SQL Anywhere - SQL Remote
文書バージョン: 17 – 2016-05-11

SQL Remote

目次

1 SQL Remote

このマニュアルでは、モバイルコンピューティング用の SQL Remote データレプリケーションシステムについて説明します。このシステムによって、SQL Anywhere の統合データベースと複数の SQL Anywhere リモートデータベースの間で、電子メールやファイル転送などの間接的リンクを使用したデータ共有が可能になります。

このセクションの内容:

1.1 SQL Remote システム

SQL Remote は、統合データベースと多数のリモートデータベース間のデータベーストランザクションの双方向レプリケーション用に設計された、メッセージベースのテクノロジです。リモートサイトにおける管理およびリソースの要件は最小限に抑えられているので、SQL Remote はモバイルデバイスに最適です。

SQL Remote では、次のような機能を提供します。

複数サブスクライバのサポート

SQL Remote を使用すると、不定期に接続するユーザが、SQL Anywhere 統合データベースと多数のリモート SQL Anywhere データベース (通常、多数のモバイルデータベースを含む) 間でデータをレプリケートできます。

トランザクションログベースのレプリケーション

SQL Remote では、トランザクションログを使用してレプリケーションを行います。このため、更新時にレプリケートされるのは変更されたデータのみで、レプリケーションシステム全体でトランザクションのアトミック性が適正に保たれ、レプリケーションに関わるデータベース間で一貫性が維持されます。

集中管理

SQL Remote は、統合データベースで集中管理されます。企業では、数多くのユニークなデータベースが存在する多数のモバイル環境を使用できますが、各リモートデータベースを個別に管理することはありません。また、エンドユーザが SQL Remote の処理を意識することはありません。

効率的なメモリの使用

効率的な実行のため、SQL Remote はメモリを効率よく使用します。このことによって、既存のリモートコンピュータとデバイス上で SQL Remote を使用できるため、新しいハードウェアに投資する必要がありません。レプリケーションは、限られた領域を使用してリモートコンピュータやデバイスと双方向に行うことができます。統合データベースからリモートデータベースにレプリケートされるのは、関連するデータのみです。

マルチプラットフォームサポート

SQL Remote は、さまざまなオペレーティングシステムとメッセージリンクでサポートされています。SQL Anywhere データベースは、1 つのファイル/オペレーティングシステムから、別のファイル/オペレーティングシステムにコピーできます。

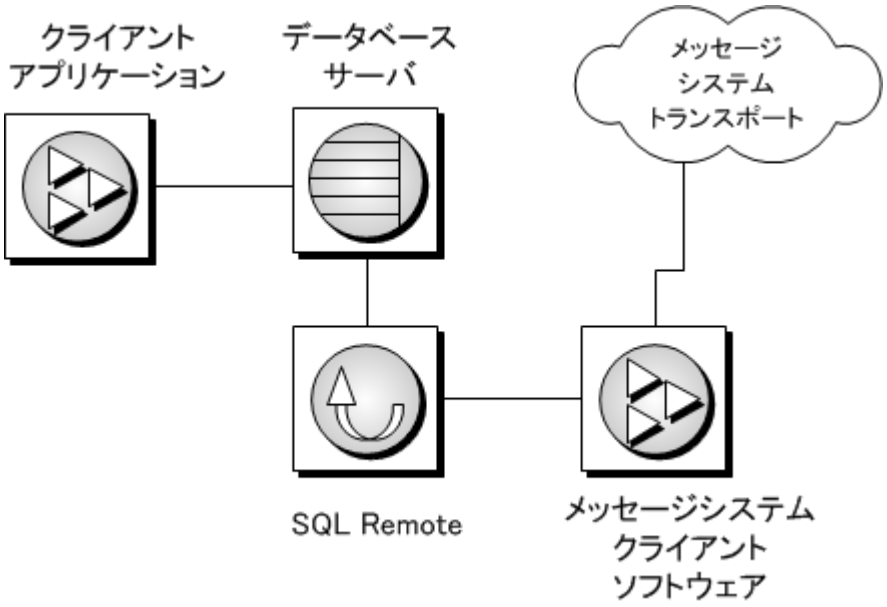
このセクションの内容:

関連情報

SQL Remote システムの作成 [11 ページ]
SQL Remote システムの管理 [81 ページ]
SQL Remote のリファレンス [208 ページ]

1.1.1 SQL Remote のコンポーネント

SQL Remote インストール環境は、データベースサーバ、SQL Remote、メッセージシステムクライアントソフトウェア、クライアントアプリケーションから成ります。



データベースサーバ

統合サイトと各リモートサイトには、SQL Anywhere データベースが必須です。
SQL Remote

データベース間でレプリケーションメッセージを送受信するには、統合サイトと各リモートサイトに SQL Remote をインストールする必要があります。

SQL Remote Message Agent は、クライアント/サーバ接続経由でデータベースサーバに接続します。SQL Remote Message Agent は、データベースサーバと同じコンピュータでも異なるコンピュータでも実行できます。
メッセージシステムクライアントソフトウェア

SQL Remote は、既存のメッセージシステムを使用して、レプリケーションメッセージを転送します。

共有ファイルまたは FTP メッセージシステムを使用している場合、メッセージシステムはオペレーティングシステムに含まれています。

SMTP 電子メールシステムを使用している場合は、統合サイトと各リモートサイトに電子メールクライアントをインストールしておく必要があります。
クライアントアプリケーション

クライアントアプリケーションでは、ODBC、Embedded SQL、またはその他のさまざまなプログラミングインタフェースを使用できます。統合データベースとリモートデータベースのどちらを使用しているかを、クライアントアプリケーション側で認識する必要はありません。クライアントアプリケーション側から見ると、どちらでも同じだからです。SQL Anywhere のプログラミングインタフェースの詳細については、次のリストを参照してください。

1.1.2 典型的な SQL Remote 設定

SQL Remote を使用するかどうかを決定するときは、多くの考慮事項があります。

SQL Remote は、レプリケーションシステム用に設計されたもので、次の要件があります。

多数のリモートデータベース

多数のリモートデータベースへのメッセージを同時に準備できるため、1 つのインストール環境で、数千のリモートデータベースをサポートできます。

随時接続

SQL Remote は、ネットワークに随時または間接的に接続されるデータベースをサポートします。SQL Remote は、各サイトのデータを常に最新に保つような設計にはなっていません。たとえば、SMTP 電子メールシステムを使用して、レプリケーションを実行することがあります。

遅延時間: 短～長

遅延時間が長いというのは、システムにおいて、あるデータベースにデータが入力されてからそのデータが各データベースにレプリケートされるまでのタイムラグが長いということです。SQL Remote の場合、レプリケーションメッセージは、秒、分、時間、または日単位の間隔で送信されます。

容量: 低～中

レプリケーションメッセージは随時配信されるため、各リモートデータベースのトランザクションの容量が大きい場合は、メッセージの容量が大きくなる可能性があります。SQL Remote は、1 つのリモートデータベースについてのレプリケーションデータが比較的低容量であるシステムに最適です。統合データベースでは、SQL Remote は複数データベースへのメッセージを同時に準備できます。

同機種データベース

システム内の各 SQL Anywhere データベースは、同様なスキーマを持つ必要があります。

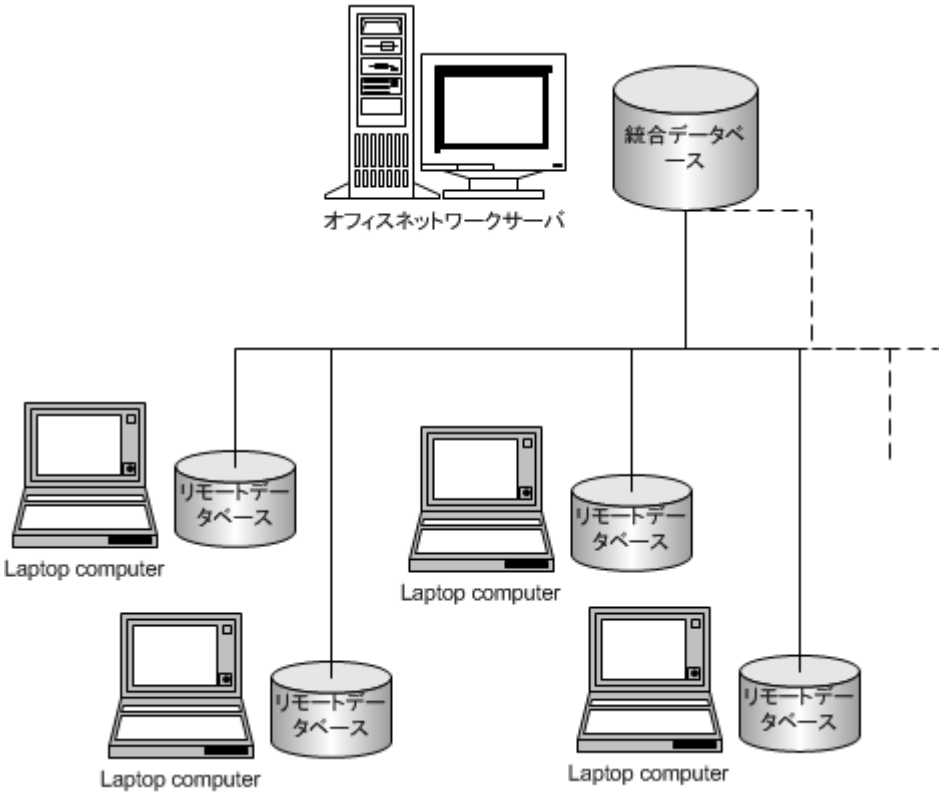
このセクションの内容:

モバイル環境でのサーバ/リモートデータベース間レプリケーション [8 ページ]
この例では、オフィスネットワーク上の統合データベースと、営業担当者のラップトップコンピュータ上にあるパーソナルデータベースとの間で、双方向にレプリケーションできます。SMTP 電子メールシステムがメッセージの伝送手段として使用されています。

複数のオフィスにわたるサーバ間データベースレプリケーション [9 ページ]
この例では、営業所や販売店のデータベースサーバと本社のデータベースサーバ間で、双方向にレプリケーションできます。

1.1.2.1 モバイル環境でのサーバ/リモートデータベース間レプリケーション

この例では、オフィスネットワーク上の統合データベースと、営業担当者のラップトップコンピュータ上にあるパーソナルデータベースとの間で、双方向にレプリケーションできます。SMTP 電子メールシステムがメッセージの伝送手段として使用されています。



統合データベースを管理するには、オフィスネットワークサーバで SQL Anywhere データベースサーバを実行します。SQL Remote は、他のクライアントアプリケーションと同じ方法で統合データベースに接続します。

各営業担当者のラップトップコンピュータには、SQL Anywhere パーソナルサーバ、SQL Anywhere リモートデータベース、SQL Remote がインストールされています。

営業担当者は、外出先からインターネットに接続して SQL Remote を実行できます。このことによって、次の機能が実行されます。

- オフィスネットワークサーバ上の統合データベースから、パブリケーションの更新を受信します。
- ローカルで行った更新内容 (新しい注文など) を、オフィスネットワークサーバ上の統合データベースに送信します。

オフィスネットワークデータベースのパブリケーションの更新には、その営業担当者が取り扱っている製品の新しい特別割引や、新価格、在庫情報などがあります。これらの更新はラップトップ上の SQL Remote によって読み込まれ、自動的に営業担当者のリモートデータベースに適用されます。営業担当者による追加操作は一切不要です。

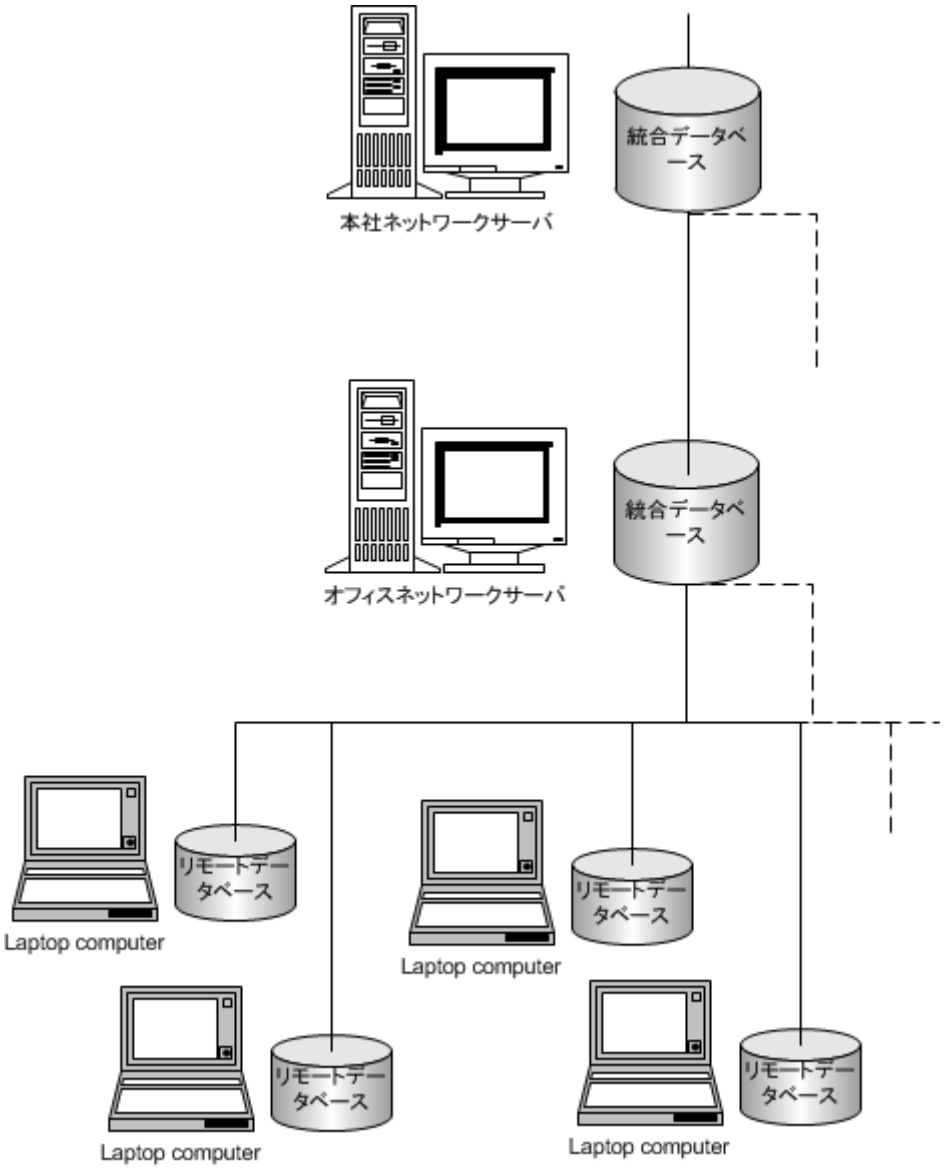
営業担当者が登録した新しい注文も、この担当者が特別な操作をすることなく自動的にオフィスネットワークデータベースに送信され、適用されます。

1.1.2.2 複数のオフィスにわたるサーバ間データベースレプリケーション

この例では、営業所や販売店のデータベースサーバと本社のデータベースサーバ間で、双方向にレプリケーションできます。

各営業所で必要となる作業は、サーバの初期設定と継続したメンテナンスのみです。

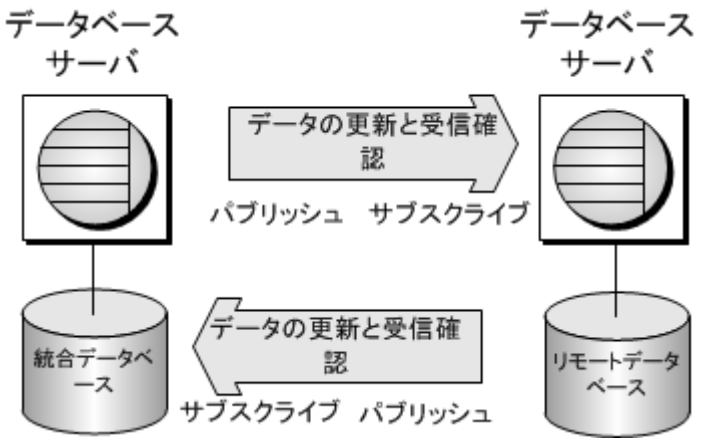
SQL Remote の階層には、レイヤを追加できます。たとえば、各営業所のサーバを統合データベースとして利用して、その営業所からのリモートサブスクライバをサポートすることができます。



SQL Remote は、各オフィスでそれぞれに適したデータセットを受信するように設定できます。スタッフレコードなどのテーブルは、レプリケーションデータと同じデータベース内にあっても機密性を保つことができます。

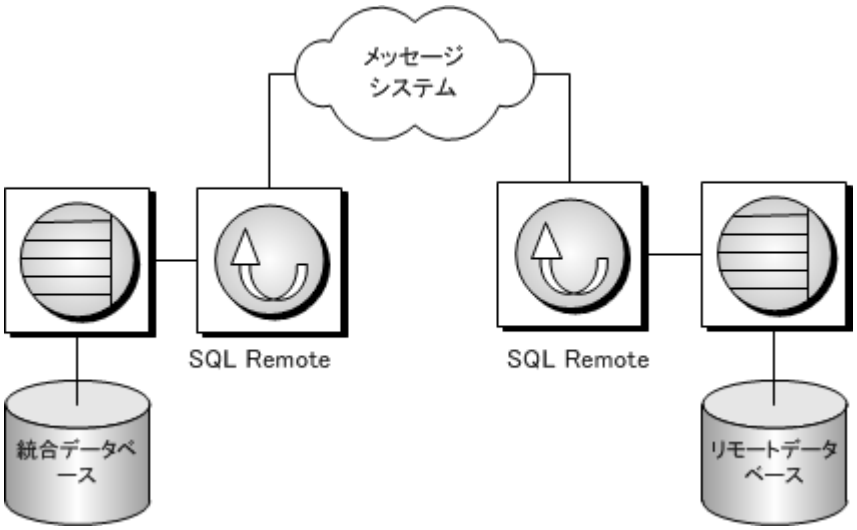
1.1.3 SQL Remote レプリケーションプロセス

SQL Remote では、メッセージは常に双方向に送信されます。統合データベースは、パブリケーションの更新を含むメッセージをリモートデータベースに送信します。リモートデータベースは、更新されたデータと受信確認メッセージを統合データベースに送信します。



リモートデータベースユーザがデータを修正すると、その変更内容が統合データベースにレプリケートされます。この変更が統合データベースで適用されると、変更は統合データベースのパブリケーションに取り込まれ、(更新元のデータベースを除く)すべてのリモートデータベースに送信される更新内容に追加されます。このように、リモートデータベース間でのレプリケーションは、統合データベースを経由して行われます。

たとえば、統合データベースのパブリケーション内でデータが更新されると、更新内容がリモートデータベースに送信されます。リモートデータベースでデータが更新されていない場合でも、レプリケーションのステータスを把握するために、確認メッセージが統合データベースに送信されます。



SQL Remote レプリケーションプロセスに関連する手順

1. レプリケーションに関係する統合データベースとリモートデータベースごとに、Message Agent とレプリケーションを管理するトランザクションログが存在します。コミットされたすべての変更は、トランザクションログに記録されて保存されます。
2. 統合データベースの SQL Remote Message Agent では、トランザクションログを定期的にスキャンして、各パブリケーション (データのセクション) に対して行われたすべてのコミット済みトランザクションをメッセージにパッケージします。次に、統合データベースの SQL Remote Message Agent は、そのパブリケーションに対してサブスクライブされているリモートユーザに、関連する変更を送信します。SQL Remote Message Agent が変更内容を送信するときは、メッセージングシステムを使用します。SQL Remote では、SMTP 電子メールシステム、FTP、HTTP/S、FILE をサポートしています。
3. リモートデータベースの SQL Remote Message Agent は、統合データベースから送信されたメッセージを受信し、メッセージの送信元である統合データベースに確認メッセージを送信します。次に、SQL Remote Message Agent はトランザクションをリモートデータベースに適用します。
4. リモートユーザは、いつでも SQL Remote Message Agent を実行して、リモートデータベースで行われたトランザクションをメッセージにパッケージし、統合データベースにそのメッセージを送信できます。
5. 統合サイトの SQL Remote Message Agent は、リモートデータベースからのメッセージを処理し、そのトランザクションを統合データベースに適用します。

関連情報

SQL Remote システムの作成 [11 ページ]
SQL Remote システムの管理 [81 ページ]

1.2 SQL Remote システムの作成

統合データベースを使用して、すべての SQL Remote 管理タスクを実行します。

SQL Remote を使用してデータベースに接続するには、SYS_RUN_REPLICATION_ROLE システムロールが必要です。

次の概要では、SQL Remote システムの作成方法について説明します。

1. SQL Anywhere 統合データベースを選択するか、または新しい SQL Anywhere データベースを作成します。統合データベースからリモートデータベース (これも SQL Anywhere データベースです) が作成されます。新しい SQL Anywhere データベースを作成する場合、SQL Remote がどのようにプライマリキーを使用するのかを念頭に置く必要があります (リモートデータベースが統合データベースをレプリケートする場合、プライマリキーが重複する可能性があります)。プライマリキーカラムのデータ型に、グローバルオートインクリメントを使用した BIGINT を選択すると実用的です。
   レプリケートするデータを決定します。
2. 効率的なレプリケーションシステムを作成するには、使用するテーブル、そのテーブルのカラム、レプリケートするローのサブセットを決定する必要があります。必要な情報のみを含めるようにしてください。
3. 統合データベースにパブリケーションを作成します。SQL Remote では、パブリッシュ/サブスクライブモデルを採用しており、適切な情報が目的のユーザに必ず届けられます。統合データベースのパブリケーションにレプリケートするデータを整理してください。
4. 統合データベースにパブリッシャユーザを作成します (パブリッシャは PUBLISH 権限を持つユーザです)。

5. 統合データベースにリモートユーザを作成します (リモートユーザは、リモートデータベースをユニークに識別するために使用します)。リモートユーザを作成する場合は、データを転送するときに使用するメッセージタイプを定義し、必要に応じてデータを送信する頻度を定義します。
6. サブスクリプションを作成し、パブリケーションに対してリモートユーザをサブスクライブします。
7. リモートユーザがデータをどのように使用するかを決定します。リモートユーザは、自分のデータを常に読み込むことができます。また、リモートユーザにはデータの更新、削除、挿入も許可できます。
8. 競合を解決する方法を選択します。リモートユーザがデータを更新、削除、挿入すると、レプリケーション時に競合が発生する可能性があります。競合を解決する方法を実装する必要があります。
9. SQL Remote システムを配備し、リモートデータベースを作成して適切なソフトウェアをインストールします。

このセクションの内容:

データ分割の切断 [65 ページ]
リモートデータベースがデータを共有していない場合、データ分割は切断となります。

重複分割 [71 ページ]
リモートデータベースがデータを共有している場合、データ分割は重複となります。たとえば、営業担当者は顧客を共有できます。

リモートデータベースのユニークな ID 番号 [77 ページ]
SQL Remote では、各リモートデータベースに異なる ID 番号を割り当てる必要があります。

