=c:\live\remotedatabase.db" e:
      \archive
      ```

      Die Transaktionslogs im *d:\live*-Verzeichnis werden durch die vollständige Sicherung nicht geändert.

c. Kopieren Sie die Sicherungsdateien, die sich im Verzeichnis *e:\archive* befinden, auf ein externes Laufwerk oder eine DVD.

d. Führen Sie den SQL Remote-Nachrichtenagenten (dbremote) mit Zugriff auf die aktuellen Transaktionslogdateien aus, indem Sie den folgenden Befehl verwenden:

```
dbremote -c "UID=DBA;PWD=sql;DBF=c:\live\remotedatabase.db" d:\live
```

Vorsicht
Führen Sie den SQL Remote-Nachrichtenagenten (dbremote) mit der Befehlszeilenoption -x *nicht* in einer Datenbank aus, die gesichert wird.

2. Richten Sie das Sicherungsdienstprogramm (dbbackup) so ein, dass inkrementelle Sicherungen des Transaktionslogs der entfernten Datenbank durchgeführt werden.

   a. Stellen Sie eine Verbindung mit der Datenbank her.

   b. Führen Sie dbbackup mit den Befehlszeilenoptionen -r, -n und -t aus.

   Zum Beispiel:

```
dbbackup -r -n -t -c "UID=DBA;PWD=sql;DBF=c:\live\remotedatabase.db"
e:\archive
```

   c. Führen Sie den SQL Remote-Nachrichtenagenten (dbremote) mit Zugriff auf die aktuellen Transaktionslogdateien aus, indem Sie den folgenden Befehl verwenden:

```
dbremote -c "UID=DBA;PWD=sql;DBF=c:\live\remotedatabase.db" d:\live
```

Ergebnisse

Die entfernte Datenbank wird gesichert.

Schutz der konsolidierten Datenbank vor Datenträgerfehlern

So schützen Sie Ihr SQL Remote-Replikationssystem vor Datenträgerfehlern

- **Replizieren Sie nur gesicherte Transaktionen** Versenden Sie Nachrichten, die nur gesicherte Transaktionen enthalten. Indem nur gesicherte Transaktionen versendet werden, ist das Replikationssystem vor Datenträgerfehlern im Transaktionslog geschützt. Sie erreichen dies folgendermaßen:

  - Den SQL Remote-Nachrichtenagenten (dbremote) mit der Befehlszeilenoption -u ausführen. Wenn der SQL Remote-Nachrichtenagent (dbremote) mit der Befehlszeilenoption -u ausgeführt wird, werden nur festgeschriebene Transaktionen, die gesichert wurden, in zu versendende Nachrichten verpackt.

    Die Option -u bietet zusätzlichen Schutz vor Systemausfällen, weil die Sicherungskopien an einen anderen Standort übertragen werden.

- **Einen Transaktionslog-Spiegel verwenden** Die Verwendung eines Transaktionslog-Spiegels schützt vor Datenträgerfehlern auf dem Transaktionslog-Computer.

- **Führen Sie den SQL Remote-Nachrichtenagenten (dbremote) nicht mit der Befehlszeilenoption -x in der konsolidierten Datenbank aus.** Führen Sie niemals den SQL Remote-Nachrichtenagenten (dbremote) mit der Befehlszeilenoption -x in einer Datenbank aus, die gesichert wird. Die Befehlszeilenoption -x verwaltet die Transaktionslogs für die Replikation, und nicht für die Sicherung oder Wiederherstellung.

Siehe auch

- „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203
- „Transaktionslog-Spiegeldateien“ [*SQL Anywhere Server - Datenbankadministration*]
- „Die konsolidierte Datenbank sichern“ auf Seite 135

Die konsolidierte Datenbank sichern

Sichern Sie Ihre konsolidierte Datenbank, indem Sie eine vollständige Sicherung der konsolidierten Datenbank und des Transaktionslogs erstellen, und führen Sie anschließend inkrementelle Sicherungen des Transaktionslogs durch.

Voraussetzungen

Sie müssen das BACKUP DATABASE-Systemprivileg haben.

Kontext und Bemerkungen

Vorsicht
Führen Sie den SQL Remote-Nachrichtenagenten (dbremote) mit der Befehlszeilenoption -x *nicht* in einer Datenbank aus, die gesichert wird.

Aufgabe

1. Erstellen Sie eine vollständige Sicherung der konsolidierten Datenbank und ihres Transaktionslogs.
   a. Stellen Sie eine Verbindung mit der Datenbank her.
   b. Führen Sie dbbackup mit den Befehlszeilenoptionen -r und -n aus.

      Zum Beispiel:

      ```
      dbbackup -r -n -c "UID=DBA;PWD=sql;DBF=c:\live\database.db" e:\archive
      ```

2. Führen Sie inkrementelle Sicherungen des Transaktionslogs der konsolidierten Datenbank durch. Bei der Sicherung des Transaktionslogs legen Sie fest, dass das Transaktionslog umbenannt und neu gestartet werden soll.
   a. Stellen Sie eine Verbindung mit der Datenbank her.
   b. Führen Sie dbbackup mit den Optionen -r, -n und -t aus.

      Zum Beispiel:

      ```
      dbbackup -r -n -t -c "UID=DBA;PWD=sql;DBF=c:\live\database.db" e:
      \archive
      ```

3. Führen Sie den SQL Remote-Nachrichtenagenten (dbremote) mit Zugriff auf das aktuelle Transaktionslog aus.

   Zum Beispiel:

   ```
   dbremote -c "UID=DBA;PWD=sql;DBF=c:\live\database.db" d:\live
   ```

Ergebnisse

Die konsolidierte Datenbank und das Transaktionslog werden gesichert.

Beispiel

Beispiel: Eine Datenbank namens *database.db* im Verzeichnis *c:\live* mit einem Transaktionslog namens *database.log* im Verzeichnis *d:\live*.

Wenn Sie das Transaktionslog im Sicherungsverzeichnis *e:\archive* sichern, indem Sie die Befehlszeilenoptionen -r und -n verwenden, um das Transaktionslog umzubenennen und neu zu starten, führt das Sicherungsdienstprogramm (dbbackup) die folgenden Aufgaben durch:

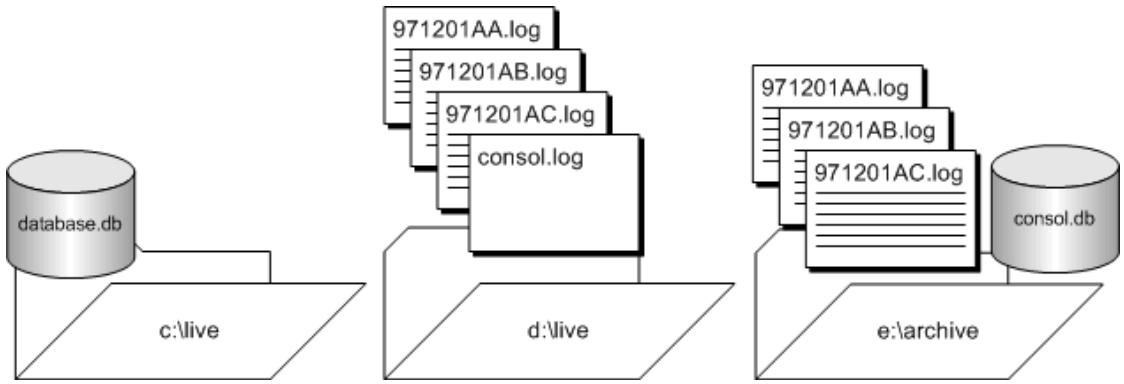
1. Die aktuelle Transaktionslogdatei auf *971201xx.log* umbenennen, wobei *xx* Buchstabenserien von *AA* bis *ZZ* sind.

2. Die Transaktionslogdatei im Sicherungsverzeichnis sichern, unter Erstellung einer Sicherungsdatei namens *971201xx.log*.

   Hinweis
   Vor SQL Anywhere 8.0.1 wurden die alten Transaktionslogdateien wie folgt benannt: *jjmmtt01.log*, *jjmmtt02.log* usw. Die Namensänderung wurde eingeführt, damit mehr alte Transaktionslogs gesichert werden können. Da der SQL Remote-Nachrichtenagent (dbremote) alle Dateien im angegebenen Verzeichnis unabhängig von ihren Namen durchsucht, sollte sich die Namensänderung nicht auf vorhandene Anwendungen auswirken.

3. Ein neues Transaktionslog als *database.log* starten.

   Nach mehreren Sicherungen enthalten das live-Verzeichnis und das archive-Verzeichnis eine Reihe von sequenziellen Transaktionslogs.



Siehe auch

- „Schutz der konsolidierten Datenbank vor Datenträgerfehlern“ auf Seite 134
- „Sicherungskopien von Transaktionslogs während einer Sicherung umbenennen (SQL)“ [*SQL Anywhere Server - Datenbankadministration*]
- „Sicherungsdienstprogramm (dbbackup)“ [*SQL Anywhere Server - Datenbankadministration*]

Eine konsolidierte Datenbank manuell wiederherstellen

Stellen Sie eine konsolidierte Datenbank wieder her, indem Sie jedes Transaktionslog in der Datenbank anwenden.

Voraussetzungen

Sie müssen das BACKUP DATABASE-Systemprivileg haben.

Erstellen Sie eine Kopie der Datenbank und der Transaktionslogdatei. In dieser Prozedur wird angenommen, dass frühere Sicherungen der Datenbankdatei erstellt wurden und verfügbar sind, zum Beispiel auf Band.

Aufgabe

1. Erstellen Sie ein temporäres Verzeichnis.
2. Stellen Sie die aktuellste Sicherung der Datenbankdatei (*.db*) und *nicht* der Transaktionslogdatei vom Band in einem temporären Verzeichnis wieder her.

   Im temporären Verzeichnis führen Sie Folgendes durch:

   a. Starten Sie die Sicherungskopie der Datenbank.
   b. Wenden Sie die alten Transaktionslogs unter Verwendung der Befehlszeilenoption -a an.
   c. Fahren Sie die Datenbank herunter.
   d. Starten Sie die Datenbank mit dem aktuellen Transaktionslog und der Befehlszeilenoption -a, um die Transaktionen anzuwenden und die Datenbankdatei auf den letzten Stand zu bringen.
   e. Fahren Sie die Datenbank herunter.
   f. Sichern Sie die Datenbank.
3. Kopieren Sie die Datenbank in das Produktionsverzeichnis.
4. Starten Sie die Datenbank.

   Jede neue Aktivität wird an das aktuelle Transaktionslog angefügt.

Ergebnisse

Die konsolidierte Datenbank wird wiederhergestellt.

Beispiel

Angenommen, Sie haben eine konsolidierte Datenbankdatei namens *c:\dbdir\cons.db*, eine Transaktionslogdatei *c:\dbdir\cons.log* und eine Transaktionslog-Spiegeldatei *d:\mirdir\cons.mlg*.

Nehmen wir überdies an, dass Sie vollständige Sicherungen wöchentlich und inkrementelle Sicherungen täglich durchführen, wobei Sie den folgenden Befehl verwenden:

```
dbbackup -c "UID=DBA;PWD=sql" -r -n -t e:\backdir
```

Dieser Befehl sichert das Transaktionslog *cons.log* im Verzeichnis *e:\backdir*. Die Transaktionslogdatei wird dann in *datexx.log* umbenannt, wobei *date* das aktuelle Datum und *xx* die nächste Buchstabenserie in der Sequenz ist, und ein neues Transaktionslog wird gestartet. Das Verzeichnis *e:\backdir* wird dann mit einem Dienstprogramm anderer Hersteller gesichert.

In diesem Szenario würden Sie den SQL Remote-Nachrichtenagenten (dbremote) mit dem optionalen Verzeichnis ausführen, sodass auf die umbenannten Transaktionslogdateien verwiesen wird. Zum Beispiel:

```
dbremote -c "UID=DBA;PWD=sql" c:\dbdir
```

Am dritten Tag nach der wöchentlichen Sicherung wird die Datenbankdatei durch einen fehlerhaften Plattenspeicherblock beschädigt. Führen Sie die folgenden Schritte aus:

1. Sichern Sie die Transaktionslog-Spiegeldatei *d:\mirdir\cons.mlg*.
2. Erstellen Sie ein temporäres Verzeichnis, in dem Sie die Wiederherstellung durchführen. In diesem Beispiel ist es das Verzeichnis *c:\recover*.
3. Stellen Sie die aktuellste Sicherung der Datenbankdatei *cons.db* mit *c:\recover\cons.db* wieder her.

   ```
   dbeng16 -a c:\dbdir\dateAA.log c:\recover\cons.db
   dbeng16 -a c:\dbdir\dateAB.log c:\recover\cons.db
   ```
4. Wenden Sie die umbenannten Transaktionslogs in der richtigen Reihenfolge an, und zwar folgendermaßen:
5. Kopieren Sie das aktuelle Transaktionslog *d:\mirdir\cons.log* in das Wiederherstellungsverzeichnis, sodass *c:\recover\cons.log* entsteht.

   ```
   dbeng16 c:\recover\cons.db
   ```
6. Starten Sie die Datenbank mit dem folgenden Befehl:
7. Fahren Sie den Datenbankserver herunter.
8. Sichern Sie die wiederhergestellte Datenbank und das Transaktionslog von *c:\recover*.
9. Kopieren Sie die Dateien von *c:\recover* in die entsprechenden Produktionsverzeichnisse:
   - Kopieren Sie *c:\recover\cons.db* in *c:\dbdir\cons.db*.

- Kopieren Sie *c:\recover\cons.log* in *c:\dbdir\cons.log* und *d:\mirdir\cons.mlg*.

Siehe auch

- „Eine konsolidierte Datenbank automatisch wiederherstellen“ auf Seite 139
- „Datenbankoption -a “ [*SQL Anywhere Server - Datenbankadministration*]

Eine konsolidierte Datenbank automatisch wiederherstellen

Sie können eine konsolidierte Datenbank automatisch wiederherstellen. Hinweise, wie Sie Transaktionslogs manuell anwenden, finden Sie unter „Eine konsolidierte Datenbank manuell wiederherstellen“ auf Seite 137.

Voraussetzungen

Sie müssen das BACKUP DATABASE-Systemprivileg haben.

Erstellen Sie eine Kopie der Datenbank und der Transaktionslogdatei. In dieser Prozedur wird angenommen, dass frühere Sicherungen der Datenbankdatei erstellt wurden und verfügbar sind, zum Beispiel auf Band.

Aufgabe

1. Stellen Sie die aktuellste Sicherungskopie der Datenbankdatei (*.db*) und *nicht* der Transaktionslogdatei vom Band in einem temporären Verzeichnis wieder her.

2. Im temporären Verzeichnis führen Sie Folgendes durch:

   a. Starten Sie die Datenbank, indem Sie die Transaktionslogs unter Verwendung der Befehlszeilenoption -ad anwenden.

      Wenn Sie die Befehlszeilenoption -ad angeben, sucht der Datenbankserver im angegebenen Verzeichnis nach den Transaktionslogs für die Datenbank. Danach bestimmt er anhand der Transaktionslog-Offsets die richtige Reihenfolge für die Anwendung der Transaktionslogs.

   b. Kopieren Sie das aktuelle Transaktionslog in das temporäre Verzeichnis.

   c. Starten Sie die Datenbank und wenden Sie das aktuelle Transaktionslog an.

   d. Fahren Sie den Datenbankserver herunter.

   e. Sichern Sie die Datenbank und das Transaktionslog.

3. Kopieren Sie die Datenbank und die Transaktionslogdateien in die entsprechenden Produktionsverzeichnisse.

4. Starten Sie Ihr System erneut auf die übliche Weise.

   Jede neue Aktivität wird an das aktuelle Transaktionslog angefügt.

Ergebnisse

Die konsolidierte Datenbank wird wiederhergestellt.

Beispiel

Angenommen, Sie haben eine konsolidierte Datenbankdatei namens *c:\dbdir\cons.db*, eine Transaktionslogdatei *c:\dbdir\cons.log* und eine Transaktionslog-Spiegeldatei *d:\mirdir\cons.mlg*.

Weiterhin wird angenommen, dass Sie einmal pro Woche eine vollständige Sicherung mit dem folgenden Befehl durchführen:

```
dbbackup -c "UID=DBA;PWD=sql" -r -n e:\backdir
```

Außerdem wird angenommen, dass Sie täglich inkrementelle Sicherungen mit dem folgenden Befehl durchführen:

```
dbbackup -c "UID=DBA;PWD=sql" -r -n -t e:\backdir
```

Dieser Befehl sichert das Transaktionslog *cons.log* im Verzeichnis *e:\backdir*. Die Transaktionslogdatei wird dann in *datexx.log* umbenannt, wobei *date* das aktuelle Datum und *xx* die nächste Buchstabenserie in der Sequenz ist, und ein neues Transaktionslog wird gestartet. Das Verzeichnis *e:\backdir* wird dann mit einem Dienstprogramm anderer Hersteller gesichert.

In diesem Szenario würden Sie den SQL Remote-Nachrichtenagenten (dbremote) mit dem optionalen Verzeichnis ausführen, sodass auf die umbenannten Transaktionslogdateien verwiesen wird. Zum Beispiel:

```
dbremote -c "UID=DBA;PWD=sql" c:\dbdir
```

Am dritten Tag nach der wöchentlichen Sicherung wird die Datenbankdatei durch einen fehlerhaften Plattenspeicherblock beschädigt. Führen Sie die folgenden Schritte aus:

1. Ersetzen Sie das Laufwerk *c:*.
2. Sichern Sie die Transaktionslog-Spiegeldatei *d:\mirdir\cons.mlg*.
3. Erstellen Sie ein temporäres Verzeichnis, in dem Sie die Wiederherstellung durchführen. In diesem Beispiel ist es *c:\recover*.
4. Stellen Sie die aktuellste Sicherung der Datenbankdatei *cons.db* mit *c:\recover\cons.db* wieder her.
5. Kopieren Sie die gesicherten Transaktionslogs auf *c:\dbdir*.
6. Wenden Sie die umbenannten Transaktionslogs an:

   ```
   dbeng16 c:\recover\cons.db -ad c:\dbdir
   ```
7. Kopieren Sie das aktuelle Transaktionslog *d:\mirdir\cons.log* in das Wiederherstellungsverzeichnis, sodass *c:\recover\cons.log* entsteht.
8. Starten Sie die Datenbank mit dem folgenden Befehl:

```
dbeng16 c:\recover\cons.db
```

9. Fahren Sie den Datenbankserver herunter.
10. Sichern Sie die wiederhergestellte Datenbank und das Transaktionslog von *c:\recover*.
11. Kopieren Sie die Dateien von *c:\recover* in die entsprechenden Produktionsverzeichnisse:
    - Kopieren Sie *c:\recover\cons.db* in *c:\dbdir\cons.db*.
    - Kopieren Sie *c:\recover\cons.log* in *c:\dbdir\cons.log* und *d:\mirdir\cons.mlg*.

Siehe auch

- „Datenbankoption -ad “ [*SQL Anywhere Server - Datenbankadministration*]
- „Sicherungsdienstprogramm (dbbackup)“ [*SQL Anywhere Server - Datenbankadministration*]

Berichterstellung und Behandlung von Replikationsfehlern

Die folgenden Fehler können in einem SQL Remote-System auftreten:

- **Zeile nicht gefunden** Siehe „Zeile nicht gefunden“ auf Seite 52.
- **Fehler bei der referenziellen Integrität** Siehe „Fehler bei der referenziellen Integrität“ auf Seite 53.
- **Mehrfach vorhandene Primärschlüssel** Siehe „Mehrfach vorhandene Primärschlüssel“ auf Seite 55.

Wenn ein Fehler auftritt, gibt der SQL Remote-Nachrichtenagent (dbremote) standardmäßig den Fehler in das Logausgabefenster aus. Der SQL Remote-Nachrichtenagent (dbremote) kann mehr Informationen in die Ausgabenachrichtendatei als in das Nachrichtenfenster ausgeben.

Die Nachrichtenlogdatei des SQL Remote-Nachrichtenagenten (dbremote) enthält die folgenden Informationen:

- Angewandte Nachrichten
- Fehlgeschlagene SQL-Anweisungen
- Andere Fehler

Um einen Fehler in die Ausgabelogdatei auszugeben, führen Sie den SQL Remote-Nachrichtenagenten (dbremote) mit der Befehlszeilenoption -o aus.

Wenn ein Fehler auftritt, können Sie SQL Remote konfigurieren, um Folgendes durchzuführen:

- **Eine Fehlerverarbeitungsprozedur ausführen** Standardmäßig wird keine Prozedur aufgerufen. Sie können allerdings die Datenbankoption replication_error verwenden, um eine gespeicherte Prozedur anzugeben, die vom SQL Remote-Nachrichtenagenten (dbremote) aufgerufen wird, wenn ein Fehler auftritt.

Sie können beispielsweise SQL Remote konfigurieren, um Folgendes durchzuführen:

- Teile des Ausgabelogs einer entfernten Datenbank an die konsolidierte Datenbank senden und in eine Datei schreiben.
- Eine E-Mail-Benachrichtigung senden, wenn in einer entfernten Datenbank ein Fehler auftritt.

- **Den Fehler ignorieren** Es kann Fälle geben, in denen es wünschenswert ist, dass der SQL Remote-Nachrichtenagent (dbremote) einen Fehler nicht meldet. Beispiel: Sie können festlegen, einen Fehler zu ignorieren, wenn Sie die Bedingungen, unter denen der Fehler auftritt, kennen und Sie sicher sind, dass der Fehler keine inkonsistenten Daten erzeugt. Siehe „Replikationsfehler ignorieren“ auf Seite 146.

Siehe auch

- „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203
- „Fehlerbehandlungsprozeduren für die Replikation“ auf Seite 142
- „Sammeln von Fehlern aus der entfernten Datenbank“ auf Seite 142
- „Empfangen von E-Mail-Benachrichtigungen zu Fehlern in entfernten Datenbanken“ auf Seite 144

Fehlerbehandlungsprozeduren für die Replikation

Setzen Sie die Option replication_error, um eine Prozedur aufzurufen, wenn ein SQL-Fehler auftritt. Standardmäßig wird keine Prozedur aufgerufen, wenn ein Fehler auftritt.

Die aufgerufene Prozedur muss ein einzelnes Argument vom Typ CHAR, VARCHAR oder LONG VARCHAR haben. Die Prozedur wird einmal mit der SQL-Fehlernachricht und ein zweites Mal mit der SQL-Anweisung aufgerufen, die den Fehler bewirkt.

Um die Option replication_error einzustellen, führen Sie die folgende Anweisung aus. *remote-user* ist der Publikationseigentümername in der Befehlszeile des SQL Remote-Nachrichtenagenten (dbremote) und *procedure-name* ist die Prozedur, die aufgerufen wird, wenn ein SQL-Fehler erkannt wird.

```
SET OPTION
remote-user.replication_error
= 'procedure-name';
```

Siehe auch

- „replication_error-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]

Sammeln von Fehlern aus der entfernten Datenbank

Teile des Ausgabelogs einer entfernten Datenbank werden an die konsolidierte Datenbank gesendet. Die Informationen werden in eine Datei geschrieben und die Datei kann Ausgabeloginformationen von einigen oder allen entfernten Datenbanken im System enthalten.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. Konfigurieren Sie die entfernten Datenbanken, um Ausgabeloginformationen an die konsolidierte Datenbank zu senden.
   a. Verwenden Sie die SET REMOTE-Anweisung mit der Option output_log_send_on_error, um Loginformationen zu senden, wenn ein Fehler auftritt.

      In der entfernten Datenbank führen Sie die folgende Anweisung aus:

      ```
      SET REMOTE link-name OPTION
      PUBLIC.output_log_send_on_error = 'Yes';
      ```

      Wenn der SQL Remote-Nachrichtenagent (dbremote) eine Nachricht liest, die mit dem Fehlerindikator **E** beginnt, sendet er die Ausgabeloginformationen an die konsolidierte Datenbank.

   b. Dieser Schritt ist optional. Setzen Sie die SET REMOTE-Anweisung mit der Option output_log_send_limit, um die Datenmenge zu beschränken, die an die konsolidierte Datenbank gesendet wird. Die Option output_log_send_limit gibt die Anzahl von Bytes am Ende des Ausgabelogs an (d.h. die aktuellsten Einträge), die an die konsolidierte Datenbank gesendet wird. Der Standardwert ist 5 kB.

      Wenn Sie einen Wert output_log_send_limit angeben, der die maximale Nachrichtengröße überschreitet, hebt SQL Remote den Wert output_log_send_limit auf und sendet nur, was in die maximale Nachrichtengröße passt.

      In der entfernten Datenbank führen Sie die folgende Anweisung aus:

      ```
      SET REMOTE link-name OPTION
      PUBLIC.output_log_send_limit = '7K';
      ```

2. Konfigurieren Sie die konsolidierte Datenbank, um Loginformationen zu empfangen.

   In der konsolidierten Datenbank führen Sie den SQL Remote-Nachrichtenagenten (dbremote) mit der Befehlszeilenoption -ro oder -rt aus.

3. Dieser Schritt ist optional. Um Ihre Konfigurationen zu testen, setzen Sie die Option output_log_send_now, um die Ausgabeloginformationen an die konsolidierte Datenbank zu senden.

   In der entfernten Datenbank setzen Sie die Option output_log_send_now auf YES.

   Beim nächsten Abruf sendet die entfernte Datenbank die Ausgabeloginformationen und setzt anschließend die Option output_log_send_now auf NO.

Ergebnisse

Ausgabelog-Informationen werden nun an die konsolidierte Datenbank gesendet.

Siehe auch

- „SET REMOTE OPTION-Anweisung [SQL Remote]“ auf Seite 238
- „SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)“ auf Seite 203
- „Empfangen von E-Mail-Benachrichtigungen zu Fehlern in entfernten Datenbanken“ auf Seite 144

Empfangen von E-Mail-Benachrichtigungen zu Fehlern in entfernten Datenbanken

Sie können eine E-Mail-Benachrichtigung senden, wenn in einer entfernten Datenbank ein Fehler auftritt. Sie können E-Mails oder ein Paging-System verwenden, um die Bestätigungen zu empfangen.

Voraussetzungen

Es gibt keine Voraussetzungen für diese Aufgabe.

Aufgabe

1. Stellen Sie in Sybase Central als Benutzer Cons eine Verbindung zur Datenbank mithilfe des Plug-Ins **SQL Anywhere 16** her.

2. Erstellen Sie eine gespeicherte Prozedur, die den DBA-Benutzer per E-Mail benachrichtigt, dass ein Fehler aufgetreten ist.

   Beispiel: Führen Sie den folgenden Befehl aus, um die sp_LogReplicationError-Prozedur zu erstellen:

   ```
   CREATE PROCEDURE cons.sp_LogReplicationError
    ( IN error_text LONG VARCHAR )
   BEGIN
    DECLARE current_remote_user CHAR( 255 );
    SET current_remote_user = CURRENT REMOTE USER;
    // Log the error
    INSERT INTO cons.replication_audit
     ( remoteuser, errormsg )
    VALUES
     ( current_remote_user, error_text );
    COMMIT WORK;
    //Now notify the DBA by email that an error has occurred
    // on the consolidated database. The email should contain the error
    // strings that the SQL Remote Message Agent is passing to the
   procedure.
    IF CURRENT PUBLISHER = 'cons' THEN
     CALL sp_notify_DBA( error_text );
    END IF
   END;
   ```

3. Erstellen Sie eine gespeicherte Prozedur, die das Versenden der E-Mails verwaltet.

   Beispiel: Führen Sie den folgenden Befehl aus, um die sp_notifiy_DBA-Prozedur zu erstellen:

   ```
   CREATE PROCEDURE sp_notify_DBA( in msg long varchar)
   BEGIN
    DECLARE rc INTEGER;
    rc=call xp_startmail( mail_user='davidf' );
    //If successful logon to mail
    IF rc=0 THEN
    rc=call xp_sendmail(
      recipient='Doe, John; Smith, Elton',
      subject='SQL Remote Error',
      "message"=msg);
    //If mail sent successfully, stop
     IF rc=0 THEN
      call xp_stopmail()
     END IF
   ```

```
  END IF
END;
```

4. Setzen Sie die Datenbankoption replication_error, um die Prozedur aufzurufen, die den DBA per E-Mail benachrichtigt, dass ein Fehler auftritt.

   Beispiel: Führen Sie die folgende Anweisung aus, um die sp_LogReplicationError-Prozedur aufzurufen, wenn ein Fehler auftritt.

   ```
   SET OPTION PUBLIC.replication_error =
    'cons.sp_LogReplicationError';
   ```

5. Eine Audit-Tabelle erstellen.

   Beispiel: Führen Sie Folgendes durch, um die Tabelle replication_audit zu erstellen:

   ```
   CREATE TABLE replication_audit (
    id   INTEGER DEFAULT AUTOINCREMENT,
    pub  CHAR(30) DEFAULT CURRENT PUBLISHER,
    remoteuser CHAR(30),
    errormsg LONG VARCHAR,
    timestamp  DATETIME DEFAULT CURRENT TIMESTAMP,
    PRIMARY KEY (id,pub)
   );
   ```

   Die folgende Tabelle beschreibt die Spalten der replication_audit-Tabelle:

Spalte	Beschreibung
pub	Aktueller Publikationseigentümer der Datenbank (identifiziert die Datenbank, in die der Publikationseigentümer eingefügt wurde).
remoteuser	Entfernter Benutzer, der die Nachricht anwendet (identifiziert die Datenbank, von der der entfernte Benutzer stammt).
errormsg	Fehlermeldung, die an die Prozedur replication_error weitergeleitet wird.

6. Testen Sie Ihre Prozeduren.

   Beispiel: Fügen Sie eine Zeile in die konsolidierte Datenbank ein, die denselben Primärschlüssel wie eine Zeile in einer entfernten Datenbank verwendet. Wenn diese Zeile aus der konsolidierten Datenbank an die entfernte Datenbank repliziert wird, tritt ein Primärschlüsselkonflikt-Fehler auf, und:

   - Der SQL Remote-Nachrichtenagent (dbremote) der entfernten Datenbank gibt die folgende Meldung in sein Ausgabelog aus:

     ```
     Nachricht von "cons" erhalten (0-0000000000-0)
     SQL-Anweisung fehlgeschlagen: (-193) Primärschlüssel für Tabelle
     'reptable' ist nicht eindeutig
     INSERT INTO cons.reptable( id,text,last_contact )
     VALUES (2,'dave','1997/apr/21 16:02:38.325')
     COMMIT WORK
     ```

   - Die folgende INSERT-Anweisung wird an die konsolidierte Datenbank gesendet:

```
INSERT INTO cons.replication_audit
 ( id,
   pub,
   remoteuser,
   errormsg,
   "timestamp")
VALUES
 ( 1,
  'cons',
  'sales',
  'primary key for table ''reptable'' is not unique (-193)',
  '1997/apr/21 16:03:13.836');
COMMIT WORK;
```

- Eine E-Mail wird an John Doe und Elton Smith mit der folgenden Nachricht gesendet:

```
primary key for table 'reptable' is not unique (-193)
INSERT INTO cons.reptable( id,text,last_contact )
            VALUES (2,'dave','1997/apr/21 16:02:52.605')
```

Ergebnisse

Nun werden E-Mail Benachrichtigungen gesendet, wenn Fehler in einer entfernten Datenbank auftreten.

Siehe auch

- „Sammeln von Fehlern aus der entfernten Datenbank“ auf Seite 142

Replikationsfehler ignorieren

Erstellen Sie einen BEFORE-Trigger für die Aktion, die den bekannten Fehler verursacht. Dieser Trigger sollte einen Fehler signalisieren. Beispiel: Um INSERT-Anweisungsfehler zu ignorieren, die auftreten, wenn in einer Tabelle eine referenzierte Spalte fehlt, erstellen Sie einen BEFORE INSERT-Trigger, der den SQL-Zustand SQLE_REMOTE_STATEMENT_FAILED signalisiert, wenn die referenzierte Spalte nicht existiert. Die INSERT-Anweisung schlägt fehl, aber das Scheitern wird im Ausgabelog des SQL Remote-Nachrichtenagenten (dbremote) nicht protokolliert.

Siehe auch

- „Entfernte Anweisung fehlgeschlagen“ [*Fehlermeldungen*]

Sicherheit

Zum Schutz Ihrer Daten stehen die folgenden Funktionen zur Verfügung.

- **SYS_RUN_REPLICATION_ROLE-Systemrolle** Es wird empfohlen, dass Sie die Verbindung zum SQL Remote-Nachrichtenagenten (dbremote) als Benutzer mit der SYS_RUN_REPLICATION_ROLE-Systemrolle herstellen. Siehe „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*].

- **Datenbankverschlüsselung** Sie können Ihre Datenbank verschlüsseln, indem Sie die Befehlszeilenoption -ek verwenden. Siehe „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214.

- **Nachrichtenverschlüsselung** Der SQL Remote-Nachrichtenagent (dbremote) verwendet einen einfachen Verschlüsselungsalgorithmus, um Nachrichten vor zufälliger Ausspähung zu schützen. Dieses Verschlüsselungsschema ist allerdings nicht dazu gedacht, einen vollständigen Schutz vor zielgerichteten Entschlüsselungsversuchen zu bieten. Hinweise zur Datenbankverschlüsselung finden Sie unter „Datenbankverschlüsselung und -entschlüsselung“ [*SQL Anywhere Server - Datenbankadministration*].

Upgrades und Resynchronisation

Sie müssen über die SYS_RUN_REPLICATION_ROLE-Systemrolle verfügen.

Seien Sie vorsichtig, wenn Sie ein Upgrade eines SQL Remote-Systems durchführen. Sie können das Upgrade eines SQL Remote-Systems auf eine der folgenden Arten durchführen:

- **Upgrade der Software** Hinweise zum Upgrade von SQL Remote finden Sie unter „SQL Remote-Upgrades“ [*SQL Anywhere 16 - Änderungen und Upgrades*].
- **Datenbankschema ändern** Um Änderungen am Datenbankschema durchzuführen, können Sie Folgendes durchführen:
  - **Den Passthrough-Modus verwenden** Der Passthrough-Modus ermöglicht es, Schemaänderungen an einige oder an alle Datenbanken in einem SQL Remote-System zu senden, erfordert aber eine umsichtige Planung und Ausführung.
  - **Subskriptionen resynchronisieren** Eine Resynchronisation besteht darin, neue Kopien der Daten in die entfernten Datenbanken zu kopieren. Wenn es viele entfernte Datenbanken gibt, kann die Resynchronisation zu einem zeitaufwendigen Prozess werden, der zu Arbeitsunterbrechungen und Datenverlusten führen kann.

Siehe auch

- „PASSTHROUGH-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SQL Remote-Passthrough-Modus“ auf Seite 148
- „Resynchronisation von Subskriptionen“ auf Seite 151

In einem laufenden System zu vermeidende Änderungen

Die folgenden Änderungen sollten nicht bei einem bereitgestellten und laufenden SQL Remote-System durchgeführt werden, außer unter den angeführten Bedingungen:

- **Publikationseigentümer ändern** Es können Probleme auftreten, wenn Sie den Publikationseigentümer-Benutzernamen in einer konsolidierten Datenbank eines bereitgestellten und laufenden SQL Remote-System ändern. Wenn Sie den Publikationseigentümer-Benutzernamen einer konsolidierten Datenbank ändern müssen, müssen Sie das SQL Remote-System herunterfahren und alle entfernten Benutzer resynchronisieren.

  Das Ändern des Benutzernamens eines Publikationseigentümers in einer entfernten Datenbank verursacht Probleme bei Subskriptionen, die die entfernte Datenbank betreffen, bis hin zum

Datenverlust. Wenn Sie den Publikationseigentümer-Benutzernamen einer entfernten Datenbank ändern müssen, fahren Sie die entfernte Datenbank herunter und resynchronisieren Sie den entfernten Benutzer.

- **Restriktive Änderungen an Tabellen durchführen** Sie können keine restriktiven Änderungen an Tabellen durchführen. Löschen Sie beispielsweise keine Spalte oder ändern Sie keine Spalte, um NULL-Werte zu verbieten, weil es Nachrichten im System geben kann, die diese Spalten referenzieren.

- **Tolerante Änderungen an Tabellen durchführen** Sie können tolerante Änderungen durchführen, indem Sie den Passthrough-Modus verwenden. Verwenden Sie den Passthrough-Modus, um Änderungen am Schema von entfernten Datenbanken und an Publikationen durchzuführen. Tolerante Änderungen umfassen das Hinzufügen einer neuen Tabelle oder Spalte, die Resynchronisation von Benutzern, das Löschen von Benutzern und das Ändern der Adresse, des Nachrichtentyps oder der Sendefrequenz bei einem entfernten Benutzer.

- **Publikationen ändern** Publikationsdefinitionen müssen sowohl in der konsolidierten als auch in entfernten Datenbanken gepflegt werden. Das Ändern von Publikationen in einem laufenden SQL Remote-System kann zu Replikationsfehlern und zu Datenverlusten im Replikationssystem führen.

- **Subskriptionen löschen** Sie können eine Subskription löschen, aber Sie müssen den Passthrough-Modus verwenden, um die Daten in der entfernten Datenbank zu entfernen.

- **Datenbanken entladen und neuladen** Sie müssen sicherstellen, das das Transaktionslog korrekt gepflegt wird.

- **Änderungen in einer mehrschichtigen Hierarchie durchführen** Hinweise zum erneuten Extrahieren von Datenbankschemas in einer vielschichtigen Hierarchie finden Sie unter „Datenbankextraktion für ein vielstufiges Hierarchiesystem“ auf Seite 87.

Siehe auch

- „ALTER PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SQL Remote-Passthrough-Modus“ auf Seite 148
- „PASSTHROUGH-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Einschränkungen beim Passthrough-Modus“ auf Seite 149
- „Resynchronisation von Subskriptionen“ auf Seite 151
- „An Replikations- oder Synchronisationssystemen beteiligte Datenbanken neu aufbauen (Befehlszeile)“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

SQL Remote-Passthrough-Modus

Verwenden Sie den Passthrough-Modus, um Standard-SQL-Anweisungen an eine entfernte Datenbank zu übergeben, wo sie ausgeführt werden können.

Sie müssen über die SYS_RUN_REPLICATION_ROLE-Systemrolle verfügen.

Sie können den Passthrough-Modus verwenden, um die folgenden Aufgaben auf einem laufenden SQL Remote-System durchzuführen:

- Neue Benutzer hinzufügen
- Benutzer resynchronisieren
- Benutzer aus dem System löschen.
- Die Adresse, den Nachrichtentyp oder die Frequenz bei einem entfernten Benutzer ändern
- Eine Spalte einer Tabelle hinzufügen

Vorsicht

- SQL Remote verlangt, dass jede Datenbank im System dieselben Objekte enthält. Wenn eine Tabelle an einigen, aber nicht an allen Standorten geändert wird, werden Versuche, die Datenänderungen zu replizieren, fehlschlagen. Zusätzliche Schemaänderungen, die an einem laufenden SQL Remote-System durchgeführt werden, können Probleme verursachen.

- Testen Sie Ihre Passthrough-Vorgänge in einer Kopie der konsolidierten Datenbank mit einer Kopie der subskribierten entfernten Datenbank. Führen Sie niemals ungetestete Passthrough-Skripten in einer Produktionsdatenbank aus.

- Qualifizieren Sie Objektnamen mit dem Eigentümernamen. PASSTHROUGH-Anweisungen vom selben Benutzernamen werden auf entfernten Datenbanken nicht ausgeführt. Objektnamen ohne Eigentümernamen-Qualifizierer werden möglicherweise nicht korrekt aufgelöst.

Siehe auch

- „In einem laufenden System zu vermeidende Änderungen“ auf Seite 147
- „PASSTHROUGH-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Einschränkungen beim Passthrough-Modus

- **Passthrough funktioniert nur auf einer Ebene in einer Hierarchie** In einem mehrschichtigen SQL Remote-System ist es wichtig, dass Passthrough-Anweisungen unmittelbar unterhalb der aktuellen Ebene durchgeführt werden. In einem mehrschichtigen System müssen Passthrough-Anweisungen für die darunter liegende Ebene jeweils in der konsolidierten Datenbank eingegeben werden.

- **Prozeduren aufrufen** Wenn eine gespeicherte Prozedur im Passthrough-Modus mit einer CALL- oder EXEC-Anweisung aufgerufen wird, gilt Folgendes:

  - Die Prozedur muss in der konsolidierten Datenbank, die den Passthrough-Befehl aufruft, vorhanden sein, auch wenn die Prozedur nicht in der konsolidierten Datenbank ausgeführt wird.

  - Die Prozedur muss auch in der entfernten Datenbank vorhanden sein. Die CALL- oder EXEC-Anweisung wird repliziert, aber Anweisungen in der Prozedur nicht. Es wird angenommen, dass die Prozedur in der replizierten Datenbank ordnungsgemäß funktioniert.

- **Steueranweisungen** Steuerungsanweisungen wie IF und LOOP sowie jeder Cursorvorgang werden im Passthrough-Modus nicht repliziert. Alle Anweisungen innerhalb der Schleife oder Steuerungsstruktur *werden* repliziert.

- **Cursorvorgänge** Vorgänge mit Cursorn werden nicht repliziert.

- **SQL SET OPTION-Anweisungen** Statische Embedded SQL SET OPTION-Anweisungen werden nicht repliziert. Hingegen werden dynamische SQL-Anweisungen repliziert. Siehe „Static und Dynamic SQL“ [*SQL Anywhere Server - Programmierung*].

  Beispiel: Die folgende Anweisung wird im Passthrough-Modus nicht repliziert:

  ```
  EXEC SQL SET OPTION ...
  ```

  Allerdings wird die folgende Dynamic SQL-Anweisung repliziert:

  ```
  EXEC SQL EXECUTE IMMEDIATE "SET OPTION ... "
  ```

- **Batchanweisungen** Batchanweisungen (eine Gruppe von Anweisungen innerhalb von BEGIN und END) werden im Passthrough-Modus nicht repliziert. Wenn Sie versuchen, Batchanweisungen im Passthrough-Modus zu verwenden, tritt ein Fehler auf.

Siehe auch

- „PASSTHROUGH-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Steueranweisungen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Resynchronisation von Subskriptionen“ auf Seite 151

Passthrough-Modus starten und stoppen

Der Passthrough-Modus wird mit der PASSTHROUGH-Anweisung gestartet und der PASSTHROUGH STOP-Anweisung gestoppt. Eine Passthrough-Sitzung bezieht sich auf die Anweisungen, die zwischen den PASSTHROUGH-Anweisungen eingegeben werden. Für Anweisungen, die in eine Passthrough-Sitzung eingegeben werden, gilt Folgendes:

- Sie werden auf Syntaxfehler überprüft.

- Sie werden in der konsolidierten Datenbank ausgeführt, außer Sie geben das ONLY-Schlüsselwort an. Wenn ONLY angegeben ist, werden die Anweisungen an die entfernte Datenbank gesendet, ohne in der konsolidierten Datenbank ausgeführt zu werden.

  Die folgende Anweisung startet eine Passthrough-Sitzung, die die Anweisungen an eine Liste von zwei benannten Subskribenten übermittelt, ohne in der aktuellen Datenbank ausgeführt zu werden.

  ```
  PASSTHROUGH ONLY
  FOR userid_1, userid_2;
  ```

- Sie werden an die gekennzeichnete Subskribenten-Datenbank übermittelt. Passthrough-Anweisungen werden in der gleichen Sequenz wie normale Replikatnachrichten in der Reihenfolge repliziert, in der die Anweisungen im Transaktionslog aufgezeichnet sind.

- Sie werden in der Subskribenten-Datenbank ausgeführt.

Direkte Passthrough-Anweisungen

Die folgende Anweisung startet eine Passthrough-Sitzung, die die Anweisungen an alle Benutzer übermittelt, die für die pubname-Publikation subskribiert sind.

```
PASSTHROUGH ONLY
FOR SUBSCRIPTION TO [owner].pubname statement1;
```

Der Passthrough-Modus ist additiv. Im folgenden Beispiel wird Anweisung_1 an user_1 und Anweisung_2 sowohl an user_1 als auch an user_2 gesendet.

```
PASSTHROUGH ONLY FOR user_1 ;
statement_1;
PASSTHROUGH ONLY FOR user_2 ;
statement_2;
```

Die folgende Anweisung stoppt eine Passthrough-Sitzung für alle entfernten Benutzer:

```
PASSTHROUGH STOP;
```

Datenmanipulationssprache (DML)

Der Passthrough-Modus wird üblicherweise verwendet, um Datenmanipulationsanweisungen zu versenden. In diesem Fall verwenden DML-Anweisungen das *before*-Schema vor dem Passthrough und das *after*-Schema nach dem Passthrough.

Das folgende Beispiel löscht eine Tabelle in der entfernten Datenbank und in der konsolidierten Datenbank.

```
-- Drop a table on the remote database
-- and at the consolidated database
PASSTHROUGH FOR Joe_Remote;
DROP TABLE CrucialData;
PASSTHROUGH STOP;
```

Das folgende Beispiel löscht eine Tabelle nur in der entfernten Datenbank.

```
-- Drop a table on the remote database only
PASSTHROUGH ONLY FOR Joe_Remote;
DROP TABLE CrucialData;
PASSTHROUGH STOP;
```

Siehe auch

- „PASSTHROUGH-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Datenmanipulationsanweisungen“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

Resynchronisation von Subskriptionen

Wenn Sie eine entfernte Datenbank erstellen, extrahieren Sie sowohl das Schema als auch die Daten aus der konsolidierten Datenbank und verwenden sie, um die entfernte Datenbank aufzubauen. Dieser Prozess stellt sicher, dass jede Datenbank eine Erstkopie der Daten hat.

Nach dem Deployment kann es sinnvoll sein, Subskriptionen unter den folgenden Bedingungen zu resynchronisieren:

- **Nachdem Sie signifikante Wartungsarbeiten an der konsolidierten Datenbank durchgeführt haben.** Beispiel: Sie führen Änderungen in der konsolidierten Datenbank durch, die jede Zeile in der Datenbank aktualisiert. Standardmäßig erstellt SQL Remote Aktualisierungsnachrichten und sendet sie an jede subskribierte entfernte Datenbank. Diese Aktualisierungsnachrichten könnten die UPDATE-, DELETE- und INSERT-Anweisungen für jede Zeile einbeziehen.

  Wenn Sie wählen, die Subskription mit einer SYNCHRONIZE SUBSCRIPTION-Anweisung zu synchronisieren, senden Sie nur die Anweisungen, die für das Löschen aller Zeilen in den subskribierten Tabellen erforderlich sind, sowie die INSERT-Anweisungen, um alle neuen Zeilen einzufügen.

- **Wenn eine entfernte Datenbank mit einer konsolidierten Datenbank aus dem Takt ist** Wenn eine entfernte Datenbank mit der konsolidierten Datenbank aus dem Takt gerät, können Sie versuchen, den Passthrough-Modus zu verwenden.

  Wenn die Verwendung des Passthrough-Modus nicht funktioniert, können Sie die Subskriptionen synchronisieren. Wenn Sie Subskriptionen synchronisieren, zwingen Sie die entfernte Datenbank wieder in den Takt mit der konsolidierten Datenbank. Eine SYNCHRONIZE SUBSCRIPTION-Anweisung enthält Anweisungen zum Löschen des Inhalts der subskribierten Tabellen in der entfernten Datenbank und Anweisungen zum Einfügen der Zeilen der Subskription von der konsolidierten Datenbank in die entfernte Datenbank.

Einschränkungen

- **Die Synchronisation gilt für die gesamte Subskription** Sie können nicht eine einzelne Tabelle synchronisieren.

- **Datenverlust bei Synchronisation** In der entfernten Datenbank gehen alle Daten verloren, die Teil der Subskription sind, die nicht in die konsolidierte Datenbank repliziert wurde.

  Bevor Sie die Datenbank synchronisieren, verwenden Sie den **Assistenten zum Entladen einer Datenbank** in Sybase Central oder das Dienstprogramm Entladen (dbunload), um die entfernte Datenbank zu entladen oder zu sichern.

Siehe auch

- „SQL Remote-Passthrough-Modus“ auf Seite 148
- „PASSTHROUGH-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYNCHRONIZE SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Tipps zum Exportieren von Daten mit dem Assistenten zum Entladen einer Datenbank“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]
- „Dienstprogramm zum Entladen (dbunload)“ [*SQL Anywhere Server - Datenbankadministration*]

Subskriptionen synchronisieren

Publikationen für subskribierte Benutzer manuell synchronisieren

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Verwenden Sie entweder das Extraktionsdienstprogramm (dbxtract) oder den **Assistenten zum Extrahieren einer Datenbank**, um die Daten für die angegebene entfernte Datenbank zu extrahieren und anschließend die Daten manuell in die entfernte Datenbank zu laden.

Vorsicht
Führen Sie nicht den SQL Remote-Nachrichtenagenten (dbremote) aus, wenn Sie das Extraktionsdienstprogramm (dbxtract) oder den **Assistenten zum Extrahieren einer Datenbank** ausführen.

Aufgabe

1. Fahren Sie den SQL Remote-Nachrichtenagenten in der entfernten Datenbank und in der konsolidierten Datenbank herunter.
2. Stellen Sie eine Verbindung mit der konsolidierten Datenbank her.
3. Doppelklicken Sie auf **Publikationen**.
4. Doppelklicken Sie auf eine Publikation.
5. Klicken Sie auf die Registerkarte **SQL Remote-Subskriptionen**.
6. Synchronisieren Sie Subskriptionen manuell:
   a. Rechtsklicken Sie in der Liste **Subskribenten** auf den Benutzer und klicken Sie auf **Eigenschaften**.
   b. Klicken Sie auf die Registerkarte **Erweitert**.
   c. Klicken Sie auf **Jetzt synchronisieren**.

      Die Subskriptionen sind betroffen, sobald Sie auf die Schaltfläche **Jetzt synchronisieren** klicken. Ein nachfolgendes Klicken auf **Abbrechen** bricht die Synchronisationsaktion *nicht* ab.
7. Klicken Sie auf **OK**.

Ergebnisse

Die Subskriptionen werden synchronisiert.

Siehe auch

- „SYNCHRONIZE SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214

Synchronisieren mit dem SQL Remote-Nachrichtenagenten (dbremote)

Ersetzen Sie den aktuellen Inhalt der subskribierten Tabellen durch die neue Kopie.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Verwenden Sie das Extraktionsdienstprogramm (dbxtract) oder den **Assistenten zum Extrahieren einer Datenbank**, um Subskriptionen zu synchronisieren. Siehe „Subskriptionen synchronisieren“ auf Seite 152.

Das Extrahieren einer großen Anzahl von Subskriptionen oder das Synchronisieren von Subskriptionen für umfangreiche, häufig benutzte Tabellen kann den Datenbankzugriff verlangsamen. Sie können die SEND AT-Klausel verwenden, um einen Zeitpunkt für die Synchronisation anzugeben, an dem die konsolidierte Datenbank gering beansprucht wird.

Aufgabe

1. Stellen Sie eine Verbindung mit der konsolidierten Datenbank her.
2. Führen Sie eine SYNCHRONIZE SUBSCRIPTION-Anweisung aus.

   Der SQL Remote-Nachrichtenagent (dbremote) in der konsolidierten Datenbank sendet eine Kopie aller Zeilen in der Subskription an den Subskribenten. Der SQL Remote-Nachrichtenagent (dbremote) nimmt an, dass ein entsprechendes Datenbankschema in der entfernten Datenbank vorhanden ist.

   Der SQL Remote-Nachrichtenagent (dbremote) in der Subskribenten-Datenbank empfängt die Synchronisationsnachricht und *ersetzt* den aktuellen Inhalt der subskribierten Tabellen durch die neue Kopie.

Ergebnisse

Die aktuellen Inhalte der subskribierten Tabellen werden durch die neue Kopie ersetzt.

Vorsicht

- **Führen Sie keine SYNCHRONIZE SUBSCRIPTION-Anweisung in einer entfernten Datenbank aus.** Führen Sie SYNCHRONIZE SUBSCRIPTION-Anweisungen in der konsolidierten Datenbank aus.
- **Große Mengen von Nachrichten möglich.** Das Synchronisieren von Datenbanken über ein Nachrichtensystem kann zu hohem Nachrichtenvolumen führen. Auch kann die Größe der Nachricht die Größe der entfernten Datenbank überschreiten. Das Synchronisieren von vielen Subskriptionen über eine Nachrichtenverbindung kann die Menge des Nachrichten-Verkehrsaufkommen erhöhen.

  Es ist in vielen Fällen empfehlenswert, dass Sie die entfernten Datenbanken extrahieren und anschließend die Daten manuell laden.

Siehe auch

- „SYNCHRONIZE SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Sendefrequenz“ auf Seite 93

Starten von Subskriptionen

Verwenden Sie diese Aufgabe, um Subskriptionen für eine Publikation für einen subskribierten Benutzer zu starten.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Um mehrere Subskriptionen in einer einzigen Transaktion zu starten, verwenden Sie die REMOTE RESET-Anweisung.

Aufgabe

1. Stellen Sie in Sybase Central eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf **Publikationen**.
3. Doppelklicken Sie auf eine Publikation.
4. Klicken Sie auf die Registerkarte **SQL Remote-Subskriptionen**.
5. Synchronisieren Sie Subskriptionen manuell:
   a. Rechtsklicken Sie in der Liste **Subskribent** auf den Benutzer und klicken Sie auf **Eigenschaften**.
   b. Klicken Sie auf die Registerkarte **Erweitert**.
   c. Klicken Sie auf **Jetzt synchronisieren**.

      Die Subskriptionen sind betroffen, sobald Sie auf die Schaltfläche **Jetzt synchronisieren** klicken. Ein nachfolgendes Klicken auf **Abbrechen** bricht die Synchronisationsaktion *nicht* ab.

Ergebnisse

Die Subskriptionen für eine Publikation für einen subskribierten Benutzer werden gestartet.

Siehe auch

- „START SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REMOTE RESET-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Stoppen von Subskriptionen

Verwenden Sie diese Aufgabe, um eine Subskription für einen Benutzer aufzuheben.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Stellen Sie eine Verbindung zur Datenbank mithilfe des **SQL Anywhere 16**-Plug-Ins her.
2. Doppelklicken Sie auf **Publikationen**.
3. Doppelklicken Sie auf die gewünschte Publikation.
4. Klicken Sie auf die Registerkarte **SQL Remote-Subskriptionen**.
5. Um Subskriptionen manuell zu synchronisieren, rechtsklicken Sie in der Liste **Subskribent**auf den Benutzer und klicken anschließend auf **Eigenschaften**.

   Klicken Sie auf die Registerkarte **Erweitert**. Auf dieser Registerkarte klicken Sie auf **Jetzt stoppen**, um Subskriptionen zu stoppen.

   Die Subskriptionen sind betroffen, sobald Sie auf die Schaltfläche **Jetzt stoppen** klicken. Ein nachfolgendes Klicken auf **Abbrechen** im Eigenschaftsfenster bricht Ihre Synchronisationsaktion *nicht* ab.

Ergebnisse

Die Subskription wird aufgehoben.

Praktische Einführung: SQL Remote-System erstellen

In den Lektionen dieser praktischen Einführung erfahren Sie, wie Sie ein SQL Remote-Replikationssystem einrichten, das sowohl eine konsolidierte SQL Anywhere-Datenbank als auch eine entfernte Datenbank verwendet.

In dieser praktischen Einführung führen Sie folgende Schritte durch:

- Erstellen einer konsolidierten SQL Anywhere-Datenbank und einer entfernten SQL Anywhere-Datenbank, die eine Teilmenge der Daten in der konsolidierten Datenbank enthält.
- Erstellen eines Replikationssystems zur gemeinsamen Dateibenutzung mit der einzelnen entfernten SQL Anywhere-Datenbank.
- Replizieren von Daten zwischen der konsolidierten Datenbank und der entfernten Datenbank.

Lektion 1: Erstellen der konsolidierten Datenbank

Erstellen der konsolidierten Datenbank und der Verzeichnisse für die praktische Einführung.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Viele.

Aufgabe

1. Erstellen Sie die Verzeichnisse *c:\tutorial*, *c:\tutorial\hq* und *c:\tutorial\field*.
2. Führen Sie im Verzeichnis *c:\tutorial* den folgenden Befehl aus, um die konsolidierte Datenbank (hq) zu erstellen:

   ```
   dbinit -dba DBA,sql hq.db
   ```
3. Stellen Sie in Interactive SQL eine Verbindung mit der konsolidierten Datenbank her.

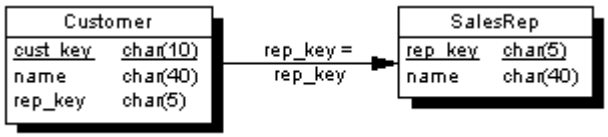
   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=server_hq;DBF=c:\tutorial\hq.db"
   ```
4. Führen Sie die folgenden Anweisungen aus, um zwei Tabellen in der konsolidierten Datenbank (hq) zu erstellen:

   ```
   CREATE TABLE SalesReps (
    rep_key CHAR(12) NOT NULL,
    name CHAR(40) NOT NULL,
    PRIMARY KEY ( rep_key )
   ```

```
);

CREATE TABLE Customers (
 cust_key CHAR(12) NOT NULL,
 name CHAR(40) NOT NULL,
 rep_key CHAR(12) NOT NULL,
 FOREIGN KEY ( rep_key )
  REFERENCES SalesReps (rep_key ),
 PRIMARY KEY (cust_key)
);
```

Die folgende Abbildung zeigt das Schema der konsolidierten Datenbank (hq) für die praktische Einführung:



- Jeder Handelsvertreter wird durch eine Zeile in der SalesReps-Tabelle dargestellt.
- Jeder Kunde wird durch eine Zeile in der Customers-Tabelle dargestellt.
- Jeder Kunde ist einem einzelnen Handelsvertreter zugeordnet und diese Zuordnung wird in die Datenbank integriert als Fremdschlüssel von der Customers-Tabelle zur SalesReps-Tabelle. Die Beziehung zwischen der Customers-Tabelle und der SalesReps-Tabelle ist eine Viele-zu-Eins-Beziehung.

Tabellenname	Beschreibung
SalesRep	Die SalesReps-Tabelle enthält eine Zeile für jeden Handelsvertreter, der für das Unternehmen arbeitet. Die SalesReps-Tabelle hat die folgenden Spalten: • **rep_key** Ein Bezeichner für jeden Handelsvertreter. Dies ist der Primärschlüssel. • **name** Der Name jedes Handelsvertreters
Customers	Die Customers-Tabelle enthält eine Zeile für jeden Kunden, der eine Geschäftsbeziehung mit dem Unternehmen unterhält. Die Customers-Tabelle enthält die folgenden Spalten: • **cust_key** Ein Bezeichner für jeden Kunden. Dies ist der Primärschlüssel. • **name** Der Name jedes Kunden • **rep_key** Ein Bezeichner für den Handelsvertreter in einer Geschäftsbeziehung. Dies ist ein Fremdschlüssel zur Tabelle "SalesReps".

5. Führen Sie die folgenden Anweisungen aus, um der SalesReps- und Customers-Tabelle Beispieldaten hinzuzufügen:

```
INSERT INTO SalesReps (rep_key, name)
VALUES ('rep1', 'Field User');
INSERT INTO SalesReps (rep_key, name)
```

```
VALUES ('rep2', 'Another User');
COMMIT;

INSERT INTO Customers (cust_key, name, rep_key)
VALUES ('cust1', 'Ocean Sports', 'rep1' );
INSERT INTO Customers (cust_key, name, rep_key)
VALUES ('cust2', 'Sports Plus', 'rep2' );
COMMIT;
```

6. Führen Sie die folgenden Anweisungen aus, um sicherzustellen, dass die Tabellen erstellt wurden:

   ```
   SELECT * FROM SalesReps;
   ```

   Die oben stehende Abfrage gibt die folgenden Daten aus der SalesReps-Tabelle zurück:

rep_key	name
rep1	Field User
rep2	Another User

   ```
   SELECT * FROM Customers;
   ```

   Die oben stehende Abfrage gibt die folgenden Daten aus der Customers-Tabelle zurück:

cust_key	name	rep_key
cust1	Ocean Sports	rep1
cust2	Sports Plus	rep2

Ergebnisse

Die Tabellen werden erstellt und mit Daten gefüllt.

Nächste Schritte

„Lektion 2: PUBLISH- und REMOTE-Privilegien in der konsolidierten Datenbank erteilen“ auf Seite 159.

Lektion 2: PUBLISH- und REMOTE-Privilegien in der konsolidierten Datenbank erteilen

Publikationseigentümer für die konsolidierte Datenbank Interactive SQL erstellen.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Jede Datenbank in einem SQL Remote-System erfordert einen Publikationseigentümer, der ein eindeutiger Benutzer mit PUBLISH-Privileg ist. Alle ausgehenden SQL Remote-Nachrichten, einschließlich Publikationsaktualisierungen und Empfangsbestätigungen, werden durch ihren Publikationseigentümer gekennzeichnet. Jede Datenbank in einem SQL Remote-System versendet Empfangsbestätigungen.

Aufgabe

1. Wenn Sie derzeit nicht mit der konsolidierten Datenbank (hq) verbunden sind, führen Sie folgenden Befehl aus:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=server_hq;DBF=c:\tutorial\hq.db"
   ```

2. Führen Sie die folgende Anweisung aus, um den Benutzer hq_user mit den Privilegien CONNECT und PUBLISH zu erstellen:

   ```
   CREATE USER hq_user IDENTIFIED BY hq_pwd;
   GRANT CONNECT TO hq_user IDENTIFIED BY hq_pwd;
   GRANT PUBLISH TO hq_user;
   ```

3. Führen Sie die folgende Anweisung aus, um die Benutzer-ID für den Publikationseigentümer der Datenbank zu prüfen:

   ```
   SELECT CURRENT PUBLISHER;
   ```

4. Eine Datenbank, wie z.B. eine konsolidierte Datenbank, die Nachrichten an andere Datenbanken sendet, muss angeben, an welche Datenbanken sie Nachrichten sendet. Um diese entfernten Datenbanken in der konsolidierten Datenbank anzugeben, erteilen Sie den Publikationseigentümern der entfernten Datenbanken das REMOTE-Privileg. Das REMOTE-Privileg identifiziert Datenbanken, die Nachrichten von der aktuellen Datenbank erhalten. Führen Sie die folgenden Anweisungen aus, um den entfernten Benutzer field_user mit dem Kennwort "field_pwd" zu erstellen, der CONNECT- und REMOTE-Privilegien hat:

   ```
   CREATE USER field_user IDENTIFIED BY field_pwd;
   GRANT CONNECT TO field_user IDENTIFIED BY field_pwd;
   GRANT REMOTE TO field_user
   TYPE file
   ADDRESS 'field';
   ```

Ergebnisse

Der Benutzer hat CONNECT- sowie PUBLISH- und REMOTE-Privilegien.