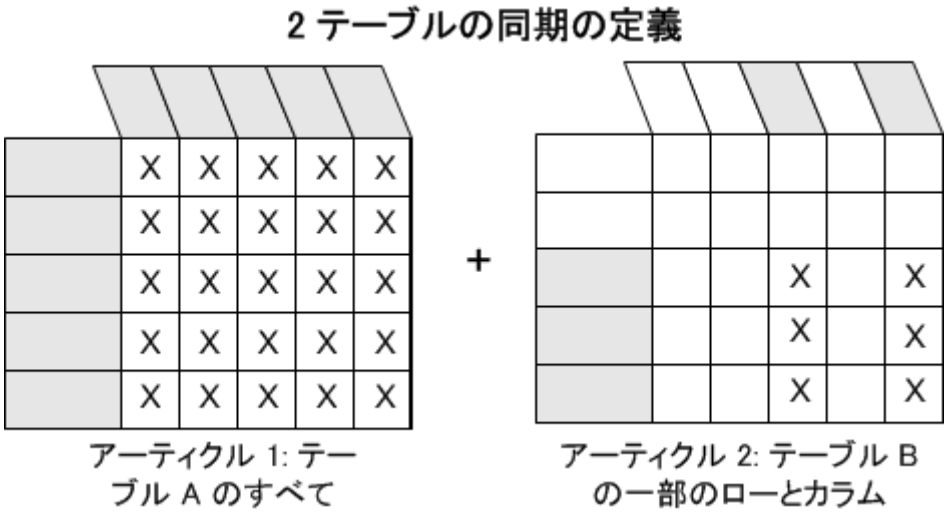
関連情報

更新の競合に対するデフォルトの解決 [47 ページ]
SQL Remote システムの管理 [81 ページ]

1.2.1 パブリケーションとアーティクル

パブリケーションでは、レプリケートされるデータセットを定義します。1 つのパブリケーションは、複数のデータベーステーブルから取得したデータを含むことができます。

アーティクルは、パブリケーションに含まれるテーブルを表します。パブリケーション内の各アーティクルは、テーブル全体またはテーブル内のローとカラムのサブセットで構成されます。



制限事項

パブリケーションには、ビューまたはストアドプロシージャを含めることができません。

パブリケーションとアーティクルの表示 (SQL Central)

SQL Central では、パブリケーションは左ウィンドウ枠のパブリケーションフォルダに表示されます。パブリケーションに作成するアーティクルはすべて、パブリケーションを選択すると、右ウィンドウ枠のアーティクルタブに表示されます。

このセクションの内容:

関連情報

プロシージャのレプリケーション [41 ページ]
トリガのレプリケーション [41 ページ]

1.2.1.1 パブリケーションの作成 (SQL Central)

統合データベースの既存のテーブルに基づいて、テーブル内のすべてのカラムとローで構成されるパブリケーションを作成します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用して統合データベースに接続します。
2. 左ウィンドウ枠で、パブリケーションフォルダをクリックします。
3. ファイル > 新規 > パブリケーション をクリックします。
4. 新しいパブリケーションの名前を指定してくださいフィールドに、パブリケーションの名前を入力します。次へをクリックします。
5. パブリケーションタイプの選択ウィンドウで、適切なパブリケーションタイプを選択します。次へをクリックします。
6. テーブルの指定リストで、1 つ以上のテーブルをクリック (複数の場合は Ctrl キーを押しながらクリック) します。追加をクリックします。
7. 完了をクリックします。

結果

指定したとおりにパブリケーションが作成されます。

1.2.1.2 テーブル内の一部のカラムだけをパブリッシュする (SQL Central)

テーブルのすべてのローと一部のカラムだけが含まれたパブリケーションを作成します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用して統合データベースに接続します。
2. 左ウィンドウ枠で、パブリケーションフォルダを展開します。
3. ファイル > 新規 > パブリケーション をクリックします。
4. 新しいパブリケーションの名前を指定してくださいフィールドに、パブリケーションの名前を入力します。次へをクリックします。
5. パブリケーションタイプの選択ウィンドウで、適切なパブリケーションタイプを選択します。次へをクリックします。
6. テーブルの指定リストで、1 つ以上のテーブルをクリック (複数の場合は Ctrl キーを押しながらクリック) します。追加をクリックします。

7. 使用可能なカラムタブで、テーブルのアイコンをクリックし、使用可能なカラムのリストを展開します。パブリッシュする各カラムをクリックし、追加をクリックします。次へをクリックします。
8. 必要に応じて、アップロードプロシージャの作成ウィンドウで、作成するストアドプロシージャを選択します。
9. 完了をクリックします。

結果

指定したとおりにパブリケーションが作成されます。

次のステップ

サブスクリプションを作成します。

関連情報

サブスクリプション [35 ページ]
参照整合性エラー [55 ページ]

1.2.1.3 テーブル内の一部のローだけをパブリッシュする (SQL Central)

テーブルの一部のローのみが含まれるパブリケーションを作成するには、パブリッシュするローのみに一致する検索条件を記述する必要があります。

検索条件で次のいずれかの句を使用します。

SUBSCRIBE BY 句

SUBSCRIBE BY 句は、パブリケーションに対する複数のサブスクライバが、テーブルから異なるローを受信する場合に使用します。

SQL Remote システムで多数のサブスクリプションが必要な場合に、SUBSCRIBE BY 句をおすすめします。SUBSCRIBE BY 句を使用すると、複数のサブスクリプションを単一のパブリケーションに関連付けできます。WHERE 句では、この関連付けができません。サブスクライバが受信するローは、指定された式の値によって異なります。

SUBSCRIBE BY 句を使用すると、より簡潔で理解しやすいパブリケーションを作成できます。また、WHERE 句を使用した複数のパブリケーションを管理するよりも、優れたパフォーマンスが得られます。

WHERE 句

WHERE 句は、アーティクルにローのサブセットを追加する場合に使用します。このアーティクルを含むパブリケーションのすべてのサブスクライバは、WHERE 句を満たすローを受信します。

パブリッシュ対象外のすべてのローにはデフォルト値を設定しておきます。デフォルト値が設定されていない場合は、リモートデータベースが統合データベースから新しいローを挿入しようとすると、エラーが発生します。

アーティクルでは WHERE 句を結合できます。

データベースサーバは、パブリケーションの数に正比例して、トランザクションログに情報を追加し、そのログをスキャンしてメッセージを送信する必要があります。WHERE 句を使用しても、複数のサブスクリプションを単一のパブリケーションに関連付けることはできません。ただし、SUBSCRIBE BY 句では、この関連付けができます。

例

各営業担当者が次の操作を実行できるパブリケーションが必要です。

- 受注に対してサブスクライブします。
- 受注をローカルで更新します。
- 統合データベースに売り上げをレプリケートします。

WHERE 句を使用すると、営業担当者ごとに個別のパブリケーションを作成する必要があります。次のパブリケーションは、Sam Singer という名前の営業担当者用です。他のそれぞれの営業担当者にも、同様のパブリケーションが必要になります。

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

次の文では、PubsOrdersSamSinger パブリケーションに対して Sam Singer をサブスクライブします。

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

SUBSCRIBE BY 句を使用する場合、必要なパブリケーションは 1 つのみです。すべての営業担当者が、次のパブリケーションを使用できます。

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

次の文では、Sam Singer の ID 8887 で、PubsOrders パブリケーションに対して Sam Singer をサブスクライブします。

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

このセクションの内容:

SUBSCRIBE BY 句を使用して一部のローのみをパブリッシュする (SQL Central) [18 ページ]
SUBSCRIBE BY 句を使用してパブリケーションを作成します。

WHERE 句を使用して一部のローだけをパブリッシュする (SQL Central) [19 ページ]
WHERE 句を使用するパブリケーションを作成し、テーブルのすべてのカラムと一部のローのみを追加します。

1.2.1.3.1 SUBSCRIBE BY 句を使用して一部のローのみをパブリッシュする (SQL Central)

SUBSCRIBE BY 句を使用してパブリケーションを作成します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用して統合データベースに接続します。
2. 左ウィンドウ枠で、パブリケーションをダブルクリックします。
3. ファイル > 新規 > パブリケーション をクリックします。
4. 新しいパブリケーションの名前を指定してくださいフィールドに、パブリケーションの名前を入力します。次へをクリックします。
5. 次へをクリックします。
6. 使用可能なテーブルリストでテーブルをクリックします。追加をクリックします。次へをクリックします。
7. 使用可能なカラムタブで、テーブルのアイコンをクリックし、使用可能なカラムのリストを展開します。パブリッシュする各カラムをクリックし、追加をクリックします (一度に複数のカラムを選択する場合は、Ctrl キーを押しながらクリックします)。次へをクリックします。
8. 次へをクリックします。
9. *SUBSCRIBE BY* 制限の指定ページで、次の手順を実行します。
   a. アーティクルリストでテーブルをクリックします。
   b. カラムをクリックし、ドロップダウンリストからカラムをクリックします。
10. 完了をクリックします。

結果

指定したとおりにパブリケーションが作成されます。

次のステップ

そのパブリケーションに対するユーザのサブスクリプションを作成します。

関連情報

SQL Remote サブスクリプションの作成 (SQL Central) [36 ページ]
WHERE 句を使用して一部のローだけをパブリッシュする (SQL Central) [19 ページ]
データ分割の切断 [65 ページ]

1.2.1.3.2 WHERE 句を使用して一部のローだけをパブリッシュする (SQL Central)

WHERE 句を使用するパブリケーションを作成し、テーブルのすべてのカラムと一部のローのみを追加します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

ユーザは、複数のパブリケーションに対してサブスクライブできます。また、単一のパブリケーションに対して複数のサブスクリプションを作成することもできます。

手順

1. SQL Central を使用して統合データベースに接続します。
2. パブリケーションフォルダをダブルクリックします。
3. ファイル > 新規 > パブリケーション をクリックします。
4. 新しいパブリケーションの名前を指定してくださいフィールドに、パブリケーションの名前を入力します。次へをクリックします。
5. 次へをクリックします。
6. 使用可能なテーブルリストでテーブルをクリックします。追加をクリックします。次へをクリックします。
7. 使用可能なカラムタブで、テーブルのアイコンをダブルクリックし、使用可能なカラムのリストを展開します。パブリッシュする各カラムをクリックし、追加をクリックします。次へをクリックします。
8. *WHERE* 句の指定ページで、次の手順に従います。
   a. アーティクルリストでテーブルをクリックします。
   b. 選択したアーティクルには次の *WHERE* 句がありますフィールドに WHERE 句を入力します。
9. 完了をクリックします。

結果

指定したとおりにパブリケーションが作成されます。

次のステップ

サブスクリプションを追加します。

関連情報

サブスクリプション [35 ページ]
ローが見つからないエラー [54 ページ]

1.2.1.4 パブリケーションの変更 (SQL Central)

パブリケーションを変更するには、アーティクルを追加、修正、または削除するか、パブリケーションの名前を変更します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

> ⚠ 警告
>
> 動作中の SQL Remote システムでパブリケーションを変更すると、レプリケーションエラーが発生し、レプリケーションシステムのデータが失われるおそれがあります。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. パブリケーションをダブルクリックします。

3. 変更するパブリケーションを右クリックし、プロパティをクリックしてパブリケーションを編集します。

結果

パブリケーションが変更されます。

関連情報

アップグレードと再同期 [151 ページ]

1.2.1.5 パブリケーションの削除 (SQL Central)

パブリケーションを削除します (そのパブリケーションに対するサブスクリプションがすべて自動的に削除されます)。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

> ⚠ 警告
>
> 動作中の SQL Remote システムでパブリケーションを削除すると、レプリケーションエラーが発生し、レプリケーションシステムのデータが失われるおそれがあります。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. パブリケーションをダブルクリックします。
3. 目的のパブリケーションを右クリックして、削除をクリックします。

結果

パブリケーションが削除され、そのパブリケーションに対するすべてのサブスクリプションが削除されます。

関連情報

アップグレードと再同期 [151 ページ]

1.2.2 ユーザ権限

SQL Remote では、リモートデータベースと統合データベースの権限を持つユーザを管理するための、一貫性のあるシステムを採用しています。

SQL Remote のレプリケーションに関係するデータベースのユーザが、次の権限を付与したり取り消すには、MANAGE REPLICATION システム権限 (SYS_REPLICATION_ADMIN_ROLE システムロールの一部) が必要です。

PUBLISH

SQL Remote システム内のすべてのデータベースが情報をパブリッシュします。したがって、どのデータベースにもパブリッシャが必要です。パブリッシャを作成するには、1 人のユーザに PUBLISH 権限を付与します。パブリッシャユーザは、SQL Remote システム全体でユニークとする必要があります。データを送信するとき、パブリッシャはそのデータベースを表しています。たとえば、データベースがメッセージを送信するとき、メッセージにはデータベースのパブリッシャユーザ名が含まれています。データベースがメッセージを受信する場合は、メッセージに含まれるパブリッシャ名によって、メッセージを送信したデータベースを識別できます。

REMOTE

統合データベースなど、他のデータベースにメッセージを送信するデータベースでは、メッセージの送信先となるリモートデータベースを指定する必要があります。統合データベースでこれらのリモートデータベースを指定するには、リモートデータベースのパブリッシャに REMOTE 権限を付与します。REMOTE 権限によって、現在のデータベースからメッセージを受信するデータベースが識別されます。

CONSOLIDATE

各リモートデータベースでは、メッセージを受信する統合データベースを指定する必要があります。リモートデータベースで統合データベースを指定するには、統合データベースのパブリッシャに CONSOLIDATE 権限を付与します。リモートデータベースは、1 つの統合データベースからのメッセージのみを受信できます。CONSOLIDATE 権限によって、リモートデータベースにメッセージを送信するデータベースが識別されます。

抽出ユーティリティ (dbxtract) による権限の自動設定

抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードでは、デフォルトで、適切な PUBLISH 権限と CONSOLIDATE 権限が、リモートデータベース内のユーザに付与されます。

このセクションの内容:

単層階層 [23 ページ]
単層階層では、1 つの統合データベースの下に 1 つ以上のリモートデータベースが存在します。このような階層では、統合データベースによって、リモートデータベースのパブリッシャに REMOTE 権限が付与され、

多層階層 [24 ページ]
多層階層では、現在のデータベースの直下にあるすべてのリモートデータベースに REMOTE 権限が付与されます。階層内の現在のデータベースの直上にあるデータベースには CONSOLIDATE 権限が付与されます。

PUBLISH 権限 [25 ページ]
SQL Remote システムのすべてのデータベースにパブリッシャが必要です。パブリッシャは、PUBLISH 権限を持つユニークなユーザです。パブリッシャを作成するには、ユーザに PUBLISH 権限を付与します。

REMOTE 権限 [28 ページ]
REMOTE 権限を付与することは、データベースにリモートユーザを追加することに相当します。SQL Remote 階層内で現在のデータベースの直下にあるデータベースのパブリッシャには、現在のデータベースによって REMOTE 権限が付与されます。

CONSOLIDATE 権限 [31 ページ]
SQL Remote 階層内で現在のデータベースの直上にあるデータベースには、現在のデータベースによって CONSOLIDATE 権限が付与されます。各リモートデータベースでは、統合データベースに CONSOLIDATE 権限を付与する必要があります。

SQL Remote のレプリケーションに関連するシステムロール [34 ページ]
SQL Remote でのアクセスと機能を制御する、レプリケーションに関連する SQL Anywhere のシステムロールが 2 つあります。SYS_RUN_REPLICATION_ROLE と SYS_REPLICATION_ADMIN_ROLE です。

関連情報

抽出ユーティリティ (dbxtract) [219 ページ]

1.2.2.1 単層階層

単層階層では、1 つの統合データベースの下に 1 つ以上のリモートデータベースが存在します。このような階層では、統合データベースによって、リモートデータベースのパブリッシャに REMOTE 権限が付与され、

各リモートデータベースによって、統合データベースのパブリッシャに CONSOLIDATE 権限が付与されます。

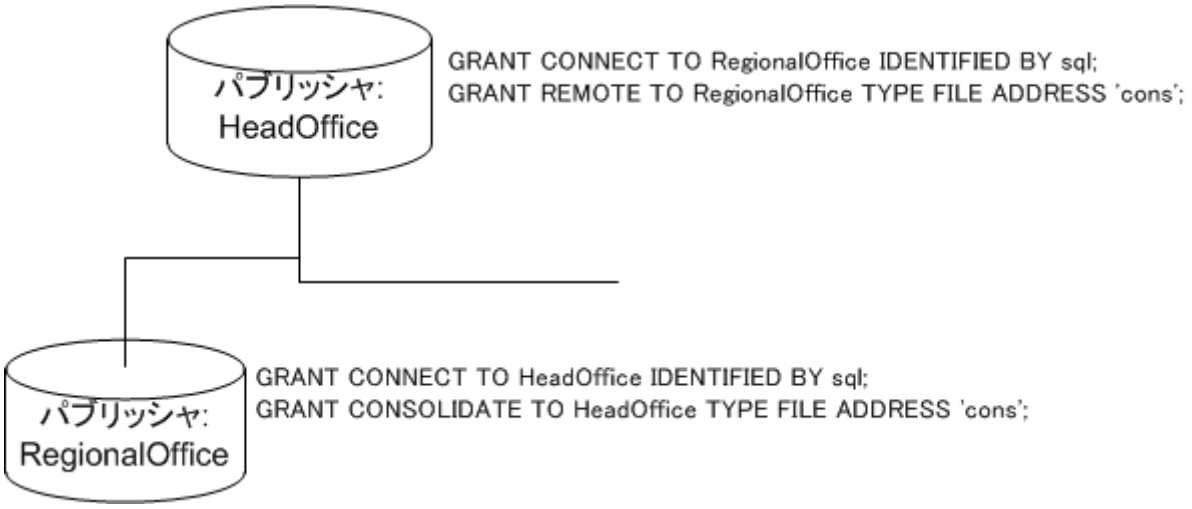
たとえば、統合データベースのパブリッシャによって識別される統合データベース HeadOffice と、リモートデータベースのパブリッシャによって識別されるリモートデータベース RegionalOffice があるとします。

統合データベース HeadOffice で、次の処理を実行します。

- リモートデータベースのパブリッシャと同じ名前のユーザを作成します。RegionalOffice.
- RegionalOffice に REMOTE 権限を付与します。このことによって、RegionalOffice が HeadOffice からメッセージを受信するデータベースとして識別されます。

リモートデータベース RegionalOffice で、次の処理を実行します。

- 統合データベースのパブリッシャと同じ名前のユーザを作成します。HeadOffice.
- HeadOffice に CONSOLIDATE 権限を付与します。このことによって、HeadOffice が RegionalOffice の統合データベースとして識別されます。つまり、HeadOffice が、RegionalOffice にメッセージを送信するデータベースとなります。



Dbxtract による権限の自動設定

抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードでは、デフォルトで、適切な PUBLISH 権限と CONSOLIDATE 権限が、リモートデータベース内のユーザに付与されます。

関連情報

抽出ユーティリティ (dbxtract) [219 ページ]

1.2.2.2 多層階層

多層階層では、現在のデータベースの直下にあるすべてのリモートデータベースに REMOTE 権限が付与されます。階層内の現在のデータベースの直上にあるデータベースには CONSOLIDATE 権限が付与されます。

たとえば、統合データベースのパブリッシャ HeadOffice によって識別される統合データベースがあり、このデータベースにはリモートデータベース RegionalOffice が存在するとします。ただし、RegionalOffice データベースにもリモートデータベース Office が存在します。

統合データベース HeadOffice で、次の処理を実行します。

- リモートデータベース RegionalOffice のパブリッシャと同じ名前のユーザを作成します。

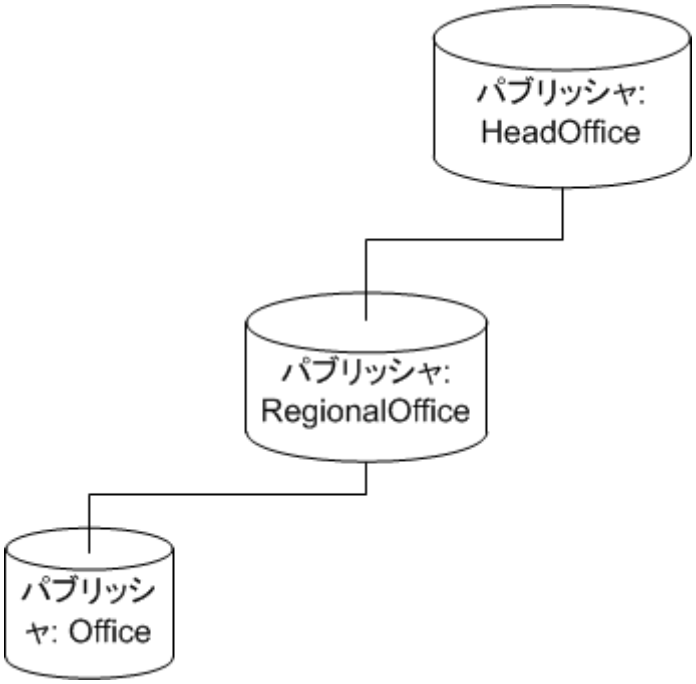
- ユーザ RegionalOffice に REMOTE 権限を付与します。このことによって、RegionalOffice が HeadOffice からメッセージを受信するデータベースとして識別されます。

RegionalOffice データベースで、次の処理を実行します。

- 統合データベース HeadOffice のパブリッシャと同じ名前のユーザを作成します。
- HeadOffice に CONSOLIDATE 権限を付与します。このことによって、HeadOffice が RegionalOffice の統合データベースとして識別されます。つまり、HeadOffice が、RegionalOffice にメッセージを送信するデータベースとなります。
- RegionalOffice の直下にあるデータベース Office と同じ名前のユーザを作成します。
- Office に REMOTE 権限を付与します。このことによって、Office が RegionalOffice からメッセージを受信するデータベースとして識別されます。

Office データベースで、次の処理を実行します。

- 統合データベース RegionalOffice のパブリッシャと同じ名前のユーザを作成します。
- RegionalOffice ユーザに CONSOLIDATE 権限を付与します。このことによって、RegionalOffice が Office の統合データベースとして識別されます。つまり、RegionalOffice が Office にメッセージを送信します。



1.2.2.3 PUBLISH 権限

SQL Remote システムのすべてのデータベースにパブリッシャが必要です。パブリッシャは、PUBLISH 権限を持つユニークなユーザです。パブリッシャを作成するには、ユーザに PUBLISH 権限を付与します。

パブリケーションの更新と受信確認を含む、SQL Remote のすべての出力メッセージは、パブリッシャによって識別されます。SQL Remote システムのすべてのデータベースは、受信確認を送信します。

PUBLISH 権限には、出力メッセージのパブリッシャを識別する以外の権限はありません。

ユーザ拡張ロールに PUBLISH 権限を付与しても、そのロールのメンバーには PUBLISH 権限は継承されません。

このセクションの内容:

1.2.2.3.1 パブリッシャの作成 (SQL Central)

ユーザを作成し、それぞれに PUBLISH 権限を付与します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

抽出ユーティリティ (dbxtract) またはデータベース抽出ウィザードでリモートデータベースを抽出すると、リモートユーザがリモートデータベースのパブリッシャになり、PUBLISH 権限が付与されます。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. データベース名をダブルクリックします。
3. ユーザをダブルクリックします。
4. ユーザ名を右クリックします。
5. パブリッシャに変更をクリックします。

結果

ユーザが作成され、それぞれに PUBLISH 権限が付与されます。

関連情報

PUBLISH 権限の取り消し (SQL Central) [27 ページ]
パブリッシャの表示 (SQL Central) [28 ページ]

1.2.2.3.2 PUBLISH 権限の取り消し (SQL Central)

ユーザの PUBLISH 権限を取り消します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

> **警告**
>
> リモートデータベースまたは統合データベースでパブリッシャを変更すると、情報が失われるなど、そのデータベースを含むサブスクリプションのいずれかに深刻な問題が発生するおそれがあります。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. データベース名をダブルクリックします。
3. ユーザをダブルクリックします。
4. ユーザ名を右クリックします。
5. パブリッシャの取り消しをクリックします。

結果

ユーザは PUBLISH 権限を持たなくなります。

関連情報

パブリッシャの表示 (SQL Central) [28 ページ]
実行中のシステムに行ってはならない変更 [151 ページ]

1.2.2.3.3 パブリッシャの表示 (SQL Central)

パブリッシャのユーザを識別します。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. ユーザをダブルクリックします。

   パブリッシャとは、タイプがパブリッシャとなっているユーザのことです。

結果

パブリッシャのユーザは表示できます。

関連情報

PUBLISH 権限の取り消し (SQL Central) [27 ページ]

1.2.2.4 REMOTE 権限

REMOTE 権限を付与することは、データベースにリモートユーザを追加することに相当します。SQL Remote 階層内で現在のデータベースの直下にあるデータベースのパブリッシャには、現在のデータベースによって REMOTE 権限が付与されます。

ユーザに REMOTE 権限を付与する場合は、次の設定を行う必要があります。

メッセージシステム

データベース内で最低 1 つのメッセージシステムが定義されるまで、新規リモートユーザを作成することはできません。

送信頻度

SQL 文を使用して REMOTE 権限を付与する場合、送信頻度の設定はオプションです。

ユーザに REMOTE 権限を付与するには、次の処理を実行します。

- ユーザをリモートユーザとして識別します。
- このリモートユーザでメッセージを交換するときに使用するメッセージタイプを指定します。
- メッセージの送信先アドレスを指定します。
- メッセージがリモートユーザに送信される頻度を指示します。

データベースのパブリッシャは、同じデータベースの REMOTE 権限と CONSOLIDATE 権限を持つことはできません。これにより、パブリッシャが、出力メッセージの送信者と受信者の両方として識別されます。

グループへの **REMOTE** 権限の付与

ユーザ拡張ロールに REMOTE 権限を付与することはできますが、そのロールの被付与者には REMOTE 権限は継承されません。ユーザ拡張ロールのそれぞれの被付与者に対して、リモート権限を明示的に付与する必要があります。

このセクションの内容:

REMOTE 権限の付与 (SQL Central) [29 ページ]
リモートユーザを追加するか、既存のユーザをリモートユーザに変更します。
REMOTE 権限の取り消し (SQL Central) [30 ページ]
SQL Remote システムからユーザまたはロールを削除して、そのユーザまたはロールを通常のユーザ/ロールに戻し、そのユーザまたはロールのサブスクリプションをすべてのパブリケーションから削除します。

関連情報

SQL Remote メッセージシステム [117 ページ]
送信頻度 [95 ページ]

1.2.2.4.1 REMOTE 権限の付与 (SQL Central)

リモートユーザを追加するか、既存のユーザをリモートユーザに変更します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. データベース名をダブルクリックします。
3. **ユーザ**をダブルクリックします。
4. ユーザ名を右クリックします。
5. 次のうちの 1 つを選択してください。

オプション	アクション
新しいリモートユーザの追加	1. **ファイル** > **新規** > **ユーザ** をクリックします。 2. **リモートユーザ作成ウィザード**の指示に従います。 3. ユーザを右クリックして、**リモートユーザに変更**をクリックします。
既存のユーザをリモートユーザにする	1. ユーザを右クリックして、**リモートユーザに変更**をクリックします。 2. このウィンドウで、メッセージタイプのクリック、アドレスの入力、送信頻度のクリックを行い、*OK* をクリックします。

結果

リモートユーザが作成されます。

関連情報

REMOTE 権限の取り消し (SQL Central) [30 ページ]

1.2.2.4.2 REMOTE 権限の取り消し (SQL Central)

SQL Remote システムからユーザまたはロールを削除して、そのユーザまたはロールを通常のユーザ/ロールに戻し、そのユーザまたはロールのサブスクリプションをすべてのパブリケーションから削除します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

ユーザまたはグループの REMOTE 権限を取り消すには、次の処理を実行します。

- SQL Remote システムからユーザを削除します。
- ユーザまたはグループを通常のユーザまたはグループに戻します。
- すべてのパブリケーションから、ユーザまたはグループのサブスクリプションを削除します。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. データベース名をダブルクリックします。
3. **ユーザ**をダブルクリックします。
4. ユーザ名を右クリックします。
5. **リモートの取り消し**をクリックします。

結果

そのユーザ/グループは、通常のユーザ/グループに戻り、SQL Remote システムから削除され、すべてのパブリケーションからサブスクリプションが削除されます。

関連情報

REMOTE 権限の付与 (SQL Central) [29 ページ]

1.2.2.5 CONSOLIDATE 権限

SQL Remote 階層内で現在のデータベースの直上にあるデータベースには、現在のデータベースによって CONSOLIDATE 権限が付与されます。各リモートデータベースでは、統合データベースに CONSOLIDATE 権限を付与する必要があります。

CONSOLIDATE 権限は、読み込み専用リモートデータベースから統合データベースにも付与する必要があります。これは、リモートデータベースから統合データベースに受信確認が送信されるためです。

ユーザに CONSOLIDATE 権限を付与する場合は、次の設定を行う必要があります。

メッセージシステム

データベース内で最低 1 つのメッセージシステムが定義されるまで、新規統合ユーザを作成することはできません。

送信頻度

SQL 文を使用して CONSOLIDATE 権限を付与する場合、送信頻度の設定はオプションです。

CONSOLIDATE 権限を付与するには、次の処理を実行します。

- ユーザを統合ユーザとして識別します。
- この統合ユーザでメッセージを交換するときに使用するメッセージタイプを指定します。
- メッセージの送信先アドレスを指定します。
- メッセージが統合ユーザに送信される頻度を指示します。

データベースのパブリッシャは、同じデータベースの REMOTE 権限と CONSOLIDATE 権限を持つことはできません。これにより、パブリッシャが、出力メッセージの送信者と受信者の両方として識別されます。

抽出ユーティリティ (dbxtract)

抽出ユーティリティ (dbxtract) またはデータベース抽出ウィザードでリモートデータベースを抽出すると、リモートデータベースで GRANT CONSOLIDATE 文が自動的に実行されます。

このセクションの内容:

関連情報

SQL Remote メッセージシステム [117 ページ]
送信頻度 [95 ページ]

1.2.2.5.1 CONSOLIDATE 権限の付与 (SQL Central)

ユーザに CONSOLIDATE 権限を付与します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

統合データベースのパブリッシャに CONSOLIDATE 権限を付与します。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. データベース名をダブルクリックします。
3. **ユーザ**をダブルクリックします。
4. ユーザ名を右クリックします。
5. **統合ユーザに変更**をクリックします。
6. **メッセージタイプ**、**アドレス**、**送信頻度**の各設定を行います。
7. *OK* をクリックして、**統合ユーザに変更**ウィンドウを閉じます。

結果

ユーザに CONSOLIDATE 権限が付与されます。

関連情報

CONSOLIDATE 権限の取り消し (SQL Central) [33 ページ]

1.2.2.5.2 CONSOLIDATE 権限の取り消し (SQL Central)

データベースのユーザの CONSOLIDATE 権限を削除します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

ユーザの CONSOLIDATE 権限を取り消す場合、SQL Anywhere では次の処理を実行します。

- SQL Remote システムからユーザを削除します。
- ユーザまたはグループを通常のユーザまたはグループに戻します。
- すべてのパブリケーションから、ユーザまたはグループのサブスクリプションを削除します。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. データベース名をダブルクリックします。
3. **ユーザ**をダブルクリックします。
4. 統合ユーザまたはグループを右クリックして、**統合の取り消し**をクリックします。

結果

そのユーザ/グループは、通常のユーザ/グループに戻り、SQL Remote システムから削除され、すべてのパブリケーションからサブスクリプションが削除されます。

関連情報

CONSOLIDATE 権限の付与 (SQL Central) [32 ページ]

1.2.2.6 SQL Remote のレプリケーションに関連するシステムロール

SQL Remote でのアクセスと機能を制御する、レプリケーションに関連する SQL Anywhere のシステムロールが 2 つあります。SYS_RUN_REPLICATION_ROLE と SYS_REPLICATION_ADMIN_ROLE です。

SYS_RUN_REPLICATION_ROLE システムロール

SYS_RUN_REPLICATION_ROLE システムロールを持っているユーザは、SQL Remote から接続している場合にのみ、データベースに対する完全な管理権限を持ちます。SQL Remote ユーザはデータベースに対するフルアクセス権を持ち、メッセージで指定された変更を行うことができます。

SYS_RUN_REPLICATION_ROLE システムロールを持つユーザだけが SQL Remote を実行できます。
SYS_RUN_REPLICATION_ROLE システムロールは次のプロパティを備えています。

SQL Remote 以外からの接続では特別な権限はなし

SYS_RUN_REPLICATION_ROLE システムロールが付与されたユーザは、SQL Remote または dbmlsync 以外からの接続では、このロールから継承される権限を行使できません。したがって、SYS_RUN_REPLICATION_ROLE システ

ムロールを持つユーザのユーザ名とパスワードが公開されていても、セキュリティ上の問題は発生しません。そのデータベースで CONNECT より上の権限が付与されたユーザ名を使用しないかぎり、データベース内のデータにアクセスすることはできません。

SQL Remote からの完全なデータベース管理権限

SQL Remote から接続している場合、SYS_RUN_REPLICATION_ROLE システムロールを持つユーザは、データベースに対する完全なデータベース管理権限を持ちます。

統合データベースでユーザを作成するときに、統合データベースのパブリッシャと各リモートユーザに SYS_RUN_REPLICATION_ROLE システムロールを付与してください。抽出ユーティリティ (dbxtract) またはデータベース抽出ウィザードによってリモートデータベースを抽出すると、リモートユーザはリモートデータベースのパブリッシャになり、PUBLISH 権限と SYS_RUN_REPLICATION_ROLE システムロールが付与されます。

この推奨内容を実行すると、ユーザの管理が簡単になります。SQL Remote からの接続 (ユーザに SYS_RUN_REPLICATION_ROLE システムロールが付与されます)、または他のクライアントアプリケーションからの接続 (この場合、SYS_RUN_REPLICATION_ROLE システムロールによってユーザに追加の権限は付与されません) にかかわらず、各リモートユーザにはデータベースに接続するために 1 つのユーザ名のみが必要です。

> **注記**
>
> GRANT REMOTE DBA は推奨されていません。代わりに GRANT ROLE SYS_RUN_REPLICATION_ROLE を使用してください。

SYS_REPLICATION_ADMIN_ROLE システムロール

SYS_REPLICATION_ADMIN_ROLE システムロールを持っているユーザは、レプリケーションロールの付与、パブリケーションの管理、サブスクリプション、ユーザとプロファイルの同期、メッセージタイプの管理、レプリケーション関連のオプションの設定など、レプリケーションに関連する管理タスクを実行することができます。

1.2.3 サブスクリプション

パブリケーションに対してユーザをサブスクライブするには、サブスクリプションを作成します。パブリケーションに含まれる情報を共有する各データベースには、そのパブリケーションに対するサブスクリプションが必要です。

データベース内の各パブリケーションに対して変更を加えると、その変更内容はパブリケーションのすべてのサブスクライバに定期的にレプリケートされます。このようなレプリケーションを、パブリケーションの更新と呼びます。

パブリケーションに対してユーザをサブスクライブするには、次の情報が必要です。

パブリケーション名

このパブリケーション名でユーザのサブスクリプションが作成されます。

サブスクリプション値

サブスクリプション値は、パブリケーションに SUBSCRIBE BY 句が含まれている場合にのみ適用されます。サブスクリプション値とは、パブリケーションの SUBSCRIBE BY 句に対してテストを行った値です。たとえば、従業員 ID を含むカラム名をパブリケーションが SUBSCRIBE BY 句として保持する場合は、サブスクリプションを作成するときに、サブスクライブされるユーザの従業員 ID の値を指定する必要があります。サブスクリプション値は常に文字列です。

この値は、パブリケーションに SUBSCRIBE BY 句が含まれている場合にのみ必要です。

サブスクライバ ID

パブリケーションに対してサブスクライブされるユーザ。統合データベースで、リモートユーザにサブスクリプションを作成する場合、そのリモートユーザに REMOTE 権限が付与されている必要があります。リモートデータベースで、統合ユーザにサブスクリプションを作成する場合、そのユーザに CONSOLIDATED 権限が付与されている必要があります。

抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードでは、デフォルトで、適切な PUBLISH 権限と CONSOLIDATE 権限が、リモートデータベース内のユーザに付与されます。

このセクションの内容:

SQL Remote サブスクリプションの作成 (SQL Central) [36 ページ]
パブリケーションに対してユーザをサブスクライブするには、サブスクリプションを作成します。

関連情報

サブスクリプションの再同期 [155 ページ]
SUBSCRIBE BY 句を使用して一部のローのみをパブリッシュする (SQL Central) [18 ページ]
サブスクリプションの開始 (SQL Central) [159 ページ]
サブスクリプションの停止 (SQL Central) [160 ページ]

1.2.3.1 SQL Remote サブスクリプションの作成 (SQL Central)

パブリケーションに対してユーザをサブスクライブするには、サブスクリプションを作成します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールを持っていて、パブリケーションがすでに作成されている必要があります。

コンテキスト

パブリケーションに含まれる情報を共有する各データベースには、そのパブリケーションに対するサブスクリプションが必要です。データベース内の各パブリケーションに対して変更を加えると、その変更内容はパブリケーションのすべてのサブスクライバに定期的にレプリケートされます。このようなレプリケーションを、パブリケーションの更新と呼びます。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. データベース名をダブルクリックします。
3. *SQL Remote* サブスクリプションをダブルクリックします。
4. ファイル > 新規 > *SQL Remote* サブスクリプション をクリックします。
5. *SQL Remote* サブスクリプション作成ウィザードの指示に従います。

   サブスクリプションの詳細は、パブリケーションにサブスクリプション式を使用するかどうかによって異なります。

結果

パブリケーションに対するユーザのサブスクリプションが作成されます。

1.2.4 トランザクションログベースのレプリケーション

SQL Remote は、コミットされた変更とパブリケーションに含まれているデータを修正する変更をレプリケートします。

コミットされた変更

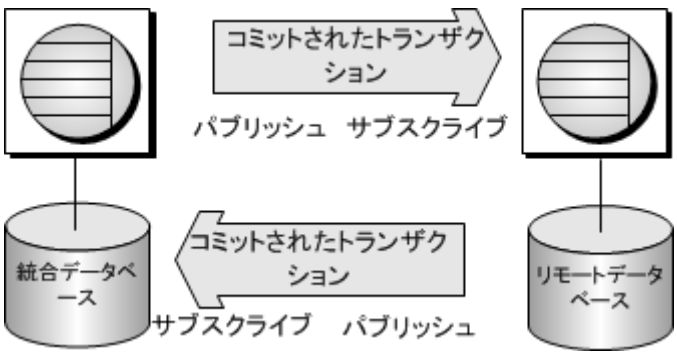
データベースに対して行われ、トランザクションログに記録された変更。

パブリケーションに含まれているデータを修正する変更

SQL Remote は、トランザクションログをスキャンして、パブリケーションに含まれているローに対する変更のうち、コミットされた変更があるかどうかを調べます。さらに、SQL 文をメッセージにパッケージし、そのメッセージをサブスクライブされたデータベースに送信します。

統合データベースでは、パブリケーションに含まれるトランザクションログ内のすべてのコミットされたトランザクションが、定期的にリモートデータベースに送信されます。

リモートデータベースでは、パブリケーションに含まれるトランザクションログ内のすべてのコミットされたトランザクションが、定期的に統合データベースに送信されます。



データベースサーバによるパブリケーションの処理方法

データベースサーバは、パブリケーションを評価してトランザクションログに情報を書き込むコンポーネントです。パブリケーションの数が多くなると、データベースサーバが実行する必要のある処理も多くなります。

データベースサーバは、パブリケーションの一部であるテーブルを更新する個々のサブスクリプション式を評価します。更新前と更新後に、式の値をトランザクションログに追加します。複数のパブリケーションの一部であるテーブルでは、個々のパブリケーションに対して更新前と更新後に、サブスクリプション式が評価されます。

トランザクションログに情報を追加すると、次の場合にパフォーマンスが低下することがあります。

高負荷の式

サブスクリプション式の評価が高い負荷を生む場合は、パフォーマンスが低下することがあります。

多数のパブリケーション

テーブルが複数のパブリケーションに属している場合は、多数の式を評価する必要があります。これに対し、サブスクリプションの数はデータベースサーバのパフォーマンスに影響しません。

複数の値が含まれる式

一部の式には複数の値が含まれていて、トランザクションログの情報が増加する場合があります。このことがパフォーマンスに影響する可能性があります。

SQL Remote で処理されるサブスクリプション

SQL Remote は、文のレプリケーションを実行するコンポーネントです。

送信フェーズの間、SQL Remote Message Agent は現在のサブスクリプションをトランザクションログ内のパブリケーション情報にマップして、リモートユーザごとに適切なメッセージを生成します。

このセクションの内容:

関連情報

SQL Remote パフォーマンス [100 ページ]

1.2.4.1 INSERT および DELETE 文のレプリケーション

統合データベースとリモートデータベースでは、INSERT 文と DELETE 文を扱う方法が異なります。

- あるデータベースで INSERT 文を実行すると、この文は INSERT 文として SQL Remote システム内でサブスクライブされたデータベースに送信されます。
- あるデータベースで DELETE 文を実行すると、この文は DELETE 文として SQL Remote システム内でサブスクライブされたデータベースに送信されます。

統合データベース

SQL Remote では、統合データベースのトランザクションログからの INSERT 文または DELETE 文をそれぞれコピーして、挿入または削除されるローに対してサブスクライブする各リモートデータベースにこれらの文を送信します。テーブルのカラムのサブセットに対してのみサブスクライブされている場合、リモートデータベースに送信される INSERT 文にはそのカラムのみが含まれます。

リモートデータベース

SQL Remote では、リモートデータベースのトランザクションログからの INSERT 文または DELETE 文をそれぞれコピーして、挿入または削除されるローに対してサブスクライブする統合データベースに、この文を送信します。次に、統合データベースによって文が適用され、その結果、トランザクションログへの書き込みが行われます。統合データベースのトランザクションログが SQL Remote によって処理されると、変更内容は最終的に他のリモートサイトに送信されます。SQL Remote は、最初に文を実行したリモートユーザにはその文を送信しません。

関連情報

重複プライマリキーエラー [57 ページ]
ローが見つからないエラー [54 ページ]

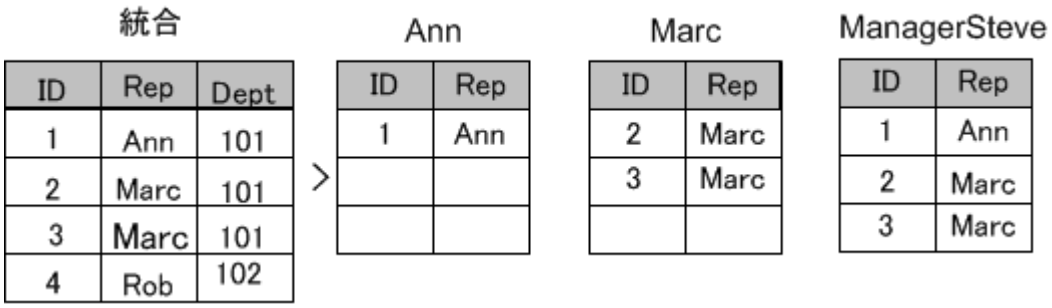
1.2.4.2 UPDATE 文のレプリケーション

UPDATE 文はデータベース内で影響を受けるローへの影響により、レプリケートの方法が異なります。

次の例で、UPDATE 文がレプリケートされる仕組みについて説明します。

- UPDATE 文は、いずれかのリモートユーザのサブスクリプションでローを更新する場合、UPDATE 文としてそのユーザに送信されます。
- UPDATE 文は、いずれかのリモートユーザのサブスクリプションからローを削除する場合、DELETE 文としてそのユーザに送信されます。
- UPDATE 文は、いずれかのリモートユーザのサブスクリプションにローを追加する場合、INSERT 文としてそのユーザに送信されます。

UPDATE 文がレプリケートされる仕組みを説明するため、次の例では、統合データベースとユーザ Ann、Marc、ManagerSteve の 3 つのリモートデータベースを使用します。



統合データベースには、次の文を使用して作成された、cons という名前のパブリケーションが存在します。

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

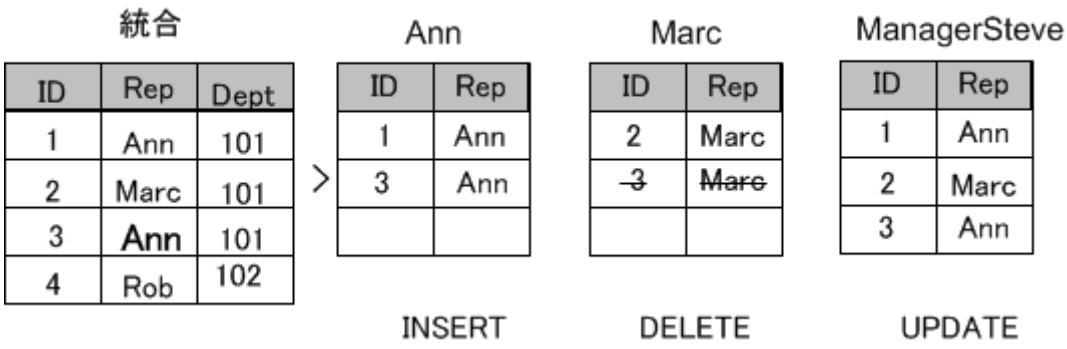
Ann と Marc は、cons パブリケーションに対して、それぞれの Rep カラム値でサブスクライブします。ManagerSteve は、cons パブリケーションに対して、Ann と Marc の Rep カラム値を使用してサブスクライブします。次の文は、パブリケーション cons に対して 3 人のユーザをサブスクライブします。

```
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";
```

統合データベースでは、Marc から Ann にローの Rep 値を変更する UPDATE 文が、次のようにレプリケートされます。

- Marc に対しては DELETE 文として。
- Ann に対しては INSERT 文として。

- ManagerSteve に対しては UPDATE 文として。



サブスクライバ間のローの再割り当ては、営業担当者間に顧客を定期的に再割り当てする営業支援アプリケーションに共通の機能で、領域の再編成とも呼ばれます。

関連情報

更新の競合 [46 ページ]

1.2.4.3 プロシージャのレプリケーション

SQL Remote は、プロシージャをレプリケートするときに、プロシージャの動作をレプリケートします。

プロシージャコールはレプリケートされません。代わりに、プロシージャの個々の動作 (INSERT 文、UPDATE 文、DELETE 文) がレプリケートされます。

1.2.4.4 トリガのレプリケーション

通常、リモートデータベースには、統合データベースに存在するトリガと同じトリガが定義されています。

デフォルトでは、SQL Remote はトリガによって実行される動作をレプリケートしません。代わりに、統合データベースでトリガを起動するアクションが、リモートデータベースにレプリケートされると、その複製トリガはリモートデータベースで自動的に起動します。これによって、権限に関する問題と各動作が 2 回発生する可能性を回避します。この原則には次のような例外があります。

RESOLVE UPDATE トリガのレプリケーション

競合解析または RESOLVE UPDATE トリガが実行する動作は、統合データベースから、競合を発生させたメッセージを送信したリモートデータベースを含む、すべてのリモートデータベースにレプリケートされます。