Nächste Schritte

„Lektion 3: Erstellen von Publikationen und Subskriptionen“ auf Seite 161.

Lektion 3: Erstellen von Publikationen und Subskriptionen

Erstellen Sie die Publikation in der konsolidierten Datenbank mit Interactive SQL.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Eine Publikation beschreibt die Datenmenge, die repliziert werden soll. In dieser Lektion erstellen Sie eine Publikation namens SalesRepData, die alle Zeilen der SalesReps-Tabelle und einige Zeilen der Customers-Tabelle repliziert. Sie subskribieren einen Benutzer für eine Publikation, indem Sie eine Subskription erstellen.

Aufgabe

1. Wenn Sie derzeit nicht mit der konsolidierten Datenbank (hq) verbunden sind, führen Sie folgenden Befehl aus:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=server_hq;DBF=c:\tutorial\hq.db"
   ```

2. Führen Sie die folgende Anweisung aus, um eine Publikation namens SalesRepData zu erstellen:

   ```
   CREATE PUBLICATION SalesRepData (
    TABLE SalesReps,
    TABLE Customers SUBSCRIBE BY rep_key
   );
   ```

   Die Publikation SalesRepData publiziert Folgendes:

   - Die gesamte SalesReps-Tabelle
   - Alle Spalten in der Customers-Tabelle, jedoch nur die Zeilen, die einem bestimmten rep_key-Wert entsprechen

3. Führen Sie die folgende Anweisung aus, um eine Subskription für SalesRepData zu erstellen:

   ```
   CREATE SUBSCRIPTION
   TO SalesRepData ('rep1')
   FOR field_user;
   ```

   Der Wert rep1 ist der rep_key-Wert für den Benutzer field_user in der SalesReps-Tabelle.

Hinweis
In dieser praktischen Einführung gibt es keinen Schutz vor mehrfach vorhandenen Einträgen von Primärschlüsselwerten. Hinweise finden Sie unter „SQL Remote-Systeme erstellen“ auf Seite 9.

Ergebnisse

Die Publikation SalesRepData wird erstellt, um alle Zeilen der SalesReps-Tabelle und einige Zeilen der Customers-Tabelle zu replizieren.

Nächste Schritte

„Lektion 4: Erstellen eines SQL Remote-Nachrichtentyps“ auf Seite 162.

Lektion 4: Erstellen eines SQL Remote-Nachrichtentyps

Alle Nachrichten, die als Teil einer Replikation gesendet werden, verwenden einen Nachrichtentyp. In dieser Lektion wird der Nachrichtentyp definiert, der beim Senden der Daten und Nachrichten verwendet werden soll.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Eine Nachrichtentypbeschreibung besteht aus zwei Teilen:

- **Ein von SQL Remote unterstütztes Nachrichtensystem** Diese praktische Einführung verwendet das FILE-Nachrichtensystem. Das FILE-Nachrichtensystem ist ein einfaches System der gemeinsamen Dateibenutzung
- **Eine FILE-Adresse** Die FILE-Adresse eines Benutzers ist ein Unterverzeichnis, in das alle seine eingehenden Nachrichten gesendet werden. Eine Anwendung ruft die Nachrichten aus diesem Verzeichnis ab. In dieser praktischen Einführung lautet die FILE-Adresse der konsolidierten Datenbank "hq" und es handelt sich um ein Unterverzeichnis von *c:\tutorial*.

Aufgabe

1. Wenn Sie derzeit nicht mit der konsolidierten Datenbank (hq) verbunden sind, führen Sie folgenden Befehl aus:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=server_hq;DBF=c:\tutorial\hq.db"
   ```

2. Führen Sie die folgende Anweisung aus, um einen FILE-Nachrichtentyp zu erstellen:

   ```
   CREATE REMOTE MESSAGE
   TYPE file
   ADDRESS 'hq';
   ```

Ergebnisse

Die FILE-Nachrichtentyp wird erstellt.

Nächste Schritte

„Lektion 5: Extrahieren der entfernten Datenbank“ auf Seite 163

Lektion 5: Extrahieren der entfernten Datenbank

Erstellen Sie eine Datenbank für einen entfernten Benutzer, indem Sie die entfernte Datenbank aus der konsolidierten Datenbank (hq) extrahieren.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Die entfernte Datenbank muss so konfiguriert werden, dass sie Nachrichten senden und empfangen und an einem SQL Remote-System teilnehmen kann. Wie die konsolidierte Datenbank (hq) benötigt auch die entfernte Datenbank einen aktuellen Publikationseigentümer (CURRENT PUBLISHER), um die Quelle der ausgehenden Nachrichten festzulegen. Außerdem muss die konsolidierte Datenbank (hq) als Subskribent gekennzeichnet sein.

Führen Sie das Dienstprogramm dbxtract aus, um eine entfernte Datenbank zu erstellen, die Folgendes enthält:

- Eine Subskription für die konsolidierte Datenbank
- Eine Publikation
- Eine aktuelle Kopie der Daten

Aufgabe

1. Extrahieren Sie das Schema der entfernten Datenbank aus der konsolidierten Datenbank (hq) für den Benutzer field_user, indem Sie den folgenden Befehl im Verzeichnis *c:\tutorial* ausführen:

   ```
   dbxtract -v -c "UID=DBA;PWD=sql;SERVER=server_hq;DBF=C:\tutorial\hq.db" c:
   \tutorial field_user
   ```

   Dieser Befehl:

   - Erstellt eine SQL-Skriptdatei namens *reload.sql* im aktuellen Verzeichnis. Die *reload.sql* Datei enthält das Schema sowie die notwendigen Anweisungen, um dieses in eine neue Datenbank zu laden.
   - Erstellt eine Datendatei im Verzeichnis *c:\tutorial*.
   - Startet die Subskriptionen für den entfernten Benutzer.

2. Führen Sie im Verzeichnis *c:\tutorial* den folgenden Befehl aus, um die entfernte Datenbank (field) zu erstellen:


```
dbinit -dba DBA,sql field.db
```

Vorsicht
Speichern Sie in einer Produktionsumgebung nicht zwei replizierende Datenbanken in demselben Verzeichnis.

3. Laden Sie die Datenbankinformationen in die entfernte Datenbank (field).

   Stellen Sie aus Interactive SQL eine Verbindung mit der entfernten Datenbank (field) als Benutzer mit der SYS_REPLICATION_ADMIN_ROLE-Systemrolle her:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=server_field;DBF=c:\tutorial\field.db"
   ```

4. Führen Sie die folgende Anweisung aus, um die Datei *reload.sql* zu lesen:

   ```
   READ C:\tutorial\reload.sql;
   ```

   Die Skriptdatei *reload.sql*:

   - Erstellt einen Nachrichtentyp in der entfernten Datenbank (field).
   - Erteilt das PUBLISH-Privileg für die entfernte Datenbank (field).
   - Erstellt die Tabellen SalesReps und Customers in der entfernten Datenbank (field). Diese Tabellen enthalten dieselben Daten wie in der konsolidierten Datenbank (hq).
   - Erstellt eine Publikation, um die Daten zu identifizieren, die repliziert werden.
   - Erstellt die Subskription für die konsolidierte Datenbank (hq) und startet die Subskription.

5. Führen Sie die folgenden Anweisungen aus, um sicherzustellen, dass die Tabellen erstellt wurden:

   ```
   SELECT * FROM SalesReps;
   ```

   Die oben stehende Abfrage gibt die folgenden Daten aus der SalesReps-Tabelle zurück:

rep_key	name
rep1	Field User
rep2	Another User

   ```
   SELECT * FROM Customers;
   ```

   Die oben stehende Abfrage gibt die folgenden Daten aus der Customers-Tabelle zurück:

cust_key	name	rep_key
cust1	Ocean Sports	rep1

Ergebnisse

Die entfernte Datenbank wird für den entfernten Benutzer erstellt.

Nächste Schritte

„Lektion 6: Senden von Daten aus der konsolidierten Datenbank in die entfernte Datenbank“ auf Seite 165.

Lektion 6: Senden von Daten aus der konsolidierten Datenbank in die entfernte Datenbank

Replizieren Sie mit Interactive SQL Daten aus der konsolidierten Datenbank (hq) in die entfernte Datenbank (field).

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Viele.

Aufgabe

1. Wenn Sie derzeit nicht mit der konsolidierten Datenbank (hq) verbunden sind, führen Sie folgenden Befehl aus:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=server_hq;DBF=c:\tutorial\hq.db"
   ```

2. Führen Sie die folgenden Anweisungen aus, um der SalesReps- und Customers-Tabelle Beispieldaten hinzuzufügen:

   ```
   INSERT INTO SalesReps ( rep_key, name )
   VALUES ( 'rep3', 'Example User' );

   INSERT INTO Customers ( cust_key, name, rep_key )
   VALUES ( 'cust3', 'Land Sports', 'rep1' );
   INSERT INTO Customers ( cust_key, name, rep_key )
   VALUES ( 'cust4', 'Air Plus', 'rep2' );
   COMMIT;
   ```

3. Führen Sie die folgenden Anweisungen aus, um festzustellen, ob die Daten eingegeben wurden:

   ```
   SELECT * FROM SalesReps;
   SELECT * FROM Customers;
   ```

4. Um die Zeilen in die entfernte Datenbank (field) zu senden, führen Sie in der konsolidierten Datenbank (hq) den Nachrichtenagenten aus dem Verzeichnis *\tutorial* aus.

   ```
   dbremote -c "UID=DBA;PWD=sql;SERVER=server_hq;DBF=c:\tutorial\hq.db"
   ```

   Bei diesem Befehl wird davon ausgegangen, dass die konsolidierte Datenbank (hq) zurzeit auf dem Standardserver läuft. Wenn die Datenbank nicht ausgeführt wird, müssen Sie statt des DBN-Parameters einen DBF-Parameter mit dem Datenbankdateinamen angeben.

5. Wenn im Nachrichtenagent-Fenster `Ausführung abgeschlossen` angezeigt wird, klicken Sie auf **Herunterfahren**.

6. Wechseln Sie zu *c:\tutorial/field*.

   Eine Datei namens *hq.0* wird im Verzeichnis aufgeführt. Diese Datei enthält die Änderungen, die aus der konsolidierten Datenbank (hq) gesendet wurden.

Ergebnisse

Die Beispieldaten werden den SalesReps- und Customers-Tabellen hinzugefügt und aus der konsolidierten Datenbank in die entfernte Datenbank gesendet.

Nächste Schritte

„Lektion 7: Empfangen von Daten in der entfernten Datenbank“ auf Seite 166.

Lektion 7: Empfangen von Daten in der entfernten Datenbank

Empfangen von Daten in der entfernten Datenbank (field), die aus der konsolidierten Datenbank (hq) gesendet wurden.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Viele.

Aufgabe

1. Wenn Sie derzeit nicht mit der entfernten Datenbank (field) verbunden sind, führen Sie folgenden Befehl aus:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=server_field;DBF=c:\tutorial\field.db"
   ```

2. Führen Sie in der entfernten Datenbank (field) den Nachrichtenagenten aus dem Verzeichnis *c:\tutorial* aus:

   ```
   dbremote -c "UID=DBA;PWD=sql;SERVER=server_field;DBF=c:\tutorial
   \field.db;"
   ```

3. Wenn im Nachrichtenagent-Fenster `Ausführung abgeschlossen` angezeigt wird, klicken Sie auf **Herunterfahren**.

   Die Datei *c:\tutorial\field\hq.0* wurde durch eine Datei namens *c:\tutorial\hq\field.0* ersetzt. Die Datei *field.0* enthält die Empfangsbestätigung.

4. Vergewissern Sie sich, dass die entfernte Datenbank (field) Daten enthält.
   a. Führen Sie die folgende Anweisung aus, um den Inhalt der SalesReps-Tabelle anzuzeigen:

      ```
      SELECT * FROM SalesReps;
      ```

      Die SalesReps-Tabelle enthält beide Zeilen, die in die konsolidierte Datenbank (hq) eingegeben wurden. Dies liegt daran, dass die SalesRepData-Publikation alle Daten aus der SalesReps-Tabelle umfasst.

rep_key	name
rep1	Field User
rep2	Another User
rep3	Example User

   b. Führen Sie die folgende Anweisung aus, um den Inhalt der Customers-Tabelle anzuzeigen:

      ```
      SELECT * FROM Customers;
      ```

      Die Customers-Tabelle enthält nun auch eine Zeile mit den Land Sports-Kundendaten, die in der konsolidierten Datenbank (hq) eingegeben wurden.

cust_key	name	rep_key
cust1	Ocean Sports	rep1
cust3	Land Sports	rep1

5. Führen Sie in der konsolidierten Datenbank (hq) den Nachrichtenagenten aus dem Verzeichnis *c:\tutorial* aus:

   ```
   dbremote -c "UID=DBA;PWD=sql;SERVER=server_hq;DBF=c:\tutorial\hq.db"
   ```

   Im Verzeichnis *c:\tutorial\hq* verschwindet die Datei *field.0*.

Ergebnisse

Die Daten, die aus der konsolidierten Datenbank gesendet wurden, werden in der entfernten Datenbank empfangen.

Nächste Schritte

„Lektion 8: Senden von Daten aus der entfernten Datenbank in die konsolidierte Datenbank“ auf Seite 168.

Lektion 8: Senden von Daten aus der entfernten Datenbank in die konsolidierte Datenbank

Replizieren von Daten aus der entfernten Datenbank (field) in die konsolidierte Datenbank (hq) mit Interactive SQL.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Viele.

Aufgabe

1. Wenn Sie derzeit nicht mit der entfernten Datenbank (field) verbunden sind, führen Sie folgenden Befehl aus:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=server_hq;DBF=c:\tutorial\field.db"
   ```

2. Führen Sie die folgende Anweisung aus, um in der entfernten Datenbank (field) eine Zeile einzufügen:

   ```
   INSERT INTO Customers ( cust_key, name, rep_key )
   VALUES ( 'cust5', 'North Land Trading', 'rep1' );
   COMMIT;
   ```

3. Führen Sie im Verzeichnis *c:\tutorial* das Dienstprogramm dbremote für die entfernte Datenbank (field) aus:

   ```
   dbremote -c "UID=DBA;PWD=sql;SERVER=server_field;DBF=c:\tutorial\field.db"
   ```

   Im Verzeichnis *c:\tutorial\hq* erscheint die Datei *field.1*.

4. Wenn Sie derzeit nicht mit der konsolidierten Datenbank (hq) verbunden sind, führen Sie folgenden Befehl aus:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=server_hq;DBF=c:\tutorial\hq.db
   ```

5. Führen Sie in der konsolidierten Datenbank (hq) den Nachrichtenagenten aus dem Verzeichnis *c:\tutorial* aus:

   ```
   dbremote -c "UID=DBA;PWD=sql;SERVER=server_hq;DBF=c:\tutorial\hq.db"
   ```

6. Wenn im Nachrichtenagent-Fenster `Ausführung abgeschlossen` angezeigt wird, klicken Sie auf **Herunterfahren**.

7. Wechseln Sie zu *c:\tutorial/field.*

   Die *hq.1* Datei wurde durch eine Datei namens *hq.2* ersetzt. Die Datei *hq.2* enthält die Empfangsbestätigung.

8. Führen Sie die folgende Anweisung aus, um die Daten in der Customers-Tabelle der konsolidierten Datenbank (hq) anzuzeigen:

   ```
   SELECT * FROM Customers;
   ```

   Diese Abfrage liefert folgende Ergebnisse:

cust_key	name	rep_key
cust1	Ocean Sports	rep1
cust2	Sports Plus	rep2
cust3	Land Sports	rep1
cust4	Air Plus	rep2
cust5	North Landing Trading	rep1

Ergebnisse

Die Daten werden von der entfernten Datenbank in die konsolidierte Datenbank repliziert.

Nächste Schritte

Keiner.

Praktische Einführung: Einrichten eines Replikationssystems unter Verwendung des HTTP-Nachrichtensystems

In den Lektionen dieser praktischen Einführung erfahren Sie, wie Sie ein SQL Remote-Replikationssystem einrichten, das sowohl eine konsolidierte SQL Anywhere-Datenbank als auch eine entfernte Datenbank verwendet. Die konsolidierte Datenbank verwendet zum Replizieren von Änderungen das FILE-Nachrichtensystem, während die entfernte Datenbank dafür das HTTP-Nachrichtensystem verwendet.

In dieser praktischen Einführung führen Sie folgende Schritte durch:

- Erstellen einer konsolidierten SQL Anywhere-Datenbank und einer entfernten SQL Anywhere-Datenbank, die alle Daten in der konsolidierten Datenbank enthält.
- Erstellen einer Verzeichnisstruktur für die Speicherung der von SQL Remote generierten Nachrichten. Die konsolidierte Datenbank verwendet für den Zugriff auf die Dateien das FILE-Nachrichtensystem, während die entfernte Datenbank dafür das HTTP-Nachrichtensystem verwendet.
- Erstellen einer SQL Anywhere-Nachrichtenserver-Datenbank, die als Webserver fungiert, um Nachrichten aus der entfernten Datenbank mit dem HTTP-Protokoll zu empfangen.
- Replizieren von Daten zwischen der konsolidierten Datenbank und der entfernten Datenbank.

Rollenanforderung
Diese praktische Einführung bezieht sich auf einen Benutzer, DBA, der über die SYS_REPLICATION_ADMIN_ROLE-Systemrolle verfügen muss.

Lektion 1: Erstellen der konsolidierten Datenbank

Erstellen Sie die Verzeichnisse, die benötigt werden, um die Datenbanken und ihre Transaktionslogs sowie die Verzeichnisstruktur für die Nachrichten zu speichern. Außerdem legen Sie das Schema der konsolidierten Datenbank fest, einschließlich der Erstellung des entfernten Benutzers sowie der Publikation und der Subskription, die für die Replikation der Daten benötigt werden.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Wenn SQL Remote in der konsolidierten Datenbank ausgeführt wird, verwendet es das FILE-Nachrichtensystem, um Nachrichten zu senden und zu empfangen, aber die entfernte Datenbank verwendet das HTTP-Nachrichtensystem.

Aufgabe

1. Erstellen Sie die folgenden Verzeichnisse für die konsolidierte Datenbank, die entfernte Datenbank und die Nachrichtenserver-Datenbank:
   - *c:\tutorial*
   - *c:\tutorial\cons*
   - *c:\tutorial\rem*
   - *c:\tutorial\msgsrv*

2. Erstellen Sie die folgenden Verzeichnisse für die von der konsolidierten Datenbank und der entfernten Datenbank generierten Nachrichtendateien:
   - *c:\tutorial\messages*
   - *c:\tutorial\messages\cons*
   - *c:\tutorial\messages\rem*

3. Führen Sie im Verzeichnis *c:\tutorial\cons* den folgenden Befehl aus, um die konsolidierte Datenbank (cons) zu erstellen:

   ```
   dbinit -dba DBA,sql cons.db
   ```

4. Verbinden Sie sich über Interactive SQL als Benutzer mit der SYS_REPLICATION_ADMIN_ROLE-Systemrolle mit der konsolidierten Datenbank (cons) und stellen Sie beim Trennen der Verbindung sicher, dass Sie die Datenbank laufen lassen, indem Sie für den AutoStop-Verbindungsparameter "AutoStop=NO" angeben:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=cons;DBF=c:\tutorial\cons
   \cons.db;autostop=no"
   ```

5. Führen Sie die folgende Anweisung aus, um die globale Datenbank-ID für die konsolidierte Datenbank (cons) festzulegen (die benötigt wird, damit bei Verwendung des GLOBAL AUTOINCREMENT-Standardwerts für alle Datenbanken unterschiedliche Primärschlüssel gewählt werden):

   ```
   SET OPTION public.global_database_id=0;
   ```

6. Das Schema für die Datenbank in dieser praktischen Einführung besteht aus einer einzelnen Tabelle, die repliziert wird, und alle Spalten und Zeilen aus der Tabelle werden für jeden entfernten Benutzer repliziert. Führen Sie die folgenden Anweisungen in der konsolidierten Datenbank (cons) aus, um die einzelne Tabelle in der Datenbank zu erstellen:

   ```
   CREATE TABLE employees (
        employee_id BIGINT NOT NULL DEFAULT GLOBAL AUTOINCREMENT(1000000)
   PRIMARY KEY,
        first_name VARCHAR(128) NOT NULL,
        last_name VARCHAR(128) NOT NULL,
        hire_date TIMESTAMP NOT NULL DEFAULT TIMESTAMP
   );
   ```

7. Führen Sie die folgenden Anweisungen in der konsolidierten Datenbank (cons) aus, um der employees-Tabelle Beispieldaten hinzuzufügen:

```
INSERT INTO employees (first_name, last_name) VALUES ('Kelly', 'Meloy');
INSERT INTO employees (first_name, last_name) VALUES ('Melisa', 'Boysen');
COMMIT;
```

8. Führen Sie die folgende Anweisung in der konsolidierten Datenbank (cons) aus, um zu bestätigen, dass die Tabelle erstellt und mit Daten gefüllt wurde:

   ```
   SELECT * FROM employees;
   ```

   Die Abfrage gibt die folgenden Daten aus der employees-Tabelle zurück, obwohl die hire_date-Spalte die Uhrzeit enthält, zu der Sie die Zeile eingefügt haben, und nicht die in der folgenden Tabelle aufgeführten Werte:

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	2011-03-25 08:27:56.310
2	Melisa	Boysen	2011-03-25 08:27:56.310

9. In dieser praktischen Einführung werden dem Publikationseigentümer und dem entfernten Benutzer keine Kennwörter zugeordnet. Solange die Benutzer in der Datenbank vorhanden sind, können Sie also für diese Benutzer keine Verbindung zur Datenbank herstellen. Führen Sie die folgende Anweisung aus, um den Benutzer cons mit den Privilegien CONNECT und PUBLISH zu erstellen:

   ```
   GRANT CONNECT TO cons;
   GRANT PUBLISH TO cons;
   ```

10. Aus Gründen der Performance kann das HTTP-Nachrichtensystem nur in der entfernten Datenbank verwendet werden und nicht in der konsolidierten Datenbank. Die folgenden Anweisungen konfigurieren die Verwendung des FILE-basierten Nachrichtensystems in der konsolidierten Datenbank:

    ```
    CREATE REMOTE MESSAGE TYPE FILE ADDRESS 'cons';
    SET REMOTE FILE OPTION public.directory='c:\\tutorial\\messages';
    SET REMOTE FILE OPTION public.debug='yes';
    ```

11. Führen Sie die folgenden Anweisungen aus, um den entfernten Benutzer rem ohne Kennwort zu erstellen und ihm REMOTE-Privilegien zu erteilen, während Sie die Adresse des Benutzers im FILE-Nachrichtensystem festlegen:

    ```
    GRANT CONNECT TO rem;
    GRANT REMOTE TO rem TYPE FILE ADDRESS 'rem';
    ```

12. Eine Publikation beschreibt die Datenmenge, die repliziert werden soll. Erstellen Sie eine Publikation namens pub_employees, die alle Zeilen der employees-Tabelle repliziert. Sie subskribieren einen Benutzer für eine Publikation, indem Sie eine Subskription erstellen.

    ```
    CREATE PUBLICATION pub_employees ( TABLE employees );
    CREATE SUBSCRIPTION TO pub_employees FOR rem;
    ```

13. Trennen Sie die Verbindung zu Interactive SQL.

Ergebnisse

In dieser Lektion erstellen Sie die Verzeichnisse, die benötigt werden, um die Datenbanken und deren Transaktionslogs sowie die Verzeichnisstruktur für die Nachrichten zu speichern. Das Schema der konsolidierten Datenbank wird definiert, einschließlich der Erstellung des entfernten Benutzers sowie der Publikation und der Subskription, die für die Replikation der Daten benötigt werden.

Nächste Schritte

Gehen Sie weiter zu „Lektion 2: Erstellen des Nachrichtenservers“ auf Seite 174.

Lektion 2: Erstellen des Nachrichtenservers

Obwohl es möglich ist, die konsolidierte Datenbank als Nachrichtenserver einzusetzen, verwenden Sie in dieser praktischen Einführung einen separaten Datenbankserver als Host für den Nachrichtenserver. Dies trägt dazu bei, die durch die Verarbeitung von Nachrichten anfallende Last zwischen den beiden Datenbankservern zu verteilen, und erzeugt außerdem eine zusätzliche Sicherheitsstufe, weil Sie den HTTP-Zugriff auf Ihre konsolidierte Datenbank nicht freigegeben haben.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Führen Sie im Verzeichnis *c:\tutorial\msgsrv* den folgenden Befehl aus, um die Nachrichtenserver-Datenbank (msgsrv) zu erstellen:

   ```
   dbinit -dba DBA,sql msgsrv.db
   ```

2. Starten Sie den Nachrichtenserver:

   ```
   dbeng16 -n msgsrv c:\tutorial\msgsrv\msgsrv.db -xs http(port=8033)
   ```

   -xs http(8033) ist in der Befehlszeile erforderlich, da dieser Datenbankserver HTTP-Anforderungen aus der entfernten Datenbank akzeptiert und auf die Nachrichtendateien im Verzeichnis *c:\tutorial\messages* zugreift. Obwohl beim Starten des Datenbankservers keine Webdienste festgelegt wurden, werden sie in dieser Lektion erstellt. Außerdem wurde nur der Personal Datenbankserver gestartet, sodass nur SQL Remote Prozesse auf diesem Computer über HTTP mit dem Nachrichtenserver kommunizieren können. In einer Produktionsumgebung wird gewöhnlich der Netzwerkserver verwendet, sodass SQL Remote-Prozesse auf anderen Computern ebenfalls Zugriff auf die Webdienste haben. Weitere Hinweise zur Verwendung der Option -xs finden Sie unter „Datenbankserveroption -xs “ [*SQL Anywhere Server - Datenbankadministration*].

3. Beim Erstellen eines separaten Nachrichtenservers müssen Sie einen großen Teil des Schemas der konsolidierten Datenbank in den Nachrichtenserver kopieren, besonders Informationen über die festgelegten entfernten Benutzer und deren Adressen. Obwohl Sie diese Aufgabe manuell ausführen können, besteht die einfachste Methode darin, mit dem Dienstprogramm dbunload eine neue Datenbank zu erstellen, die dasselbe Schema besitzt wie die konsolidierte Datenbank:

```
dbunload -n -xx -ac "SERVER=msgsrv;DBN=msgsrv;UID=DBA;PWD=sql" -c
"SERVER=cons;DBN=cons;UID=DBA;PWD=sql"
```

Die im dbunload-Befehl verwendeten Optionen bewirken Folgendes:

- **-n** Gibt an, dass nur das Schema entladen werden soll, und auf dem Nachrichtenserver werden keine Daten aus der konsolidierten Datenbank hinzugefügt.
- **-xx** Führt einen externen Entlade- und Aktualisierungsvorgang aus, der erforderlich ist, wenn beide beteiligten Datenbanken bereits laufen.
- **-ac "SERVER=msgsrv;DBN=msgsrv;UID=DBA;PWD=sql"** Legt die Zielverbindung für den Entladevorgang fest, in dieser Lektion den Nachrichtenserver.
- **-c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql"** Legt die Quellverbindung für den Entladevorgang fest, in dieser Lektion die konsolidierte Datenbank.

4. Verbinden Sie sich über Interactive SQL mit der Nachrichtenserverdatenbank (msgsrv) als Benutzer mit der entsprechenden SYS_REPLICATION_ADMIN_ROLE Systemrolle:

   ```
   dbisql -c "SERVER=msgsrv;DBN=msgsrv;UID=DBA;PWD=sql"
   ```

   In Lektion 1 haben Sie keine Kennwörter für den Publikationseigentümer (cons) und den entfernten Benutzer (rem) erstellt, sodass keiner dieser Benutzer eine Verbindung mit der konsolidierten Datenbank herstellen kann. Ein Kennwort für diese Benutzer ist im Nachrichtenserver erforderlich, da die von entfernten Benutzern kommenden HTTP-Anforderungen den Publikationseigentümer der entfernten Datenbank verwenden und das Kennwort, das Sie zum Authentifizieren beim Nachrichtenserver angeben. Führen Sie die folgenden Anweisungen in der Nachrichtenserver-Datenbank (msgsrv) aus, um Kennwörter für den Publikationseigentümer und den entfernten Benutzer festzulegen:

   ```
   GRANT CONNECT TO cons IDENTIFIED BY cons;
   GRANT CONNECT TO rem IDENTIFIED BY rem;
   ```

5. Beim ersten Initialisieren einer Datenbank wird keiner der Webdienste festgelegt, die benötigt werden, um HTTP-Anforderungen von entfernten Benutzern zu akzeptieren, und es werden keine Definitionen erstellt, die der Datenbankserver für den Zugriff auf das Verzeichnis verwenden könnte, in dem die Nachrichtendateien gespeichert werden. Die Erstellung dieser Objekte wird mit der gespeicherten Prozedur sr_add_message_server automatisiert, in der durch einen optionalen Parameter der Eigentümer aller Objekte angegeben werden kann. Führen Sie die folgenden Anweisungen für die Nachrichtenserver-Datenbank (msgsrv) aus, um alle für den Nachrichtenserver erforderlichen Objekte festzulegen, und geben Sie an, dass der Benutzer cons Eigentümer aller Objekte ist:

   ```
   CREATE ROLE FOR USER cons;
   SET REMOTE http OPTION cons.root_directory='c:\\tutorial\\messages';
   CALL sr_add_message_server( 'cons' );
   COMMIT;
   ```

   Weitere Hinweise finden Sie unter „sr_add_message_server-Systemprozedur“ auf Seite 228.

6. Trennen Sie die Verbindung zu Interactive SQL.

Ergebnisse

Ein weiterer Datenbankserver dient nun als Host für den Nachrichtenserver.

Nächste Schritte

Gehen Sie weiter zu „Lektion 3: Erstellen der entfernten Datenbank“ auf Seite 176.

Lektion 3: Erstellen der entfernten Datenbank

Extrahieren Sie die entfernte Datenbank und ersetzen Sie das FILE-Nachrichtensystem in der entfernten Datenbank durch das HTTP-Nachrichtensystem.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Führen Sie im Verzeichnis *c:\tutorial\rem* den folgenden Befehl aus, um die entfernte Datenbank (rem) zu erstellen:

   ```
   dbinit -dba DBA,sql rem.db
   ```

2. In dieser Lektion verwenden Sie dbxtract, um die entfernte Datenbank zu erstellen. Führen Sie den folgenden Befehl aus, um die Datenbank für den Benutzer rem aus der konsolidierten Datenbank zu extrahieren, und lassen Sie den Datenbankserver für die entfernte Datenbank nach der Extraktion laufen:

   ```
   dbxtract -xx -ac "SERVER=rem;DBN=rem;dbf=c:\tutorial\rem
   \rem.db;UID=DBA;PWD=sql;autostop=no" -c
   "SERVER=cons;DBN=cons;UID=DBA;PWD=sql" rem
   ```

3. Wenn Sie derzeit nicht mit der entfernten Datenbank (rem) verbunden sind, führen Sie folgenden Befehl aus:

   ```
   dbisql -c "SERVER=rem;DBN=rem;UID=DBA;PWD=sql"
   ```

4. Die konsolidierte Datenbank verwendet das FILE-Nachrichtensystem. Deshalb werden beim Ausführen von dbxtract SQL Remote-Definitionen unter der Annahme erstellt, dass die entfernte Datenbank (rem) ebenfalls das FILE-Nachrichtensystem verwendet. Wenn Sie die entfernte Datenbank für die Verwendung des HTTP-Nachrichtensystems konfigurieren möchten, müssen Sie die folgenden SQL-Anweisungen in der entfernten Datenbank (rem) ausführen, um das FILE-Nachrichtensystem für diese entfernte Datenbank zu entfernen:

   ```
   CREATE REMOTE TYPE "FILE" ADDRESS '';
   SET REMOTE FILE OPTION public.directory='';
   SET REMOTE FILE OPTION public.debug='';
   ```

5. Führen Sie die folgenden Anweisungen in der entfernten Datenbank (rem) aus, um das HTTP-Nachrichtensystem für diese entfernte Datenbank zu konfigurieren:

   ```
   CREATE REMOTE TYPE "HTTP" ADDRESS 'rem';
   GRANT CONSOLIDATE TO "cons" TYPE "HTTP" ADDRESS 'cons';
   SET REMOTE HTTP OPTION public.user_name='rem';
   SET REMOTE HTTP OPTION public.password='rem';
   SET REMOTE HTTP OPTION public.debug='yes';
   ```

```
SET REMOTE HTTP OPTION public.https='no';
SET REMOTE HTTP OPTION public.url='localhost:8033';
COMMIT;
```

6. Überprüfen Sie, ob die entfernte Datenbank (rem) nach der Extraktion die beiden Datenzeilen enthält, die bereits in der konsolidierten Datenbank vorhanden waren. Führen Sie die folgende Anweisung aus, um den Inhalt der employees-Tabelle anzuzeigen:

   ```
   SELECT * FROM employees;
   ```

   Die Abfrage gibt die folgenden Daten aus der employees-Tabelle zurück, obwohl die hire_date-Spalte die Uhrzeit enthält, zu der Sie die Zeile eingefügt haben, und nicht die in der folgenden Tabelle aufgeführten Werte:

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	2011-03-25 08:27:56.310
2	Melisa	Boysen	2011-03-25 08:27:56.310

7. Trennen Sie die Verbindung zu Interactive SQL.

Ergebnisse

Die entfernte Datenbank wird extrahiert und das FILE-Nachrichtensystem in der entfernten Datenbank wird durch das HTTP-Nachrichtensystem ersetzt.

Nächste Schritte

Gehen Sie weiter zu „Lektion 4: Hinzufügen und Replizieren von Daten in konsolidierter und entfernter Datenbank“ auf Seite 177.

Lektion 4: Hinzufügen und Replizieren von Daten in konsolidierter und entfernter Datenbank

Fügen Sie Daten zur konsolidierten und entfernten Datenbank hinzu, führen Sie SQL Remote aus, um die Änderungen zu replizieren, und bestätigen Sie anschließend, dass die Daten in beiden Datenbanken konsistent sind.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Wenn Sie derzeit nicht mit der konsolidierten Datenbank (cons) verbunden sind, führen Sie folgenden Befehl aus:

   ```
   dbisql -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql"
   ```

2. Führen Sie die folgenden Anweisungen in der konsolidierten Datenbank (cons) aus, um der employees-Tabelle zusätzliche Beispieldaten hinzuzufügen:

   ```
   INSERT INTO employees (first_name, last_name) VALUES ('Javier', 'Spoor');
   COMMIT;
   ```

3. Führen Sie die folgenden Anweisungen in der entfernten Datenbank (rem) aus, um der employees-Tabelle zusätzliche Beispieldaten hinzuzufügen:

   ```
   INSERT INTO employees (first_name, last_name) VALUES ('Nelson',
   'Kreitzer');
   COMMIT;
   ```

4. Führen Sie in der konsolidierten Datenbank (cons) den Nachrichtenagenten aus:

   ```
   dbremote -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql" -qc -v -o c:\tutorial
   \cons1.txt
   ```

   Dieser durchsucht das Transaktionslog der konsolidierten Datenbank (cons) und generiert mit dem FILE-Nachrichtensystem eine Nachricht für die entfernte Datenbank (rem). Da der Systemparameter für Fehlerbehebungsmeldungen in der konsolidierten Datenbank auf das FILE-Nachrichtensystem eingestellt wurde, können Sie die Datei *c:\tutorial\cons1.txt* auf Fehlerbehebungsmeldungen überprüfen, die anzeigen, dass Meldungen in das Verzeichnis *c:\tutorial\messages\rem* geschrieben werden. Beispiel:

   ```
   I. 2011-03-25 11:03:31. Processing transactions from active transaction
   log
   I. 2011-03-25 11:03:31. Sending message to "rem"
   (0-0000000000-0000550994-0)
   I. 2011-03-25 11:03:31. sopen "c:\tutorial\messages\rem\cons.0"
   I. 2011-03-25 11:03:31. write " c:\tutorial\messages\rem\cons.0"
   I. 2011-03-25 11:03:31. close " c:\tutorial\messages\rem\cons.0"
   ```

5. Führen Sie in der entfernten Datenbank (rem) den Nachrichtenagenten aus:

   ```
   dbremote -c "SERVER=rem;DBN=rem;UID=DBA;PWD=sql" -qc -v -o c:\tutorial
   \rem.txt
   ```

   Unter Verwendung des HTTP-Messaging-Systems wird durch diesen Befehl die soeben von der konsolidierten Datenbank generierte Nachricht empfangen und übernommen. Danach wird das Transaktionslog durchsucht und eine Nachricht mit der neuen Zeile, die in der entfernten Datenbank hinzugefügt wurde, wird an die konsolidierte Datenbank zurückgesendet. Da der Systemparameter für Fehlerbehebungsmeldungen in der konsolidierten Datenbank auf das HTTP-Nachrichtensystem eingestellt wurde, können Sie die Datei *c:\tutorial\rem.txt* auf Fehlerbehebungsmeldungen überprüfen, die anzeigen, dass das HTTP-Nachrichtensystem verwendet wird. Beispiel:

   ```
   I. 2011-03-25 11:10:02. Sending message to "cons"
   (0-0000000000-0000557411-0)
   I. 2011-03-25 11:10:02. HTTPWriteMessage "rem.0"
   I. 2011-03-25 11:10:02. HTTPWriteMessage: success -- filename  "rem.0"
   I. 2011-03-25 11:10:02. HTTPDisconnect
   ```

6. Führen Sie in der konsolidierten Datenbank (cons) den Nachrichtenagenten aus:

   ```
   dbremote -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql" -qc -v -o c:\tutorial
   \cons2.txt
   ```

Unter Verwendung des FILE-basierten Nachrichtensystems wird durch diesen Befehl die soeben von der entfernten Datenbank generierte Nachricht empfangen und übernommen.

7. Um zu überprüfen, ob die konsolidierte Datenbank alle vier Datenzeilen enthält, können Sie die folgende Anweisung ausführen und den Inhalt der employees-Tabelle anzeigen:

   ```
   SELECT * FROM employees;
   ```

   Die Abfrage gibt die folgenden Daten aus der employees-Tabelle zurück, obwohl die hire_date-Spalte die Uhrzeit enthält, zu der Sie die Zeile eingefügt haben, und nicht die in der folgenden Tabelle aufgeführten Werte:

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	2011-03-25 08:27:56.310
2	Melisa	Boysen	2011-03-25 08:27:56.310
3	Javier	Spoor	2011-03-25 08:30:26.110
102000001	Nelson	Kreitzer	2011-03-25 08:31:51.970

8. Überprüfen Sie, ob die entfernte Datenbank (rem) alle vier Datenzeilen enthält, indem Sie die folgende Anweisung ausführen, um den Inhalt der employees-Tabelle anzuzeigen:

   ```
   SELECT * FROM employees;
   ```

   Die Abfrage gibt die folgenden Daten aus der employees-Tabelle zurück, obwohl die hire_date-Spalte die Uhrzeit enthält, zu der Sie die Zeile eingefügt haben, und nicht die in der folgenden Tabelle aufgeführten Werte:

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	2011-03-25 08:27:56.310
2	Melisa	Boysen	2011-03-25 08:27:56.310
3	Javier	Spoor	2011-03-25 08:30:26.110
102000001	Nelson	Kreitzer	2011-03-25 08:31:51.970

9. Trennen Sie die Verbindung zu Interactive SQL.

Ergebnisse

Die Daten werden der konsolidierten und der entfernten Datenbank hinzugefügt, die Änderungen werden repliziert, und die Datenkonsistenz wurde überprüft.

Nächste Schritte

Gehen Sie weiter zu „Lektion 5: Aufräumen“ auf Seite 180.

Lektion 5: Aufräumen

Fahren Sie die drei Datenbankserver herunter, die Sie während dieser praktischen Einführung gestartet haben.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Führen Sie den folgenden Befehl aus, um die entfernte Datenbank herunterzufahren:

   ```
   dbstop -y -c "SERVER=rem;DBN=rem;UID=DBA;PWD=sql"
   ```

2. Führen Sie den folgenden Befehl aus, um den Nachrichtenserver herunterzufahren:

   ```
   dbstop -y -c "SERVER=msgsrv;DBN=msgsrv;UID=DBA;PWD=sql"
   ```

3. Führen Sie den folgenden Befehl aus, um die konsolidierte Datenbank herunterzufahren:

   ```
   dbstop -y -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql"
   ```

Ergebnisse

Die drei Datenbankserver werden heruntergefahren.

Nächste Schritte

Keiner.

Praktische Einführung: Einrichten eines Replikationssystems unter Verwendung des HTTP-Nachrichtensystems mit der konsolidierten Datenbank als Nachrichtenserver

In den Lektionen dieser praktischen Einführung erfahren Sie, wie Sie ein SQL Remote-Replikationssystem einrichten, das sowohl eine konsolidierte SQL Anywhere-Datenbank als auch eine entfernte Datenbank verwendet. Die konsolidierte Datenbank verwendet zum Replizieren von Änderungen das FILE-Nachrichtensystem, während die entfernte Datenbank dafür das HTTP-Nachrichtensystem verwendet.

In dieser praktischen Einführung führen Sie folgende Schritte durch:

- Erstellen einer konsolidierten SQL Anywhere-Datenbank und einer entfernten SQL Anywhere-Datenbank, die alle Daten in der konsolidierten Datenbank enthält.
- Erstellen einer Verzeichnisstruktur für die Speicherung der von SQL Remote generierten Nachrichten. Die konsolidierte Datenbank verwendet für den Zugriff auf die Dateien das FILE-Nachrichtensystem, während die entfernte Datenbank dafür das HTTP-Nachrichtensystem verwendet.
- Konfigurieren der konsolidierten Datenbank als Nachrichtenserver für das HTTP-Nachrichtensystem.
- Erstellen einer entfernten Datenbank, die Nachrichten mithilfe des HTTP-Messaging-Systems sendet.
- Replizieren von Daten zwischen der konsolidierten Datenbank und der entfernten Datenbank.

Rollenanforderung
Diese praktische Einführung bezieht sich auf einen Benutzer, DBA, der über die SYS_REPLICATION_ADMIN_ROLE-Systemrolle verfügen muss.

Lektion 1: Erstellen der konsolidierten Datenbank

Erstellen Sie die Verzeichnisse, die benötigt werden, um die Datenbanken und deren Transaktionslogs zu speichern, sowie die Verzeichnisstruktur für die Nachrichten. Außerdem legen Sie das Schema der konsolidierten Datenbank fest, einschließlich der Erstellung des entfernten Benutzers sowie der Publikation und der Subskription, die für die Replikation der Daten benötigt werden.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Wenn SQL Remote in der konsolidierten Datenbank ausgeführt wird, verwendet es das FILE-Nachrichtensystem, um Nachrichten zu senden und zu empfangen, aber die entfernte Datenbank verwendet das HTTP-Nachrichtensystem.

Aufgabe

1. Erstellen Sie die folgenden Verzeichnisse für die konsolidierte Datenbank und die entfernte Datenbank:
   - *c:\tutorial*
   - *c:\tutorial\cons*
   - *c:\tutorial\rem*
2. Erstellen Sie die folgenden Verzeichnisse für die von der konsolidierten Datenbank, der entfernten Datenbank und der Nachrichtenserver-Datenbank generierten Nachrichtendateien:
   - *c:\tutorial\messages*
   - *c:\tutorial\messages\cons*
   - *c:\tutorial\messages\rem*
3. Führen Sie im Verzeichnis *c:\tutorial\cons* den folgenden Befehl aus, um die konsolidierte Datenbank (cons) zu erstellen:

   ```
   dbinit -dba DBA,sql cons.db
   ```
4. Starten Sie die konsolidierte Datenbank:

   ```
   dbeng16 -n cons c:\tutorial\cons\cons.db -xs http(port=8033)
   ```

   -xs http(8033) ist in der Befehlszeile erforderlich, da dieser Datenbankserver HTTP-Anforderungen aus der entfernten Datenbank akzeptiert und auf die Nachrichtendateien im Verzeichnis *c:\tutorial\messages* zugreift. Während beim Start des Datenbankservers keine Webdienste definiert sind, werden sie in der nächsten Lektion erstellt. In dieser Lektion starten Sie nur den Personal Datenbankserver, sodass nur SQL Remote Prozesse auf diesem Computer über HTTP mit dem Nachrichtenserver kommunizieren können. In einer Produktionsumgebung wird gewöhnlich der Netzwerkserver verwendet, sodass SQL Remote-Prozesse auf anderen Computern ebenfalls Zugriff auf die Webdienste haben. Weitere Hinweise zur Verwendung der Option -xs finden Sie unter „Datenbankserveroption -xs “ [*SQL Anywhere Server - Datenbankadministration*].
5. Verbinden Sie sich mit Interactive SQL mit der konsolidierten Datenbank (cons) als Benutzer mit der entsprechenden SYS_REPLICATION_ADMIN_ROLE-Systemrolle:

   ```
   dbisql -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql"
   ```
6. Führen Sie die folgende Anweisung aus, um die globale Datenbank-ID für die konsolidierte Datenbank (cons) festzulegen (die benötigt wird, damit bei Verwendung des GLOBAL AUTOINCREMENT-Standardwerts für alle Datenbanken unterschiedliche Primärschlüssel gewählt werden):

   ```
   SET OPTION public.global_database_id=0;
   ```

7. Das Schema für die Datenbank in dieser praktischen Einführung besteht aus einer einzelnen Tabelle und alle Spalten und Zeilen aus der Tabelle werden für jeden entfernten Benutzer repliziert. Führen Sie die folgenden Anweisungen in der konsolidierten Datenbank (cons) aus, um die einzelne Tabelle in der Datenbank zu erstellen:

   ```
   CREATE TABLE employees (
        employee_id BIGINT NOT NULL DEFAULT GLOBAL AUTOINCREMENT(1000000)
   PRIMARY KEY,
        first_name VARCHAR(128) NOT NULL,
        last_name VARCHAR(128) NOT NULL,
        hire_date TIMESTAMP NOT NULL DEFAULT TIMESTAMP
   );
   ```

8. Führen Sie die folgenden Anweisungen in der konsolidierten Datenbank (cons) aus, um der employees-Tabelle Beispieldaten hinzuzufügen:

   ```
   INSERT INTO employees (first_name, last_name) VALUES ('Kelly', 'Meloy');
   INSERT INTO employees (first_name, last_name) VALUES ('Melisa', 'Boysen');
   COMMIT;
   ```

9. Führen Sie die folgende Anweisung in der konsolidierten Datenbank (cons) aus, um zu bestätigen, dass die Tabelle erstellt und mit Daten gefüllt wurde:

   ```
   SELECT * FROM employees;
   ```

   Die Abfrage gibt die folgenden Daten aus der employees-Tabelle zurück, obwohl die hire_date-Spalte die Uhrzeit enthält, zu der Sie die Zeile eingefügt haben, und nicht die in der folgenden Tabelle aufgeführten Werte:

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	2011-03-25 08:27:56.310
2	Melisa	Boysen	2011-03-25 08:27:56.310

10. In dieser praktischen Einführung werden dem Publikationseigentümer und dem entfernten Benutzer Kennwörter zugeordnet, da die konsolidierte Datenbank als Nachrichtenserver für das HTTP-Nachrichtensystem fungiert. Führen Sie die folgende Anweisung aus, um den Benutzer cons mit den Privilegien CONNECT und PUBLISH zu erstellen:

    ```
    GRANT CONNECT TO cons;
    GRANT PUBLISH TO cons;
    ```

11. Aus Gründen der Performance kann das HTTP-Nachrichtensystem nur in der entfernten Datenbank verwendet werden und nicht in der konsolidierten Datenbank. Die folgenden Anweisungen konfigurieren die Verwendung des FILE-basierten Nachrichtensystems in der konsolidierten Datenbank:

    ```
    CREATE REMOTE MESSAGE TYPE FILE ADDRESS 'cons';
    SET REMOTE FILE OPTION public.directory='c:\\tutorial\\messages';
    SET REMOTE FILE OPTION public.debug='yes';
    ```

12. Führen Sie die folgenden Anweisungen aus, um den entfernten Benutzer rem ohne Kennwort zu erstellen und ihm REMOTE-Privilegien zu erteilen, während Sie die Adresse des Benutzers im FILE-Nachrichtensystem festlegen:

```
GRANT CONNECT TO rem IDENTIFIED BY rem;
GRANT REMOTE TO rem TYPE FILE ADDRESS 'rem';
```

13. Eine Publikation beschreibt die Datenmenge, die repliziert werden soll. Erstellen Sie eine Publikation namens pub_employees, die alle Zeilen der employees-Tabelle repliziert. Sie subskribieren einen Benutzer für eine Publikation, indem Sie eine Subskription erstellen.

    ```
    CREATE PUBLICATION pub_employees ( TABLE employees );
    CREATE SUBSCRIPTION TO pub_employees FOR rem;
    ```

14. Trennen Sie die Verbindung zu Interactive SQL.

Ergebnisse

Die Verzeichnisse, die benötigt werden, um die Datenbanken und deren Transaktionslogs zu speichern, sowie die Verzeichnisstruktur für die Nachrichten werden erstellt. Das Schema der konsolidierten Datenbank wird definiert, einschließlich der Erstellung des entfernten Benutzers sowie der Publikation und der Subskription, die für die Replikation der Daten benötigt werden.

Nächste Schritte

Gehen Sie weiter zu „Lektion 2: Konfigurieren der konsolidierten Datenbank als Nachrichtenserver“ auf Seite 184.

Lektion 2: Konfigurieren der konsolidierten Datenbank als Nachrichtenserver

Konfigurieren der konsolidierten Datenbank als Nachrichtenserver für das HTTP-Nachrichtensystem.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Es ist auch möglich, das eine separate Datenbank und einen separaten Datenbankserver als Nachrichtenserver zu konfigurieren.

Aufgabe

1. Verbinden Sie sich über Interactive SQL als Benutzer mit der SYS_REPLICATION_ADMIN_ROLE-Systemrolle mit der Datenbank:

   ```
   dbisql -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql"
   ```

2. Beim ersten Initialisieren einer Datenbank wird keiner der Webdienste festgelegt, die benötigt werden, um HTTP-Anforderungen von entfernten Benutzern zu akzeptieren, und es werden keine Definitionen erstellt, die der Datenbankserver für den Zugriff auf das Verzeichnis verwenden könnte, in dem die Nachrichtendateien gespeichert werden. Die Erstellung dieser Objekte wird mit der gespeicherten

Prozedur sr_add_message_server automatisiert, in der durch einen optionalen Parameter der Eigentümer aller Objekte angegeben werden kann. Führen Sie die folgenden Anweisungen für die konsolidierte Datenbank (cons) aus, um alle für den Nachrichtenserver erforderlichen Objekte festzulegen, und geben Sie an, dass der Benutzer cons Eigentümer aller Objekte ist:

```
CREATE ROLE FOR USER cons;
SET REMOTE http OPTION cons.root_directory='c:\\tutorial\\messages';
CALL sr_add_message_server( 'cons' );
COMMIT;
```

3. Trennen Sie die Verbindung zu Interactive SQL.

Ergebnisse

Die konsolidierte Datenbank wird als Nachrichtenserver für das HTTP-Nachrichtensystem konfiguriert.

Nächste Schritte

Gehen Sie weiter zu „Lektion 3: Erstellen der entfernten Datenbank“ auf Seite 185.

Lektion 3: Erstellen der entfernten Datenbank

Extrahieren Sie die entfernte Datenbank und ersetzen Sie anschließend das FILE-Nachrichtensystem in der entfernten Datenbank durch das HTTP-Nachrichtensystem.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Führen Sie im Verzeichnis *c:\tutorial\rem* den folgenden Befehl aus, um die entfernte Datenbank (rem) zu erstellen:

   ```
   dbinit -dba DBA,sql rem.db
   ```

2. Starten Sie die konsolidierte Datenbank:

   ```
   dbeng16 -n rem c:\tutorial\rem\rem.db
   ```

3. In dieser Lektion verwenden Sie dbxtract, um die entfernte Datenbank zu erstellen. Führen Sie den folgenden Befehl aus, um die Datenbank für den Benutzer rem aus der konsolidierten Datenbank zu extrahieren, und lassen Sie den Datenbankserver für die entfernte Datenbank nach der Extraktion laufen:

   ```
   dbxtract -xx -ac "SERVER=rem;DBN=rem;dbf=c:\tutorial\rem
   \rem.db;UID=DBA;PWD=sql;autostop=no" -c
   "SERVER=cons;DBN=cons;UID=DBA;PWD=sql" rem
   ```

   Wenn Sie derzeit nicht mit der entfernten Datenbank (rem) verbunden sind, führen Sie folgenden Befehl aus:


```
dbisql -c "SERVER=rem;DBN=rem;UID=DBA;PWD=sql"
```

4. Die konsolidierte Datenbank verwendet das FILE-Nachrichtensystem. Deshalb werden beim Ausführen von dbxtract SQL Remote-Definitionen unter der Annahme erstellt, dass die entfernte Datenbank (rem) ebenfalls das FILE-Nachrichtensystem verwendet. Wenn Sie die entfernte Datenbank für die Verwendung des HTTP-Nachrichtensystems einstellen möchten, müssen Sie die folgenden SQL-Anweisungen in der entfernten Datenbank (rem) ausführen, um das FILE-Nachrichtensystem für diese entfernte Datenbank zu entfernen:

   ```
   CREATE REMOTE TYPE "FILE" ADDRESS '';
   SET REMOTE FILE OPTION public.directory='';
   SET REMOTE FILE OPTION public.debug='';
   ```

5. Führen Sie die folgenden Anweisungen in der entfernten Datenbank (rem) aus, um das HTTP-Nachrichtensystem für diese entfernte Datenbank zu konfigurieren:

   ```
   CREATE REMOTE TYPE "HTTP" ADDRESS 'rem';
   GRANT CONSOLIDATE TO "cons" TYPE "HTTP" ADDRESS 'cons';
   SET REMOTE HTTP OPTION public.user_name='rem';
   SET REMOTE HTTP OPTION public.password='rem';
   SET REMOTE HTTP OPTION public.debug='yes';
   SET REMOTE HTTP OPTION public.https='no';
   SET REMOTE HTTP OPTION public.url='localhost:8033';
   COMMIT;
   ```

6. Prüfen Sie, ob die employees-Tabelle in der entfernten Datenbank (rem) nach der Extraktion die beiden Datenzeilen enthält, die bereits in der konsolidierten Datenbank vorhanden waren. Führen Sie die folgende Anweisung aus, um den Inhalt der employees-Tabelle anzuzeigen:

   ```
   SELECT * FROM employees
   ```

   Die Abfrage gibt die folgenden Daten aus der employees-Tabelle zurück, obwohl die hire_date-Spalte die Uhrzeit enthält, zu der Sie die Zeile eingefügt haben, und nicht die in der folgenden Tabelle aufgeführten Daten:

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	2011-03-25 08:27:56.310
2	Melisa	Boysen	2011-03-25 08:27:56.310

7. Trennen Sie die Verbindung zu Interactive SQL.

Ergebnisse

Die entfernte Datenbank wird extrahiert und das FILE-Nachrichtensystem in der entfernten Datenbank wird durch das HTTP-Nachrichtensystem ersetzt.

Nächste Schritte

Gehen Sie weiter zu „Lektion 4: Hinzufügen und Replizieren von Daten in konsolidierter und entfernter Datenbank“ auf Seite 187.

Lektion 4: Hinzufügen und Replizieren von Daten in konsolidierter und entfernter Datenbank

Fügen Sie Daten zur konsolidierten und entfernten Datenbank hinzu, führen Sie SQL Remote aus, um die Änderungen zu replizieren, und bestätigen Sie anschließend, dass die Daten in beiden Datenbanken konsistent sind.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Wenn Sie derzeit nicht mit der konsolidierten Datenbank (cons) verbunden sind, führen Sie folgenden Befehl aus:

   ```
   dbisql -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql"
   ```

2. Führen Sie die folgenden Anweisungen in der konsolidierten Datenbank (cons) aus, um der employees-Tabelle zusätzliche Beispieldaten hinzuzufügen:

   ```
   INSERT INTO employees (first_name, last_name) VALUES ('Javier', 'Spoor');
   COMMIT;
   ```

3. Führen Sie die folgenden Anweisungen in der entfernten Datenbank (rem) aus, um der employees-Tabelle zusätzliche Beispieldaten hinzuzufügen:

   ```
   INSERT INTO employees (first_name, last_name) VALUES ('Nelson',
   'Kreitzer');
   COMMIT;
   ```

4. Führen Sie in der konsolidierten Datenbank (cons) den Nachrichtenagenten aus:

   ```
   dbremote -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql" -qc -v -o c:\tutorial
   \cons1.txt
   ```

   Dieser durchsucht das Transaktionslog der konsolidierten Datenbank (cons) und generiert mit dem FILE-Nachrichtensystem eine Nachricht für die entfernte Datenbank (rem). Da der Systemparameter für Fehlerbehebungsmeldungen in der konsolidierten Datenbank auf das FILE-Nachrichtensystem eingestellt wurde, können Sie die Datei *c:\tutorial\cons1.txt* auf Fehlerbehebungsmeldungen überprüfen, die anzeigen, dass Meldungen in das Verzeichnis *c:\tutorial\messages\rem* geschrieben werden. Beispiel:

   ```
   I. 2011-03-25 11:03:31. Processing transactions from active transaction
   log
   I. 2011-03-25 11:03:31. Sending message to "rem"
   (0-0000000000-0000550994-0)
   I. 2011-03-25 11:03:31. sopen "c:\tutorial\messages\rem\cons.0"
   I. 2011-03-25 11:03:31. write " c:\tutorial\messages\rem\cons.0"
   I. 2011-03-25 11:03:31. close " c:\tutorial\messages\rem\cons.0"
   ```

5. Führen Sie in der entfernten Datenbank (rem) den Nachrichtenagenten aus:

```
dbremote -c "SERVER=rem;DBN=rem;UID=DBA;PWD=sql" -qc -v -o c:\tutorial
\rem.txt
```

Unter Verwendung des HTTP-Nachrichtensystems wird durch diesen Befehl die soeben von der konsolidierten Datenbank generierte Nachricht empfangen und übernommen. Danach wird das Transaktionslog durchsucht und eine Nachricht mit der neuen Zeile, die in der entfernten Datenbank hinzugefügt wurde, wird an die konsolidierte Datenbank zurückgesendet. Da der Systemparameter für Fehlerbehebungsmeldungen in der konsolidierten Datenbank auf das HTTP-Nachrichtensystem eingestellt wurde, können Sie die Datei *c:\tutorial\rem.txt* auf Fehlerbehebungsmeldungen überprüfen, die anzeigen, dass das HTTP-Nachrichtensystem verwendet wird. Beispiel:

```
I. 2011-03-25 11:10:02. Sending message to "cons"
(0-00000000000-0000557411-0)
I. 2011-03-25 11:10:02. HTTPWriteMessage "rem.0"
I. 2011-03-25 11:10:02. HTTPWriteMessage: success -- filename  "rem.0"
I. 2011-03-25 11:10:02. HTTPDisconnect
```

6. Führen Sie in der konsolidierten Datenbank (cons) den Nachrichtenagenten aus:

   ```
   dbremote -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql" -qc -v -o c:\tutorial
   \cons2.txt
   ```

   Unter Verwendung des FILE-basierten Nachrichtensystems wird durch diesen Befehl die soeben von der entfernten Datenbank generierte Nachricht empfangen und übernommen.