BEFORE トリガのレプリケーション

更新中のローを変更する BEFORE トリガの動作は、UPDATE 文が動作する前にレプリケートされます。たとえば、ローの更新回数を追跡するカウンタカラムをロー内で増加させる BEFORE UPDATE トリガは、レプリケートされると 2 回カウントが行われます。これは、UPDATE 文がレプリケートされると、リモートデータベースの BEFORE UPDATE トリガが起動されるためです。

また、カラムを最終更新時間に設定する BEFORE UPDATE トリガは、UPDATE 文がレプリケートされた時間を取得します。

この問題を回避するには、サブスクライバデータベースに BEFORE UPDATE トリガが存在しないこと、または BEFORE UPDATE トリガがレプリケートされた動作を実行しないことを確認してください。

トリガの動作をレプリケートするオプション

メッセージを送信するときにトリガの動作をすべてレプリケートするには、SQL Remote Message Agent (dbremote) の -t オプションを使用します。

-t オプションを使用する場合は、トリガの動作がリモートデータベースで 2 回 (レプリケートされたトリガの動作が適用されるときと、リモートデータベースでトリガを起動するとき) 実行されないようにします。

トリガの動作が 2 回実行されないようにするには、次のいずれかのオプションを使用します。

- IF CURRENT REMOTE USER IS NULL ...END IF 文で、トリガの本文を囲みます。
- SQL Remote ユーザ名に対して、SQL Anywhere の fire_triggers オプションを Off に設定します。

トリガエラーの回避

パブリケーションにデータベースのサブセットのみが含まれる場合、統合データベースのトリガは、統合データベースには存在し、リモートデータベースには存在しないテーブルやローを参照できます。このようなトリガがリモートデータベースで起動すると、エラーが発生します。こうしたエラーを回避するには、IF 文を使用してトリガの動作を条件付きとし、次の処理を実行します。

- CURRENT PUBLISHER の値を使用して、条件付きのトリガアクションにします。
- NULL 値を返さない object_id 関数を使用して、条件付きのトリガアクションにします。object_id 関数は、テーブルやその他のオブジェクトを引数として取得し、そのオブジェクトの ID 番号か NULL 値 (オブジェクトが存在しない場合) を返します。
- ローが存在するかどうか決定する SELECT 文を使用して、条件付きのトリガアクションにします。

抽出ユーティリティ (dbxtract)

デフォルトでは、データベース抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードによってトリガ定義が抽出されます。

関連情報

CURRENT REMOTE USER 特別値 [51 ページ]
更新の競合に対するデフォルトの解決 [47 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]

1.2.4.5 データ定義文

データ定義文 (CREATE、ALTER、DROP) は、パススルーモードで実行した場合を除き、SQL Remote によってレプリケートされません。

関連情報

SQL Remote のパススルーモード [152 ページ]

1.2.4.6 SQL Remote のデータ型

SQL Remote は、文字セットを変換しません。

互換性のあるソート順と文字セットの使用

SQL Anywhere 統合データベースが使用する文字セットと照合順を、リモートデータベースと同じ設定にする必要があります。

このセクションの内容:

BLOB [44 ページ]
BLOB には、LONG VARCHAR、LONG BINARY、TEXT、IMAGE の各データ型が含まれています。

SQL Remote の日付と時刻 [45 ページ]
日付カラムまたは時刻カラムをレプリケートする場合、SQL Remote は、データベースオプションの sr_date_format、sr_time_format、sr_timestamp_format の設定を使用して、日付をフォーマットします。

1.2.4.6.1 BLOB

BLOB には、LONG VARCHAR、LONG BINARY、TEXT、IMAGE の各データ型が含まれています。

SQL Remote は、INSERT 文または UPDATE 文をレプリケートすると、BLOB 値の代わりに変数を使用します。つまり、BLOB は細かく分割して各部分がレプリケートされます。分割された部分は受信データベースで SQL 変数を使用して再構成され、連結されます。変数の値は、次のように文を結合して作成します。

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

この変数によって、長い値を含んでいる SQL 文が短くなり、1 つのメッセージ内に収まります。

SET 文は独立した SQL 文であるため、BLOB は複数の SQL Remote メッセージに効果的に分割されます。

BLOB のレプリケーションの制御

SQL Anywhere の blob_threshold オプションを使用すると、長い値のレプリケーションを詳細に制御できます。blob_threshold オプションより長い値は、BLOB 値であるかのようにレプリケートされます。

verify_threshold オプションを使用してメッセージサイズを最小にする

verify_threshold データベースオプションを使用すると、(レプリケートされた UPDATE 文の VERIFY 句による) 確認の対象から長い値を除外できます。このオプションのデフォルト値は 1000 です。カラムのデータ型がしきい値より長い場合は、カラムの古い値は UPDATE 文がレプリケートされるときに確認されません。このようにして、SQL Remote メッセージのサイズを小さくしますが、競合する長い値の更新が検出されないという短所もあります。

メッセージのサイズを小さくするために verify_threshold オプションを使用している場合に競合を検出するには、次の方法を使用します。

1. BLOB を更新するときに、同じテーブル内の last_modified カラムも必ず更新されるようにデータベースを設定します。
2. last_modified カラムが BLOB カラムと一緒にレプリケートされるようにパブリケーションを設定します。
3. BLOB カラムと last_modified カラムがレプリケートされるときに、last_modified カラムの値を確認できます。last_modified カラムと競合している場合は、BLOB カラムとも競合しています。

ワークテーブルを使用して重複する更新を防ぐ

BLOB が繰り返し更新されるときは、ワークテーブルで更新し、最終バージョンをレプリケートされたテーブルに割り当てるようにします。たとえば、作業中の資料が 1 日に 20 回更新される場合、1 日の終りに SQL Remote を 1 回実行すると、20 回の更新すべてがレプリケートされます。資料のサイズが 200 KB の場合は、4 MB のメッセージが送信されます。

`document_in_progress` テーブルを使用します。ユーザが資料の変更を終了したときに、アプリケーションがその資料を `document_in_progress` テーブルからレプリケートされたテーブルに移動します。この結果、1 回の更新 (200 KB のメッセージ) で済みます。

1.2.4.6.2 SQL Remote の日付と時刻

日付カラムまたは時刻カラムをレプリケートする場合、SQL Remote は、データベースオプションの sr_date_format、sr_time_format、sr_timestamp_format の設定を使用して、日付をフォーマットします。

たとえば、次のオプション設定は、SQL Remote に 1998 年 5 月 2 日という日付を 1998-05-02 として送信するように命令します。

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

日付と時刻をレプリケートする場合は、次の点を考慮します。

- 時刻、日付、タイムスタンプには、SQL Remote システム全体で一貫したフォーマットを使用します。
- 日付とタイムスタンプのフォーマットに使用されている年、月、日の順序が、date_order データベースオプションの設定と同じになるようにします。
- 個々の接続の間、date_order オプションを変更できます。

1.2.5 レプリケーションの競合とエラー

SQL Remote を使用すると、複数のデータベースでデータベースを更新できます。ただし、特にデータベースの構造が複雑な場合は、レプリケーションエラーを回避するように注意深く設計する必要があります。

レプリケーションの競合

レプリケーションの競合は、レプリケーションエラーとは異なります。SQL Remote では、SQL Remote システムの通常のオペレーションの一部として、トリガとプロシージャを使用して適切な競合解決を提供します。

レプリケーションエラー

次のようなレプリケーションエラーがあります。

ローが見つからないエラー

あるユーザがロー (特定のプライマリキー値を持つ) を削除します。別のユーザが異なるサイトで同じローを更新または削除すると発生します。この場合は、後者のユーザが UPDATE 文または DELETE 文を送信したときに、ローが見つからないという理由でエラーが発生します。

参照整合性エラー

外部キーを持つカラムがパブリケーションに含まれていて、関連するプライマリキーが含まれていない場合、その外部キーを参照する INSERT 文は失敗します。

また、プライマリテーブルに SUBSCRIBE BY 式が含まれていて、それと関連付けされている外部テーブルには含まれていない場合に、参照整合性エラーが発生することがあります。外部テーブルのローをレプリケートすることができても、プライマリテーブルのローはパブリケーションから除外される可能性があるためです。

重複プライマリキーエラー

2 人のユーザが同じプライマリキー値を使用してローを挿入する場合や、あるユーザがプライマリキーを更新し、別のユーザが新しい値のプライマリキーを挿入する場合に発生します。レプリケーションシステムのデータベースに 2 度目にアクセスしようとすると、プライマリキーが 2 つ作成されるため、エラーが発生します。

配信エラー

配信エラーは、保証されたメッセージ配信システムの一部である SYSREMOTEUSER システムテーブルを使用して管理されます。

関連情報

エラー処理プロシージャのレプリケーション [146 ページ]
保証されたメッセージ配信システム [111 ページ]
トリガを使用したカスタム競合解決 [50 ページ]
ローが見つからないエラー [54 ページ]
参照整合性エラー [55 ページ]
重複プライマリキーエラー [57 ページ]
更新の競合に対するデフォルトの解決 [47 ページ]
レプリケーションエラーのレポートと処理 [145 ページ]
更新の競合 [46 ページ]

1.2.6 更新の競合

データが読み取り用に共有されている場合、または各ロー (プライマリキーによって識別される) が 1 つのデータベースのみで更新される場合、更新の競合は発生しません。複数のデータベースでデータが更新された場合にのみ、更新の競合が発生します。

UPDATE 文をレプリケートするため、SQL Remote はローごとに個別の UPDATE 文を発行します。これらのシングルロー文は、次のいずれかの理由で失敗する場合があります。

更新するローが 1 つまたは複数のカラムで異なる

存在するはずの値の 1 つが他のユーザによって変更された場合、更新の競合が発生します。

リモートデータベースでは、ローの値にかかわらず更新が実行されます。

統合データベースでは、SQL Remote によって競合解決オペレーションが実行されます。たとえば、競合が検出されると、統合データベースでは次の処理を実行できます。

- デフォルトによる競合解決を使用します。
- VERIFY 句を使用する、カスタマイズされた競合解決を使用します。

- トリガを使用する、カスタマイズされた競合解決を使用します。

> **注記**
>
> UPDATE 文の競合は、プライマリキーの更新には適用されません。SQL Remote システムで、プライマリキーを更新しないでください。適切な設計を使用して、プライマリキーの競合をシステムから取り除く必要があります。

更新するローが存在しない

個々のローはプライマリキーの値によって識別されます。ローが削除されたり、プライマリキーが別のユーザによって変更されたりしている場合は、更新すべきローが見つかりません。

リモートデータベースでは、更新は行われません。

統合データベースでは、更新は行われません。
主キー制約または一意性制約のないテーブルはレプリケートされた更新の **WHERE** 句のカラムをすべて参照する

2 つのリモートデータベースで同じローが別々に更新され、その変更が統合データベースにレプリケートされた場合は、統合データベースに到着した最初の変更が適用され、2 番目のデータベースの変更は適用されません。そのため、データベースの整合性が失われます。したがって、レプリケートされたすべてのテーブルに主キー制約または一意性制約を持たせ、制約対象のカラムが更新されないようにする必要があります。

このセクションの内容:

更新の競合に対するデフォルトの解決 [47 ページ]
更新の競合は、複数のサイトでデータが更新された場合にのみ発生します。

VERIFY 句を使用したカスタム競合解決 [48 ページ]
SQL Remote は、VERIFY 句を使用するメッセージの中で UPDATE 文を生成します。UPDATE 文は、既存の 1 つ以上のローの値を新しい値に変更します。

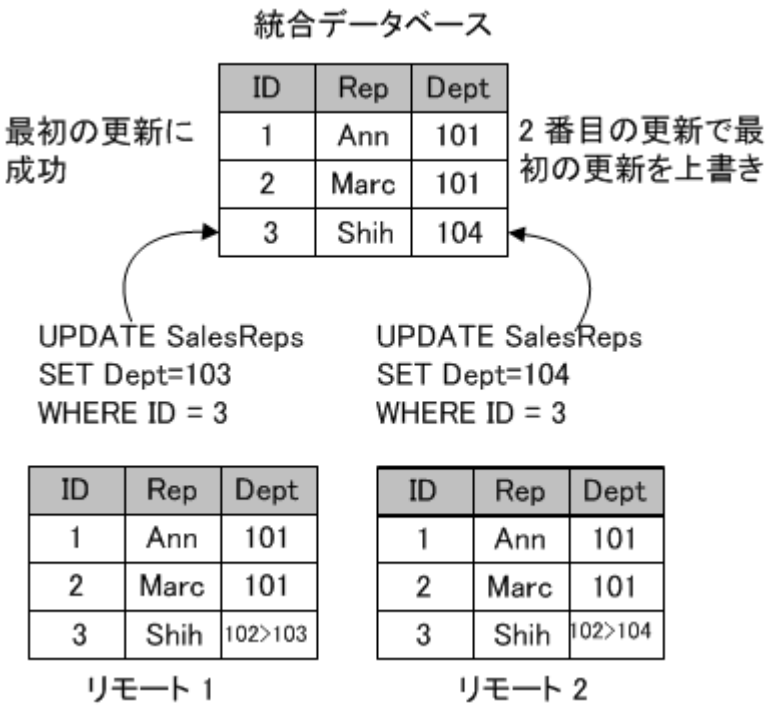
トリガを使用したカスタム競合解決 [50 ページ]
カスタム競合解決はいくつかの手法を取ることができます。たとえば、アプリケーションによっては、解決は次のようになる可能性があります。

1.2.6.1 更新の競合に対するデフォルトの解決

更新の競合は、複数のサイトでデータが更新された場合にのみ発生します。

次に例を示します。

1. ユーザ 1 が、リモートサイト 1 でローを更新します。
2. ユーザ 2 が、リモートサイト 2 で同じローを更新します。
3. ユーザ 1 による更新内容が、統合データベースに送信されて適用されます。
4. ユーザ 2 による更新内容が、統合データベースに送信されます。
   この種の更新の競合を解決するデフォルトの方法は、次のとおりです。
   1. 新しい方のオペレーション (ユーザ 2) が成功します。この値が統合データベースの値になります。また、この値が、ローに対してサブスクライブされるその他すべてのデータベースにレプリケートされます。
   2. その他の更新 (ユーザ 1) がすべて失われます。
   3. 競合はレポートされません。



ID	Rep	Dept
1	Ann	101
2	Marc	101
3	Shih	102>103

ID	Rep	Dept
1	Ann	101
2	Marc	101
3	Shih	102>104

関連情報

ローが見つからないエラー [54 ページ]

1.2.6.2 VERIFY 句を使用したカスタム競合解決

SQL Remote は、VERIFY 句を使用するメッセージの中で UPDATE 文を生成します。UPDATE 文は、既存の 1 つ以上のローの値を新しい値に変更します。

VERIFY 句を含む UPDATE 文には、そのローの既存の値も含まれています。

UPDATE 文を適用すると、統合データベースは既存のローの値を、リモートデータベースで既存のローの値が変更された後に期待される値と比較します。更新の競合は、VERIFY 句の値がデータベースのローと一致しない場合に、データベースサーバによって検出されます。

たとえば、更新の競合は、次のような順序でイベントが実行されると発生します。

1. ユーザ 1 が、リモートサイト 1 でローを更新します。
2. ユーザ 2 が、リモートサイト 2 で同じローを更新します。
3. ユーザ 1 による更新内容が、統合データベースに送信されて適用されます。
4. ユーザ 2 による更新内容が、統合データベースに送信されます。
   UPDATE 文に VERIFY 句が含まれているため、統合データベースは競合を検出できます。統合データベースでは、SQL Remote がローに含まれている値を、ユーザ 2 が送信した古いロー値と比較します。これらの値は同じではないため、更新の競合が発生します。

更新の競合が検出されると、統合データベースでは次の処理を実行します。

1. オペレーションに対して定義された、すべての競合解決トリガを起動します。
   競合解決トリガを定義して、更新の競合を処理します。競合解決トリガは、リモートユーザがメッセージを適用する場合に、統合データベースでのみ起動されます。
2. UPDATE 文を実行します。
3. 競合解決トリガのすべてのアクションと UPDATE 文が、すべてのリモートデータベースに送信されます。この中には、その競合をトリガするメッセージを送信したリモートデータベースも含まれています。
   通常、SQL Remote は、トリガアクションをレプリケートしません。これは、トリガがリモートデータベースにあることを想定しているためです。競合解決トリガは統合データベースでのみ起動されるため、このトリガアクションはリモートデータベースにレプリケートされます。
5. リモートデータベースは、統合データベースから UPDATE 文を受信します。
   リモートデータベースでは、統合データベースからのメッセージに更新の競合が含まれていると、RESOLVE UPDATE トリガは起動されません。
6. リモートデータベースで、UPDATE 文が処理されます。
   プロセスの最後では、システム全体でデータの一貫性が保たれます。

このセクションの内容:

関連情報

トリガを使用したカスタム競合解決 [50 ページ]

1.2.6.2.1 VERIFY 句を持つ UPDATE 文

SQL Remote は、VERIFY 句を使用するメッセージの中で UPDATE 文を生成します。UPDATE 文は、既存の 1 つ以上のローの値を新しい値に変更します。

次に、VERIFY 句を持つ UPDATE 文を示します。

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

VERIFY 句が使用できるのは、`table-list` パラメータにテーブルが 1 つしか存在しない場合のみです。VERIFY 句は、指定したカラムの値と、予測される値のセットを比較します。この予測値は、UPDATE 文が適用されたときにパブリッシャデータベースに存在した値です。VERIFY 句を指定すると、テーブルが一度に 1 つずつ更新されます。

VERIFY 句を使用できるのは、単一のローを更新する場合だけです。ただし、このことがクライアントアプリケーションを記述する場合に制約になることはありません。これは、複数のローの UPDATE 文がデータベースで実行されても、SQL Remote によって単一のローに対する UPDATE 文の繰り返しとして解釈されるためです。

1.2.6.2.2 verify_all_columns オプション

デフォルトでは、データベースオプション verify_all_columns は Off に設定されています。設定が Off の場合は、更新されたカラムのみが検証されます。

verify_all_columns オプションが On に設定されている場合は、次のようになります。

- レプリケートされた UPDATE 文に対して、すべてのカラムが検証されます。
- いずれかのカラムが異なっているときには必ず RESOLVE UPDATE トリガが起動します。
- 各 UPDATE 文に対して送信される情報が多くなるため、メッセージのサイズが大きくなります。

verify_all_columns オプションは、PUBLIC ロール用、または SQL Remote の接続文字列に含まれるユーザ用のいずれかに設定できます。

抽出ユーティリティ (dbxtract)

リモートデータベースを抽出する前に統合データベースで verify_all_columns オプションが設定されている場合、抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードによってリモートデータベースの verify_all_columns オプションが設定されます。

1.2.6.3 トリガを使用したカスタム競合解決

カスタム競合解決はいくつかの手法を取ることができます。たとえば、アプリケーションによっては、解決は次のようになる可能性があります。

- 元のトランザクションの日付を比較します。
- 2 つ以上の更新の結果に対して計算を実行します。
- 競合をテーブルにレポートします。

カスタム競合解決を使用するには、RESOLVE UPDATE トリガを記述する必要があります。

RESOLVE UPDATE 競合解決トリガの使用

RESOLVE UPDATE トリガの起動後に、各ローが更新されます。RESOLVE UPDATE トリガの構文は、次のとおりです。

```
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
```

REFERENCING 句を使用すると、更新するテーブルのロー値 (OLD)、更新用のロー値 (NEW)、VERIFY 句によれば存在するはずのロー (REMOTE) にアクセスできます。REMOTE AS 句の中で参照できるのは、VERIFY 句内のカラムのみです。その他のカラムは、エラーを返します。

このセクションの内容:

関連情報

トリガのレプリケーション [41 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]

1.2.6.3.1 CURRENT REMOTE USER 特別値

CURRENT REMOTE USER 特別値は、メッセージをデータベースに適用するときに、SQL Remote の受信フェーズによって設定されます。

CURRENT REMOTE USER 特別値は、適用される操作が SQL Remote の受信フェーズによって適用されるかどうか、および、そのことが当てはまる場合に、適用される操作をどのリモートユーザが生成したかをトリガで判別する場合に、非常に役立ちます。

1.2.6.3.2 日付の競合解決

日付の競合を解決する方法を説明するため、交渉管理システムのテーブルの 1 つに、各顧客との最新の交渉日のデータを保持するカラムがあるとします。

ある担当者が金曜日に顧客と交渉を行いましたが、この担当は、変更を月曜日まで統合データベースにアップロードしません。一方、別の担当者が土曜日に同じ顧客と交渉し、変更をその日の夕方に更新しました。

土曜日に更新が統合データベースにレプリケートされても競合は発生しませんが、月曜日に更新が受信された時点で、ローがすでに変更されていることがわかります。

デフォルトでは月曜日の更新が処理され、金曜日を最新の交渉日とする、間違った情報を持つカラムが残されます。ただし、このカラムの更新の競合は、最新の日付をローに挿入して解決してください。

解決の実装

次の RESOLVE UPDATE トリガは、新しい 2 つの値から最新のものを選択して、データベースにその値を入力します。

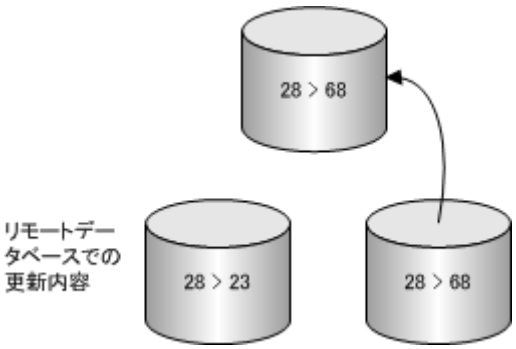
```
CREATE TRIGGER contact_date RESOLVE UPDATE
 ON Contacts
 REFERENCING OLD AS old_name
  NEW AS new_name
 FOR EACH ROW
 BEGIN
   IF new_name.contact_date <
      old_name.contact_date THEN
          SET new_name.contact_date
           = old_name.contact_date
   END IF
 END;
```

更新される値が置き換える値よりも新しい場合は、新しい値はリセットされて、変更されないままエントリが残されます。

1.2.6.3.3 在庫競合解決

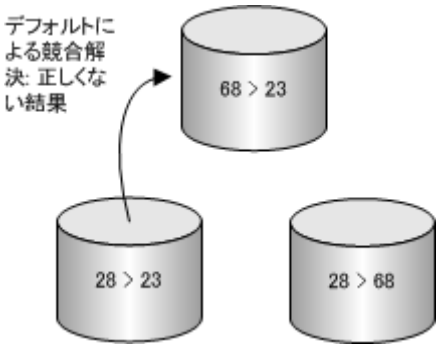
スポーツ用品メーカーのための倉庫システムがあるとします。製品情報のテーブルには、各製品の残数量を保持する Quantity カラムがあります。このカラムへの更新は、通常は在庫数を引いていくことであり、新しく製品が搬入される場合は、在庫数を追加します。

リモートデータベースで営業担当者が受注を入力し、S サイズのタンクトップ T シャツの在庫を 5 枚差し引いて 28 から 23 としました。この在庫数も担当者のデータベースに入力されます。一方、この更新内容が統合データベースにレプリケートされる前に、別の営業担当者は T シャツの返品を 40 枚受け取りました。この営業担当者は、自分のリモートデータベースに返品を入力し、倉庫での変更を統合データベースにレプリケートして、Quantity カラムの値を 40 追加して 68 とします。

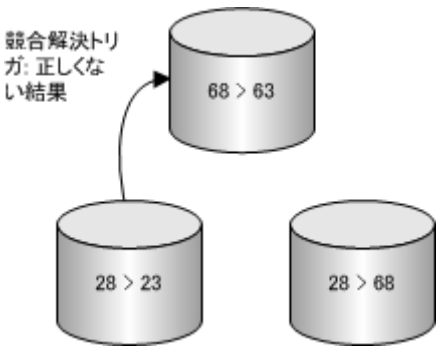


この倉庫のエントリがデータベースに追加され、現在の Quantity カラムは、在庫に 68 枚の S サイズのタンクトップ T シャツがあることを示しています。ここで最初の営業担当者からの更新を受信すると、28 から 23 に変更するという内容にもかかわらず、実際のカラム値は 68 であるという競合が SQL Anywhere で検出されます。

デフォルトでは、より新しい更新が処理され、在庫レベルが間違った値 23 に設定されます。



この例では、データベースの最終的な値が 63 になるように、在庫のカラムに対する変更を合計した結果を算出して解決する必要があります。



解決の実装

このような状況に適した RESOLVE UPDATE トリガでは、2 つの更新内容による増加分を追加します。次に例を示します。

```
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;
```

統合データベースの以前の値 (68) と、元の UPDATE 文が実行された時点のリモートデータベースの以前の値 (28) の差が、このトリガによって送信する新しい値に追加されてから、UPDATE 文が実行されます。したがって、new_name.Quantity は 63 (= 23 + 68 - 28) となり、この値が Quantity カラムに入力されます。

リモートデータベースでの一貫性は、次のように管理されます。

1. 元のリモート UPDATE 文によって、値が 28 から 23 に変更されました。
2. 倉庫システムのエントリがリモートデータベースにレプリケートされますが、以前の値が予測値と一致しないためにエラーが発生します。
3. RESOLVE UPDATE トリガによる変更内容が、リモートデータベースにレプリケートされます。

1.2.7 ローが見つからないエラー

あるユーザがロー (特定のプライマリキー値を持つ) を削除します。別のユーザが異なるサイトで同じローを更新または削除すると発生します。この場合は、後者のユーザが UPDATE 文または DELETE 文を送信したときに、ローが見つからないという理由でエラーが発生します。

UPDATE 文と DELETE 文を正しくレプリケートするには、すべてのプライマリキーカラムをアーティクルに含める必要があります。

UPDATE 文または DELETE 文をレプリケートする場合、SQL Remote はプライマリキーカラムを使用して、更新または削除を行うローをユニークに識別します。レプリケートされるすべてのテーブルには、宣言されたプライマリキーか、一意性制約がある必要があります。ユニークインデックスでは十分ではありません。

WHERE 句とプライマリキー

プライマリキーカラムは、レプリケートされた UPDATE 文と DELETE 文の WHERE 句の中で使用します。テーブルにプライマリキーがない場合、WHERE 句はテーブルのすべてのカラムを参照します。

関連情報

レプリケーションエラーのレポートと処理 [145 ページ]
更新の競合に対するデフォルトの解決 [47 ページ]
INSERT および DELETE 文のレプリケーション [39 ページ]

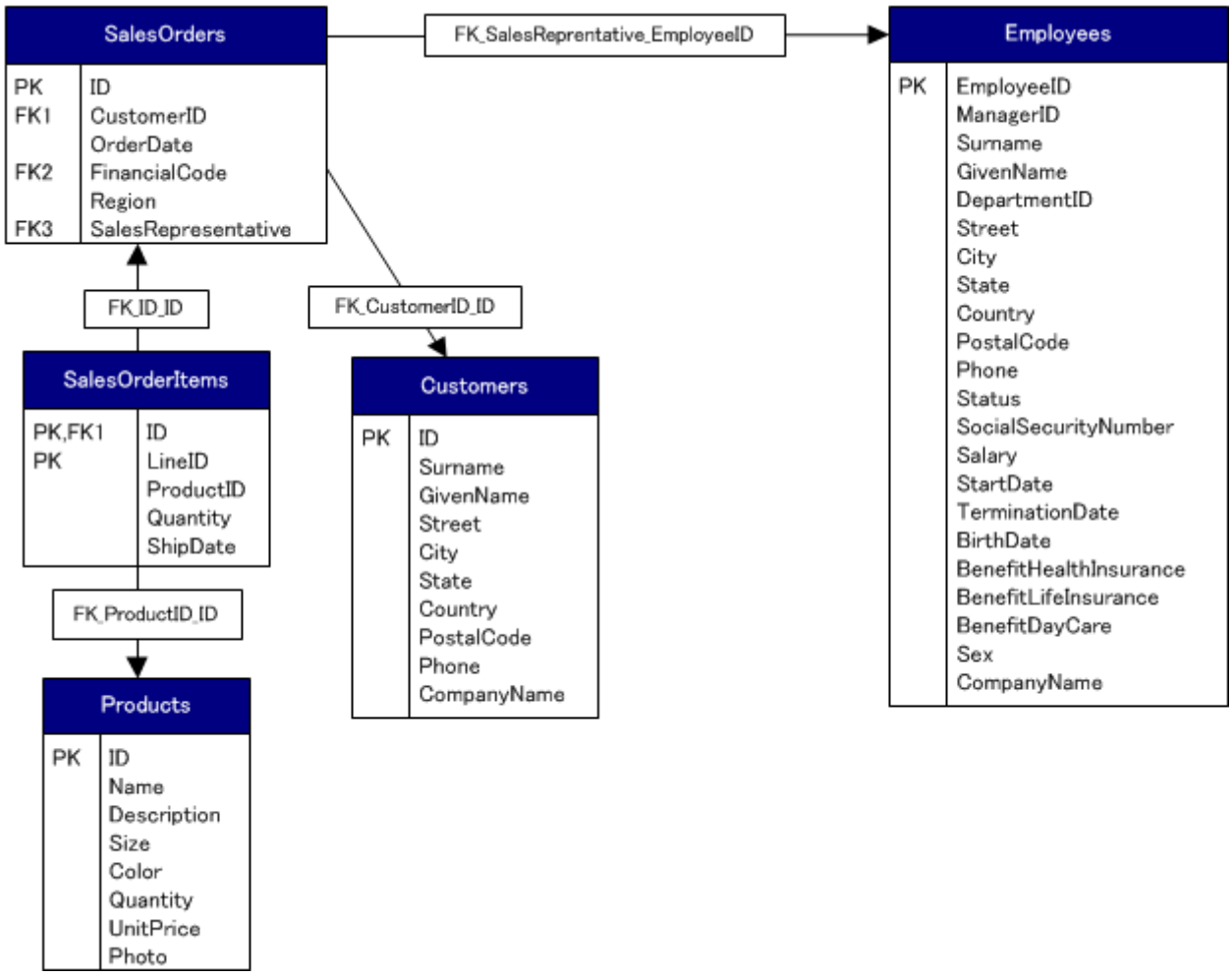
1.2.8 参照整合性エラー

リレーショナルデータベースのテーブルは、外部キーの参照で関連付けられることがよくあります。そのため、参照整合性制約によって、データベースの一貫性が保たれます。

データベースの一部のみをレプリケートする場合は、レプリケート後のデータベースでも引き続き参照整合性が保たれていることを確認する必要があります。

参照テーブルがレプリケートされないエラーを回避する必要があります。リモートデータベースには、レプリケートされないテーブルを参照する外部キーが含まれないようにしてください。

たとえば、統合データベースの SalesOrders テーブルには、Employees テーブルに対する外部キーがあります。SalesOrders.SalesRepresentative は、プライマリキー Employees.EmployeeID を参照する外部キーです。



次の例では、Employees テーブルを含まず、SalesOrder テーブル全体を含むパブリケーション PubSales を作成します。

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

リモートユーザ Rep1 は、PubSales パブリケーションに対してサブスクライブします。次に、統合データベースから Rep1 を抽出して、Rep1 用のデータベースの作成を試みます。ただし、Rep1 には Employees テーブルがないため、データベースの作成に失敗します。この問題を回避するには、次の処理を実行します。

外部キー参照を削除する

外部キー参照を除外するには、抽出ユーティリティ (dbxtract) を使用するときに -xf オプションを指定します。

ただし、リモートデータベースから外部キー参照を削除する場合、リモートデータベースには、SalesOrders テーブルの SalesRepresentative カラムに無効な値が挿入されることを防ぐ制約はありません。

リモートデータベースで SalesRepresentative カラムに無効な値が挿入された場合、レプリケートされた INSERT 文は統合データベースで失敗します。

不足しているテーブルをパブリケーションに追加する

パブリケーションに Employees テーブル (または少なくともプライマリキー) を追加します。次に例を示します。

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

このセクションの内容:

関連情報

レプリケーションエラーのレポートと処理 [145 ページ]

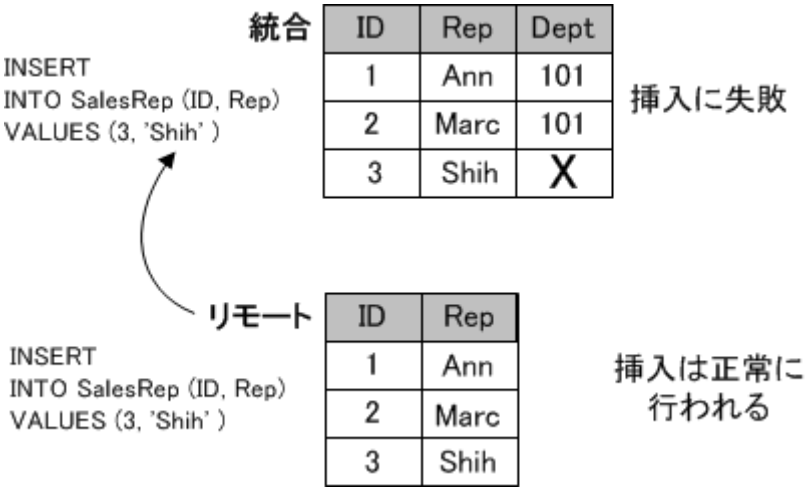
1.2.8.1 挿入時のエラー

リモートデータベースから統合データベースに INSERT 文をレプリケートする場合は、パブリケーションから一部のカラムのみを除外できます。それ以外のカラムを除外すると、挿入は失敗します。

除外できるのは次のものです。

- NULL を使用できるカラム
- デフォルトのカラム

これらの条件を満たさないカラムを除外すると、統合データベースへのレプリケート時にリモートデータベースで実行された INSERT 文が失敗します。



> **注記**
>
> この例の例外として、BEFORE トリガを使用して、INSERT 文に含まれていないカラムを管理する場合があります。

関連情報

レプリケーションの競合とエラー [45 ページ]

1.2.9 重複プライマリキーエラー

すべてのユーザが同じデータベースに接続する場合は、各 INSERT 文がユニークなプライマリキーを使用することが保証されるため、問題は発生しません。ユーザがプライマリキーの再使用を試みる場合に、INSERT 文のエラーが発生します。

レプリケーションシステムにおいては状況が異なります。これは、ユーザが複数のデータベースに接続しているためです。異なるリモートデータベースに接続している 2 人のユーザが、同じプライマリキーの値を使用してローを挿入すると問題が発生します。各リモートデータベースでは、プライマリキーの値がユニークであるため、各文が正常に実行されます。

ただし、この 2 人のユーザが自分のデータベースを同じ統合データベースにレプリケートすると、問題が発生します。統合データベースに対する最初のデータベースのレプリケーションは、正常に実行されます。ただし、レプリケーションシステムの指定されたデータベースで受信される、2 番目の挿入の実行は失敗します。

必ずユニークとするプライマリキー値

プライマリキーのエラーを回避するには、データベースでローが挿入されるときに、そのプライマリキーがシステムのすべてのデータベース間でユニークであることが必ず保証されている必要があります。この目的を達成するには、次のようないくつかの方法があります。

1. SQL Anywhere データベースのデフォルトのグローバルオートインクリメント機能を使用します。
2. プライマリキープールを使用して、各サイトで未使用のユニークなプライマリキー値のリストを管理します。

これらの方法のいずれか一方または両方を使用すると、プライマリキーの重複を回避できます。

このセクションの内容:

関連情報

レプリケーションエラーのレポートと処理 [145 ページ]

1.2.9.1 SQL Remote の GLOBAL AUTOINCREMENT カラム

GLOBAL AUTOINCREMENT のデフォルトを使用して、リモートデータベースごとにユニークなグローバルデータベース ID 番号を割り当てます。

カラムに GLOBAL AUTOINCREMENT のデフォルトを指定すると、そのカラムの値のドメインが分割されます。各分割には同じ数の値が含まれます。たとえば、データベース内の整数カラムの分割サイズを 1000 に設定した場合、1 つの分割が 1001 から 2000 まで拡大します。また、2 つ目の分割は 2001 から 3000 まで拡大し、以降、同じように拡大していきます。

SQL Anywhere では、データベースのデフォルト値は、そのデータベース番号でユニークに識別された分割からのみ設定されます。たとえば、リモートデータベースに ID 番号 10 を割り当てた場合、このデータベースのデフォルト値は 10001 ～ 11000 の範囲から選択されます。ID 番号 11 が割り当てられた別のリモートデータベースでは、11001 ～ 12000 の範囲にある同一カラムのデフォルト値が指定されます。

このセクションの内容:

1.2.9.1.1 デフォルト GLOBAL AUTOINCREMENT 宣言

SQL Central でカラムのプロパティを選択するか、CREATE TABLE 文または ALTER TABLE 文に DEFAULT GLOBAL AUTOINCREMENT 句を組み込むことで、作業データベースにデフォルト値を設定できます。

分割サイズ

オプションで、AUTOINCREMENT キーワードの直後に括弧で分割サイズを指定できます。この分割サイズには任意の負でない整数を設定できますが、通常、分割サイズは、サイズの値がすべての分割で不足しないように選択されます。

カラムの型が INT または UNSIGNED INT である場合、デフォルトの分割サイズは 2^{16} = 65536 です。それ以外の型のカラムの場合、デフォルトの分割サイズは 2^{32} = 4294967296 です。特に、カラムの型が INT または BIGINT ではない場合は、これらのデフォルト値が適切ではないことがあるため、分割サイズを明示的に指定してください。

例

次の文では、2 つのカラム (顧客 ID 番号を保持する整数カラムと顧客名を保持する文字列カラム) を持つテーブルが作成されます。ID 番号カラムである ID では GLOBAL AUTOINCREMENT のデフォルトを使用し、その分割サイズは 5000 です。

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

1.2.9.2 SQL Remote のプライマリキープール

プライマリキープールは、SQL Remote システムで、各データベースのプライマリキー値のセットを格納するテーブルです。

マスタプライマリキープールテーブルが作成され、統合データベースに格納されます。リモートユーザは、統合データベースのプライマリキープールテーブルに対してサブスクライブし、自分のプライマリキー値のセットを受信します。リモートユーザが新しいローをテーブルに挿入する場合は、ストアドプロシージャを使用してプールから有効なプライマリキーを選択します。プールは、使用できる値を補充するプロシージャを、統合データベースで定期的に実行することによって管理されます。

プライマリキープールを使用するには、次のコンポーネントが必要です。

プライマリキープールテーブル

統合データベースでは、システム内のデータベースごとに有効なプライマリキー値を保持するテーブルが必要です。

補充プロシージャ

統合データベースでは、値を補充したキープールテーブルを維持するストアドプロシージャが必要です。

キープールの共有

システムの各リモートデータベースは、統合データベースのキープールテーブルから取得した、そのデーターベース自体の有効な値のセットに対してサブスクライブする必要があります。

データ入力プロシージャ

リモートデータベースでは、次に有効なプライマリキー値をプールから選択し、キープールからその値を削除するストアドプロシージャを使用して、新しいローを入力します。

このセクションの内容:

1.2.9.2.1 プライマリキープールテーブルの作成 (SQL)

プライマリキープールテーブルを作成します。

手順

1. 統合データベースで次の文を実行して、プライマリキープールテーブルを作成します。

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

カラム	説明
table_name	プライマリキープールで管理するテーブルの名前を保持します。たとえば、統合データベースのみに営業担当者が新しく追加されると、Customers テーブルのみがプライマリキープールを必要とし、このカラムは重複します。
value	プライマリキー値のリストを保持します。それぞれの値は、table_name にリストされた各テーブルに対してユニークです。
location	受信者の識別子。システムによっては、この値が SalesReps テーブルの rep_key 値と同じになる場合があります。また、営業担当者以外のユーザが存在するシステムもあり、このようなシステムでは、この 2 つの識別子は別々である必要があります。

2. パフォーマンスを向上させるには、次の文を実行して、プライマリキーテーブルにインデックスを作成します。

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

結果

プライマリキープールが作成されます。

1.2.9.2.2 プライマリキープールのレプリケーション (SQL)

プライマリキープールに個別のパブリケーションを作成し、そのパブリケーションに対してユーザをサブスクライブします。

前提条件

事前にプライマリキープールテーブルが作成されている必要があります。

コンテキスト

プライマリキープールは、既存のパブリケーションに組み込むか、または別のパブリケーションとして共有できます。

手順

1. 統合データベースで、プライマリキープールのデータ用のパブリケーションを作成します。

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

2. 各リモートデータベースのサブスクリプションを、KeyPoolData パブリケーションに作成します。

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

サブスクリプションの引数は、location の識別子です。

結果

プライマリキープールがレプリケートされます。

1.2.9.2.3 キープールの入力と補充 (SQL)

統合データベースのプライマリキープールテーブルの内容を補充して、リモートデータベースに新しいプライマリキーをレプリケートします。

前提条件

事前にプライマリキープールテーブルが作成されている必要があります。

手順

1. 統合データベースで、プライマリキープールを値で埋めるためのプロシージャを作成します。

   > **注記**
   >
   > トリガアクションはレプリケートされないため、キープールの補充にはトリガを使用できません。

   次に例を示します。

```
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. 各ユーザのプライマリキープールにプライマリキーの初期値を挿入します。

   ReplenishPool プロシージャには、各サブスクライバに対して少なくとも 1 つのプライマリキー値が必要です。これによって、最大値を検索し、値を追加して次のセットを生成できます。

   はじめにプールを値で埋めるには、各ユーザに対して値を 1 つ挿入してから、ReplenishPool を呼び出して残りを埋めます。次の例では、3 人のリモートユーザと単一の統合ユーザ Office について示します。

   ```
   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();
   ```

   ReplenishPool プロシージャによって、各ユーザのプールの値が 100 個まで増えます。指定する値は、ユーザがデータベースのテーブルにローを挿入する頻度によって異なります。

3. 定期的に ReplenishPool を実行します。

   キープールテーブルのプライマリキー値のプールを再補充するため、ReplenishPool プロシージャを統合データベースで定期的に実行してください。

結果

統合データベースのプライマリキープールテーブルが補充され、新しいプライマリキーがリモートデータベースにレプリケートされます。

1.2.9.2.4 キープールのプライマリキーの使用 (SQL)

プライマリキーを使用して、リモートユーザが新しい顧客を追加したときにリモートユーザが使用できるプライマリキーのプールを作成します。

前提条件

事前にプライマリキープールテーブルが作成されている必要があります。

コンテキスト

営業担当者が Customers テーブルに新しく顧客を追加する場合、挿入するプライマリキー値は、ストアドプロシージャを使用して取得します。次の例では、プライマリキー値を提供するストアドプロシージャと、挿入を実行するストアドプロシージャを使用します。

手順

1. リモートデータベース上で実行するプロシージャを作成して、プライマリキープールテーブルからプライマリキーを取得します。

   たとえば、NewKey プロシージャは、キープールにある整数値を提供し、プールからその値を削除します。

```
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;
```

   NewKey プロシージャは、営業担当者の識別子がリモートデータベースの CURRENT PUBLISHER であることを利用します。

2. リモートデータベース上で実行するプロシージャを作成して、サブスクライブされたテーブルに新しいローを挿入します。

   たとえば、NewCustomers プロシージャは、プライマリキーを構成する NewKey が取得した値を使用して、テーブルに新しい顧客を挿入します。

```
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
```

NewKey から取得される new_cust_key 値をテストしてこの値が NULL でないことを確認し、値が NULL の場合には挿入しないように、プロシージャを強化できます。

結果

プライマリキーが設定されます。

1.2.10 リモートデータベース間でのローの分割

各リモートデータベースは、統合データベースに格納されたデータの異なるサブセットを持つことができます。パブリケーションとサブスクリプションを作成すると、リモートデータベース間でデータが分割されます。

共通部分がないように切断分割にすることも、重複を持たせて分割することもできます。たとえば、従業員ごとに独自の顧客セットを持っていて、かつ顧客を共有していない場合は、切断分割になります。複数のリモートデータベースに存在するように顧客を共有している場合、分割は重複を含みます。

場合によっては、テーブルにサブスクリプション式がなくても、テーブルのローを分割する必要があります。

場合によっては、データベースに多対多の関係が存在する場合、テーブルを分割する必要があります。

関連情報

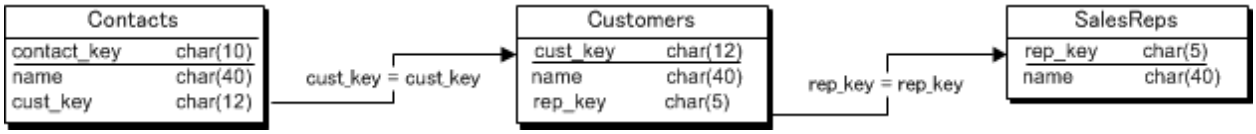
データ分割の切断 [65 ページ]
重複分割 [71 ページ]

1.2.11 データ分割の切断

リモートデータベースがデータを共有していない場合、データ分割は切断となります。

たとえば、各営業担当者には独自の顧客セットがあり、他の営業担当者と顧客を共有していません。

次の例では、3 つのテーブル Customers、Contacts、SalesReps に、営業担当者と顧客間の対話に関する情報が格納されています。各営業担当者は、複数の顧客に対して販売活動を行います。連絡先が 1 箇所だけの顧客もいれば、複数ある顧客もいます。



Contacts、Customers、SalesReps テーブルの説明

次の表では、Customers、Contacts、SalesReps の各データベーステーブルについて説明します。

テーブル	説明	テーブル定義
Contacts	会社と取引があるすべての個別の連絡先。連絡先はそれぞれ 1 人の顧客に属します。Contacts テーブルには、次のカラムがあります。 **contact_key** 各連絡先の識別子。これがプライマリキーです。 **name** 各連絡先の名前。 **cust_key** 連絡先が関連する顧客の識別子。これが Customers テーブルへの外部キーです。	`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) );`
Customers	会社と取引があるすべての顧客。Customers テーブルには、次のカラムがあります。 **cust_key** 各顧客の識別子。これがプライマリキーです。 **name** 各顧客の名前。 **rep_key** 取引を行う営業担当者の識別子。これが SalesReps テーブルへの外部キーです。	`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	社内のすべての営業担当者。SalesReps テーブルには、次のカラムがあります。 **rep_key** 各営業担当の識別子。これがプライマリキーです。 **name** 各営業担当の名前。	`CREATE TABLE SalesReps ( rep_key CHAR(12) NOT NULL, name CHAR(40) NOT NULL, PRIMARY KEY (rep_key) );`

営業担当者は、次の情報を提供するパブリケーションに対してサブスクライブする必要があります。

社内のすべての営業担当者のリスト

次の文では、SalesRep テーブル全体をパブリッシュするパブリケーションを作成します。

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

営業担当者に割り当てられている顧客のリスト

この情報は、Customers テーブルで取得できます。次の文では、Customers テーブルをパブリッシュするパブリケーションを作成します。このパブリケーションには、Customers テーブルの rep_key カラムの値に一致するローが含まれています。

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

割り当てられている顧客の窓口情報のリスト

この情報は、Contacts テーブルで取得できます。Contacts テーブルは営業担当者の間で分割する必要がありますが、SalesRep テーブルの rep_key 値への参照は含みません。この問題を解決するには、Customers テーブルの rep_key カラムを参照する Contacts アーティクルで、サブクエリを使用します。

次の文では、Contacts テーブルをパブリッシュするパブリケーションを作成します。このパブリケーションには、Customers テーブルの rep_key カラムを参照するローが含まれています。

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

Customers テーブルの 1 つのローには、Contacts テーブルの現在のローの cust_key 値が含まれています。SUBSCRIBE BY 文の中で WHERE 句を使用すると、サブクエリは必ず単一の値のみを返します。

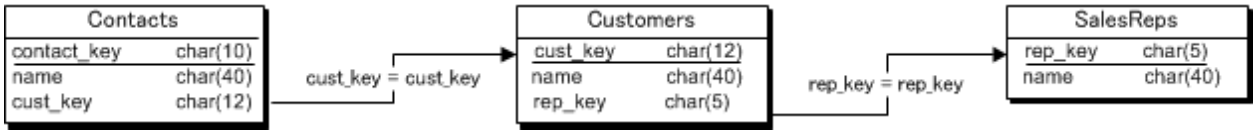
次の文では、完全なパブリケーションが作成されます。