7. Um zu überprüfen, ob die konsolidierte Datenbank alle vier Datenzeilen enthält, können Sie die folgende Anweisung ausführen und den Inhalt der employees-Tabelle anzeigen:

   ```
   SELECT * FROM employees
   ```

   Die Abfrage gibt die folgenden Daten aus der employees-Tabelle zurück, obwohl die hire_date-Spalte die Uhrzeit enthält, zu der Sie die Zeile eingefügt haben, und nicht die in der folgenden Tabelle aufgeführten Werte:

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	2011-03-25 08:27:56.310
2	Melisa	Boysen	2011-03-25 08:27:56.310
3	Javier	Spoor	2011-03-25 08:30:26.110
102000001	Nelson	Kreitzer	2011-03-25 08:31:51.970

8. Überprüfen Sie, ob die entfernte Datenbank (rem) alle vier Datenzeilen enthält, indem Sie die folgende Anweisung ausführen, um den Inhalt der employees-Tabelle anzuzeigen:

   ```
   SELECT * FROM employees
   ```

   Die Abfrage gibt die folgenden Daten aus der employees-Tabelle zurück, obwohl die hire_date-Spalte die Uhrzeit enthält, zu der Sie die Zeile eingefügt haben, und nicht die in der folgenden Tabelle aufgeführten Daten:

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	2011-03-25 08:27:56.310
2	Melisa	Boysen	2011-03-25 08:27:56.310
3	Javier	Spoor	2011-03-25 08:30:26.110
102000001	Nelson	Kreitzer	2011-03-25 08:31:51.970

9. Trennen Sie die Verbindung zu Interactive SQL.

Ergebnisse

Die Daten werden der konsolidierten und der entfernten Datenbank hinzugefügt, die Änderungen werden repliziert, und die Datenkonsistenz wurde überprüft.

Nächste Schritte

Gehen Sie weiter zu „Lektion 5: Aufräumen“ auf Seite 189.

Lektion 5: Aufräumen

Fahren Sie die entfernte und die konsolidierte Datenbank herunter.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Führen Sie den folgenden Befehl aus, um die entfernte Datenbank herunterzufahren:

   ```
   dbstop -y -c "SERVER=rem;DBN=rem;UID=DBA;PWD=sql"
   ```

2. Führen Sie den folgenden Befehl aus, um die konsolidierte Datenbank herunterzufahren:

   ```
   dbstop -y -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql"
   ```

3. Löschen Sie die Verzeichnisse, die Sie in Lektion 1 erstellt haben.

Ergebnisse

Die entfernte und die konsolidierte Datenbank werden heruntergefahren.

Nächste Schritte

Keiner.

Praktische Einführung: Einrichten eines Replikationssystems unter Verwendung des HTTP-Nachrichtensystems und mit der konsolidierten Datenbank als Nachrichtenserver über Relay Server

In den Lektionen dieser praktischen Einführung erfahren Sie, wie Sie ein SQL Remote-Replikationssystem einrichten, das eine konsolidierte SQL Anywhere-Datenbank verwendet, Relay Server zum Weiterleiten des HTTP-Datenverkehrs an die konsolidierte Datenbank sowie eine entfernte Datenbank. Die konsolidierte Datenbank verwendet zum Replizieren von Änderungen das FILE-Nachrichtensystem, während die entfernte Datenbank dafür das HTTP-Nachrichtensystem verwendet.

In dieser praktischen Einführung führen Sie folgende Schritte durch:

- Erstellen einer konsolidierten SQL Anywhere-Datenbank und einer entfernten SQL Anywhere-Datenbank, die alle Daten in der konsolidierten Datenbank enthält.
- Erstellen einer Verzeichnisstruktur für die Speicherung der von SQL Remote generierten Nachrichten. Die konsolidierte Datenbank verwendet für den Zugriff auf die Dateien das FILE-Nachrichtensystem, während die entfernte Datenbank dafür das HTTP-Nachrichtensystem verwendet.
- Konfigurieren eines vorhandenen Relay Servers zum Weiterleiten des HTTP-Datenverkehrs an die konsolidierte Datenbank.
- Konfigurieren der konsolidierten Datenbank, sodass sie als Nachrichtenserver für das HTTP-Nachrichtensystem fungiert und weitergeleiteten HTTP-Datenverkehr vom Relay Server akzeptiert.
- Erstellen einer entfernten Datenbank, die Nachrichten mithilfe des HTTP-Messaging-Systems sendet.
- Replizieren von Daten zwischen der konsolidierten Datenbank und der entfernten Datenbank.

Rollenanforderung
Diese praktische Einführung bezieht sich auf einen Benutzer, DBA, der über die SYS_REPLICATION_ADMIN_ROLE-Systemrolle verfügen muss.

Lektion 1: Erstellen der konsolidierten Datenbank

Erstellen Sie die Verzeichnisse, die benötigt werden, um die Datenbanken und deren Transaktionslogs zu speichern, sowie die Verzeichnisstruktur für die Nachrichten. Außerdem legen Sie das Schema der konsolidierten Datenbank fest, einschließlich der Erstellung des entfernten Benutzers sowie der Publikation und der Subskription, die für die Replikation der Daten benötigt werden.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Wenn SQL Remote in der konsolidierten Datenbank ausgeführt wird, verwendet es das FILE-Nachrichtensystem, um Nachrichten zu senden und zu empfangen, aber die entfernte Datenbank verwendet das HTTP-Nachrichtensystem.

Für diese praktische Einführung lautet der Name des Computers, auf dem die konsolidierte Datenbank und damit der Nachrichtenserver läuft, **machine_cons**.

Aufgabe

1. Erstellen Sie die folgenden Verzeichnisse für die konsolidierte Datenbank und die entfernte Datenbank:
   - *c:\tutorial*
   - *c:\tutorial\cons*
   - *c:\tutorial\rem*

2. Erstellen Sie die folgenden Verzeichnisse für die von der konsolidierten Datenbank und der entfernten Datenbank generierten Nachrichtendateien:
   - *c:\tutorial\messages*
   - *c:\tutorial\messages\cons*
   - *c:\tutorial\messages\rem*

3. Führen Sie im Verzeichnis *c:\tutorial\cons* den folgenden Befehl aus, um die konsolidierte Datenbank (cons) zu erstellen:

   ```
   dbinit -dba DBA,sql cons.db
   ```

4. Starten Sie die konsolidierte Datenbank:

   ```
   dbsrv16 -n cons c:\tutorial\cons\cons.db -xs http(port=8033)
   ```

   -xs http(8033) ist in der Befehlszeile erforderlich, da dieser Datenbankserver HTTP-Anforderungen aus der entfernten Datenbank akzeptiert und auf die Nachrichtendateien im Verzeichnis *c:\tutorial\messages* zugreift. Während beim Start des Datenbankservers keine Webdienste definiert sind, werden sie in der nächsten Lektion erstellt. In dieser Lektion starten Sie nur den Personal Datenbankserver, sodass nur SQL Remote Prozesse auf diesem Computer über HTTP mit dem Nachrichtenserver kommunizieren können. In einer Produktionsumgebung wird gewöhnlich der Netzwerkserver verwendet, sodass SQL Remote-Prozesse auf anderen Computern ebenfalls Zugriff auf die Webdienste haben. In dieser Lektion haben Sie einen Netzwerkserver gestartet und ihm den Namen "cons" gegeben. Wenn in Ihrem Netzwerk bereits ein anderer Datenbankserver mit diesem Namen läuft, müssen Sie einen anderen Namen für den Netzwerkserver wählen und die Verbindungszeichenfolgen im Rest dieser praktischen Einführung so ändern, dass der andere Name verwendet wird. Weitere Hinweise zur Verwendung der Option -xs finden Sie unter „Datenbankserveroption -xs “ [*SQL Anywhere Server - Datenbankadministration*].

5. Verbinden Sie sich mit Interactive SQL mit der konsolidierten Datenbank (cons) als Benutzer mit der entsprechenden SYS_REPLICATION_ADMIN_ROLE-Systemrolle:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=cons;DBN"
   ```

6. Führen Sie die folgende Anweisung aus, um die globale Datenbank-ID für die konsolidierte Datenbank (cons) festzulegen (die benötigt wird, damit bei Verwendung des GLOBAL AUTOINCREMENT-Standardwerts für alle Datenbanken unterschiedliche Primärschlüssel gewählt werden):

   ```
   SET OPTION public.global_database_id=0;
   ```

7. Das Schema für die Datenbank in dieser praktischen Einführung besteht aus einer einzelnen Tabelle und alle Spalten und Zeilen aus der Tabelle werden für jeden entfernten Benutzer repliziert. Führen Sie die folgenden Anweisungen in der konsolidierten Datenbank (cons) aus, um die einzelne Tabelle in der Datenbank zu erstellen:

   ```
   CREATE TABLE employees (
        employee_id BIGINT NOT NULL DEFAULT GLOBAL AUTOINCREMENT(1000000)
   PRIMARY KEY,
        first_name VARCHAR(128) NOT NULL,
        last_name VARCHAR(128) NOT NULL,
        hire_date TIMESTAMP NOT NULL DEFAULT TIMESTAMP
   );
   ```

8. Führen Sie die folgenden Anweisungen in der konsolidierten Datenbank (cons) aus, um der employees-Tabelle Beispieldaten hinzuzufügen:

   ```
   INSERT INTO employees (first_name, last_name) VALUES ('Kelly', 'Meloy');
   INSERT INTO employees (first_name, last_name) VALUES ('Melisa', 'Boysen');
   COMMIT;
   ```

9. Führen Sie die folgende Anweisung in der konsolidierten Datenbank (cons) aus, um zu bestätigen, dass die Tabelle erstellt und mit Daten gefüllt wurde:

   ```
   SELECT * FROM employees;
   ```

   Die Abfrage gibt die folgenden Daten aus der employees-Tabelle zurück, obwohl die hire_date-Spalte die Uhrzeit enthält, zu der Sie die Zeile eingefügt haben, und nicht die in der folgenden Tabelle aufgeführten Werte:

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	2011-03-25 08:27:56.310
2	Melisa	Boysen	2011-03-25 08:27:56.310

10. In dieser praktischen Einführung werden dem Publikationseigentümer und dem entfernten Benutzer Kennwörter zugeordnet, da die konsolidierte Datenbank als Nachrichtenserver für das HTTP-Nachrichtensystem fungiert. Führen Sie die folgende Anweisung aus, um den Benutzer cons mit den Privilegien CONNECT und PUBLISH zu erstellen:

    ```
    GRANT CONNECT TO cons;
    GRANT PUBLISH TO cons;
    ```

11. Aus Gründen der Performance kann das HTTP-Nachrichtensystem nur in der entfernten Datenbank verwendet werden und nicht in der konsolidierten Datenbank. Die folgenden Anweisungen konfigurieren die Verwendung des FILE-basierten Nachrichtensystems in der konsolidierten Datenbank:

    ```
    CREATE REMOTE MESSAGE TYPE FILE ADDRESS 'cons';
    SET REMOTE FILE OPTION public.directory='c:\\tutorial\\messages';
    SET REMOTE FILE OPTION public.debug='yes';
    ```

12. Führen Sie die folgenden Anweisungen aus, um den entfernten Benutzer rem ohne Kennwort zu erstellen und ihm REMOTE-Privilegien zu erteilen, während Sie die Adresse des Benutzers im FILE-Nachrichtensystem festlegen:

    ```
    GRANT CONNECT TO rem IDENTIFIED BY rem;
    GRANT REMOTE TO rem TYPE FILE ADDRESS 'rem';
    ```

13. Eine Publikation beschreibt die Datenmenge, die repliziert werden soll. Erstellen Sie eine Publikation namens pub_employees, die alle Zeilen der employees-Tabelle repliziert. Sie subskribieren einen Benutzer für eine Publikation, indem Sie eine Subskription erstellen.

    ```
    CREATE PUBLICATION pub_employees ( TABLE employees );
    CREATE SUBSCRIPTION TO pub_employees FOR rem;
    ```

14. Trennen Sie die Verbindung zu Interactive SQL.

Ergebnisse

Die Verzeichnisse, die benötigt werden, um die Datenbanken und deren Transaktionslogs zu speichern, sowie die Verzeichnisstruktur für die Nachrichten werden erstellt. Das Schema der konsolidierten Datenbank wird definiert, einschließlich der Erstellung des entfernten Benutzers sowie der Publikation und der Subskription, die für die Replikation der Daten benötigt werden.

Nächste Schritte

Gehen Sie weiter zu „Lektion 2: Konfigurieren des Relay Servers“ auf Seite 194.

Lektion 2: Konfigurieren des Relay Servers

Ändern Sie die Konfiguration des Relay Servers so, dass HTTP-Anforderungen an die SQL Anywhere-Backend-Datenbank weitergeleitet werden.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Für diese praktische Einführung wurden die folgenden Annahmen getroffen:

1. Der Name des Computers, auf dem der Relay Server ausgeführt wird, lautet **machine_iis**, und es handelt sich um ein Windows 2008 Server R2-System mit IIS 7.5.

2. Die Setup-Anweisungen für den Relay Server wurden exakt befolgt, wie in der Dokumentation beschrieben:

   - Relay Server
   - „Relay Server-Deployment“ [*Relay Server*]

3. Sie haben die Relay Server-Komponenten für Microsoft IIS 7.0 oder 7.5 unter Windows Server 2008/ Windows Server 2008 R2 bereitgestellt. Siehe „Deployment der Relay Server-Komponenten für Microsoft IIS 7.0 unter Windows Server 7.0 oder 8.0“ [*Relay Server*].

Kontext und Bemerkungen

Das Konfigurieren des Relay Servers besteht nur aus dem Ändern der vom Relay Server verwendeten Datei *rs.config*. Anschließend muss der rshost-Prozess aktualisieren oder neu gestartet werden.

Aufgabe

1. Fügen Sie in der vom rshost-Prozess auf machine_iis verwendeten Datei *rs.config* Einträge hinzu, die angeben, dass der SQL Anywhere-Backend-Datenbankserver als Nachrichtenserver fungiert:

   ```
   [backend_farm]
   id=srhttp_tutorial_farm
   description=SQL Anywhere Web Services farm for tutorial
   active_cookie=yes
   active_header=no
   enable=yes
   verbosity=5

   [backend_server]
   id= srhttp_tutorial_server
   description=SQL Anywhere Web Services server for tutorial
   farm= srhttp_tutorial_farm
   enable=yes
   verbosity=5
   ```

2. Führen Sie im Verzeichnis *%SQLANY16%\RelayServer\IIS\Bin64\Server* die folgende Befehlszeile aus, um die Aktualisierung der Konfiguration zu übernehmen:

   ```
   rshost -u -f rs.config
   ```

Ergebnisse

Die Konfiguration des Relay Servers wird geändert, sodass HTTP-Anforderungen an die SQL Anywhere-Backend-Datenbank weitergeleitet werden.

Nächste Schritte

Gehen Sie weiter zu „Lektion 3: Konfigurieren der konsolidierten Datenbank als Nachrichtenserver“ auf Seite 196.

Lektion 3: Konfigurieren der konsolidierten Datenbank als Nachrichtenserver

Konfigurieren Sie die konsolidierte Datenbank als Nachrichtenserver für das HTTP-Nachrichtensystem.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Kontext und Bemerkungen

Es ist auch möglich, das eine separate Datenbank und einen separaten Datenbankserver als Nachrichtenserver zu konfigurieren.

Aufgabe

1. Verbinden Sie sich über Interactive SQL als Benutzer mit der SYS_REPLICATION_ADMIN_ROLE-Systemrolle:

   ```
   dbisql -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql"
   ```

2. Beim ersten Initialisieren einer Datenbank wird keiner der Webdienste festgelegt, die benötigt werden, um HTTP-Anforderungen von entfernten Benutzern zu akzeptieren, und es werden keine Definitionen erstellt, die der Datenbankserver für den Zugriff auf das Verzeichnis verwenden könnte, in dem die Nachrichtendateien gespeichert werden. Die Erstellung dieser Objekte wird mit der gespeicherten Prozedur sr_add_message_server automatisiert, in der durch einen optionalen Parameter der Eigentümer aller Objekte angegeben werden kann. Führen Sie die folgenden Anweisungen für die konsolidierte Datenbank (cons) aus, um alle für den Nachrichtenserver erforderlichen Objekte festzulegen, und geben Sie an, dass der Benutzer cons Eigentümer aller Objekte ist:

   ```
   CREATE ROLE FOR USER cons;
   SET REMOTE http OPTION cons.root_directory='c:\\tutorial\\messages';
   CALL sr_add_message_server( 'cons' );
   COMMIT;
   ```

3. Einige zusätzliche Konfiguration ist erforderlich, wenn der Relay Server an die SQL Anywhere-Backend-Datenbank weiterleiten soll. Es ist möglich, eine Umgebung mit hoher Verfügbarkeit für Ihre SQL Anywhere-Backend-Server einzurichten, in der einige Knoten als schreibgeschützt und einige als nicht schreibgeschützt festgelegt sind. In dieser praktischen Einführung gibt es nur einen einzelnen Datenbankserver im System. Daher müssen Sie die Datenbank als nicht schreibgeschützten Knoten festlegen. Führen Sie die folgenden Anweisungen in der konsolidierten Datenbank (cons) aus, um alle Objekte festzulegen, die benötigt werden, damit der Relay Server diese Datenbankserver als nicht schreibgeschützten Knoten erkennt:

   ```
   CREATE PROCEDURE sp_oe_read_status()
   RESULT (doc LONG VARCHAR)
   BEGIN
   DECLARE res LONG VARCHAR;
   SET res='AVAILABLE=TRUE';
   CALL sa_set_http_header('Content-Length', LENGTH(res) );
   SELECT res;
   END;
   ```

```
GO

CREATE SERVICE oe_read_status
TYPE 'raw'
AUTHORIZATION OFF
SECURE OFF
USER DBA
AS CALL sp_oe_read_status();
GO
```

4. Trennen Sie die Verbindung zu Interactive SQL.

5. Der Outbound Enabler fungiert als Kanal zwischen dem Relay Server und der SQL Anywhere-Backend-Datenbank. Starten Sie auf dem Computer machine_cons den Relay Server Outbound Enabler (RSOE) mit folgender Befehlszeile:

   ```
   rsoe -cr "host=machine_iis;port=80;url_suffix=/rs/server/rs_server.dll"
   -cs "host=machine_cons;port=8033;status_url=/oe_read_status"
   -f srhttp_tutorial_farm -id srhttp_tutorial_server -v 5 -o rsoe.log
   ```

Ergebnisse

Die konsolidierte Datenbank wird als Nachrichtenserver für das HTTP-Nachrichtensystem konfiguriert.

Nächste Schritte

Gehen Sie weiter zu „Lektion 4: Erstellen der entfernten Datenbank“ auf Seite 197.

Lektion 4: Erstellen der entfernten Datenbank

Extrahieren Sie die entfernte Datenbank und ersetzen Sie das FILE-Nachrichtensystem in der entfernten Datenbank durch das HTTP-Nachrichtensystem.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Führen Sie im Verzeichnis *c:\tutorial\rem* den folgenden Befehl aus, um die entfernte Datenbank (rem) zu erstellen:

   ```
   dbinit -dba DBA,sql rem.db
   ```

2. In dieser Lektion verwenden Sie dbxtract, um die entfernte Datenbank zu erstellen. Führen Sie den folgenden Befehl aus, um die Datenbank für den Benutzer rem aus der konsolidierten Datenbank zu extrahieren, und lassen Sie den Datenbankserver für die entfernte Datenbank nach der Extraktion laufen:

   ```
   dbxtract -xx -ac "SERVER=rem;DBN=rem;DBF=c:\tutorial\rem
   \rem.db;UID=DBA;PWD=sql;autostop=no" -c
   "SERVER=cons;DBN=cons;UID=DBA;PWD=sql" rem
   ```

3. Wenn Sie derzeit nicht über Interactive SQL mit der entfernten Datenbank (rem) verbunden sind, führen Sie folgenden Befehl aus:

   ```
   dbisql -c "SERVER=rem;DBN=rem;UID=DBA;PWD=sql"
   ```

4. Die konsolidierte Datenbank verwendet das FILE-Nachrichtensystem. Deshalb werden beim Ausführen von dbxtract SQL Remote-Definitionen unter der Annahme erstellt, dass die entfernte Datenbank (rem) ebenfalls das FILE-Nachrichtensystem verwendet. Wenn Sie die entfernte Datenbank für die Verwendung des HTTP-Nachrichtensystems konfigurieren möchten, müssen Sie die folgenden SQL-Anweisungen in der entfernten Datenbank (rem) ausführen, um das FILE-Nachrichtensystem für diese entfernte Datenbank zu entfernen:

   ```
   CREATE REMOTE TYPE "FILE" ADDRESS '';
   SET REMOTE FILE OPTION public.directory='';
   SET REMOTE FILE OPTION public.debug='';
   ```

5. Führen Sie die folgenden Anweisungen in der entfernten Datenbank (rem) aus, um das HTTP-Nachrichtensystem für diese entfernte Datenbank zu konfigurieren:

   ```
   CREATE REMOTE TYPE "HTTP" ADDRESS 'rem';
   GRANT CONSOLIDATE TO "cons" TYPE "HTTP" ADDRESS 'cons';
   SET REMOTE HTTP OPTION public.user_name='rem';
   SET REMOTE HTTP OPTION public.password='rem';
   SET REMOTE HTTP OPTION public.debug='yes';
   SET REMOTE HTTP OPTION public.https='no';
   SET REMOTE HTTP OPTION public.url='machine_iis:80/rs/client/rs_client.dll/
   srhttp_tutorial_farm';
   COMMIT;
   ```

6. Überprüfen Sie, ob die entfernte Datenbank (rem) nach der Extraktion die beiden Datenzeilen enthält, die bereits in der konsolidierten Datenbank vorhanden waren. Führen Sie die folgende Anweisung aus, um den Inhalt der employees-Tabelle anzuzeigen:

   ```
   SELECT * FROM employees;
   ```

   Die Abfrage gibt die folgenden Daten aus der employees-Tabelle zurück, obwohl die hire_date-Spalte die Uhrzeit enthält, zu der Sie die Zeile eingefügt haben, und nicht die in der folgenden Tabelle aufgeführten Daten:

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	2011-03-25 08:27:56.310
2	Melisa	Boysen	2011-03-25 08:27:56.310

7. Trennen Sie die Verbindung zu Interactive SQL.

Ergebnisse

Die entfernte Datenbank wird extrahiert und das FILE-Nachrichtensystem in der entfernten Datenbank wird durch das HTTP-Nachrichtensystem ersetzt.

Nächste Schritte

Gehen Sie weiter zu „Lektion 5: Hinzufügen und Replizieren von Daten in konsolidierter und entfernter Datenbank“ auf Seite 199.

Lektion 5: Hinzufügen und Replizieren von Daten in konsolidierter und entfernter Datenbank

Fügen Sie Daten zur konsolidierten und entfernten Datenbank hinzu, führen Sie SQL Remote aus, um die Änderungen zu replizieren, und bestätigen Sie anschließend, dass die Daten in beiden Datenbanken konsistent sind.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Wenn Sie derzeit nicht mit der konsolidierten Datenbank (cons) verbunden sind, führen Sie folgenden Befehl aus:

   ```
   dbisql -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql"
   ```

2. Führen Sie die folgenden Anweisungen in der konsolidierten Datenbank (cons) aus, um der employees-Tabelle zusätzliche Beispieldaten hinzuzufügen:

   ```
   INSERT INTO employees (first_name, last_name) VALUES ('Javier', 'Spoor');
   COMMIT;
   ```

3. Führen Sie die folgenden Anweisungen in der entfernten Datenbank (rem) aus, um der employees-Tabelle zusätzliche Beispieldaten hinzuzufügen:

   ```
   INSERT INTO employees (first_name, last_name) VALUES ('Nelson',
   'Kreitzer');
   COMMIT;
   ```

4. Führen Sie in der konsolidierten Datenbank (cons) den Nachrichtenagenten aus:

   ```
   dbremote -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql" -qc -v -o c:\tutorial
   \cons1.txt
   ```

   Dieser durchsucht das Transaktionslog der konsolidierten Datenbank (cons) und generiert mit dem FILE-Nachrichtensystem eine Nachricht für die entfernte Datenbank (rem). Da der Systemparameter für Fehlerbehebungsmeldungen in der konsolidierten Datenbank auf das FILE-Nachrichtensystem eingestellt wurde, können Sie die Datei *c:\tutorial\cons1.txt* auf Fehlerbehebungsmeldungen überprüfen, die anzeigen, dass Meldungen in das Verzeichnis *c:\tutorial\messages\rem* geschrieben werden. Beispiel:

   ```
   I. 2011-04-12 09:33:03. Processing transactions from active transaction
   log
   I. 2011-04-12 09:33:03. Sending message to "rem"
   ```