```
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 )
);
```

このセクションの内容:

1.2.11.1 BEFORE UPDATE トリガ

次の例では、3 つのテーブル Customers、Contacts、SalesReps に、営業担当者と顧客間の対話に関する情報が格納されています。各営業担当者は、複数の顧客に対して販売活動を行います。連絡先が 1 箇所だけの顧客もいれば、複数ある顧客もいます。



営業担当者は、SalesRep テーブルのコピー、営業担当者に割り当てられた顧客の詳細が格納されている Customers テーブルのコピー、顧客に対応する窓口の詳細が格納された Contacts テーブルのコピーを提供するパブリケーションに対してサブスクライブします。たとえば、各営業担当者は、次のパブリケーションに対してサブスクライブします。

```
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 )
);
```

参照整合性の維持

サブスクライバ間のローの再割り当ては、営業担当者間に顧客を定期的に再割り当てする営業支援アプリケーションに共通の機能で、領域の再編成とも呼ばれます。

統合データベースでは、新しい営業担当者に顧客が再割り当てされると、Customers テーブルの rep_key 値が更新されます。

次の文では、顧客 cust1 を別の営業担当者 rep2 に再割り当てします。

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

この更新は、次のようにレプリケートされます。

- 以前の営業担当者のリモートデータベースの Customers テーブルに対する DELETE 文として。
- 新しい営業担当者のリモートデータベースの Customers テーブルに対する INSERT 文として。

Contacts テーブルは変更されません。統合データベースのトランザクションログには、Contacts テーブルに関するエントリがありません。そのため、リモートデータベースの SQL Remote は、Contacts テーブルの cust_key ローを再割り当てできません。この再割り当てができないことによって、以前の営業担当者のリモートデータベースの Contacts テーブルには、すでに存在しない顧客の cust_key 値が含まれるという参照整合性の問題が発生します。

解決法としては、BEFORE UPDATE トリガを使用します。BEFORE UPDATE トリガはデータベーステーブルをまったく変更しませんが、統合データベースのトランザクションログにエントリを作成します。

この BEFORE UPDATE トリガは、次のように起動する必要があります。

- UPDATE 文が実行される前。したがって、ローの BEFORE 値が評価され、トランザクションログに追加されます。
- 各文ではなく FOR EACH ROW。トリガの提供する情報を新しいサブスクリプション式とする必要があります。

たとえば、次の文では、BEFORE UPDATE トリガを作成します。

```
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 は、トランザクションログに記録された情報を使用して、どのサブスクライバがどのローを受信するかを決定します。

この文を実行した後、統合データベースのトランザクションログには 2 つのエントリが含まれています。

- BEFORE UPDATE トリガによって生成された、Contacts テーブルの INSERT 文と DELETE 文。

```
--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
```

- 実行された元の UPDATE 文と、それぞれローが挿入または削除されたユーザの INSERT 文と DELETE 文。

```
--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 は、BEFORE タグと AFTER タグのトランザクションログをスキャンします。この情報に基づいて、どのリモートユーザが INSERT 文、UPDATE 文、または DELETE 文を取得するかが決定されます。

- ユーザが BEFORE list に含まれていて AFTER list に含まれていない場合は、DELETE 文が Contacts テーブルに送信されます。
- ユーザが AFTER list に含まれていて BEFORE list に含まれていない場合は、INSERT 文が Contacts テーブルに送信されます。
- ユーザが BEFORE list と AFTER list の両方に含まれている場合は、Contacts テーブルに対して何も実行されませんが、Customers テーブルの UPDATE 文は送信されます。

BEFORE list と AFTER list の値が同じ場合は、リモートユーザがすでにローを保有しているため、UPDATE 文が送信されます。

トリガに関する注意事項

次の例では、BEFORE UPDATE トリガを使用する必要があります。他のコンテキストでは、BEFORE DELETE トリガと BEFORE INSERT トリガが必要です。

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

この例では、BEFORE トリガを使用します。

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

UPDATE 文は、変更が適用されるパブリケーションとテーブルをリストします。文中の WHERE 句は、変更が適用されるローを示します。この UPDATE 文では、テーブルのデータは変更されませんが、トランザクションログにエントリが作成されます。

この例では、サブスクリプション式は 1 つの値を返します。ただし、複数の値を返すサブクエリも使用できます。サブスクリプション式の値は、更新を実行した後の値とする必要があります。

この例では、ローへのサブスクライバのみが新しい営業担当者になります。

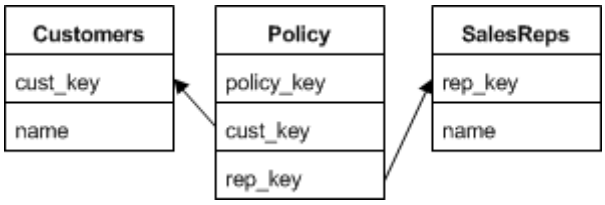
関連情報

重複分割 [71 ページ]

1.2.12 重複分割

リモートデータベースがデータを共有している場合、データ分割は重複となります。たとえば、営業担当者は顧客を共有できます。

3 つのテーブル Customers、Policy、SalesReps に、営業担当者と顧客間の対話に関する情報が格納されているとします。各営業担当者は複数の顧客を担当していますが、複数の営業担当者と取り引きしている顧客もいます。Policy テーブルには、Customers と SalesReps の両テーブルへの外部キーがあります。Customers と SalesReps 間には、多対多の関係があります。



Customers、Policy、SalesReps テーブルの説明

次の表では、Customers、Policy、SalesReps の各データベーステーブルについて説明します。

テーブル	説明

Customers	会社と取引があるすべての顧客。Customers テーブルには、次のカラムがあります。
	cust_key
	各顧客の識別子を含むプライマリキーのカラム。
	name
	各顧客の名前を含むカラム。
	次の文で、このテーブルが作成されます。
	`CREATE TABLE Customers (`
	`  cust_key CHAR(12) NOT NULL,`
	`  name CHAR(40) NOT NULL,`
	`  PRIMARY KEY (cust_key)`
	`);`

テーブル	説明

Policy	顧客と営業担当者間の多対多の関係を管理する、3 つのカラムで構成されたテーブル。Policy テーブルには次のカラムがあります。
	policy_key
	取り引きの識別子を含んだ、プライマリキーのカラム
	cust_key
	取り引きを行う顧客の外部キーを含むカラム。
	rep_key
	取り引きを行う営業担当者の外部キーを含むカラム。
	次の文で、このテーブルが作成されます。
	`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	社内のすべての営業担当者。SalesReps テーブルには、次のカラムがあります。
	rep_key
	各営業担当の識別子。これがプライマリキーです。
	name
	各営業担当の名前。
	次の文で、このテーブルが作成されます。
	`CREATE TABLE SalesReps (`
	`  rep_key CHAR(12) NOT NULL,`
	`  name CHAR(40) NOT NULL,`
	`  PRIMARY KEY (rep_key)`
	`);`

データの分割

顧客と営業担当者間の多対多の関係では、適切に情報を共有するための新しい問題が発生します。

営業担当者は、次の情報を提供するパブリケーションに対してサブスクライブする必要があります。

SalesReps テーブル全体

このアーティクルに対応した修飾子はありません。このため、SalesReps テーブル全体がパブリケーションに含まれます。

```
...
```

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

データに対してサブスクライブされた営業担当者を含む、取り引きを記録した **Policy** テーブルのロー

このアーティクルは、SUBSCRIBE BY サブスクリプション式を使用して、営業担当者間でデータの分割に使用するカラムを指定します。

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

このサブスクリプション式によって、rep_key カラムの値がサブスクリプションで指定された値に一致するテーブルのローのみを、各営業担当者が受信します。

Policy テーブルの分割は切断です。複数のサブスクライバが共有するローはありません。

データに対してサブスクライブされた営業担当者と取り引きする顧客をリストした、**Customers** テーブルのロー

Customers テーブルには、データを分割するサブスクリプションで使用される営業担当者の値への参照はありません。パブリケーションでサブクエリを使用して、この問題に対処できます。

Customers テーブルの各ローが、SalesReps テーブルの複数のローと関連付けされ、複数の営業担当者のデータベースで共有されている場合があります。つまり、重複しているサブスクリプションがあります。

サブクエリを持つサブスクリプション式は、分割を定義するときに使用します。このアーティクルは、次のように定義されます。

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

Customers テーブルの分割は非切断です。複数のサブスクライバが共有するローがいくつかあります。

次の文では、完全なパブリケーションが作成されます。

```
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
   )
);
```

複数の値を返すパブリケーションのサブクエリ

Customers アーティクルのサブクエリは、その結果セットに単一のカラム (rep_key) を返します。ただし、特定の顧客を担当する営業担当者全員に対応して、複数のローを返す場合があります。サブスクリプション式に複数の値がある場合は、この値のいずれかに一致するサブスクリプションを持つすべてのサブスクライバにローがレプリケートされます。複数の値を持つサブスクリプション式の場合、テーブルの分割を重複にできます。

このセクションの内容:

1.2.12.1 サブスクライバ間でローを再割り当てする場合の参照整合性の維持

テーブルでローが追加、更新、または削除される場合、関連テーブルにレプリケートされない可能性があります。このシナリオはトリガを使用して処理します。

たとえば、顧客と営業担当者間の取引を取り消すには、Policy テーブルのローを削除します。この例の Policy テーブルの変更内容は、以前の営業担当者に正しくレプリケートされます。ただし、Customers テーブルは変更されません。このため、Customers テーブルに対する変更は、以前の営業担当者にレプリケートされません。

トリガがない場合は、Customers テーブルに不正確なデータを持つサブスクライバが残される可能性があります。Policy テーブルに新しいローを追加する場合にも、同様の問題が発生します。

トリガを使用した問題の解決法

この問題を解決するには、Policy テーブルを変更する処理の前に起動されるトリガを記述してください。これらの特別なトリガによって、データベーステーブルは変更されません。代わりに、サブスクライバデータベースでのデータ管理用に SQL Remote が使用するトランザクションログに、エントリが作成されます。

BEFORE INSERT トリガ

たとえば、次の文では、Policy テーブルに対する挿入を追跡する BEFORE INSERT トリガを作成し、リモートデータベースに適切なデータが含まれることを保証します。

```
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;
```

BEFORE DELETE トリガ

次の文では、Policy テーブルからの削除を追跡する BEFORE DELETE トリガを作成します。

```
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;
```

UPDATE PUBLICATION 文の SUBSCRIBE BY 句にはサブクエリが含まれており、このサブクエリは複数の値を返すことができます。

複数の値を返すサブクエリ

UPDATE PUBLICATION 文の SUBSCRIBE 句に含まれているサブクエリは UNION 式であり、複数の値を返すことができます。

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

- UNION 式の最初の部分は、Policy テーブルから取得した、顧客と取り引きする既存の営業担当者のセットです。サブスクリプションクエリの結果セットには、新しい営業担当者だけでなく、ローを受信する営業担当者全員が含まれている必要があります。
- UNION 式の 2 番目の部分は、INSERT 文から取得した、顧客と取り引きする新しい営業担当者の rep_key 値です。

BEFORE DELETE トリガ内のサブクエリは、複数の値を返します。

```
...
SELECT rep_key
FROM Policy
WHERE cust_key = OldRow.cust_key
```

```
AND rep_key <> OldRow.rep_key
...
```

- サブクエリは、Policy テーブルから rep_key の値を取得します。削除される値 (AND *rep_key* <> *OldRow.rep_key*) を除き、移動される顧客 (WHERE *cust_key* = *OldRow.cust_key*) と取り引きする営業担当者全員のプライマリキー値が、その値に含まれます。
  サブスクリプションクエリの結果セットには、削除に続くローを受信する営業担当者と一致した値のすべてが含まれている必要があります。

注記

- Customers テーブルのデータは、(プライマリキーの値などによって) 個々のサブスクライバと関連付けられることはなく、複数のサブスクライバ間で共有されます。このため、レプリケーションメッセージ間の複数のリモートサイトでデータが更新される可能性があり、レプリケーションの競合が発生することがあります。(特定のユーザだけに Customers テーブルの更新権を付与するなどの方法で) 権限を使用するか、データベースに RESOLVE UPDATE トリガを追加してプログラム上で競合を処理することによって、この問題に対処できます。
- Policy テーブル上の更新については、ここでは説明していません。そのような操作は避けるか、例で示したように、BEFORE INSERT と BEFORE DELETE の各トリガの機能を組み合わせる BEFORE UPDATE トリガを作成する必要があります。

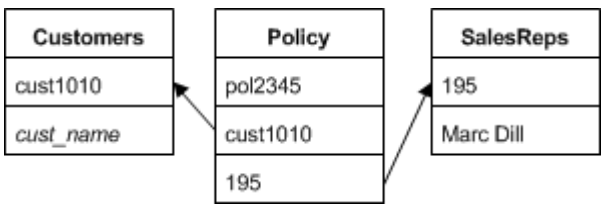
1.2.12.2 多対多の関係における subscribe_by_remote オプション

subscribe_by_remote オプションを On に設定すると、SUBSCRIBE BY 値が NULL または空の文字列であるローに対してリモートデータベースから操作が行われた場合、リモートユーザがローに対するサブスクリプションを作成するとみなされます。デフォルトでは、subscribe_by_remote オプションは On に設定されています。

subscribe_by_remote オプションは、これを使用しない場合にいくつかのパブリケーションで発生する問題を解決します。顧客が複数の営業担当者に所属できるため、次のパブリケーションでは、Customers テーブルのサブスクリプション式にサブクエリを使用します。