```
(0-0000000000-0000550994-0)
I. 2011-04-12 09:33:03. sopen "c:\tutorial\messages\rem\cons.0"
I. 2011-04-12 09:33:03. write " c:\tutorial\messages\rem\cons.0"
I. 2011-04-12 09:33:03. close " c:\tutorial\messages\rem\cons.0"
```

5. Führen Sie in der entfernten Datenbank (rem) den Nachrichtenagenten aus:

   ```
   dbremote -c "SERVER=rem;DBN=rem;UID=DBA;PWD=sql" -qc -v -o c:\tutorial
   \rem.txt
   ```

   Unter Verwendung des HTTP-Nachrichtensystems wird durch diesen Befehl die soeben von der konsolidierten Datenbank generierte Nachricht empfangen und übernommen. Danach wird das Transaktionslog durchsucht und eine Nachricht mit der neuen Zeile, die in der entfernten Datenbank hinzugefügt wurde, wird an die konsolidierte Datenbank zurückgesendet. Da der Systemparameter für Fehlerbehebungsmeldungen in der konsolidierten Datenbank auf das HTTP-Nachrichtensystem eingestellt wurde, können Sie die Datei *c:\tutorial\rem.txt* auf Fehlerbehebungsmeldungen überprüfen, die anzeigen, dass das HTTP-Nachrichtensystem verwendet wird. Beispiel:

   ```
   I. 2011-04-12 09:34:03. Sending message to "cons"
   (0-0000000000-0000576448-0)
   I. 2011-04-12 09:34:03. HTTPWriteMessage "rem.0"
   I. 2011-04-12 09:34:03. HTTPWriteMessage: success -- filename  "rem.0"
   I. 2011-04-12 09:34:03. HTTPDisconnect
   ```

6. Sie können auch bestätigen, dass die Anforderung über den Relay Server geleitet wurde, indem Sie anhand der vom RSOE generierten Ausgabedatei überprüfen, ob Informationen in das Log geschrieben werden.

   ```
   I. 2011-04-12 09:34:03. <UpChannel-0000> PacketRead packet-len:257
   I. 2011-04-12 09:34:03. <UpChannel-0000> PacketRead packet-opcode:0xf004
   I. 2011-04-12 09:34:03. <UpChannel-0000> packet read..
   I. 2011-04-12 09:34:03. <UpChannel-0000> successful packet read..
   processing it..
   I. 2011-04-12 09:34:03. <UpChannel-0000> 259
   RS_CLI_SESSION_BEGIN(snum=0006 sfp=4e0e5291 ridx=0)
   I. 2011-04-12 09:34:03. <UpChannel-0000> Notifying worker thread
   ```

7. Führen Sie in der konsolidierten Datenbank (cons) den Nachrichtenagenten aus:

   ```
   dbremote -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql" -qc -v -o c:\tutorial
   \cons2.txt
   ```

   Unter Verwendung des FILE-basierten Nachrichtensystems wird durch diesen Befehl die soeben von der entfernten Datenbank generierte Nachricht empfangen und übernommen.

8. Um zu überprüfen, ob die konsolidierte Datenbank alle vier Datenzeilen enthält, können Sie die folgende Anweisung ausführen und den Inhalt der employees-Tabelle anzeigen:

   ```
   SELECT * FROM employees;
   ```

   Die Abfrage gibt die folgenden Daten aus der employees-Tabelle zurück, obwohl die hire_date-Spalte die Uhrzeit enthält, zu der Sie die Zeile eingefügt haben, und nicht die in der folgenden Tabelle aufgeführten Werte:

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	2011-03-25 08:27:56.310
2	Melisa	Boysen	2011-03-25 08:27:56.310
3	Javier	Spoor	2011-03-25 08:30:26.110
102000001	Nelson	Kreitzer	2011-03-25 08:31:51.970

9. Überprüfen Sie, ob die entfernte Datenbank (rem) alle vier Datenzeilen enthält, indem Sie die folgende Anweisung ausführen, um den Inhalt der employees-Tabelle anzuzeigen:

   ```
   SELECT * FROM employees;
   ```

   Die Abfrage gibt die folgenden Daten aus der employees-Tabelle zurück, obwohl die hire_date-Spalte die Uhrzeit enthält, zu der Sie die Zeile eingefügt haben, und nicht die in der folgenden Tabelle aufgeführten Daten:

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	2011-03-25 08:27:56.310
2	Melisa	Boysen	2011-03-25 08:27:56.310
3	Javier	Spoor	2011-03-25 08:30:26.110
102000001	Nelson	Kreitzer	2011-03-25 08:31:51.970

10. Trennen Sie die Verbindung zu Interactive SQL.

Ergebnisse

Die Daten werden der konsolidierten und der entfernten Datenbank hinzugefügt, die Änderungen werden repliziert, und die Datenkonsistenz wurde überprüft.

Nächste Schritte

Gehen Sie weiter zu „Lektion 6: Aufräumen“ auf Seite 201.

Lektion 6: Aufräumen

Fahren Sie den RSOE sowie die konsolidierte und die entfernte Datenbank herunter.

Voraussetzungen

Sie müssen die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Aufgabe

1. Führen Sie den folgenden Befehl aus, um den RSOE herunterzufahren:

   ```
   rsoe -s -cr "host=machine_iis;port=80;url_suffix=/rs/server/
   rs_server.dll"
   -cs "host=machine_cons-t3500;port=8033;status_url=/oe_read_status"
   -f srhttp_tutorial_farm -id srhttp_tutorial_server
   ```

2. Führen Sie den folgenden Befehl aus, um die konsolidierte Datenbank herunterzufahren:

   ```
   dbstop -y -c "SERVER=cons;DBN=cons;UID=DBA;PWD=sql"
   ```

   Führen Sie den folgenden Befehl aus, um die entfernte Datenbank herunterzufahren:

   ```
   dbstop -y -c "SERVER=rem;UID=DBA;PWD=sql"
   ```

Ergebnisse

Der RSOE sowie die konsolidierte und die entfernte Datenbank werden heruntergefahren.

Nächste Schritte

Keiner.

SQL Remote-Referenz

Dieser Abschnitt enthält Referenzmaterial zu SQL Remote.

SQL Remote-Dienstprogramme und Optionen

SQL Remote-Nachrichtenagent-Dienstprogramm (dbremote)

Das Dienstprogramm versendet SQL Remote-Nachrichten bzw. wendet sie an und hält das Nachrichtenprotokollierungssystem aufrecht, um die Nachrichtenzustellung sicherzustellen Um SQL Remote auszuführen, müssen Sie über die SYS_RUN_REPLICATION_ROLE-Systemrolle verfügen.

Syntax

dbremote [*options*] [*directory*]

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*].
	
	Wenn Sie Kennwörter oder andere Informationen in einer Konfigurationsdatei schützen möchten, können Sie das Dienstprogramm zum Ausblenden von Dateien zum Verschleiern des Inhalts von Konfigurationsdateien verwenden. Siehe „Dienstprogramm zum Verschleiern von Dateien (dbfhide)“ [*SQL Anywhere Server - Datenbankadministration*].
	
	Die Umgebungsvariable kann eine beliebige Menge von Befehlszeilenoptionen enthalten. Die erste der folgenden zwei Anweisungen setzt zum Beispiel eine Umgebungsvariable, die eine Reihe von Befehlszeilenoptionen für einen SQL Remote-Prozess enthält, der mit einer Cachegröße von 4 MB startet, nur Nachrichten empfängt und mit einer Datenbank namens **field** auf einem Datenbankserver namens **myserver** verbunden wird. Die SET-Anweisung muss in einer einzigen Zeile eingegeben werden:
	
	`SET envvar=-m 4096 -r -c "Server=myserver;DBN=field;UID=sa;PWD=sysadmin"`
	`dbremote @envvar`
	
	Die Konfigurationsdatei enthält Zeilenumbrüche und kann eine beliebige Menge von Optionen enthalten. Die folgende Befehlsdatei enthält zum Beispiel eine Reihe von Befehlszeilenoptionen für einen SQL Remote-Nachrichtenagenten, der mit einer Cachegröße von 4 MB startet, nur Nachrichten versendet und eine Verbindung mit einer Datenbank namens **field** auf einem Datenbankserver mit Namen **myserver** herstellt:
	
	`-m 4096`
	`-s`
	`-c "Server=myserver;DBN=field;UID=sa;PWD=sysadmin"`
	
	Wenn diese Konfigurationsdatei als *c:\config.txt* gespeichert wird, kann sie folgendermaßen in einem Befehl verwendet werden:
	
	`dbremote @c:\config.txt`

Option	Beschreibung
-a	Verarbeitet empfangene Nachrichten (im Posteingang), ohne sie in die Datenbank zu übernehmen. Zusammen mit -v (ausführlich dargestellte Ausgabe) und -p (keine Bereinigung der Nachrichten) hilft Ihnen diese Markierung dabei, Probleme bei eintreffenden Nachrichten zu ermitteln. Wenn sie ohne -p verwendet wird, bereinigt diese Markierung den Posteingang, ohne die Nachrichten anzuwenden. Dies ist nützlich, wenn eine Subskription erneut gestartet wird.
-b	Wird im Batchmodus ausgeführt. In diesem Modus verarbeitet der SQL Remote-Nachrichtenagent eintreffende Nachrichten, durchsucht einmal das Transaktionslog und verarbeitet ausgehende Nachrichten. Danach stoppt er.
-c *"keyword=value;..."*	Legt Verbindungsparameter fest. Wenn diese Option nicht angegeben ist, wird die Umgebungsvariable SQLCONNECT verwendet. Die folgende Anweisung führt beispielsweise dbremote in einer Datenbank unter *c:\mydata.db* aus und stellt eine Verbindung mit Benutzer-ID **DBA** und Kennwort **sql** her (Benutzer DBA muss über die SYS_RUN_REPLICATION_ROLE-Systemrolle verfügen): `dbremote -c "UID=DBA;PWD=sql;DBF=c:\mydata.db"` Der SQL Remote-Nachrichtenagent (dbremote) muss von einem Benutzer mit der SYS_RUN_REPLICATION_ROLE-Systemrolle ausgeführt werden. Siehe „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]. Der SQL Remote-Nachrichtenagent unterstützt alle Bereiche der SQL Anywhere-Verbindungsparameter. Siehe „Verbindungsparameter“ [*SQL Anywhere Server - Datenbankadministration*].
-dl	Zeigt Nachrichten im Fenster des SQL Remote-Nachrichtenagenten oder an einer Eingabeaufforderung an und überträgt sie, falls angegeben, in die Logdatei.

Option	Beschreibung
-ek *key*	Gibt an, dass an der Eingabeaufforderung zur Eingabe des Chiffrierschlüssels für stark verschlüsselte Datenbanken aufgefordert werden soll. Wenn Sie mit einer stark verschlüsselten Datenbank arbeiten, müssen Sie den Chiffrierschlüssel übergeben, um die Datenbank oder das Transaktionslog, einschließlich der Offline-Transaktionslogs, überhaupt verwenden zu können. Bei stark verschlüsselten Datenbanken müssen Sie entweder -ek oder -ep, dürfen aber nicht beide angeben. Wenn Sie bei einer stark verschlüsselten Datenbank keinen Schlüssel angeben, schlägt der Befehl fehl.
-ep	Gibt an, dass zur Eingabe des Chiffrierschlüssel aufgefordert werden soll. Diese Option öffnet ein Fenster, in das Sie den Chiffrierschlüssel eingeben. Diese zusätzliche Sicherheitsmaßnahme verhindert, dass der Chiffrierschlüssel in lesbarer Form angezeigt wird. Bei stark verschlüsselten Datenbanken müssen Sie entweder -ek oder -ep, dürfen aber nicht beide angeben. Wenn Sie bei einer stark verschlüsselten Datenbank keinen Schlüssel angeben, schlägt der Befehl fehl.
-g *n*	Weist den SQL Remote-Nachrichtenagenten an, Transaktionen, die weniger als *n* Vorgänge enthalten, mit den nachfolgenden Transaktionen zu gruppieren. Der Standardwert ist zwanzig Vorgänge. Eine Erhöhung von *n* kann die Prozessverarbeitung von eintreffenden Nachrichten beschleunigen, indem weniger Festschreibungen durchgeführt werden. Allerdings kann das auch zu Deadlocks und Blockierungen führen, weil die Größe der Transaktionen erhöht wird.
-l *length*	Gibt die Maximallänge der einzelnen zu versendenden Nachrichten in Byte an. Die *size* ist die Speichermenge in Byte. Verwenden Sie **k**, **m** oder **g**, um die Einheit in kB, MB bzw. GB anzugeben. Längere Transaktionen werden in mehr als eine Nachricht aufgeteilt. Der Standardwert ist 50000 Byte und die Minimallänge ist 10000 Byte. **Vorsicht** Die maximale Nachrichtenlänge muss bei allen Computern in einer Installation dieselbe sein. Bei Plattformen mit eingeschränkter Speicherzuordnung muss dieser Wert kleiner als die maximale Speicherzuordnung des Betriebssystems sein.

Option	Beschreibung
-m *size*	Bestimmt die Höchstmenge an Speicher, die vom SQL Remote-Nachrichtenagenten zum Aufbau der Nachrichten und für das Caching von eintreffenden Nachrichten verwendet werden soll. Die *size* ist die Speichermenge in Byte. Verwenden Sie **k**, **m** oder **g**, um die Einheit in kB, MB bzw. GB anzugeben. Der Standardwert ist 2048 Kilobyte (2 MB). Wenn alle entfernten Datenbanken eindeutige Teilmengen der Vorgänge erhalten, die repliziert werden, wird eine separate Nachricht für jede entfernte Datenbank gleichzeitig aufgebaut. Für eine Gruppe von entfernten Benutzern, die dieselben Vorgänge erhalten, wird nur eine Nachricht aufgebaut. Wenn der verwendete Speicher den Wert -m überschreitet, werden Nachrichten versendet, bevor sie ihren Höchstwert (wie von der Option -l festgelegt) erreichen. Wenn Nachrichten eintreffen, werden sie vom SQL Remote-Nachrichtenagenten im Speicher abgelegt, bis sie übernommen werden. Dieses Caching von Nachrichten verhindert ein nochmaliges Lesen von Nachrichten, die die falsche Reihenfolge haben, was möglicherweise die Performance auf großen Installationen vermindert. Wenn die mit der Option -m festgelegte Speicherzuweisung überschritten wird, werden Nachrichten gelöscht, und zwar jene zuerst, die am längsten nicht verwendet wurden.
-ml *directory*	Gibt den Speicherort von Offline-Transaktionslog-Spiegeldateien an. Mit dieser Option kann dbremote alte Transaktionslog-Spiegeldateien löschen, wenn einer der beiden folgenden Umstände eintritt: ● Der Offline-Transaktionslogspiegel befindet sich in einem anderen Verzeichnis als der Transaktionslogspiegel. ● dbremote läuft nicht auf demselben Computer wie der entfernte Datenbankserver In einer normalen Installation befinden sich der aktive Transaktionslogspiegel und die umbenannten Transaktionslogspiegel in demselben Verzeichnis und dbremote läuft auf demselben Computer wie die entfernte Datenbank, sodass diese Option nicht erforderlich ist und alte Transaktionslog-Spiegeldateien automatisch gelöscht werden. Transaktionslogs in diesem Verzeichnis sind nur betroffen, wenn die Datenbankoption delete_old_logs auf einen anderen Wert als Off eingestellt ist. Siehe „delete_old_logs-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*].

Option	Beschreibung
-o *filename*	Gibt die Nachrichten in die angegebene Logdatei aus. Standardmäßig werden Ausgaben auf dem Bildschirm angezeigt.
-os *size*	Legt die maximale Dateigröße zur Protokollierung von Ausgabenachrichten fest. Die *size* ist die Speichermenge in Byte. Verwenden Sie **k**, **m** oder **g**, um die Einheit in kB, MB bzw. GB anzugeben. Standardmäßig gibt es kein Limit, und die Untergrenze ist 10000 Byte. Bevor SQL Remote Ausgabenachrichten in eine Ausgabelogdatei protokolliert, wird die aktuelle Dateigröße überprüft. Wenn die Dateigröße durch die Lognachricht den festgelegten Wert überschreitet, benennt SQL Remote die Ausgabedatei in *jjmmttxx.dbs* um, wobei *xx* eine Zahl ist, die bei 00 beginnt und weiter erhöht wird (und auch mehr als 2 Stellen lang sein kann), und *jjmmtt* für das aktuelle Jahr sowie den aktuellen Monat und Tag steht. Wenn der SQL Remote-Nachrichtenagent für lange Zeit im kontinuierlichen Modus läuft, können Sie mit dieser Option alte Ausgabelogdateien manuell löschen und dadurch Speicherplatz freisetzen.
-ot *file*	Kürzt die Ausgabelogdatei und fügt anschließend Ausgabemeldungen an. Standardmäßig werden Ausgaben auf dem Bildschirm angezeigt.
-p	Bereinigt Nachrichten nicht.
-q	Startet den SQL Remote-Nachrichtenagenten mit einem minimierten Fenster. Diese Option gilt nur für Windows-Betriebssysteme.
-qc	Schließt nach Abschluss das SQL Remote-Fenster.
-r	Empfängt Nachrichten. Wenn weder -r noch -s angegeben ist, führt der SQL Remote-Nachrichtenagent beide Phasen aus. Sonst werden nur die angegebenen Phasen ausgeführt. Der SQL Remote-Nachrichtenagent läuft im kontinuierlichen Modus, wenn er mit der Option -r gestartet wird. Damit der SQL Remote-Nachrichtenagent nach dem Nachrichtenempfang herunterfährt, verwenden Sie zusätzlich zu -r auch die Option -b.

Option	Beschreibung
-rd *minutes*	Legt die Abrufhäufigkeit für eingehende Nachrichten fest. Standardmäßig sucht der SQL Remote-Nachrichtenagent jede Minute nach ankommenden Nachrichten. Diese Option (rd steht für **receive delay**) ermöglicht die Konfiguration der Abrufhäufigkeit, die nützlich ist, wenn der Abruf die Systemressourcen stark in Anspruch nimmt. Der Standardwert ist eine Minute. Sie können das Suffix **s** nach der Zahl benutzen, um Sekunden zu definieren. Dies kann nützlich sein, wenn häufige Abrufe gewünscht werden. Beispiel: Der folgende Befehl führt einen Abruf alle dreißig Sekunden durch: `dbremote -rd 30s` Die Option -rd wird häufig zusammen mit der Option -rp verwendet, die die Anzahl der Abrufe festlegt, die der SQL Remote-Nachrichtenagent abwartet, bevor das Neusenden einer fehlenden Nachricht angefordert wird. Siehe „Performance beim Nachrichtenempfang“ auf Seite 99.
-ro *filename*	Protokolliert die entfernte Ausgabe in einer Datei. Diese Option wird in der konsolidierten Datenbank verwendet. Wenn entfernte Datenbanken so konfiguriert sind, dass sie Ausgabeloginformationen an die konsolidierte Datenbank senden, schreibt diese Option die Informationen in eine Datei. Die Option wird bereitgestellt, damit Administratoren Fehler an entfernten Standorten beheben können. Siehe „Sammeln von Fehlern aus der entfernten Datenbank“ auf Seite 142.

Option	Beschreibung
-rp *number*	Gibt die Anzahl der Empfangsabrufe an, bevor eine Nachricht als verloren angesehen wird. Wenn der SQL Remote-Nachrichtenagent kontinuierlich läuft, prüft der Nachrichtenagent die Nachrichten in bestimmten Abständen. Wenn nach einer bestimmten Anzahl von Abfragen (standardmäßig nach einer) eine Nachricht fehlt, nimmt der SQL Remote-Nachrichtenagent an, dass sie verloren gegangen ist, und fordert an, dass die Nachricht erneut gesendet wird. In langsamen Nachrichtensystemen kann dieses Verhalten zu vielen unnötigen Neusendeanforderungen führen. Sie können mit dieser Option festlegen, wie oft abgerufen wird, bevor eine Neusendeanforderung erfolgt, um die Anzahl der Neusendeanforderungen zu minimieren. Weitere Hinweise zum Konfigurieren dieser Option finden Sie unter „Performance beim Nachrichtenempfang“ auf Seite 99. Die Befehlszeilenoption -rp wird oft in Verbindung mit der Option -rd verwendet, mit der die Abrufhäufigkeit für eintreffende Nachrichten eingestellt wird.
-rt *filename*	Kürzt die Ausgabelogdatei beim Start und hängt dann die Logausgabe von der entfernten Datenbank an die Datei an. Diese Option wird in der konsolidierten Datenbank verwendet. Sie ist fast identisch mit der Option -ro, nur wird die Datei beim Start gekürzt.
-ru *time*	Gibt die Warteperiode zum neuerlichen Durchsuchen des Logs bei Empfang einer Neusendeanforderung an. Diese Option steuert den Parameter **resend urgency**. Diese Angabe legt die Periode zwischen der Erkennung einer Neusendeanforderung und der Verarbeitung dieser Neusendeanforderung durch den SQL Remote-Nachrichtenagenten fest. Benutzen Sie diesen Parameter, um den SQL Remote-Nachrichtenagenten zu unterstützen, Neusendeanforderungen von vielen Benutzern abzuwickeln, bevor das Log nochmals durchsucht wird. Die Zeiteinheit kann **s** (Sekunden), **m** (Minuten), **h** (Stunden) oder **d** (Tage) sein.
-s	Versendet Nachrichten. Wenn weder -r noch -s angegeben ist, führt der SQL Remote-Nachrichtenagent beide Phasen aus. Sonst werden nur die angegebenen Phasen ausgeführt.

Option	Beschreibung
-sd *time*	Steuert die Verzögerung zwischen Abrufen des Datenbank-Transaktionslogs. Die Befehlszeilenoption -sd wird nur bei Ausführung im kontinuierlichen Modus verwendet. Der Standardwert ist eine Minute. Steuert den Parameter **send delay**, der angibt, wie lange zwischen Abrufen abgewartet wird, um die aus dem Transaktionslog entnommene Datenmenge zu vergrößern.
-t	Repliziert alle Trigger. Wenn Sie die Option verwenden, müssen Sie sicherstellen, dass die Triggeraktionen in den entfernten Datenbanken nicht zweimal ausgeführt werden, nämlich einmal vom Trigger, der am entfernten Computer ausgelöst wird, und das andere Mal von der expliziten Anwendung der replizierten Aktionen aus der konsolidierten Datenbank. Um sicherzustellen, dass Triggeraktionen nicht zweimal ausgeführt werden, können Sie die Trigger mit der Anweisung IF CURRENT REMOTE USER IS NULL…END IF umschließen. Siehe Den CURRENT REMOTE USER-Spezialwert verwenden auf Seite 49.
-ts *session-name*(*session-option*=[*option-value*;...])	Richtet eine SQL Remote-Protokollierungssitzung ein. Der Sitzungsname muss **logging** lauten. Alle Daten, die nach dem Teil `-ts logging` der Option angegeben werden, müssen ohne Leerzeichen angegeben werden. Detaillierte Hinweise über die Optionen für Protokollierungssitzungen finden Sie unter „mlsrv16-Option -ts“ [*MobiLink - Serveradministration*]. Siehe auch „Dienstprogramm für die Verwaltung von ETD-Dateien (dbmanageetd)“ [*SQL Anywhere Server - Datenbankadministration*].
-u	Verarbeitet nur Transaktionen, die in Offline-Transaktionslogs vorhanden sind. Die Option verhindert, dass der SQL Remote-Nachrichtenagent Transaktionen nach der letzten Sicherung verarbeitet. Wenn Sie diese Option verwenden, werden ausgehende Transaktionen und die Bestätigung von eintreffenden Transaktionen erst versendet, wenn sie in Offline-Transaktionslogs vorhanden sind. Es werden nur Transaktionen von umbenannten Logs bearbeitet.

Option	Beschreibung
-ud	Führt den SQL Remote-Nachrichtenagenten als Daemon auf Unix-Plattformen aus. Wenn Sie den SQL Remote-Nachrichtenagenten als Daemon ausführen, müssen Sie auch die Befehlszeilenoption -o oder -ot benutzen, um die Ausgabeinformationen zu protokollieren. Wenn Sie den SQL Remote-Nachrichtenagenten als Daemon ausführen und FTP- oder SMTP-Nachrichtenverbindungen verwenden, müssen Sie die Nachrichtenverbindungsparameter in der Datenbank speichern, weil der SQL Remote-Nachrichtenagent den Benutzer nicht zur Eingabe dieser Optionen auffordert, wenn er als Daemon läuft. Hinweise zu Nachrichtenverbindungsparametern finden Sie unter „Festlegen von Steuerungsparametern für den entfernten Nachrichtentyp“ auf Seite 118. Starten Sie den SQL Remote-Nachrichtenagenten als Daemon, werden seine Berechtigungen durch die unmask-Einstellung des aktuellen Benutzers gesteuert. Es wird empfohlen, den umask-Wert vor dem Start des SQL Remote-Nachrichtenagenten festzulegen, um sicherzustellen, dass er die entsprechenden Berechtigungen hat.
-ui	Startet den SQL Remote-Nachrichtenagenten unter Linux mit X Window-Serverunterstützung im Shell-Modus, wenn keine nutzbare Anzeige verfügbar ist.
-ux	Öffnet das SQL Remote-Nachrichtenagent-Fenster unter Solaris und Linux. Wenn -ux angegeben ist, muss dbremote in der Lage sein, eine verwendbare Anzeige zu finden. Wenn keine gefunden wird, weil z.B. die DISPLAY-Umgebungsvariable nicht eingestellt ist oder der X Window-Server nicht läuft, schlägt der Start von dbremote fehl. Unter Windows erscheint das SQL Remote-Meldungsfenster automatisch.
-v	Zeigt eine ausführliche Ausgabe an. Diese Option übermittelt die in den Nachrichten enthaltenen SQL-Anweisungen an das Meldungsfenster und, falls die Option -o oder -ot verwendet wird, an eine Logdatei.

Option	Beschreibung
-w *n*	Legt die Anzahl der Datenbank-Worker-Threads zur Übernahme eintreffender Nachrichten fest. Diese Option wird unter Windows Mobile nicht unterstützt. Der Standardwert ist Null, was heißt, dass alle Nachrichten durch den (einzigen) Haupt-Thread übernommen werden. Ein Wert von 1 (eins) legt fest, dass ein Thread die Nachrichten vom Nachrichtensystem empfängt, und ein zweiter die Nachrichten in die Datenbank übernimmt. Die maximale Anzahl von Datenbank-Worker-Threads ist 50. Die Option -w ermöglicht eine Erhöhung des Durchsatzes von eintreffenden Nachrichten mit Hardware-Upgrades. Wenn Sie die konsolidierte Datenbank auf ein Gerät legen, das viele gleichzeitige Vorgänge durchführen kann (eine RAID-Anordnung mit einem logischen Laufwerk im Striped-Modus), kann das den Durchsatz von eintreffenden Nachrichten erhöhen. Mehrere Prozessoren auf dem Computer, auf dem der SQL Remote-Nachrichtenagent läuft, können ebenfalls den Durchsatz von eintreffenden Nachrichten erhöhen. Die Option -w erhöht die Performance auf einer Hardware, die nicht mehrere Vorgänge gleichzeitig ausführen kann, nicht merklich. Eintreffende Meldungen von einer einzigen entfernten Datenbank werden nie mit mehreren Threads angewendet. Meldungen von einer einzigen entfernten Datenbank werden stets seriell in der richtigen Reihenfolge angewendet.
-x [*size*]	Benennt das Transaktionslog um und startet es neu, nachdem es nach neuen ausgehenden Meldungen durchsucht wurde. Unter gewissen Umständen kann die Replikation von Daten in eine konsolidierte Datenbank die Sicherung entfernter Datenbanken oder das Umbenennen des Transaktionslogs bei heruntergefahrenem Datenbankserver ersetzen. Wenn der optionale Qualifizierer *size* angegeben wird, erfolgt die Umbenennung des Transaktionslogs nur bei Überschreitung der angegebenen Größe. Die *size* ist die Speichermenge in Byte. Verwenden Sie **k**, **m** oder **g**, um die Einheit in kB, MB bzw. GB anzugeben. Der Standardwert ist 0.

Option	Beschreibung
transaction-logs-directory	Gibt das Verzeichnis an, in dem alte Transaktionslogs gespeichert werden. Der optionale *directory*-Parameter bestimmt ein Verzeichnis, in dem alte Transaktionslogs aufbewahrt werden, damit der SQL Remote-Nachrichtenagent Zugriff auf Ereignisse hat, die vor dem Start des aktuellen Logs stattgefunden haben.

Bemerkungen

Sie können den SQL Remote-Nachrichtenagenten auch aus Ihrer eigenen Anwendung ausführen, indem Sie die DBTools-Bibliothek aufrufen. Weitere Hinweise finden Sie in der Datei *dbrmt.h* im Verzeichnis *%SQLANY16%\SDK\Include*.

Die Benutzer-ID in der Befehlszeile des SQL Remote-Nachrichtenagenten muss über die SYS_RUN_REPLICATION_ROLE-Systemrolle verfügen.

Der SQL Remote-Nachrichtenagent verwendet mehrere Datenbankverbindungen.

- **Nachrichtensystem-Steuerungsparameter** SQL Remote verwendet mehrere Registrierungseinstellungen, um das Verhalten von Nachrichtenverbindungen zu steuern.

  Unter Windows werden die Steuerungsparameter der Nachrichtenverbindung in der Registrierung an der folgenden Position gespeichert:

  ```
  \\HKEY_CURRENT_USER
   \Software
    \Sybase
      \SQL Remote
  ```

Siehe auch

- „SQL Remote-Nachrichtenagent (dbremote)“ auf Seite 90
- „SQL Remote-Nachrichtensysteme“ auf Seite 114
- „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Extraktionsdienstprogramm (dbxtract)

Dieses Dienstprogramm extrahiert eine entfernte Datenbank aus einer konsolidierten SQL Anywhere-Datenbank. Benutzer müssen über die SYS_REPLICATION_ADMIN_ROLE-Systemrolle verfügen.

Syntax

dbxtract [*options*] [*directory*] *subscriber*

Option	Beschreibung
@*data*	Liest Parameter aus einer Konfigurationsdatei ein. Siehe „@data - Datenbankserveroption“ [*SQL Anywhere Server - Datenbankadministration*]. Verwenden Sie diese Option, um Optionen aus der angegebenen Umgebungsvariablen oder Konfigurationsdatei einzulesen. Wenn beide mit demselben Namen vorhanden sind, wird die Umgebungsvariable verwendet. Siehe „Konfigurationsdateien“ [*SQL Anywhere Server - Datenbankadministration*]. Wenn Sie Kennwörter oder andere Informationen in einer Konfigurationsdatei schützen möchten, können Sie das Dienstprogramm zum Ausblenden von Dateien zum Verschleiern des Inhalts von Konfigurationsdateien verwenden. Siehe „Dienstprogramm zum Verschleiern von Dateien (dbfhide)“ [*SQL Anywhere Server - Datenbankadministration*].
-ac *"keyword=value;..."*	Stellt eine Verbindung mit der in der Verbindungszeichenfolge angegebenen Datenbank zum Neuladen her. Sie können die Vorgänge kombinieren, die eine Datenbank entladen und die Ergebnisse in eine vorhandene Datenbank laden, indem Sie diese Option verwenden. Beispiel: Der folgende Befehl (in einer einzigen Zeile eingegeben) lädt eine Kopie der Daten für den field_user-Subskribenten in eine vorhandene Datenbankdatei namens *c:\field.db*. Der Benutzer DBA muss das SERVER OPERATOR-Systemprivileg haben. `dbxtract -c "UID=DBA;PWD=sql;DBF=c:\cons.db" -ac "UID=DBA;PWD=sql;DBF=c:\field.db" field_user` Wenn Sie diese Option verwenden, wird keine Zwischenkopie der Daten auf dem Plattenspeicher erstellt, und daher geben Sie kein Entladeverzeichnis in der Befehlszeile ein. Dies bietet größere Sicherheit für Ihre Daten, beeinträchtigt aber die Performance.
-al *filename*	Gibt den Namen der Transaktionslogdatei für die neue Datenbank an, falls die Befehlszeilenoption -an verwendet wird.

Option	Beschreibung
-an *database*	Erstellt eine Datenbankdatei mit denselben Einstellungen wie die Datenbank, die extrahiert wird, und lädt sie automatisch neu. Sie können die Vorgänge "Entladen einer Datenbank", "Erstellen einer neuen Datenbank" und "Laden der Daten" mit dieser Option kombinieren. Beispiel: Der folgende Befehl z.B. (in einer einzigen Zeile eingegeben) erstellt eine neue Datenbankdatei namens *c:\field.db* und kopiert das Schema und die Daten für den field_user-Subskribenten von *c:\cons.db* hinein. Der Benutzer DBA muss das SERVER OPERATOR-Systemprivileg haben. `dbxtract -c "UID=DBA;PWD=sql;DBF=c:\cons.db" -an c:\field.db field_user` Wenn Sie diese Option verwenden, wird keine Zwischenkopie der Daten auf dem Plattenspeicher erstellt, und daher geben Sie kein Entladeverzeichnis in der Befehlszeile ein. Dies bietet größere Sicherheit für Ihre Daten, beeinträchtigt aber die Performance.
-ap *size* [**k**]	Legt die Seitengröße der neuen Datenbank fest. Diese Option wird ignoriert, außer -an wird verwendet. Die Seitengröße für eine Datenbank (in Byte) kann 2048, 4096, 8192, 16384 oder 32768 betragen, wobei der Standardwert die Seitengröße der ursprünglichen Datenbank ist. Verwenden Sie **k**, um die Einheit in Kilobyte anzugeben (zum Beispiel -ap 4k). Falls auf dem Datenbankserver bereits Datenbanken ausgeführt werden, muss die Seitengröße des Servers (festgelegt mit der Option -gp) groß genug für diese neue Seitengröße sein. Siehe „Datenbankserveroption -gp “ [*SQL Anywhere Server - Datenbankadministration*].
-b	Startet keine Subskriptionen. Wenn diese Option angegeben ist, müssen Subskriptionen in der konsolidierten Datenbank (für die entfernte Datenbank) und in der entfernten Datenbank (für die konsolidierte Datenbank) explizit mit der START SUBSCRIPTION-Anweisung gestartet werden, damit die Replikation beginnt. Siehe „START SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*].

Option	Beschreibung
-c *"keyword=value;..."*	Gibt Parameter für die Datenbankverbindung in einer Zeichenfolge an. Die Benutzer-ID muss die SYS_RUN_REPLICATION_ROLE-Systemrolle haben, damit gewährleistet ist, dass der Benutzer die Privilegien für alle Tabellen in der Datenbank besitzt. Beispiel: Die folgende Anweisung (in einer einzigen Zeile eingegeben) extrahiert eine Datenbank für die entfernte Benutzer-ID joe_remote von der Beispieldatenbank, die auf dem sample_server-Datenbankserver läuft, indem mit der Benutzer-ID "DBA" und dem Kennwort "sql" eine Verbindung hergestellt wird. Der Benutzer DBA muss das SERVER OPERATOR-Systemprivileg haben. Die Daten werden in das Verzeichnis *c:\extract* entladen. `dbxtract -c "Server=sample_server;DBN=demo;UID=DBA;PWD=sql" c:\extract joe_remote` Wenn keine Verbindungsparameter angegeben sind, werden die Verbindungsparameter aus der SQLCONNECT-Umgebungsvariablen verwendet, falls diese eingestellt ist.
-d	Extrahiert nur Daten. Wenn diese Option angegeben ist, wird die Schemadefinition nicht entladen, und es werden keine Publikationen oder Subskriptionen in der entfernten Datenbank erstellt. Diese Option wird verwendet, wenn bereits eine entfernte Datenbank mit dem richtigen Schema vorhanden ist und nur mit den Daten angefüllt werden muss.

Option	Beschreibung
-ea *alg*	Gibt Verschlüsselungsalgorithmus für die neue Datenbank an. Mit dieser Option können Sie bestimmen, dass ein starker Verschlüsselungsalgorithmus zum Verschlüsseln Ihrer neuen Datenbank verwendet werden soll. Für den FIPS-zertifizierten Algorithmus können Sie entweder AES (Standard) oder AES_FIPS wählen. AES_FIPS benutzt eine eigene Bibliothek und ist mit AES nicht kompatibel. Um die Sicherheit zu erhöhen, geben Sie AES oder AES256 für starke 128-Bit- bzw. 256-Bit-Verschlüsselung an. Geben Sie AES_FIPS oder AES256_FIPS für FIPS-zertifizierte 128-Bit- bzw. 256-Bit-Verschlüsselung an. Bei starker Verschlüsselung müssen Sie auch die Option -ek oder -ep angeben. Weitere Hinweise zur starken Verschlüsselung finden Sie unter „Einfache Verschlüsselung und starke Verschlüsselung“ [*SQL Anywhere Server - Datenbankadministration*]. Um eine nicht verschlüsselte Datenbank zu erstellen, geben Sie die Option -ea none an oder geben Sie weder die Option -ea noch -e, -ep oder -et an. Wenn Sie die Option -ea nicht angeben, ist das Standardverhalten wie folgt: ● -ea none, wenn -ek, -ep oder -et nicht angegeben ist ● -ea AES, wenn -ek oder -ep angegeben ist (mit oder ohne -et) ● -ea simple, wenn -et ohne -ek oder -ep verwendet wird Algorithmusnamen berücksichtigen die Groß- und Kleinschreibung nicht. **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*].

Option	Beschreibung
-ek *key*	Gibt Chiffrierschlüssel für die neue Datenbank an. Mit dieser Option können Sie eine stark verschlüsselte Datenbank erstellen, indem Sie einen Chiffrierschlüssel direkt in die Befehlszeile angeben. Der Algorithmus, der zur Verschlüsselung der Datenbank verwendet wird, ist AES oder AES_FIPS, wie in der Option -ea angegeben. Wenn Sie die Option -ek angeben, ohne -ea anzugeben, wird der AES-Algorithmus verwendet. **Vorsicht** Bei stark verschlüsselten Datenbanken achten Sie darauf, eine Kopie des Schlüssels an einem sicheren Ort zu verwahren. Wenn Sie den Chiffrierschlüssel verlieren, gibt es keine Möglichkeit, auf die Daten zuzugreifen, auch nicht mit Unterstützung durch den technischen Support. Die Datenbank muss verworfen und eine neue Datenbank muss erstellt werden.
-ep	Fordert zur Eingabe des Chiffrierschlüssels für die neue Datenbank auf. Diese Option gibt an, dass Sie eine stark verschlüsselte Datenbank anlegen wollen, wobei der Chiffrierschlüssel in einem Fenster eingegeben wird. Diese zusätzliche Sicherheitsmaßnahme verhindert, dass der Chiffrierschlüssel in lesbarer Form angezeigt wird. Sie müssen den Chiffrierschlüssel ein zweites Mal eingeben, um zu bestätigen, dass die erste Eingabe korrekt war. Wenn die Eingaben nicht übereinstimmen, schlägt die Initialisierung fehl. Siehe „Einfache Verschlüsselung und starke Verschlüsselung“ [*SQL Anywhere Server - Datenbankadministration*].

Option	Beschreibung
-er	Entfernt Verschlüsselung aus verschlüsselten Tabellen während eines Entladevorgangs. Beim Extrahieren aus einer Datenbank mit aktivierter Tabellenverschlüsselung müssen Sie entweder -er oder -et angeben, um anzuzeigen, ob bei der neuen Datenbank die Tabellenverschlüsselung aktiviert ist. Andernfalls erhalten Sie einen Fehler, wenn Sie versuchen, Daten in die neue Datenbank zu laden. Der folgende Befehl wird komplett auf einer Zeile eingegeben und extrahiert eine Datenbank (*cons.db*) mit verschlüsselten Tabellen in eine neue Datenbank (*field.db*), bei der die Tabellenverschlüsselung nicht aktiviert ist, wobei die Verschlüsselung von verschlüsselten Tabellen entfernt wird. Der Benutzer DBA muss das SERVER OPERATOR-Systemprivileg haben. `dbxtract -an c:\field.db -er -c "UID=DBA;PWD=sql;DBF=c:\cons.db;DBKEY=29bN8cj1z field_user"`
-et	Aktiviert Datenbanktabellen-Verschlüsselung in der neuen Datenbank (-an oder -ar muss auch angegeben sein). Wenn Sie die Option -et ohne die Option -ea angeben, wird der AES-Algorithmus verwendet. Wenn Sie die Option -et angeben, müssen Sie auch die Optionen -ep oder -ek angeben. Sie können die Tabellenverschlüsselungs-Einstellungen für die neue Datenbank ändern, sodass sie sich von den Einstellungen der entladenen Datenbank unterscheiden. Beim Neuaufbau einer Datenbank mit aktivierter Tabellenverschlüsselung müssen Sie entweder -er oder -et angeben, um anzuzeigen, ob bei der neuen Datenbank die Tabellenverschlüsselung aktiviert ist. Sonst erhalten Sie einen Fehler, wenn Sie versuchen, Daten in die neue Datenbank zu laden. Das folgende Beispiel muss komplett auf einer Zeile eingegeben werden und extrahiert eine Datenbank (*cons.db*), die mit dem Algorithmus für einfache Verschlüsselung chiffrierte Tabellen enthält, in eine neue Datenbank (*field.db*), bei der die Tabellenverschlüsselung aktiviert ist, und verwendet AES_FIPS-Verschlüsselung mit dem Schlüssel 34jh. Der Benutzer DBA muss das SERVER OPERATOR-Systemprivileg haben. `dbxtract -an c:\field.db -et -ea AES_FIPS -ek 34jh -c "UID=DBA;PWD=sql;DBF=c:\cons.db field_user"`

Option	Beschreibung
-f	Extrahiert voll qualifizierte Publikationen. Normalerweise müssen Sie voll qualifizierte Publikationsdefinitionen für die entfernte Datenbank nicht extrahieren, da üblicherweise alle Zeilen in die konsolidierte Datenbank zurück repliziert werden. Es kann allerdings sinnvoll sein, vollqualifizierte Publikationen in mehrstufigen Systemeinrichtungen oder bei Systemeinrichtungen zu verwenden, in denen die entfernte Datenbank Zeilen enthält, die sich nicht in der konsolidierten Datenbank befinden.
-g	• **Materialisierte Ansichten** Standardmäßig werden als MANUAL REFRESH definierte materialisierte Ansichten nach einem Neuladen nicht initialisiert. Wenn Sie festlegen wollen, dass diese materialisierten Ansichten im Zuge des Neuladeverfahrens initialisiert werden, geben Sie die Option -g an. Wenn Sie -g angeben, führt der Datenbankserver die Systemprozedur sa_refresh_materialized_views aus. Siehe „sa_refresh_materialized_views-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]. Bei der Verwendung der Option -g sollten Sie bedenken, dass die Initialisierung aller materialisierten Ansichten dazu führen kann, dass der Neuladeprozess signifikant länger dauert. Andererseits bewirkt ein Weglassen der Option -g, dass die erste Abfrage, die eine nicht initialisierte Ansicht zu verwenden versucht, warten muss, während der Datenbankserver die Ansicht initialisiert, was zu einer unerwarteten Verzögerung führen kann. Wenn Sie die Option -g nicht verwenden, können Sie die materialisierten Ansichten auch manuell initialisieren, nachdem das Neuladen abgeschlossen ist. Siehe „Initialisieren einer materialisierten Ansicht“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]. • **Textindizes** Standardmäßig werden als MANUAL REFRESH definierte Textindizes nach einem Neuladen nicht initialisiert. Wenn Sie definieren möchten, dass Textindizes im Zuge des Neuladeverfahrens initialisiert werden, geben Sie die Option -g an. Wenn Sie -g angeben, führt der Datenbankserver die Systemprozedur sa_refresh_text_indexes aus. Siehe „sa_refresh_text_indexes-Systemprozedur“ [*SQL Anywhere Server - SQL-Referenzhandbuch*].

Option	Beschreibung
-ii	Führt ein internes Entladen und ein internes Neuladen durch. Diese Option zwingt das Reload-Skript, die internen UNLOAD- und LOAD TABLE-Anweisungen an Stelle der Interactive SQL OUTPUT- und INPUT-Anweisungen zu verwenden, um Daten zu entladen bzw. zu laden. Diese Kombination der Vorgänge ist das Standardverhalten. Externe Vorgänge verwenden den Pfad der Datendateien relativ zum aktuellen Arbeitsverzeichnis von dbxtract, während interne Anweisungen den Pfad relativ zum Datenbankserver benutzen.
-ix	Führt ein internes Entladen und ein externes Neuladen durch. Diese Option zwingt das Reload-Skript dazu, die interne UNLOAD-Anweisung zu verwenden, um Daten zu entladen, und die Interactive SQL INPUT-Anweisung dazu, die Daten in die neue Datenbank zu laden. Externe Vorgänge verwenden den Pfad der Datendateien relativ zum aktuellen Arbeitsverzeichnis von dbxtract, während interne Anweisungen den Pfad relativ zum Datenbankserver benutzen.
-l *level*	Führt alle Extraktionsvorgänge auf der angegebenen Isolationsstufe aus. Die Standardeinstellung ist die Isolationsstufe 0. Wenn Sie eine Datenbank aus einem aktiven Server extrahieren, sollten Sie dies mit der Isolationsstufe 3 tun, um sicherzustellen, dass die Daten in der extrahierten Datenbank mit den Daten auf dem Datenbankserver konsistent sind. Ein Erhöhen der Isolationsstufe kann zu einer großen Anzahl von Sperren führen, die vom Extraktionsdienstprogramm (dbxtract) verwendet werden. Dies schränkt möglicherweise die Datenbanknutzung für andere Benutzer ein. Siehe „Extraktionsdienstprogramm (dbxtract)“ auf Seite 214.
-n	Extrahiert nur die Schemadefinition. Bei dieser Option werden keine Daten entladen. Die Reload-Datei enthält nur SQL-Anweisungen, um das Datenbankschema aufzubauen. Sie können die SYNCHRONIZE SUBSCRIPTION-Anweisung verwenden, um die Daten über das Nachrichtensystem zu laden. Siehe „SYNCHRONIZE SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]. Publikationen, Subskriptionen sowie PUBLISH- und SUBSCRIBE-Privilegien sind Teil des Schemas. In diesem Beispiel muss der Benutzer DBA das SERVER OPERATOR-Systemprivileg haben. `dbxtract -c "UID=DBA;PWD=sql;DBF=c:\remote\cons\cons.db" -n "c:\remote\reload.sql" UserName`

Option	Beschreibung
-nl	Extrahiert die Struktur (dasselbe Verhalten wie bei der Option -n), aber die resultierende Datei *reload.sql* umfasst auch LOAD TABLE- oder INPUT-Anweisungen für jede Tabelle. Wenn diese Option verwendet wird, werden keine Benutzerdaten extrahiert. Wenn Sie die Option -nl angeben, müssen Sie auch ein Datenverzeichnis aufnehmen, damit die LOAD/INPUT-Anweisungen generiert werden können, auch wenn keine Dateien in dieses Verzeichnis geschrieben werden. Mit dieser Option können Sie ein Entlade-Skript generieren, ohne dass Daten entladen werden. Sie können die Daten extrahieren, indem Sie die Option -d angeben. Wenn eine Datenbank eine Tabelle enthält, deren Daten nicht entladen werden sollen, kann das Entladen der Daten bei dieser Tabelle vermieden werden, indem `dbxtract -d -e table-name` verwendet wird .
-o *filename*	Gibt die Nachrichten in die angegebene Logdatei aus.
-p *character*	Gibt ein Escapezeichen an. Der Standardwert für das Escapezeichen (\) kann mit dieser Option durch ein anderes Zeichen ersetzt werden.
-q	Dialogfreier Modus: Es werden keine Nachrichten ausgegeben oder Fenster angezeigt. Wenn diese Option angegeben ist, muss -y ebenfalls angegeben sein, weil sonst der Vorgang fehlschlägt. Diese Option ist nur im Befehlszeilen-Dienstprogramm verfügbar.
-r *file*	Gibt den Namen der generierten Reload-Skriptdatei für Interactive SQL an. Der Standardname der Reload-Skriptdatei ist *reload.sql* im aktuellen Verzeichnis. Sie können mit dieser Option einen anderen Dateinamen angeben.
-u	Sortiert während des Entladungsprozesses keine Daten. Standardmäßig sind die Daten in jeder Tabelle nach dem Primärschlüssel aufgelistet. Entladungen laufen mit der Option -u schneller ab, zum Laden der Daten in die entfernte Datenbank ist jedoch mehr Zeit erforderlich.
-v	Zeigt Meldungen ausführlich an. Der Name der Tabelle, die entladen wird, die Anzahl der entladenen Zeilen und die verwendete SELECT-Anweisung.
-xf	Schließt Fremdschlüssel aus. Diese Option kann verwendet werden, wenn eine entfernte Datenbank eine Teilmenge des Schemas der konsolidierten Datenbank enthält und einige Fremdschlüsselreferenzen nicht in der entfernten Datenbank vorhanden sind.

Option	Beschreibung
-xh	Schließt Prozedur-Hooks aus.
-xi	Führt ein externes Entladen und ein internes Neuladen durch. Standardmäßig wird beim Entladen der Datenbank die UNLOAD-Anweisung verwendet, die vom Datenbankserver ausgeführt wird. Wenn Sie ein externes Entladen wählen, verwendet dbxtract stattdessen die OUTPUT-Anweisung. Die OUTPUT-Anweisung wird im Client ausgeführt. Externe Vorgänge verwenden den Pfad der Datendateien relativ zum aktuellen Arbeitsverzeichnis von dbxtract, während interne Anweisungen den Pfad relativ zum Datenbankserver benutzen.
-xp	Extrahiert keine gespeicherten Prozeduren aus der Datenbank.
-xt	Extrahiert keine Trigger aus der Datenbank
-xv	Extrahiert keine Ansichten aus der Datenbank.
-xx	Führt ein externes Entladen und ein externes Laden durch. Verwenden Sie die OUTPUT-Anweisung, um die Daten zu entladen, und die INPUT-Anweisung, um die Daten in die neue Datenbank zu laden. Standardmäßig wird beim Entladen die UNLOAD-Anweisung und beim Laden die LOAD TABLE-Anweisung verwendet. Die internen UNLOAD- und LOAD TABLE-Anweisungen sind schneller als OUTPUT und INPUT. Externe Vorgänge verwenden den Pfad der Datendateien relativ zum aktuellen Arbeitsverzeichnis von dbxtract, während interne Anweisungen den Pfad relativ zum Datenbankserver benutzen.
y	Ersetzen Sie die vorhandene SQL-Skriptdatei ohne Bestätigung.
directory	Gibt das Verzeichnis an, in das die Dateien geschrieben werden. Nicht erforderlich, wenn Sie -an oder -ac verwenden.
subscriber	Gibt den Subskribenten an, für den die Datenbank extrahiert wird.

Bemerkungen

Standardmäßig wird das Extraktionsdienstprogramm (dbxtract) mit der Isolationsstufe 0 ausgeführt. Wenn Sie eine Datenbank aus einem aktiven Server extrahieren, sollten Sie dies mit der Isolationsstufe 3 tun, um sicherzustellen, dass die Daten in der extrahierten Datenbank mit den Daten auf dem Datenbankserver konsistent sind. Ein Ausführen mit Isolationsstufe 3 kann möglicherweise den Zugriff auf den Datenbankserver von anderen Benutzern aufgrund der großen Anzahl von Sperren verzögern. Es

wird empfohlen, dass Sie das Extraktionsdienstprogramm (dbxtract) ausführen, wenn der Server nicht zu beschäftigt ist, oder dafür eine Kopie der Datenbank verwenden.

Das Extraktionsdienstprogramm (dbxtract) erstellt eine SQL-Skriptdatei und eine Reihe von dazugehörenden Datendateien. Die Skriptdatei kann für eine neu initialisierte Datenbank ausgeführt werden, um die Datenbankobjekte zu erstellen und die Daten für die entfernte Datenbank zu laden.

Standardmäßig heißt die Befehlsdatei *reload.sql*.

Wenn der entfernte Benutzer eine Gruppe ist, dann werden alle Benutzer-IDs, die Mitglieder dieser Gruppe sind, extrahiert. Das ermöglicht mehrere Benutzer in einer entfernten Datenbank mit unterschiedlichen Benutzer-IDs, ohne eine benutzerdefinierte Extraktion verwenden zu müssen.

Wenn Sie das Extraktionsdienstprogramm (dbxtract) oder den **Assistenten zum Extrahieren einer Datenbank** mit einer Datenbank der Version 10.0.0 oder später verwenden, muss die Version von dbxtract mit der Version des Datenbankservers übereinstimmen, der für den Zugriff auf die Datenbank verwendet wird. Wenn eine ältere Version von dbxtract mit einem neueren Datenbankserver verwendet wird oder umgekehrt, wird ein Fehler gemeldet.

Das Extraktionsdienstprogramm (dbxtract) und der **Assistent zum Extrahieren einer Datenbank** entladen nicht die für die Benutzer-ID **dbo** während der Erstellung der Datenbank erzeugten Objekte. Änderungen an diesen Objekten, wie das Neudefinieren einer Systemprozedur, gehen beim Entladen der Daten verloren. Die von der Benutzer-ID **dbo** seit der Initialisierung der Datenbank erstellten Objekte werden vom Extraktionsdienstprogramm (dbxtract) entladen und gehen daher nicht verloren.

Beispiel

Entfernte Datenbanken automatisch extrahieren:

1. Stellen Sie eine Verbindung mit der konsolidierten Datenbank als Benutzer mit der SYS_RUN_REPLICATION_ROLE-Systemrolle her.

2. Führen Sie das Extraktionsdienstprogramm (dbxtract) aus und verwenden Sie die Option -ac, um in eine vorhandene Datenbank zu extrahieren, oder die Option -an, um in eine neue Datenbank zu extrahieren. Sie müssen das SERVER OPERATOR-Systemprivileg haben.

   Wenn Sie die Befehlszeilenoption -an angeben, müssen Sie eine leere Datenbank erstellen, bevor Sie das Extraktionsdienstprogramm (dbxtract) ausführen. Der folgende Befehl erstellt beispielsweise eine leere Datenbank namens *mydata.db*.

   ```
   dbinit -dba DBA,sql c:\remote\mydata.db
   ```

   Führen Sie den folgenden Befehl aus, um eine neue entfernte Datenbank aus einer konsolidierten Datenbank zu extrahieren, die sich unter *\consolidateddata.db* befindet. Die neue Datenbank ist für den entfernten Benutzer namens field_user bestimmt und wird unter *c:\remote\mydata.db* erstellt. Der Benutzer DBA muss das SERVER OPERATOR-Systemprivileg haben.

   ```
   dbxtract -c "UID=DBA;PWD=sql;DBF=c:\consolidateddata.db" -an c:\remote
   \mydata.db field_user
   ```

   Die neue entfernte Datenbank *mydata.db* wird mit dem entsprechenden Schema und den entsprechenden entfernten Benutzern, Publikationen, Subskriptionen und Triggern erstellt.

Standardmäßig werden die Daten aus der konsolidierten Datenbank in die entfernten Datenbanken extrahiert und die Subskriptionen gestartet. Das Extraktionsdienstprogramm (dbxtract) startet allerdings nicht den SQL Remote-Nachrichtenagenten, daher werden keine Nachrichten ausgetauscht.

Siehe auch

- „Extraktion von entfernten Datenbanken“ auf Seite 82
- „Datenbankextraktion“ [*SQL Anywhere Server - SQL-Benutzerhandbuch*]

SQL Remote-Optionen

Replikationsoptionen sind Datenbankoptionen, die das Verhalten der Replikation steuern.

Syntax

SET [**TEMPORARY**] **OPTION**
[*userid*. | **PUBLIC**.]*option-name* **=** [*option-value*]

Parameter	Beschreibung
option-name	Der Name der Option, die geändert wird
option-value	Eine Zeichenfolge, die die Einstellung für die Option enthält

Bemerkungen

Diese Optionen werden vom SQL Remote-Nachrichtenagenten verwendet und sollten für die Benutzer-ID eingestellt sein, die in der Befehlszeile des SQL Remote-Nachrichtenagenten angegeben wird. Sie können auch für die öffentliche Verwendung eingestellt werden.

Option	Werte	Standardwert
„blob_threshold-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	Ganzzahl (in kB)	256
„compression-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	Ganzzahl von -1 bis 9	6
„delete_old_logs-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	On, Off, Delay, *n* days	Off
„external_remote_options-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	On, Off	Off

Option	Werte	Standardwert
„qualify_owners-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	On, Off	On
„quote_all_identifiers-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	On, Off	Aus
„replication_error-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	Name der gespeicherten Prozedur	(keine Prozedur)
„replication_error_piece-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	Name der gespeicherten Prozedur	(keine Prozedur)
„save_remote_passwords-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	On, Off	On
„sr_date_format-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	*Datumszeichenfolge*	yyyy/mm/dd
„sr_time_format-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	*Uhrzeit-Zeichenfolge*	hh:nn:ss.Ssssss
„sr_timestamp_format-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	*Zeitstempel-Zeichenfolge*	yyyy/mm/dd hh:nn:ss.Ssssss
„sr_timestamp_with_time_zone_format-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	*Zeitstempel-mit-Zeitzonen-Zeichenfolge*	yyyy/mm/dd hh:nn:ss.Ssssss +hh:nn
„subscribe_by_remote-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	On, Off	On
„verify_all_columns-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	On, Off	Off

Option	Werte	Standardwert
„verify_threshold-Option [SQL Remote]“ [*SQL Anywhere Server - Datenbankadministration*]	Ganzzahl (in Byte)	1000

Gespeicherte SQL Remote-Prozeduren

Sie können die folgenden gespeicherten Prozeduren verwenden, um ein HTTP-Messaging-System zu verwalten.

sr_add_message_server-Systemprozedur

Diese Prozedur legt die Webdienste fest, die benötigt werden, um HTTP-Anforderungen von entfernten Benutzern akzeptieren, sowie die erforderlichen Definitionen, damit der Datenbankserver Zugriff auf das Verzeichnis erhält, in dem die Nachrichtendateien gespeichert sind.

Syntax

CALL sr_add_message_server('*owner*' **);**

Rückgabewert

Keiner. Ein Fehler wird zurückgegeben, wenn beim Erstellen der erforderlichen Objekte für die Definition des Nachrichtenservers Probleme auftreten.

Bemerkungen

Beim ersten Initialisieren einer Datenbank wird keiner der Webdienste festgelegt, die benötigt werden, um HTTP-Anforderungen von entfernten Benutzern zu akzeptieren, und es werden keine Definitionen erstellt, die der Datenbankserver für den Zugriff auf das Verzeichnis verwenden könnte, in dem die Nachrichtendateien gespeichert werden. Die Erstellung dieser Objekte wird mit der gespeicherten Prozedur sr_add_message_server automatisiert, in der durch einen optionalen Parameter der Eigentümer aller Objekte angegeben werden kann. Objektnamen dürfen nicht mehrfach vorkommen.

Siehe auch

- „sr_drop_message_server-Systemprozedur“ auf Seite 229
- „sr_update_message_server-Systemprozedur“ auf Seite 229

Beispiel

Die folgenden Anweisungen führen dazu, dass die Nachrichtenserver-Datenbank (msgsrv), alle für den Nachrichtenserver benötigten Objekte festlegt und angibt, dass der Benutzer cons (in diesem Fall die konsolidierte Datenbank) Eigentümer aller dieser Objekte sein soll.

```
CREATE ROLE FOR USER cons;
SET REMOTE http OPTION cons.root_directory='c:\\tutorial\\messages';
CALL sr_add_message_server( 'cons' );
COMMIT;
```

sr_drop_message_server-Systemprozedur

Diese Prozedur löscht alle durch sr_add_message_server erstellten Objekte.

Syntax

CALL sr_drop_message_server;

Rückgabewert

Keiner. Ein Fehler wird zurückgegeben, wenn beim Erstellen der erforderlichen Objekte für die Definition des Nachrichtenservers Probleme auftreten.

Bemerkungen

Diese gespeicherte Prozedur wird verwendet, um alle durch sr_add_message_server erstellten Objekte löschen.

Siehe auch

- „sr_add_message_server-Systemprozedur“ auf Seite 228
- „sr_update_message_server-Systemprozedur“ auf Seite 229

sr_update_message_server-Systemprozedur

Diese Prozedur muss immer dann aufgerufen werden, wenn sich die SQL Remote-Definitionen im Nachrichtenserver ändern.

Syntax

CALL sr_update_message_server('*owner*' **);**

Rückgabewert

Keiner. Ein Fehler wird zurückgegeben, wenn beim Erstellen der erforderlichen Objekte für die Definition des Nachrichtenservers Probleme auftreten.

Bemerkungen

Für diese Prozedur können Sie mit einem optionalen Parameter den Benutzer angeben, der Eigentümer der in der gespeicherten Prozedur erstellten Objekte sein soll.

Siehe auch

- „sr_add_message_server-Systemprozedur“ auf Seite 228
- „sr_drop_message_server-Systemprozedur“ auf Seite 229

SQL Remote-Systemprozeduren

Die folgenden Namen und Argumente für gespeicherte Prozeduren bilden die Schnittstelle für die benutzerdefinierte Replikation in SQL Remote-Datenbanken.

Hinweise

Wenn nicht anders festgelegt, gelten folgende Bedingungen für Ereignis-Hook-Prozeduren:

- Die gespeicherten Prozeduren müssen über die SYS_REPLICATION_ADMIN_ROLE-Systemrolle verfügen.
- Die Prozedur darf keine Vorgänge festschreiben oder rückgängig machen oder Aktionen setzen, die ein implizites Festschreiben zur Folge haben. Die Aktionen der Prozedur werden automatisch von der aufrufenden Anwendung festgeschrieben.
- Sie können eine Fehlersuche bei Hooks vornehmen, indem Sie die ausführliche Darstellung des SQL Remote-Nachrichtenagenten aktivieren.

Die Tabelle #hook_dict

Die Tabelle #hook_dict wird erstellt, unmittelbar bevor mit der folgenden CREATE-Anweisung ein Hook aufgerufen wird:

```
CREATE TABLE #hook_dict(
NAME VARCHAR(128) NOT NULL UNIQUE,
value VARCHAR(255) NOT NULL );
```

Der SQL Remote-Nachrichtenagent benutzt die Tabelle #hook_dict, um Werte an die Hook-Funktionen zu übergeben. Hook-Funktionen benutzen die Tabelle #hook_dict, um Werte an den SQL Remote-Nachrichtenagenten zurück zu melden.

Systemprozedur sp_hook_dbremote_begin

Verwenden Sie diese Systemprozedur, um benutzerdefinierte Aktionen am Beginn des Replikationsprozesses hinzuzufügen.

Zeilen in der Tabelle #hook_dict

Name	Werte	Beschreibung
send	**true** oder **false**	Gibt an, ob der Prozess den Sendevorgang der Replikation durchführt.
receive	**true** oder **false**	Gibt an, ob der Prozess den Empfangsvorgang der Replikation durchführt.

Privilegien

Hook-Prozeduren können von jedem Benutzer mit dem MANAGE REPLICATION-Systemprivileg erstellt werden. Um jedoch zu gewährleisten, dass ein Hook auf die #hook_dict-Tabelle zugreifen kann, die verwendet wird, um Daten an und aus Hooks zu übergeben, müssen Hooks eine der folgenden Anforderungen erfüllen:

- Eigentümer muss ein Benutzer sein, der die Systemprivilegien SELECT ANY TABLE und UPDATE ANY TABLE hat.

- Sie müssen durch die SQL SECURITY INVOKER-Klausel der CREATE PROCEDURE-Anweisung definiert sein.

Bemerkungen

Wenn eine Prozedur dieses Namens vorhanden ist, wird sie aufgerufen, sobald der SQL Remote-Nachrichtenagent startet.

Systemprozedur sp_hook_dbremote_end

Benutzen Sie diese Systemprozedur, um benutzerdefinierte Aktionen hinzuzufügen, bevor der SQL Remote-Nachrichtenagent beendet.

Zeilen in der Tabelle #hook_dict

Name	Werte	Beschreibung
send	**true** oder **false**	Gibt an, ob der Prozess den Sendevorgang der Replikation durchführt.
receive	**true** oder **false**	Gibt an, ob der Prozess den Empfangsvorgang der Replikation durchführt.
exit code	**integer**	Ein von Null verschiedener Beendigungscode weist auf einen Fehler hin.

Privilegien

Hook-Prozeduren können von jedem Benutzer mit dem MANAGE REPLICATION-Systemprivileg erstellt werden. Um jedoch zu gewährleisten, dass ein Hook auf die #hook_dict-Tabelle zugreifen kann, die verwendet wird, um Daten an und aus Hooks zu übergeben, müssen Hooks eine der folgenden Anforderungen erfüllen:

- Eigentümer muss ein Benutzer sein, der die Systemprivilegien SELECT ANY TABLE und UPDATE ANY TABLE hat.
- Sie müssen durch die SQL SECURITY INVOKER-Klausel der CREATE PROCEDURE-Anweisung definiert sein.

Bemerkungen

Wenn eine Prozedur dieses Namens vorhanden ist, wird sie als letztes Ereignis des Synchronisationsprozesses aufgerufen, bevor der SQL Remote-Nachrichtenagent heruntergefahren wird.

Systemprozedur sp_hook_dbremote_shutdown

Benutzen Sie diese Systemprozedur, um das Herunterfahren des SQL Remote-Nachrichtenagenten einzuleiten.

Zeilen in der Tabelle #hook_dict

Name	Werte	Beschreibung
send	**true** oder **false**	Gibt an, ob der Prozess den Sendevorgang der Replikation durchführt.
receive	**true** oder **false**	Gibt an, ob der Prozess den Empfangsvorgang der Replikation durchführt.
shutdown	**true** oder **false**	Diese Zeile ist **false**, wenn die Prozedur aufgerufen wird. Wenn die Prozedur die Zeile auf **true** aktualisiert, wird der SQL Remote-Nachrichtenagent heruntergefahren.

Privilegien

Hook-Prozeduren können von jedem Benutzer mit dem MANAGE REPLICATION-Systemprivileg erstellt werden. Um jedoch zu gewährleisten, dass ein Hook auf die #hook_dict-Tabelle zugreifen kann, die verwendet wird, um Daten an und aus Hooks zu übergeben, müssen Hooks eine der folgenden Anforderungen erfüllen:

- Eigentümer muss ein Benutzer sein, der die Systemprivilegien SELECT ANY TABLE und UPDATE ANY TABLE hat.
- Sie müssen durch die SQL SECURITY INVOKER-Klausel der CREATE PROCEDURE-Anweisung definiert sein.

Bemerkungen

Wenn eine Prozedur dieses Namens vorhanden ist, wird sie aufgerufen, wenn der SQL Remote-Nachrichtenagent Nachrichten weder sendet noch empfängt, und ermöglicht ein durch den Hook eingeleitetes Herunterfahren des SQL Remote-Nachrichtenagenten.

sp_hook_dbremote_receive_begin

Benutzen Sie diese Systemprozedur, um Aktionen vornehmen zu lassen, bevor der Empfangsteil der Replikation beginnt.

Zeilen in #hook_dict

Keine

Privilegien

Hook-Prozeduren können von jedem Benutzer mit dem MANAGE REPLICATION-Systemprivileg erstellt werden. Um jedoch zu gewährleisten, dass ein Hook auf die #hook_dict-Tabelle zugreifen kann, die verwendet wird, um Daten an und aus Hooks zu übergeben, müssen Hooks eine der folgenden Anforderungen erfüllen:

- Eigentümer muss ein Benutzer sein, der die Systemprivilegien SELECT ANY TABLE und UPDATE ANY TABLE hat.

- Sie müssen durch die SQL SECURITY INVOKER-Klausel der CREATE PROCEDURE-Anweisung definiert sein.

sp_hook_dbremote_receive_end

Benutzen Sie diese Systemprozedur, um Aktionen vornehmen zu lassen, nachdem der Empfangsteil der Replikation abgeschlossen ist.

Zeilen in #hook_dict

Keine

Privilegien

Hook-Prozeduren können von jedem Benutzer mit dem MANAGE REPLICATION-Systemprivileg erstellt werden. Um jedoch zu gewährleisten, dass ein Hook auf die #hook_dict-Tabelle zugreifen kann, die verwendet wird, um Daten an und aus Hooks zu übergeben, müssen Hooks eine der folgenden Anforderungen erfüllen:

- Eigentümer muss ein Benutzer sein, der die Systemprivilegien SELECT ANY TABLE und UPDATE ANY TABLE hat.
- Sie müssen durch die SQL SECURITY INVOKER-Klausel der CREATE PROCEDURE-Anweisung definiert sein.

sp_hook_dbremote_send_begin

Benutzen Sie diese gespeicherte Prozedur, um Aktionen vornehmen zu lassen, bevor der Sendeteil der Replikation beginnt.

Zeilen in #hook_dict

Keine

Privilegien

Hook-Prozeduren können von jedem Benutzer mit dem MANAGE REPLICATION-Systemprivileg erstellt werden. Um jedoch zu gewährleisten, dass ein Hook auf die #hook_dict-Tabelle zugreifen kann, die verwendet wird, um Daten an und aus Hooks zu übergeben, müssen Hooks eine der folgenden Anforderungen erfüllen:

- Eigentümer muss ein Benutzer sein, der die Systemprivilegien SELECT ANY TABLE und UPDATE ANY TABLE hat.
- Sie müssen durch die SQL SECURITY INVOKER-Klausel der CREATE PROCEDURE-Anweisung definiert sein.

sp_hook_dbremote_send_end

Benutzen Sie diese gespeicherte Prozedur, um Aktionen vornehmen zu lassen, nachdem der Sendeteil der Replikation abgeschlossen ist.

Zeilen in #hook_dict

Keine

Privilegien

Hook-Prozeduren können von jedem Benutzer mit dem MANAGE REPLICATION-Systemprivileg erstellt werden. Um jedoch zu gewährleisten, dass ein Hook auf die #hook_dict-Tabelle zugreifen kann, die verwendet wird, um Daten an und aus Hooks zu übergeben, müssen Hooks eine der folgenden Anforderungen erfüllen:

- Eigentümer muss ein Benutzer sein, der die Systemprivilegien SELECT ANY TABLE und UPDATE ANY TABLE hat.
- Sie müssen durch die SQL SECURITY INVOKER-Klausel der CREATE PROCEDURE-Anweisung definiert sein.

sp_hook_dbremote_message_sent

Benutzen Sie diese gespeicherte Prozedur, um Aktionen durchzuführen, nachdem eine Nachricht gesendet wurde.

Zeilen in #hook_dict

Name	Werte
remote user	Der Nachrichtenadressat.

Privilegien

Hook-Prozeduren können von jedem Benutzer mit dem MANAGE REPLICATION-Systemprivileg erstellt werden. Um jedoch zu gewährleisten, dass ein Hook auf die #hook_dict-Tabelle zugreifen kann, die verwendet wird, um Daten an und aus Hooks zu übergeben, müssen Hooks eine der folgenden Anforderungen erfüllen:

- Eigentümer muss ein Benutzer sein, der die Systemprivilegien SELECT ANY TABLE und UPDATE ANY TABLE hat.
- Sie müssen durch die SQL SECURITY INVOKER-Klausel der CREATE PROCEDURE-Anweisung definiert sein.

sp_hook_dbremote_message_missing

Benutzen Sie diese gespeicherte Prozedur, um Aktionen durchzuführen, wenn der SQL Remote-Nachrichtenagent ermittelt hat, dass eine oder mehr Nachrichten eines entfernten Benutzers fehlen.

Zeilen in #hook_dict

Name	Werte
remote user	Der Name des entfernten Benutzers, der Nachrichten nochmals senden muss

Privilegien

Hook-Prozeduren können von jedem Benutzer mit dem MANAGE REPLICATION-Systemprivileg erstellt werden. Um jedoch zu gewährleisten, dass ein Hook auf die #hook_dict-Tabelle zugreifen kann, die verwendet wird, um Daten an und aus Hooks zu übergeben, müssen Hooks eine der folgenden Anforderungen erfüllen:

- Eigentümer muss ein Benutzer sein, der die Systemprivilegien SELECT ANY TABLE und UPDATE ANY TABLE hat.
- Sie müssen durch die SQL SECURITY INVOKER-Klausel der CREATE PROCEDURE-Anweisung definiert sein.

sp_hook_dbremote_message_apply_begin

Benutzen Sie diese gespeicherte Prozedur, um Aktionen durchzuführen, unmittelbar bevor der SQL Remote-Nachrichtenagent eine Nachricht von einem Benutzer in die Datenbank übernimmt.

Zeilen in #hook_dict

Name	Werte
remote user	Der Name des entfernten Benutzers, der die Nachrichten gesendet hat, die jetzt in die Datenbank übernommen werden sollen

Privilegien

Hook-Prozeduren können von jedem Benutzer mit dem MANAGE REPLICATION-Systemprivileg erstellt werden. Um jedoch zu gewährleisten, dass ein Hook auf die #hook_dict-Tabelle zugreifen kann, die verwendet wird, um Daten an und aus Hooks zu übergeben, müssen Hooks eine der folgenden Anforderungen erfüllen:

- Eigentümer muss ein Benutzer sein, der die Systemprivilegien SELECT ANY TABLE und UPDATE ANY TABLE hat.
- Sie müssen durch die SQL SECURITY INVOKER-Klausel der CREATE PROCEDURE-Anweisung definiert sein.

sp_hook_dbremote_message_apply_end

Benutzen Sie diese gespeicherte Prozedur, um Aktionen durchzuführen, unmittelbar nachdem der SQL Remote-Nachrichtenagent eine Nachricht von einem Benutzer in die Datenbank übernommen hat.

Zeilen in #hook_dict

Name	Werte
remote user	Der Name des entfernten Benutzers, der die Nachrichten gesendet hat, die gerade in die Datenbank übernommen wurden

Privilegien

Hook-Prozeduren können von jedem Benutzer mit dem MANAGE REPLICATION-Systemprivileg erstellt werden. Um jedoch zu gewährleisten, dass ein Hook auf die #hook_dict-Tabelle zugreifen kann, die verwendet wird, um Daten an und aus Hooks zu übergeben, müssen Hooks eine der folgenden Anforderungen erfüllen:

- Eigentümer muss ein Benutzer sein, der die Systemprivilegien SELECT ANY TABLE und UPDATE ANY TABLE hat.
- Sie müssen durch die SQL SECURITY INVOKER-Klausel der CREATE PROCEDURE-Anweisung definiert sein.

Systemtabellen von SQL Remote

Systeminformationen von SQL Remote werden im SQL Anywhere-Katalog aufbewahrt. Eine einfachere Version dieser Daten wird in einer Reihe von Systemansichten aufbewahrt. Mit den folgenden Ansichten können Sie auf SQL Remote-Daten zugreifen:

- „SYSARTICLE-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSARTICLECOL-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSPUBLICATION-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSREMOTEOPTION-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSREMOTEOPTIONTYPE-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSREMOTETYPE-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSREMOTEUSER-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYSSUBSCRIPTION-Systemansicht“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

SQL Remote-SQL-Anweisungen

Verwenden Sie die folgenden SQL-Anweisungen, um SQL Remote-Befehle auszuführen:

- „ALTER PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „ALTER REMOTE MESSAGE TYPE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE REMOTE [MESSAGE] TYPE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „CREATE TRIGGER-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „DROP PUBLICATION-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „DROP REMOTE MESSAGE TYPE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „DROP SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „GRANT CONSOLIDATE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „GRANT PUBLISH-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „GRANT REMOTE-Anweisung [SQLRemote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „GRANT ROLE SYS_REPLICATION_ADMIN_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „GRANT ROLE SYS_RUN_REPLICATION_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „PASSTHROUGH-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REMOTE RESET-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REVOKE CONSOLIDATE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REVOKE PUBLISH-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REVOKE REMOTE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „REVOKE ROLE SYS_REPLICATION_ADMIN_ROLE-Anweisung [MobiLink] [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SET REMOTE OPTION-Anweisung [SQL Remote]“
- „START SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „STOP SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SYNCHRONIZE SUBSCRIPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „UPDATE-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

SET REMOTE OPTION-Anweisung [SQL Remote]

Legt einen Nachrichtensteuerungsparameter für eine SQL Remote-Nachrichtenverbindung fest.

Syntax

SET REMOTE *link-name* **OPTION**
[*userid*.| **PUBLIC**.]*link-option-name* = *link-option-value*

link-name :
file
| **ftp**
| **http**
| **smtp**

link-option-name :
common-options
| *file-options*
| *ftp-options*
| *smtp-options*

common-options :
debug
| **encode_dll**
| **max_retries**
| **output_log_send_on_error**
| **output_log_send_limit**
| **output_log_send_now**
| **pause_after_failure**

file-options :
directory
| **invalid_extensions**
| **unlink_delay**

ftp-options :
active_mode
| **host**
| **invalid_extensions**
| **password**
| **port**
| **root_directory**
| **reconnect_retries**
| **reconnect_pause**
| **suppress_dialogs**
| **user**

http-options :
| **certificate**
| **client_port**
| **https**
| **password**
| **proxy**
| **reconnect_retries**
| **reconnect_pause**
| **root_directory**

| url
| user

smtp-options :
local_host
| pop3_host
| pop3_password
| pop3_port
| pop3_userid
| smtp_authenticate
| smtp_option
| smtp_password
| smtp_port
| smtp_userid
| suppress_dialogs
| top_supported

link-option-value : *string*

Parameter

userid Wenn Sie keine *userid* eingeben, wird der aktuelle Publikationseigentümer genommen.

common-options Folgende Optionen gelten für die FILE-, FTP-, HTTP- und SMTP-Nachrichtensysteme gleichermaßen:

- **debug** Dieser Parameter ist entweder auf YES oder NO gesetzt. Der Standardwert ist NO. Wenn auf YES gesetzt, werden spezifische Debug-Ausgaben des Nachrichtensystems angezeigt. Diese Informationen können verwendet werden, um Probleme im Nachrichtensystem zu beheben.

- **max_retries** Standardmäßig gilt: Wenn SQL Remote im kontinuierlichen Modus ausgeführt wird und ein Fehler tritt auf, wenn auf das Nachrichtensystem zugegriffen wird, wird es automatisch nach den Sende- und/oder Empfangsphasen heruntergefahren. Verwenden Sie diesen Parameter, um die Häufigkeit festzulegen, mit der Sie aus SQL Remote zu wiederholen Sie den Sende- und/oder von der Synchronisation, bevor sie beendet.

- **output_log_send_on_error** Sendet Loginformationen, wenn ein Fehler auftritt.

- **output_log_send_limit** Begrenzt die Menge der Informationen, die an die konsolidierte Datenbank gesendet werden. Die Option output_log_send_limit gibt die Anzahl von Bytes am Ende des Ausgabelogs an (d.h. die aktuellsten Einträge), die an die konsolidierte Datenbank gesendet wird. Der Standardwert ist 5 kB.

- **output_log_send_now** Wenn auf YES gesetzt, werden Ausgabeloginformationen an die konsolidierte Datenbank gesendet. Beim nächsten Abruf sendet die entfernte Datenbank die Ausgabeloginformationen und setzt anschließend die Option output_log_send_now auf NO.

- **pause_after_failure** Dieser Parameter gilt, wenn der max_retries-Parameter auf einen anderen Wert als Null gesetzt ist, und SQL Remote im kontinuierlichen Modus ausgeführt wird. Wenn ein Fehler im Nachrichtensystem auftritt, definiert dieser Parameter die Anzahl der Sekunden, die SQL Remote wartet, bis die Sende- und/oder Empfangsphasen wiederholt werden.

- **encode_dll** Wenn Sie ein benutzerdefiniertes Kodierungsschema implementiert haben, müssen Sie dies auf den vollständigen Pfad der von Ihnen erstellten angepassten Kodierungs-DLL einstellen.

file-options Folgende Optionen gelten nur für das FILE-Nachrichtensystem:

- **directory** Das Verzeichnis, in dem die Nachrichten gespeichert werden. Dieser Parameter ist eine Alternative zur SQLREMOTE-Umgebungsvariablen.

- **invalid_extensions** Eine durch Kommas getrennte Liste von Dateierweiterungen, die vom SQL Remote-Nachrichtenagenten (dbremote) beim Generieren von Dateien im Nachrichtensystem nicht verwendet werden sollen.

- **unlink_delay** Die Sekunden, die abgewartet werden sollen, bevor eine Datei gelöscht wird, falls der vorherige Versuch, die Datei zu löschen, fehlgeschlagen ist. Wenn für unlink_delay kein Wert definiert ist, wird als Standardverhalten 1 Sekunde nach dem ersten fehlgeschlagenen Versuch, 2 Sekunden nach dem zweiten, 3 Sekunden nach dem dritten und 4 Sekunden nach dem vierten gewartet.

ftp-options Folgende Optionen gelten nur für das FTP-Nachrichtensystem:

- **active_mode** Dieser Parameter steuert, wie SQL Remote die Client-/Server-Verbindung herstellt. Dieser Parameter ist entweder auf YES oder NO gesetzt. Der Standardwert ist NO (passiver Modus). Der passive Modus ist der bevorzugte Übermittlungsmodus und der Standardwert für die FTP-Nachrichtenverbindung. Im passiven Modus werden alle Datenübertragungsverbindungen durch den Client initiiert, in diesem Fall durch die Nachrichtenverbindung. Im aktiven Modus initiiert der FTP-Server alle Datenverbindungen.

- **host** Der Hostname des Computers, auf dem der FTP-Server läuft. Dieser Parameter kann ein Hostname (z.B. FTP.ianywhere.com) oder eine IP-Adresse (z.B. 192.138.151.66) sein.

- **invalid_extensions** Eine durch Kommas getrennte Liste von Dateierweiterungen, die von dbremote beim Generieren von Dateien im Nachrichtensystem nicht verwendet werden sollen.

- **password** Das Kennwort für den Zugriff auf den FTP-Host

- **port** Die IP-Portnummer, die für die FTP-Verbindung verwendet wird. Dieser Parameter wird gewöhnlich nicht benötigt.

- **reconnect_retries** Gibt an, wie häufig die Verbindung versuchen soll, einen Socket zu öffnen, bevor der Versuch abgebrochen wird. Der Standardwert ist "4". Wenn Sie diesen Parameter setzen, sind nur erneute Verbindungen betroffen. Die ursprünglich vom FTP-Link erstellte Verbindung ist nicht betroffen.

- **reconnect_pause** Die Zeit in Sekunden, die zwischen den Verbindungsversuchen abgewartet wird. Der Standardwert ist 30 Sekunden. Wenn Sie diesen Parameter setzen, sind nur erneute Verbindungen betroffen. Die ursprünglich vom FTP-Link erstellte Verbindung ist nicht betroffen.

- **root_directory** Das Stammverzeichnis auf der FTP-Host-Site, unter dem die Nachrichten gespeichert werden.

- **suppress_dialogs** Dieser Parameter ist auf TRUE oder FALSE gesetzt. Wenn dieser Parameter auf TRUE gesetzt ist, wird das Fenster **Verbinden** nach gescheiterten Verbindungsversuchen zum FTP-Server nicht angezeigt. Stattdessen wird eine Fehlermeldung ausgegeben.

- **user** Der Benutzername für den Zugriff auf den FTP-Host

http-options Diese Optionen gelten nur für das HTTP-Nachrichtensystem:

- **certificate** Um eine sichere (HTTPS-)Anforderung durchführen zu können, muss der Client Zugriff auf das vom HTTPS-Server verwendete Zertifikat haben. Die benötigten Informationen werden in einer Zeichenfolge von durch Semikola getrennten Schlüssel/Werte-Paaren angegeben. Sie können mithilfe des Dateischlüssels den Dateinamen für das Zertifikat angeben. Sie können nicht sowohl einen Dateischlüssel als auch einen Zertifikatschlüssel angeben. Folgende Schlüssel sind verfügbar:

Schlüssel	Abkürzung	Beschreibung
file		The file name of the certificate
certificate	cert	The certificate itself
company	co	The company specified in the certificate
unit		The company unit specified in the certificate
name		The common name specified in the certificate

  Zertifikate sind nur bei Anforderungen erforderlich, die entweder an den HTTPS-Server gerichtet sind oder von einem nicht-sicheren zu einem sicheren Server umgeleitet werden können. Nur PEM-formatierte Zertifikate werden unterstützt. **certificate=**'*Datei=Dateiname*'

  Zertifikatnamen in SQL Anywhere-Datenbanken erstellen

```
CREATE OR REPLACE CERTIFICATE certificate_name FROM FILE
'certificate_file';
```

  Zertifikatnamen für einen HTTPS-Nachrichtentyp verwenden

```
SET REMOTE HTTP OPTION user_name.certificate =
'cert_name=certificate_name';
```

- **client_port** Kennzeichnet die Portnummer, auf der SQL Remote unter Verwendung von HTTP kommuniziert. Sie wird nur für Verbindungen über Firewalls bereitgestellt und empfohlen, die "ausgehende" TCP/IP-Verbindungen filtern. Sie können eine einzelne Portnummer, Bereiche von Portnummern oder eine Kombination aus beiden angeben. Die Angabe einer niedrigen Anzahl von Clientports kann dazu führen, dass SQL Remote nicht in der Lage ist, Nachrichten zu senden und zu empfangen, wenn das Betriebssystem die Ports nicht rechtzeitig freigibt, nachdem SQL Remote den Port bei einer früheren Ausführung geschlossen hat.

- **debug** Wenn dieser Parameter auf YES eingestellt ist, werden alle HTTP-Befehle und -Antworten im Ausgabelog angezeigt. Diese Informationen können zur Fehlerbehandlung von HTTP-Problemen verwendet werden. Der Standardwert ist NO.

- **https** Geben Sie an, ob HTTPS (**https=yes**) oder HTTP (**https=no**) verwendet werden soll.

- **password** Das Kennwort für die Nachrichtenserver-Datenbank. Das Kennwort authentifiziert bei HTTP-Servern und Gateways anderer Hersteller, die RFC 2617 Basic-Authentifizierung verwenden.

- **proxy_host** Gibt den URI eines Proxyservers an. Wird verwendet, wenn SQL Remote über einen Proxyserver auf das Netzwerk zugreifen muss. Zeigt an, dass SQL Remote eine Verbindung zum Proxyserver herstellen soll, um die Anforderung durch ihn an den Nachrichtenserver zu senden.

- **reconnect_retries** Gibt an, wie häufig die Verbindung versuchen soll, einen Socket zu öffnen, bevor der Versuch abgebrochen wird. Der Standardwert ist "4". Wenn Sie diesen Parameter setzen, sind nur erneute Verbindungen betroffen. Die ursprünglich vom FTP-Link erstellte Verbindung ist nicht betroffen.

- **reconnect_pause** Die Zeit in Sekunden, die zwischen den Verbindungsversuchen abgewartet wird. Der Standardwert ist 30 Sekunden. Wenn Sie diesen Parameter setzen, sind nur erneute Verbindungen betroffen. Die ursprünglich vom FTP-Link erstellte Verbindung ist nicht betroffen.

- **root_directory** Dieser HTTP-Steuerungsparameter wird ignoriert, wenn er auf Clientseite festgelegt wurde. Sie müssen diesen Steuerungsparameter im Nachrichtenserver festlegen, bevor Sie die gespeicherte Prozedur sr_add_message_server oder sr_update_message_server aufrufen. Bei der Verwendung des HTTP-Nachrichtensystems kann die für einen entfernten Benutzer oder Publikationseigentümer angegebene Adresse nur ein einziges Unterverzeichnis enthalten und nicht mehrere Unterverzeichnisse.

- **url** Geben Sie den Servernamen oder die IP-Adresse sowie optional die Portnummer für den verwendeten HTTP-Server an, durch Semikola getrennt. Wenn Anforderungen durch den Relay Server übergeben werden, können Sie optional eine URL-Erweiterung hinzufügen, um anzuzeigen, an welche Serverfarm die Anforderung übergeben werden soll.

- **user** Die Benutzer-ID für die Nachrichtenserver-Datenbank. Dient zum Authentifizieren bei HTTP-Servern und Gateways anderer Hersteller, die RFC 2617 Basic-Authentifizierung verwenden.

smtp-options Folgende Optionen gelten nur für das SMTP-Nachrichtensystem:

- **local_host** Der Name des lokalen Computers. Er ist auf Computern nützlich, bei denen SQL Remote nicht in der Lage ist, den lokalen Hostnamen zu bestimmen. Der lokale Hostname wird benötigt, um eine Sitzung mit einem SMTP-Server zu initialisieren. In den meisten Netzwerkumgebungen kann der Hostname automatisch ermittelt werden. In diesem Fall wird dieser Eintrag nicht gebraucht.

- **pop3_host** Der Name des Computers, auf dem der POP-Host läuft. Üblicherweise ist dies derselbe Name wie der SMTP-Host:. Er entspricht dem POP3-Hostfeld im SMTP/POP3-Login-Fenster.

- **pop3_password** Das Kennwort, das zum Abrufen der Mail verwendet wird. Der Wert entspricht dem Kennwortfeld im SMTP/POP3-Login-Fenster.

- **pop3_port** Die Nummer des Ports, den der POP3-Server abhört. Der Standardwert ist 110. Dies entspricht dem Portfeld im SMTP/POP3-Login-Fenster.

- **pop3_userid** Die Benutzer-ID, die zum Abrufen der Mail verwendet wird. Die POP3-Benutzer-ID entspricht dem Benutzer-ID-Feld im SMTP/POP3-Login-Fenster. Sie erhalten eine Benutzer-ID von Ihrem POP3-Host-Administrator.

- **smtp_host** Der Name des Computers, auf dem der SMTP-Server läuft. Er entspricht dem SMTP-Hostfeld im SMTP/POP3-Login-Fenster

- **top_supported** SQL Remote verwendet einen POP3-Befehl namens TOP, um eintreffende Nachrichten zu verarbeiten. Der TOP-Befehl wird möglicherweise nicht von allen POP3-Servern unterstützt. Wenn Sie den Parameter "top_supported" auf NO setzen, verwendet SQL Remote den RETR-Befehl, der weniger effizient ist, aber mit allen POP-Servern funktioniert. Der Standardwert ist YES.

- **smtp_authenticate** Legt fest, ob die SMTP-Verbindung den Benutzer authentifiziert. Der Standardwert ist YES. Setzen Sie diesen Parameter auf NO, um die SMTP-Authentifizierung zu deaktivieren.

- **smtp_userid** Die Benutzer-ID für die SMTP-Authentifizierung. Standardmäßig übernimmt dieser Parameter denselben Wert wie der Parameter pop3_userid. Der Parameter "smtp_userid" muss nur gesetzt werden, wenn die Benutzer-ID von der des POP-Servers abweicht.

- **smtp_password** Das Kennwort für die SMTP-Authentifizierung. Standardmäßig übernimmt dieser Parameter denselben Wert wie der Parameter pop3_password. Der Parameter "smtp_password" muss nur gesetzt werden, wenn die Benutzer-ID von der des POP-Servers abweicht.

- **smtp_port** Die Nummer des Ports, den der SMTP-Server derzeit abhört. Standardwert ist "25". Dies entspricht dem Portfeld im SMTP/POP3-Login-Fenster.

- **suppress_dialogs** Wenn dieser Parameter auf TRUE gesetzt ist, wird das Fenster **Verbinden** nach gescheiterten Verbindungsversuchen zum Mail-Server nicht angezeigt. Stattdessen wird eine Fehlermeldung ausgegeben.

Bemerkungen

Der SQL Remote (dbremote)-Nachrichtenagent speichert Nachrichtenverbindungsparameter, die der Benutzer in das Fenster für den Nachrichtenagenten eingibt, wenn die Nachrichtenverbindung das erste Mal verwendet wird. In diesem Fall ist es nicht nötig, diese Anweisung explizit zu verwenden. Diese Anweisung ist besonders von Nutzen, wenn eine konsolidierte Datenbank für das Extrahieren von zahlreichen Datenbanken vorbereitet wird.

Bei den Optionsnamen wird die Groß-/Kleinschreibung berücksichtigt. Ob die Optionswerte auf Groß-/Kleinschreibung reagieren, hängt von der Option ab: Boolesche Werte reagieren nicht, während Kennwörter, Verzeichnisnamen und andere Zeichenfolgen abhängig von der Reaktion auf Groß-/Kleinschreibung des Dateisystems (bei Verzeichnisnamen) oder der Datenbank (bei Benutzer-IDs und Kennwörtern) sind.

Privilegien

Der Publikationseigentümer kann seine eigenen Optionen setzen. Andernfalls müssen Sie die SYS_REPLICATION_ADMIN_ROLE-Systemrolle haben.

Nebenwirkungen

Automatisches Festschreiben (Autocommit).

Siehe auch

- „Sammeln von Fehlern aus der entfernten Datenbank“ auf Seite 142
- „Festlegen von Steuerungsparametern für den entfernten Nachrichtentyp“ auf Seite 118
- „Benutzerdefinierte Kodierungsschemata“ auf Seite 114
- „SET OPTION-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „Das FTP-Nachrichtensystem“ auf Seite 121
- „Das FILE-Nachrichtensystem“ auf Seite 119
- „Das HTTP-Nachrichtensystem“ auf Seite 124
- „Praktische Einführung: Einrichten eines Replikationssystems unter Verwendung des HTTP-Nachrichtensystems“ auf Seite 171
- „Das SMTP-Nachrichtensystem“ auf Seite 128
- „CREATE CERTIFICATE-Anweisung“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]
- „SET REMOTE OPTION-Anweisung [SQL Remote]“ [*SQL Anywhere Server - SQL-Referenzhandbuch*]

Standards und Kompatibilität

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

Beispiele

Die folgende Anweisung legt den FTP-Host für die FTP-Verbindung des Benutzers Sam_Singer auf *ftp.mycompany.com* fest:

```
SET REMOTE FTP OPTION Sam_Singer.host = 'ftp.mycompany.com';
```

Die folgende Anweisung verhindert, dass SQL Remote die angegebenen Dateierweiterungen für Meldungen verwendet, die generiert werden:

```
SET REMOTE FTP OPTION
"Public"."invalid_extensions"='exe,pif,dll,bat,cmd,vbs';
```

Die folgende Anweisung legt die URL so fest, dass sie auf den Localhost für die HTTP-Verbindung des Benutzers Sam_Singer zeigt:

```
SET REMOTE HTTP OPTION Sam_Singer.url='localhost:8033';
```

Die folgende Anweisung legt die HTTP-URL so fest, dass sie auf einen Relay Server verweist, der die Anforderung an die SRHTTP-Farm weiterleitet:

```
SET REMOTE HTTP OPTION "public"."url"='iis7.company.com:80/rs/client/
rs_client.dll/srhttp';
```

Index

Symbole

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W