```
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
   ),
);
```

たとえば、営業担当者 Marc Dill が、新しい顧客との取り引きをデータに入力します。まず、Marc Dill は Customers テーブルに新しいローを挿入し、Policy テーブルにもローを挿入して新規の顧客を自分自身に割り当てます。



統合データベースでは、SQL Remote によって Customers ローの挿入が実行され、この挿入の実行時に SQL Anywhere によってトランザクションログにサブスクリプション値が記録されます。

あとで SQL Remote がトランザクションログをスキャンすると、サブスクリプション式からサブスクライバのリストが構築されます。顧客を割り当てた Policy テーブルでローがまだ適用されていないため、この場合 Marc Dill はリストされません。subscribe_by_remote が Off に設定されていると、この新しい顧客は DELETE 文として Marc Dill に送信されます。

subscribe_by_remote が On に設定されているかぎり、SQL Remote は、ローの所属先はローを挿入した営業担当者であるとみなし、INSERT 文は Marc Dill にレプリケートされません。また、レプリケーションシステムは影響を受けません。

subscribe_by_remote オプションが Off に設定されている場合は、必ず Policy ローを挿入してから Customers ローを挿入し、トランザクションの最後までチェックを延期することによって参照整合性違反を回避してください。

1.2.13 リモートデータベースのユニークな ID 番号

SQL Remote では、各リモートデータベースに異なる ID 番号を割り当てる必要があります。

ID 番号はさまざまな方法で作成して配布できます。テーブルに値を設定し、ユーザ名など、ユニークなプロパティに基づいて、各データベースに適切なローをダウンロードするのも 1 つの方法です。

global_database_id オプションの使用

各データベース内のパブリックオプション global_database_id は、ユニークな正の整数に設定してください。特定のデータベースのデフォルト値の範囲は、pn + 1 から p(n + 1) です。ここで、p は分割サイズ、n はパブリックオプション global_database_id の値を表します。たとえば、分割サイズを 1000、global_database_id を 3 に設定すると、範囲は 3001 ～ 4000 になります。

global_database_id が負ではない整数に設定されている場合、SQL Anywhere は次のルールを適用してデフォルト値を選択します。

- カラムに現在の分割の値が含まれていない場合、最初のデフォルト値は pn + 1 です。
- カラムに現在の分割の値が含まれていても、そのすべてが p(n + 1) 未満であれば、この範囲内でこれまで使用した最大値より 1 大きい値が次のデフォルト値になります。
- デフォルトのカラム値は、現在の分割以外のカラムの値の影響を受けません。つまり、pn + 1 より小さいか p(n + 1) より大きい数には影響されません。Mobile Link 同期を介して別のデータベースからレプリケートされた場合に、このような値が存在する可能性があります。

public オプション global_database_id がデフォルト値の 2147483647 に設定されると、NULL 値がカラムに挿入されます。NULL 値が許可されていない場合にローを挿入しようとすると、エラーが発生します。たとえば、テーブルのプライマリキーにカラムが含まれている場合に、この状況が発生します。

public オプション global_database_id は、負の値に設定できないため、選択された値は常に正になります。ID 番号の最大値を制限するのは、カラムデータ型と分割サイズだけです。

デフォルトの NULL 値は、分割で値が不足したときにも生成されます。この例では、別の分割からデフォルト値を選択できるように、データベースに global_database_id の新しい値を割り当ててください。カラムで NULL が許可されていない場合、

NULL 値を挿入しようとするとエラーが発生します。未使用の値が残り少ないことを検出し、このような状態を処理するには、GlobalAutoincrement タイプのイベントを作成します。

特定の分割で値が不足する場合は、新しいデータベース ID をそのデータベースに割り当てることができます。方法が適切なものであれば、新しいデータベース ID 番号を割り当てることができます。未使用のデータベース ID 値のプールを管理する方法も、その 1 つです。このプールは、プライマリキープールと同じ方法で管理されます。

分割で値が不足しそうな場合に、自動的にデータベース管理者へ通知する (またはその他のアクションを実行する) ようにイベントハンドラを設定できます。

このセクションの内容:

関連情報

SQL Remote のプライマリキープール [59 ページ]
デフォルト GLOBAL AUTOINCREMENT 宣言 [59 ページ]

1.2.13.1 global_database_id 値の設定 (SQL)

リモートデータベースの ID 番号を設定します。

コンテキスト

ID 番号は正の整数にします。

手順

global_database_id オプションの値を設定します。

たとえば、次の文ではデータベースの ID 番号が 20 に設定されます。

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

特定カラムの分割サイズが 5000 の場合、このデータベースのデフォルト値は 100001 ～ 105000 の範囲から選択されます。

結果

global_database_id 番号が設定されます。

1.2.13.2 データベース抽出時のデータベース ID 番号の設定 (SQL)

ストアドプロシージャを作成して、ユニークなデータベース ID 番号を設定するタスクを自動化できます。

前提条件

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

手順

1. *sp_hook_dbxtract_begin* という名前のストアドプロシージャを作成します。

   たとえば、user_id が 1001 であるリモートユーザ user2 のデータベースを抽出するには、次の文を実行します。

   ```
   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';
   ```

   この設定を使用すると、データベースが抽出されるたびに、global_database_id の値が毎回異なります。1 回目は 1001、2 回目は 2001 という具合です。
2. -v オプションを指定した抽出ユーティリティ (dbxtract) またはデータベース抽出ウィザードを実行して、リモートデータベースを抽出します。抽出ユーティリティは、次のタスクを実行します。
   a. #hook_dict という名前のテンポラリテーブルを作成し、次の内容を追加します。

name	value
extracted_db_global_id	抽出されるユーザ ID

sp_hook_dbxtract_begin プロシージャを作成してローの value カラムを修正する場合、その値は抽出されたデータベースの global_database_id オプションとして使用され、DEFAULT GLOBAL AUTOINCREMENT 値のプライマリキー値範囲の開始位置にマークを付けます。

- sp_hook_dbxtract_begin プロシージャを定義しない場合、抽出されるデータベースは、global_database_id が 101 に設定されます。
- sp_hook_dbxtract_begin プロシージャを定義しても、このプロシージャが #hook_dict のいずれのローも修正しない場合、global_database_id は 101 に設定されたままになります。

b. *sp_hook_dbxtract_begin* を呼び出します。
c. プロシージャフックのデバッグに役立つように、次の情報を出力します。
   - 検出されたプロシージャフック
   - プロシージャフックが呼び出される前の #hook_dict の内容
   - プロシージャフックが呼び出された後の #hook_dict の内容

結果

ユニークなデータベース ID 番号が設定されます。

関連情報

#Hook_dict テーブル [233 ページ]
SQL Remote システムプロシージャ [232 ページ]
抽出ユーティリティ (dbxtract) [219 ページ]

1.3 SQL Remote セキュリティに関する考慮事項

SQL Remote には、データ保護を支援する多数の機能が用意されています。

SYS_RUN_REPLICATION_ROLE システムロール

SYS_RUN_REPLICATION_ROLE システムロールを持つユーザを使用して SQL Remote Message Agent (dbremote) に接続してください。SYS_RUN_REPLICATION_ROLE システムロールが付与されたユーザは、SQL Remote または dbmlsync 以外からの接続では、このロールから継承される権限を行使できません。したがって、SYS_RUN_REPLICATION_ROLE システムロールを持つユーザのユーザ名とパスワードが公開されていても、セキュリティは損なわれません。そのデータベースで CONNECT より上の権限が付与されたユーザ名を使用しないかぎり、データベース内のデータにアクセスすることはできません。

データベースの暗号化

抽出ユーティリティ dbxtract の -ea オプションと -ek オプションを使用してデータベースを暗号化できます。dbxtract の -et オプションを使用してテーブルレベルでデータを暗号化することもできます。

-ea

新しいデータベースで使用する暗号化アルゴリズムを指定します。このオプションにより、新しいデータベースの暗号化に強力な暗号化アルゴリズムを選択できます。AES (デフォルト) または AES_FIPS (FIPS 認定のアルゴリズム) のどちらかを選択できます。AES_FIPS は個別のライブラリを使用するため、AES との互換性はありません。

セキュリティを強化するには、128 ビットの場合は AES、256 ビットの場合は AES256 の強力な暗号化を指定します。FIPS 認定の暗号化を使用するには、128 ビットの場合は AES_FIPS、256 ビットの場合は AES256_FIPS をそれぞれ指定してください。強力な暗号化を使用するためには、-ek または -ep オプションも指定する必要があります。

-ek

新しいデータベースで使用する暗号化キーを指定します。このオプションを使用すると、コマンドに暗号化キーを直接指定することで、強力に暗号化されたデータベースを作成できます。データベースの暗号化に使用されるアルゴリズムは、-ea オプションで指定した AES または AES_FIPS です。-ek オプションを指定して、-ea オプションを指定しないと、AES アルゴリズムが使用されます。

> ⚠ 警告
>
> 強力な暗号化が適用されたデータベースの場合、キーのコピーは必ず安全な場所に保管してください。暗号化キーがわからなくなった場合は、テクニカルサポートに依頼してもデータにはアクセスできません。アクセスできなくなったデータベースは、廃棄して、新しくデータベースを作成する必要があります。

-et

新しいデータベースのデータベーステーブル暗号化を有効にします。-ea オプションを指定せずに -et オプションを指定すると、AES アルゴリズムが使用されます。-et オプションを指定する場合、-ep または -ek も指定しないと操作は失敗します。新しいデータベースのテーブル暗号化設定を変更して、アンロードしているデータベースのテーブル暗号化設定とは異なるものに設定できます。

メッセージの暗号化

SQL Remote Message Agent (dbremote) は、単純難読化アルゴリズムを使用して、不意なスヌーピングからメッセージを保護します。しかし、この暗号化スキームでは、強力な暗号解読方法からメッセージを完全に保護することはできません。ネットワーク上のデータのセキュリティが問題である場合は、独自のカスタムエンコードスキームを実装できます。

1.4 SQL Remote システムの管理

統合データベースから SQL Remote システムを配備して管理します。

SQL リモートシステムを管理する場合は、SYS_REPLICATION_ADMIN_ROLE システムロールを持っている必要があります。

SQL Remote システムを配備して管理するには、次の手順に従います。

1. 統合データベースを設定します。
2. SQL Remote システムを確認してテストします。

SQL Remote システムを配備する前に、特に多数のリモートデータベースがある場合には、そのシステムを十分にテストしてください。

3. リモートデータベースを作成し、設計の配備を行います。
   統合データベースの DBA として、次の手順で SQL Remote を配備します。
   1. 各リモートユーザ用の SQL Anywhere データベースとそのデータの最初のコピーを作成し、そのサブスクリプションを開始します。
   2. 各リモートユーザのコンピュータに、SQL Anywhere データベースサーバ、リモートデータベース、SQL Remote、クライアントアプリケーションをインストールします。
4. SQL Remote Message Agent (dbremote) を実行してメッセージを交換します。
   メッセージを交換するには、次の手順を実行する必要があります。
   1. 統合データベースとリモートデータベースで SQL Remote Message Agent (dbremote) を継続モードとバッチモードのどちらで実行するかを決定します。
   2. ユーザ名、SQL Remote Message Agent (dbremote) 接続文字列、権限などが正しく、システムが適切に設定されていることを確認します。
5. メッセージを管理します。
   保証されたメッセージ配信システムを使用して、数多くのデータベース間でやりとりされるメッセージを管理します。
6. パフォーマンスを向上させます。
7. バックアップとリカバリの方式を実装します。
   統合データベースにバックアップとリカバリの方式を作成し、実装してください。
8. エラーを処理します。
9. 必要に応じて、ソフトウェアとデータベーススキーマをアップグレードします。

このセクションの内容:

SQL Remote システムバックアップ [134 ページ]
バックアップアクティビティを実行する前に、多くの考慮事項を認識している必要があります。

統合データベースの手動リカバリ (コマンドラインの場合) [141 ページ]
各トランザクションログをデータベースに適用して、統合データベースをリカバリします。

統合データベースの自動リカバリ (コマンドラインの場合) [143 ページ]
統合データベースを自動でリカバリします。

レプリケーションエラーのレポートと処理 [145 ページ]
SQL Remote には、発生する可能性のあるエラーを処理するための方法が多数用意されています。

SQL Remote セキュリティに関する考慮事項 [150 ページ]
SQL Remote には、データを保護するための多数の機能が用意されています。

アップグレードと再同期 [151 ページ]
さまざまな方法で SQL Remote データベースをアップグレードおよび再同期できます。

SQL Remote のパススルーモード [152 ページ]
パススルーモードを使用して、標準 SQL 文をそれらの実行が可能なリモートデータベースに渡します。

サブスクリプションの再同期 [155 ページ]
リモートデータベースを作成する場合は、統合データベースからスキーマとデータの両方を抽出し、リモートデータベースの構築に使用します。この処理により、各データベースにデータの初期コピーが確実に格納されます。

関連情報

SQL Remote システムの作成 [11 ページ]
SQL Remote Message Agent (dbremote) モード [93 ページ]

1.4.1 リモートデータベースの抽出

リモートユーザ用のデータベースを作成するには、統合データベースからリモートデータベースを抽出します。

データベース抽出ウィザードまたは抽出ユーティリティ (dbxtract) のいずれかを使用して、特定のリモートユーザ用のリモートデータベースを統合データベースから抽出できます。どちらの方法を使用しても、次の 1 つ以上のタスクを実行できます。

自動的にスキーマとデータを抽出して新規または既存のデータベースに直接再ロードする

これは、SQL Remote を学習する場合に適した方法です。この方法を使用した場合、データの中間コピーはディスク上に作成されません。この方法により、データのセキュリティが向上します。ただし、実装にかかる時間は長くなります。
スキーマとデータをファイルに抽出してから、新規または既存のデータベースにこれらをロードする

SQL Remote を配備する場合は、この方法をお奨めします。スキーマファイルを編集して、リモートデータベースの抽出と作成をカスタマイズできます。

効率を高める 1 つの方法は、複数のリモートデータベースを作成することです。

このセクションの内容:

リモートデータベースの自動抽出 (SQL Central) [84 ページ]
統合データベースを抽出して、スキーマとデータを新しいデータベースにリロードします。データの中間コピーはディスク上に作成されません。

関連情報

再ロードファイルへのリモートデータベースの抽出 [85 ページ]
複数のリモートデータベースの作成 (コマンドラインの場合) [90 ページ]

1.4.1.1 リモートデータベースの自動抽出 (SQL Central)

統合データベースを抽出して、スキーマとデータを新しいデータベースにリロードします。データの中間コピーはディスク上に作成されません。

前提条件

EXECUTE ANY PROCEDURE および SELECT ANY TABLE 権限が必要です。SELECT ANY TABLE 権限は SYS_REPLICATION_ADMIN_ROLE ロールに付属しています。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用して統合データベースに接続します。
2. ツール > *SQL Anywhere17* > データベース抽出 をクリックします。
3. 接続する統合データベースを選択して、抽出します。
4. プロンプトが表示されたら、新しいデータベースへの抽出と再ロードをクリックします。

   プロンプトが表示されたら、構造とデータを抽出をクリックします。
5. ウィザードの指示に従い、デフォルト値をそのまま使用します。

結果

適切なスキーマ、リモートユーザ、パブリケーション、サブスクリプション、トリガを含む新しいリモートデータベースが作成されます。デフォルトでは、統合データベースのデータがリモートデータベースに抽出され、サブスクリプションが開始されます。ただし、ウィザードでは、SQL Remote Message Agent が開始されないため、メッセージが交換されません。

関連情報

SQL Remote Message Agent (dbremote) [92 ページ]
再ロードファイルへのリモートデータベースの抽出 [85 ページ]
抽出ユーティリティ (dbxtract) [219 ページ]

1.4.2 再ロードファイルへのリモートデータベースの抽出

ほとんどの配備シナリオでは、リモートデータベースの抽出と作成をカスタマイズする必要があります。カスタム抽出を作成するには、スクリプトファイルと一連のテキストファイルへのデータベースの抽出を選択します。

データベースをファイルに抽出する場合は、作成するファイルを次のいずれかから選択します。

reload.sql という名前の **SQL** スクリプトファイル。このファイルには、リモートデータベーススキーマの構築に必要な文が含まれています。

抽出ユーティリティ (dbxtract) の -n オプションを使用します。

たとえば、次のコマンドを実行します。

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

一連のデータファイル。それぞれに、データベーステーブルの内容が含まれています。

抽出ユーティリティ (dbxtract) の -d オプションを使用します。データファイルが存在しない場合、それが `c:¥remote1` 内に作成されます。

たとえば、次のコマンドを実行します。

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

reload.sql ファイルとデータファイルの両方。

`reload.sql` とデータファイルが存在しない場合、それらが `c:¥remote1` 内に作成されます。`reload.sql` ファイルには、データファイルをロードする命令が含まれています。

たとえば、次のコマンドを実行します。

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

`reload.sql` ファイル

`reload.sql` ファイルには、データベーススキーマを構築するのに必要な SQL 文が記述され、次のオブジェクトを作成する文が含まれています。

- パブリッシャ、リモートユーザ、統合ユーザ
- パブリケーションとサブスクリプション

- メッセージ型
- テーブル
- ビュー
- トリガ
- プロシージャ

> **注記**
>
> リモートデータベースを作成する場合は、`reload.sql` を編集する必要があるかもしれません。抽出ユーティリティ (dbxtract) はリモートデータベースの準備を支援するためのものですが、すべての場合においてブラックボックスソリューションとはなりません。

このセクションの内容:

reload.sql ファイル (コマンドライン) からのリモートデータベースの作成 [87 ページ]
: 既存のデータベースをモデルとしてリモートデータベースを作成します。スキーマ、リモートユーザ、パブリケーション、サブスクリプション、トリガも生成されます。

reload.sql ファイルを編集するタイミング [88 ページ]
: リモートデータベースを作成するときは、必要に応じて `reload.sql` スクリプトファイルを編集してください。

多層階層システムのデータベースの抽出 [89 ページ]
: この例では、抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードを使用して多層階層内でデータベースを作成する方法について説明します。

複数のリモートデータベースの作成 (コマンドラインの場合) [90 ページ]
: 複数のデータベースの同時作成を効率化します。

関連情報

SQL Remote Message Agent (dbremote) [92 ページ]
リモートデータベースの抽出 [83 ページ]
リモートデータベースの自動抽出 (SQL Central) [84 ページ]
抽出ユーティリティ (dbxtract) [219 ページ]

1.4.2.1 reload.sql ファイル (コマンドライン) からのリモートデータベースの作成

既存のデータベースをモデルとしてリモートデータベースを作成します。スキーマ、リモートユーザ、パブリケーション、サブスクリプション、トリガも生成されます。

手順

1. 抽出ユーティリティ (dbxtract) を使用して、データベーススキーマとデータをファイルに抽出します。たとえば、次のコマンドを実行します。

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

   デフォルトでは、指定されたリモートユーザのサブスクリプションが自動的に開始されます。
2. 必要に応じて、reload.sql を編集します。
3. 空の SQL Anywhere データベースを作成します。

   たとえば、次のコマンドを実行します。

   ```
   dbinit -dba DBA,passwd c:¥remote1¥rem1.db
   ```
4. Interactive SQL からデータベースに接続して、reload.sql スクリプトファイルを実行します。

   たとえば、次の文を実行します。

   ```
   READ remote1¥reload.sql
   ```

   適切なスキーマ、リモートユーザ、パブリケーション、サブスクリプション、トリガを含む新しいリモートデータベース rem1.db が作成されます。ただし、抽出ユーティリティ (dbxtract) では、SQL Remote Message Agent が開始されないため、メッセージが交換されません。

結果

リモートデータベースが作成されます。

関連情報

抽出ユーティリティ (dbxtract) [219 ページ]

1.4.2.2 reload.sql ファイルを編集するタイミング

リモートデータベースを作成するときは、必要に応じて reload.sql スクリプトファイルを編集してください。

たとえば、次の場合に reload.sql ファイルを編集します。

リモートデータベースへのレプリケートされないテーブルの追加

レプリケーションに関係しないテーブルであれば、統合データベースにないテーブルをリモートデータベースに追加できます。抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードは、レプリケートされないテーブルを統合データベースから抽出できません。

データベースを抽出した後、reload.sql を編集してこのようなテーブルを追加してください。

プロシージャ、トリガ、ビューの抽出

デフォルトでは、抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードは、すべてのストアドプロシージャ、トリガ、ビューをデータベースから抽出します。ビューとプロシージャには、リモートサイトで必要なものと必要でないものがあります。たとえば、プロシージャには、データベースの、リモートサイトに含まれない部分を参照するものがあります。

データベースを抽出した後、reload.sql を編集して、不必要なプロシージャ、トリガ、ビューを削除してください。

多層システムでの抽出ユーティリティ (dbxtract) の使用

多層システム内に第 2 レベルから第 n レベルのデータベースを作成するには、抽出ユーティリティ (dbxtract) またはデータベース抽出ウィザードを使用します。

デフォルトでは、dbxtract とデータベース抽出ウィザードは、すべてのストアドプロシージャ、トリガ、ビューをデータベースから抽出します。ビューとプロシージャには、リモートサイトで必要なものと必要でないものがあります。たとえば、プロシージャには、データベースの、リモートサイトに含まれない部分を参照するものがあります。

データベースを抽出した後、reload.sql を編集して、不必要なプロシージャ、トリガ、ビューを削除してください。

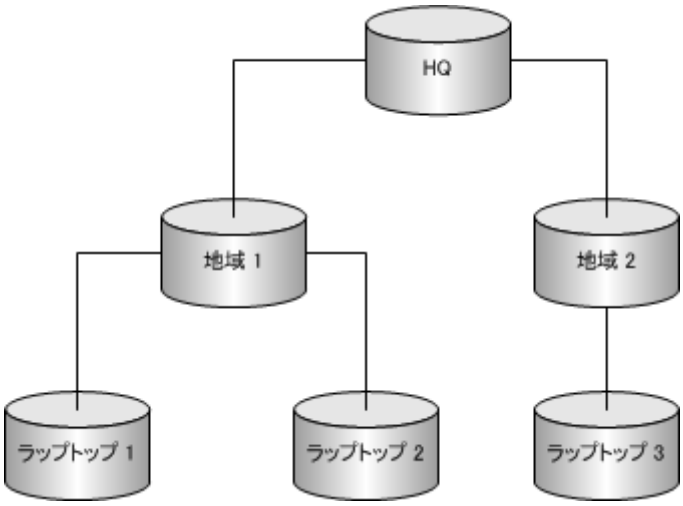
関連情報

複数のリモートデータベースの作成 (コマンドラインの場合) [90 ページ]
多層階層システムのデータベースの抽出 [89 ページ]

1.4.2.3 多層階層システムのデータベースの抽出

この例では、抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードを使用して多層階層内でデータベースを作成する方法について説明します。

多層配置での抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードの役割について理解するために、3 層の SQL Remote システムについて検討してみます。次の図は、このシステムを表したものです。



3 層のシステムにリモートデータベースを作成するには、次の手順に従います。

1. 最上位レベルの統合データベース HQ で抽出ユーティリティ (dbxtract) を使用して、第 2 レベルのデータベース Region 1 と Region 2 を作成します。
2. 第 2 レベルのデータベース Region 1 と Region 2 で抽出ユーティリティ (dbxtract) を使用して、ユーザ Laptop 1、Laptop 2、Laptop 3 用の第 3 レベルのデータベースを作成します。第 2 レベルのデータベースは、第 1 レベルのデータベース HQ のリモートデータベースであり、第 3 レベルのデータベース Laptop 1、Laptop 2、Laptop 3 の統合データベースです。

多層階層システムでのデータベースの再抽出

最上位レベルの統合データベースから第 2 レベルのデータベース用のスキーマを再抽出する必要がある場合、抽出ユーティリティ (dbxtract) によってリモートユーザ (Laptop 1、Laptop 2、Laptop 3) がそのサブスクリプションと権限とともに削除されます。このため、これらの第 3 レベルのユーザとそのサブスクリプションを手動で再作成してください。

最上位レベルの統合データベースから第 2 レベルのデータベースのデータのみを再抽出する必要がある場合、抽出ユーティリティ (dbxtract) はリモートユーザに影響しません。

完全に修飾されたパブリケーション定義

完全に修飾されたパブリケーション定義には、WHERE 句と SUBSCRIBE BY 句があります。ほとんどの場合、完全に修飾されたパブリケーション定義をリモートデータベース用に抽出する必要はありません。通常、リモートデータベースは、すべてのローをレプリケートし、統合データベースに戻します。

関連情報

複数のリモートデータベースの作成 (コマンドラインの場合) [90 ページ]
抽出ユーティリティ (dbxtract) [219 ページ]

1.4.2.4 複数のリモートデータベースの作成 (コマンドラインの場合)

複数のデータベースの同時作成を効率化します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. 統合データベースのコピーを作成し、統合データベースからリモートユーザのサブスクリプションを開始します。次に例を示します。
   a. 統合データベースと SQL Remote Message Agent (実行されている場合) を停止します。
   b. dbeng17 と異なるサーバ名を使用して統合データベースのローカルコピーを開始して、他のプロセスがローカルコピーに接続していないことを確認します。
   c. サブスクリプションを開始します。

      統合データベースのコピー作成と同時に、サブスクリプションを開始する必要があります。データベースのコピー作成とサブスクリプション開始の間に発生したオペレーションはすべて失われ、これが原因でリモートデータベースでのエラーが発生する恐れがあります。統合データベースでサブスクリプションを開始すると、サブスクライバのデータベースがまだ存在しない場合でも、メッセージをパッケージしてサブスクライバに送信できます。

      1 つのトランザクション内でいくつかのサブスクリプションを開始するには、REMOTE RESET 文を使用します。
   d. ただちに統合データベースを停止します。
   e. 統合データベースをコピーします。

      デフォルトでは、抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードはともに、独立性レベル 3 で実行されます。この独立性レベルでは、抽出されたデータベース内のデータは、データベースサーバのデータと一致します

が、他のユーザがデータベースを使用できなくなる場合があります。統合データベースのコピーに対してリモートデータベースを抽出してください。

f. 統合データベースを再起動します。統合データベースで SQL Remote Message Agent が実行されていた場合は、SQL Remote Message Agent も再起動します。

2. 統合データベースのコピーからリモートデータベースのスキーマを抽出します。データベースはコピーであるため、ロックと同時性の問題は発生しません。ただし、多数のリモートデータベースがある場合は、この処理に時間がかかることがあります。

   リモートデータベースのスキーマを抽出する場合、次のオプションを選択します。

   a. リモートデータベースのスキーマのみを抽出します。

      デフォルトでは、抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードはともに、各ユーザ用のスキーマとデータを含め、一度にデータベースを 1 つだけ処理します。ただし、ほとんどの配備シナリオでは、各リモートデータベースで使用するデータは異なりますが、スキーマは同じです。抽出ユーティリティ (dbxtract) またはデータベース抽出ウィザードを使用してユーザごとにスキーマとデータの両方を抽出すると、同じスキーマが繰り返し抽出されることになります。

   b. プライマリキーを基準にデータを順序付けます。

      デフォルトでは、各テーブルのデータはプライマリキーを基準に順序付けられます。抽出ユーティリティ (dbxtract) の-u オプションを使用してデータがプライマリキーを基準に順序付けられると、リモートデータベースへのデータのロード処理が高速になります。

3. `reload.sql` ファイルを使用して空のリモートデータベースを作成します。このデータベースファイルをコピーして必要な数のリモートデータベースを作成します。
4. リモートデータベースごとに、各リモートユーザに固有の SQL Remote 定義を定義します。
5. リモートユーザごとに、抽出ユーティリティ (dbxtract) の-d オプションを使用して統合データベースからユーザに対応するデータのみを抽出します。
6. 各リモートユーザのデータを対応するリモートデータベースにロードします。

   各リモートデータベースが作成されると、その情報はライブ統合データベースより古いものになります。

   しかし、SQL Remote Message Agent (dbremote) を実行すると、各ユーザはライブ統合データベースから送信されたメッセージを受信して適用し、そのリモートデータベースを最新の情報に更新できます。

結果

リモートデータベースが作成されます。

関連情報

reload.sql ファイルを編集するタイミング [88 ページ]
SQL Remote Message Agent (dbremote) [92 ページ]
ユーザ権限 [22 ページ]
サブスクリプションの開始 (SQL Central) [159 ページ]
多層階層システムのデータベースの抽出 [89 ページ]

抽出ユーティリティ (dbxtract) [219 ページ]

1.4.3 SQL Remote Message Agent (dbremote)

SQL Remote Message Agent (dbremote) は、SQL Remote レプリケーションの主要コンポーネントです。SQL Remote Message Agent (dbremote) をインストールし、システム内の各データベースで実行してください。

SQL Remote Message Agent (dbremote) では、メッセージの送受信処理を行います。

dbremote を実行するには、SYS_RUN_REPLICATION_ROLE システムロールが必要です。

SQL Remote Message Agent の機能は次のとおりです。

メッセージ送信時の **SQL Remote Message Agent (dbremote)** のタスク

- 各パブリッシャデータベースのトランザクションログをスキャンして、トランザクションログのエントリをサブスクライバへのメッセージに変換します。
- サブスクライバにメッセージを送信します。
- SQL Remote Message Agent (dbremote) がメッセージの再送要求を受信した場合は、要求を作成したデータベースにメッセージを再送します。
- システムテーブルのメッセージ情報を保持し、保証されたメッセージ配信システムを管理します。

メッセージ受信時の **SQL Remote Message Agent (dbremote)** のタスク

- 受信メッセージを処理して、データベースに適切な順序で適用します。
- 欠落しているメッセージの再送を要求します。
- システムテーブルのメッセージ情報を保持し、保証されたメッセージ配信システムを管理します。

接続

SQL Remote Message Agent (dbremote) は、データベースサーバへの接続をいくつか使用します。それらは次のとおりです。

汎用的な接続

SQL Remote Message Agent (dbremote) の起動中は常に接続されています。
トランザクションログをスキャンするための接続

スキャンのフェーズ中のみ接続されています。
トランザクションログスキャンスレッドからコマンドを実行するための接続

スキャンのフェーズ中のみ接続されています。
同期サブスクリプション要求を処理するための接続

送信フェーズ中のみ接続されています。
各ワーカスレッドのための接続

受信フェーズ中のみ接続されています。

このセクションの内容:

SQL Remote Message Agent (dbremote) モード [93 ページ]
SQL Remote Message Agent (dbremote) は、継続モードまたはバッチモードのいずれかで実行できます。

継続モードでの SQL Remote Message Agent の実行 (dbremote) [94 ページ]
各リモートユーザのプロパティを設定することによって、SEND AT または SEND EVERY の頻度で指定された時刻にメッセージを送信します。

バッチモードでの SQL Remote Message Agent の実行 (dbremote) [97 ページ]
SQL Remote Message Agent (dbremote) を使用して、受信メッセージの受信と処理を行い、トランザクションログをスキャンし、送信メッセージの作成と送信を行います。

Mac OS X での SQL Remote Message Agent (dbremote) の実行 [98 ページ]
SyncConsole を使用して Mac OS X で SQL Remote Message Agent (dbremote) を起動します。

UNIX での SQL Remote Message Agent (dbremote) の実行 [99 ページ]
UNIX プラットフォーム上では、-ud オプションを指定して、SQL Remote Message Agent (dbremote) をデーモンとして実行します。

関連情報

メッセージを送信するタスク [107 ページ]
メッセージを受信するタスク [101 ページ]

1.4.3.1 SQL Remote Message Agent (dbremote) モード

SQL Remote Message Agent (dbremote) は、継続モードまたはバッチモードのいずれかで実行できます。

継続モード

継続モードでは、SQL Remote Message Agent (dbremote) は各リモートユーザの送信頻度プロパティで指定された時刻に、定期的にメッセージを送信します。メッセージを送信していないときは、メッセージが到着すると受信します。

継続モードは、メッセージが随時送受信される統合データベースにおいて有用です。負荷を分散させて迅速なレプリケーションを確実にするためです。

バッチモード

バッチモードでは、SQL Remote Message Agent (dbremote) は受信メッセージを受信して処理し、トランザクションログを 1 回スキャンし、出力メッセージを作成して送信した後、停止します。

バッチモードは、不定期に接続するリモートデータベースにおいて有用です。接続したときにだけ、統合データベースとメッセージを交換できるためです。たとえば、リモートデータベースがメインネットワークにダイヤルアップするようなときです。

このセクションの内容:

SQL Remote Message Agent (dbremote) の稼働条件 [94 ページ]
SQL Remote には、高い柔軟性があります。システム内では、複数のデバイスと複数のオペレーティングシステム上で両方のモードの SQL Remote Message Agent (dbremote) を実行できます。

関連情報

継続モードでの SQL Remote Message Agent の実行 (dbremote) [94 ページ]
バッチモードでの SQL Remote Message Agent の実行 (dbremote) [97 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]

1.4.3.1.1 SQL Remote Message Agent (dbremote) の稼働条件

SQL Remote には、高い柔軟性があります。システム内では、複数のデバイスと複数のオペレーティングシステム上で両方のモードの SQL Remote Message Agent (dbremote) を実行できます。

ただし、SQL Remote には、次の稼働条件があります。

SYS_RUN_REPLICATION_ROLE システムロール

SQL Remote Message Agent (dbremote) は、SYS_RUN_REPLICATION_ROLE システムロールを持つユーザが実行する必要があります。

システム内にある各 SQL Remote Message Agent (dbremote) のメッセージの最大長は、すべて同じ値にする

このメッセージ長は、オペレーティングシステムのメモリ割り当て制限によって制限されることがあります。制限より長い受信メッセージは、矛盾したメッセージとして削除されます。デフォルト値は 50000 バイトです。このメッセージ長は、SQL Remote Message Agent (dbremote) の -l オプションを使用して変更できます。

1.4.3.2 継続モードでの SQL Remote Message Agent の実行 (dbremote)

各リモートユーザのプロパティを設定することによって、SEND AT または SEND EVERY の頻度で指定された時刻にメッセージを送信します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

統合データベースなどで、SQL Remote Message Agent (dbremote) を継続モードで実行する場合は、すべての REMOTE ユーザが送信頻度を指定していることを確認してください。

-l オプションで定義されているメッセージの最大長は、システム内のすべてのデータベースで同じである必要があります。

コンテキスト

通常、統合データベースは、継続モードで実行されます。継続モードでは、SQL Remote Message Agent (dbremote) は SEND AT または SEND EVERY プロパティで指定された時刻に、メッセージを送信します。

手順

1. すべての REMOTE ユーザが SEND AT または SEND EVERY の頻度を指定していることを確認します。
2. -b オプションを使用しないで、SQL Remote Message Agent (dbremote) を起動します。

   Windows では、SQL Remote Message Agent (dbremote) の名称は `dbremote.exe` です。UNIX では、この名前は dbremote です。Mac OS X では、SyncConsole を使用して SQL Remote Message Agent (dbremote) を起動することもできます。

   たとえば、次のコマンドではデータベースファイル `c:¥mydata.db` で dbremote を継続モードで実行します。接続にはユーザ名 ManagerSteve とパスワード passwd を使用しています。

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

結果

SQL Remote Message Agent (dbremote) は、継続モードで実行するように設定されています。

このセクションの内容:

送信頻度 [95 ページ]
統合データベースなどで、SQL Remote Message Agent (dbremote) を継続モードで実行する場合は、すべての REMOTE ユーザが送信頻度を指定していることを確認してください。

メッセージの送信頻度の設定 [96 ページ]
統合データベースとリモートデータベースで SQL Remote Message Agent (dbremote) を継続モードとバッチモードのどちらで実行するかを決定します。

1.4.3.2.1 送信頻度

統合データベースなどで、SQL Remote Message Agent (dbremote) を継続モードで実行する場合は、すべての REMOTE ユーザが送信頻度を指定していることを確認してください。

継続モードでは、SQL Remote Message Agent (dbremote) は SEND AT または SEND EVERY プロパティで指定された時刻に、メッセージを送信します。

SQL Remote Message Agent (dbremote) では、次の送信頻度の値をサポートしています。

SEND EVERY

メッセージを送信する間隔の待機時間を指定します。

SEND EVERY を設定したユーザにメッセージを送信すると、同じ頻度が設定されているすべてのユーザにメッセージが同時に送信されます。たとえば、12 時間ごとに更新内容を受信するリモートユーザ全員に、時間をずらすことなく同時に更新内容が送信されます。これにより、SQL Anywhere のトランザクションログを処理する回数を減らすことができます。ユーザに固有の頻度を設定することは、できるだけ避けてください。

送信頻度を時、分、秒 (HH:MM:SS フォーマット) で指定できます。

SEND AT

メッセージを送信する時刻を指定します。

毎日、指定した時刻に更新内容が送信されます。ユーザに固有の時刻を設定し、送信時間をずらすことは、できるだけ避けてください。また、データベースがビジーでない時刻を選択してください。

デフォルト設定 **(SEND** 句なし**)**

ユーザが SEND AT 句または SEND EVERY 句を指定していない場合は、SQL Remote Message Agent (dbremote) はバッチモードで起動し、起動のたびにメッセージを送信して停止します。

非常に頻繁なメッセージの送信

頻繁にメッセージを送信すると、小さいメッセージが送信されることが多くなります。メッセージの送信頻度を下げると、より多くの命令を 1 つのメッセージにグループ化できます。お使いのメッセージシステムで小さいメッセージを大量に送信しなければならない場合は、送信間隔をあまり短くしないでください。

関連情報

バッチモードでの SQL Remote Message Agent の実行 (dbremote) [97 ページ]

1.4.3.2.2 メッセージの送信頻度の設定

統合データベースとリモートデータベースで SQL Remote Message Agent (dbremote) を継続モードとバッチモードのどちらで実行するかを決定します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

継続モードでは、SQL Remote Message Agent (dbremote) は SEND AT または SEND EVERY プロパティで指定された時刻に、メッセージを送信します。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. *SQL Remote* ユーザをダブルクリックします。
3. ユーザを右クリックし、プロパティをクリックします。
4. *SQL Remote* タブをクリックします。
5. 次の間隔で送信または毎日次の時刻に送信のいずれかをクリックし、時間を指定します。*OK* をクリックします。

結果

頻度が設定されます。

1.4.3.3 バッチモードでの SQL Remote Message Agent の実行 (dbremote)

SQL Remote Message Agent (dbremote) を使用して、受信メッセージの受信と処理を行い、トランザクションログをスキャンし、送信メッセージの作成と送信を行います。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。-l オプションで定義されているメッセージの最大長は、システム内のすべてのデータベースで同じである必要があります。

手順

1. リモートのプロパティに SEND AT オプションも SEND EVERY オプションも設定していないリモートユーザが最低 1 つあることを確認します。

   リモートユーザのすべてに SEND AT 句または SEND EVERY 句が定義されており、メッセージを送受信してから停止する必要がある場合は、-b オプションを使用して SQL Remote Message Agent (dbremote) を起動してください。

2. SQL Remote Message Agent (dbremote) を起動します。

   Windows では、SQL Remote Message Agent (dbremote) の名称は `dbremote.exe` です。UNIX では、この名前は dbremote です。Mac OS X では、*SyncConsole* を使用して SQL Remote Message Agent (dbremote) を起動することもできます。

   たとえば、次のコマンドではデータベースファイル `c:¥mydata.db` で dbremote をバッチモードで実行します。接続にはユーザ名 ManagerSteve とパスワード passwd を使用しています。

   ```
   dbremote -b -c "UID=ManagerSteve;PWD=passwd;DBF=c:¥mydata.db"
   ```

   SQL Remote Message Agent (dbremote) は受信メッセージを受信して処理し、トランザクションログを 1 回スキャンし、出力メッセージを作成して送信した後、停止します。

結果

SQL Remote Message Agent (dbremote) が起動します。

関連情報

SQL Remote Message Agent (dbremote) の稼働条件 [94 ページ]
UNIX での SQL Remote Message Agent (dbremote) の実行 [99 ページ]
Mac OS X での SQL Remote Message Agent (dbremote) の実行 [98 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]

1.4.3.4 Mac OS X での SQL Remote Message Agent (dbremote) の実行

SyncConsole を使用して Mac OS X で SQL Remote Message Agent (dbremote) を起動します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。-l オプションで定義されているメッセージ長は、システム内のすべてのデータベースで同じである必要があります。

コンテキスト

dbremote ユーティリティを使用して、Mac OS X で SQL Remote Message Agent を起動することもできます。

手順

1. [Finder] で、/Applications/SQLAnywhere*17* に移動します。
2. *SyncConsole* をダブルクリックします。
3. ファイル > 新規 > *SQL Remote* をクリックします。

   クライアントオプションのウィンドウが表示されます。
4. dbremote の接続情報を指定します。

   たとえば、次の接続パラメータでは、SQL Anywhere サンプルデータベースの ODBC データソースが使用されます。

   ```
   DSN="SQL Anywhere 17 Demo;PWD=sql"
   ```

結果

SQL Remote Message Agent (dbremote) が起動します。

関連情報

SQL Remote Message Agent (dbremote) の稼働条件 [94 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]

1.4.3.5 UNIX での SQL Remote Message Agent (dbremote) の実行

UNIX プラットフォーム上では、-ud オプションを指定して、SQL Remote Message Agent (dbremote) をデーモンとして実行します。

SYS_RUN_REPLICATION_ROLE システムロールが必要です。-l オプションで定義されているメッセージの最大長は、システム内のすべてのデータベースで同じである必要があります。

関連情報

SQL Remote Message Agent (dbremote) の稼働条件 [94 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]

1.4.4 SQL Remote パフォーマンス

SQL Remote は、データベースサーバと SQL Remote Message Agent (dbremote) の間でサブスクリプションとパブリケーションについての通知の負荷を分割し、ネットワークパフォーマンスを管理します。

テーブルのローに挿入、削除、または更新が行われるたびに、ローに対してサブスクライブされたユーザにメッセージが作成されます。また、更新の場合はサブスクリプション式が変更されることがあるため、あるサブスクライバには削除文、別のサブスクライバには更新文、また別のサブスクライバには挿入文が送信されます。

データベースサーバ

データベースサーバは、パブリケーションを処理します。

SQL Remote Message Agent (dbremote)

SQL Remote Message Agent (dbremote) は、サブスクリプションを処理します。

SQL Remote Message Agent (dbremote) は、トランザクションログから評価済みのサブスクリプション式またはサブスクリプションカラムへのエントリを読み込み、更新前の値と更新後の値をパブリケーションの個々のサブスクライバのサブスクリプション値と照合します。SQL Remote Message Agent (dbremote) は、このようにして適切なオペレーションを個々のサブスクライバに送信します。

サブスクライバの数が非常に多くてもデータベースサーバのパフォーマンスは低下しませんが、SQL Remote Message Agent (dbremote) のパフォーマンスは低下する場合があります。サブスクリプション値を大量のサブスクリプション値と照合する作業と、メッセージを送信する作業は、大きな負荷となる場合があります。

このセクションの内容:

メッセージを受信するタスク [101 ページ]
SQL Remote Message Agent (dbremote) は、メッセージの受信時に一連のタスクを実行します。

メッセージ受信時のパフォーマンスのトラブルシューティング [102 ページ]
SQL Remote システムの全体のスループットにおける主なボトルネックは、一般的に、多くのリモートデータベースからメッセージを受信して、それをデータベースに適用することです。

メッセージを送信するタスク [107 ページ]
SQL Remote Message Agent (dbremote) は、メッセージの送信時に一連のタスクを実行します。

メッセージを送信する場合のパフォーマンス [108 ページ]
メッセージの送信におけるパフォーマンス上の主な問題は、あるサイトにデータが入力されてから他のサイトに表示されるまでのターンアラウンドタイムです。

関連情報

トランザクションログベースのレプリケーション [37 ページ]

1.4.4.1 メッセージを受信するタスク

SQL Remote Message Agent (dbremote) は、メッセージの受信時に一連のタスクを実行します。

受信メッセージのポーリング

データベースに着信した新しいメッセージをチェックするために、SQL Remote Message Agent (dbremote) によって新しいメッセージがポーリングされます。

メッセージの読み込み

メッセージが着信すると、SQL Remote Message Agent (dbremote) によって読み込まれ、適用可能になるまでキャッシュメモリ内に格納されます。

欠落しているメッセージがあり、SQL Remote Message Agent (dbremote) が継続モードで実行されている場合、SQL Remote Message Agent (dbremote) は、後続のポーリングでメッセージの着信を待機します。SQL Remote Message Agent (dbremote) が待機するポーリング回数は、その待機時間と呼ばれ、-rp オプションで指定されます。

- SQL Remote Message Agent (dbremote) の待機時間が切れる前に、欠落していたメッセージが着信した場合、このメッセージは正しい順序でキャッシュに追加されます。
- 欠落しているメッセージが着信しないまま、SQL Remote Message Agent (dbremote) の待機時間が切れた場合、SQL Remote Message Agent (dbremote) は、パブリッシャデータベースからのメッセージの再送要求を送信します。

キャッシュメモリ使用量を超えるまで、メッセージは継続して読み込まれ、キャッシュに追加されます。-m オプションで指定したメモリ使用量を超過すると、メッセージは削除されます。

メッセージの適用

SQL Remote Message Agent (dbremote) は、サブスクライバデータベースに正しい順序でメッセージを適用します。

サブスクライバデータベースへのメッセージ適用の確認メッセージの待機

メッセージが受信されてサブスクライブされたデータベースに適用されると、パブリッシャには確認メッセージが返送されます。パブリッシャの SQL Remote Message Agent (dbremote) は確認メッセージを受信すると、システムテーブル内で確認メッセージを追跡します。

関連情報

新しいメッセージをチェックするための間隔調整のポーリング [102 ページ]
受信メッセージのキャッシュによるスループットの調整 [103 ページ]
メッセージを再送するための要求調整 [104 ページ]
データベースワーカスレッド [106 ページ]
保証されたメッセージ配信システム [111 ページ]

1.4.4.2 メッセージ受信時のパフォーマンスのトラブルシューティング

SQL Remote システムの全体のスループットにおける主なボトルネックは、一般的に、多くのリモートデータベースからメッセージを受信して、それをデータベースに適用することです。

このラグタイムを短縮するには、SQL Remote Message Agent (dbremote) を継続モードで実行している場合に、次の変数を調整します。

- SQL Remote Message Agent (dbremote) が受信メッセージを確認する頻度。
- SQL Remote Message Agent (dbremote) が送信するメッセージの格納に使用するメモリ量。
- SQL Remote Message Agent (dbremote) が順序不整合のメッセージの再送を要求するまでメッセージの着信を待機する時間。
- 受信したメッセージの処理に使用されるワーカスレッドの数。

このセクションの内容:

1.4.4.2.1 新しいメッセージをチェックするための間隔調整のポーリング

データベースに着信した新しいメッセージをチェックするために、SQL Remote Message Agent (dbremote) によって新しいメッセージがポーリングされます。このポーリングの頻度は調整できます。

ポーリングが終了してから次回のポーリングが開始されるまでのポーリング間隔のデフォルトは、1 分です。ポーリング間隔は、-rd オプションを使用して設定できますが、通常はデフォルトで十分です。

ポーリング間隔の延長

秒単位の値を使用することによって、ポーリング頻度を上げることができます。たとえば、次のコマンドでは、30 秒ごとにポーリングが行われます。

```
dbremote -c "DSN=SQL Anywhere 17 Demo;PWD=sql" -rd 30s
```

一般的には、メッセージへの素早い応答が要求される特別な場合でないかぎり、ポーリング間隔は短くしないでください。間隔を非常に短く設定すると、システムのスループット全体に悪影響を及ぼすことがあります。それは次のような理由によります。

- キューにメッセージがないときもポーリングするため、リソースが無駄になることがあります。たとえば、電子メールを使用している場合は、メールサーバをポーリングするごとにメッセージシステムに負荷がかかります。あまり頻繁にポーリングを行うと、メッセージシステムに悪影響を及ぼし、利点はまったくありません。
- 再送要求によってシステムに過負荷がかかる場合があります。ポーリング間隔を調整する場合は、SQL Remote Message Agent (dbremote) の待機時間も調整してください。待機時間とは、SQL Remote Message Agent (dbremote) が順序不整合のメッセージの再送を要求するまでメッセージの着信を待機するポーリング回数のことです。

ポーリング間隔の短縮

ポーリングの頻度を下げることができます。次のコマンドでは、ポーリング間隔を 5 分に設定します。

```
dbremote -c "DSN=SQL Anywhere 17 Demo;PWD=sql" -rd 5
```

長めのポーリング間隔を設定すると、システムのメッセージスループット全体が向上しますが、個別のメッセージの適用にかかる時間が長くなることがあります。たとえば、メッセージの受信頻度と比較して受信メッセージのポーリング間隔が長すぎる場合、キューに入っているメッセージを終了して、処理されるまで待機できます。

関連情報

受信メッセージのキャッシュによるスループットの調整 [103 ページ]
メッセージを再送するための要求調整 [104 ページ]
データベースワーカスレッド [106 ページ]
メッセージを送信する場合のパフォーマンス [108 ページ]

1.4.4.2.2 受信メッセージのキャッシュによるスループットの調整

メッセージが着信すると、SQL Remote Message Agent (dbremote) はメッセージを読み込み、メッセージが適用されるまでキャッシュメモリ内に格納します。

このようなメッセージのキャッシュによって、次の状態を回避できます。

- 規模の大きいシステムにおいてパフォーマンスを低下させるおそれがある、メッセージシステムからの順序不整合のメッセージの再読み込み。メッセージのキャッシュは、メッセージを WAN (リモートアクセスサービスやモデム経由の POP3 など) 上で読み込む場合に役立ちます。
- メッセージを読み込む (単一スレッドタスク) データベースワーカスレッド間の競合。メッセージの内容がキャッシュされるためです。

メッセージをキャッシュする方法

次のいずれかの状況が生じると、SQL Remote Message Agent (dbremote) によって適用されるまで、メッセージはメモリ内に格納されます。

- トランザクションが非常に大きく、マルチパートのメッセージを必要とします。
- メッセージが順序不整合の状態で着信します。

メッセージキャッシュサイズの指定

SQL Remote Message Agent (dbremote) の -m オプションを使用して、メッセージキャッシュのサイズを指定します。-m オプションは、SQL Remote Message Agent (dbremote) がメッセージの格納に使用するメモリの最大容量を指定します。使用できるサイズは、n (バイト)、nK、nM で指定します。デフォルトは 2048 K (2 M) です。指定したキャッシュメモリ使用量を超過すると、メッセージは削除されます。

-m オプションは、単一の統合データベースと多数のリモートデータベースを使用する場合に役立ちます。

例

次のコマンドは、12 MB のメモリをメッセージキャッシュとして使用して SQL Remote Message Agent (dbremote) を起動します。

```
dbremote -c "DSN=SQL Anywhere 17 Demo;PWD=sql" -m 12M
```

関連情報

新しいメッセージをチェックするための間隔調整のポーリング [102 ページ]
メッセージを再送するための要求調整 [104 ページ]
データベースワーカスレッド [106 ページ]
メッセージを送信する場合のパフォーマンス [108 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]

1.4.4.2.3 メッセージを再送するための要求調整

メッセージがシーケンスから欠落している場合、SQL Remote Message Agent (dbremote) は、指定されたポーリング回数を待機してから、欠落しているメッセージの再送を要求します。

SQL Remote Message Agent (dbremote) が待機するポーリング回数は、その待機時間と呼ばれます。デフォルトでは、SQL Remote Message Agent (dbremote) の待機時間は 1 です。

SQL Remote Message Agent (dbremote) の待機時間が 1 であり、メッセージ 6 を受信するはずが、メッセージ 7 を受信した場合、SQL Remote Message Agent (dbremote) は何もしません。代わりに、SQL Remote Message Agent

(dbremote) は、次のポーリングの結果を待機します。次のポーリングの後、メッセージ 6 が欠落したままである場合、SQL Remote Message Agent (dbremote) はメッセージ 6 の再送要求を発行します。

再送待機時間の延長

ポーリング間隔が非常に短く、メッセージの到着順を維持しないメッセージシステムを使用しているとします。一般的には、順序不整合のメッセージが到着するのは 2 ～ 3 回のポーリングが完了してからになります。この例では、-rp オプションを使用して SQL Remote Message Agent (dbremote) の待機時間を長くし、不要な再送要求が大量に送信されないようにしてください。多くの場合、-rp オプションは、ポーリング間隔を設定する -rd オプションとともに使用されます。

例

user1 と user2 という 2 人のリモートユーザがいて、両ユーザともポーリング間隔を 30 秒、待機時間を 3 回のポーリングに設定して SQL Remote Message Agent (dbremote) を実行します。たとえば、これらのユーザは、次のコマンドを使用してそれぞれの SQL Remote Message Agent (dbremote) を実行します。

```
dbremote -c "DSN=SQL Anywhere 17 Demo;PWD=sql"  -rd 30s -rp 3
```

次の一連の操作によってメッセージは `userX.n` とマーク付けされます。X はユーザ名であり、`n` はメッセージ番号です。たとえば、user1.5 は、user1 からの 5 番目のメッセージになります。SQL Remote Message Agent (dbremote) では、メッセージは両方のユーザについて 1 番から開始するものと考えます。

0 秒後

1. SQL Remote Message Agent (dbremote) が user1.1 と user2.4 を読み込みます。
2. SQL Remote Message Agent (dbremote) が user1.1 を適用します。
3. SQL Remote Message Agent (dbremote) の現在の待機時間: user1: N/A、user2: 3 です。

30 秒後

1. SQL Remote Message Agent (dbremote) でのメッセージの読み込み: 新着メッセージなし
2. SQL Remote Message Agent (dbremote) での適用: なし
3. SQL Remote Message Agent (dbremote) の現在の待機時間: user1: N/A、user2: 2

60 秒後

1. SQL Remote Message Agent (dbremote) でのメッセージの読み込み: user1.3
2. SQL Remote Message Agent (dbremote) での適用: 新着メッセージなし
3. SQL Remote Message Agent (dbremote) の待機時間: user1: 3、user2: 1

90 秒後

1. SQL Remote Message Agent (dbremote) でのメッセージの読み込み: user1.4
2. SQL Remote Message Agent (dbremote) での適用: なし
3. SQL Remote Message Agent (dbremote) の待機時間: user1: 3、user2: 0
4. SQL Remote Message Agent (dbremote) が、user2 に再送を要求します。

ユーザが新着メッセージを受信すると、それが予期していたメッセージでなくても SQL Remote Message Agent (dbremote) の待機時間はリセットされます。

120 秒後

1. SQL Remote Message Agent (dbremote) が user1.2 と user2.2 を読み込みます。
2. SQL Remote Message Agent (dbremote) が user1.2、user1.3、user1.4、user2.2 を適用します。
3. SQL Remote Message Agent (dbremote) の待機時間: user1: N/A、user2: N/A

関連情報

新しいメッセージをチェックするための間隔調整のポーリング [102 ページ]
受信メッセージのキャッシュによるスループットの調整 [103 ページ]
データベースワーカスレッド [106 ページ]
メッセージを送信する場合のパフォーマンス [108 ページ]

1.4.4.2.4 データベースワーカスレッド

データベースワーカスレッドの数を設定する場合は、考慮すべき点がいくつかあります。

1. メッセージを読み込みます。メッセージが読み込まれ、ヘッダ情報が検索されます (正しい適用順を決定するため)。メッセージシステムからのメッセージの読み込みは、1 つのスレッドになります。
2. メッセージを適用します。読み込まれたメッセージが、適用されるデータベースワーカスレッドに渡されます。
   通常、リモートデータベースでは、メッセージは直列で適用されます。多層システムでは、リモートデータベースが他のリモートの統合データベースになることもあります。このタイプのリモートデータベースでは、統合データベースと同様にメッセージが適用されます。
   統合データベースでは、デフォルトでメッセージが直列に適用されます。追加のデータベースワーカスレッドを使用すると、SQL Remote Message Agent ユーティリティ (dbremote) の w オプションを使用してリモートユーザからの受信メッセージを並列で適用できます。
   統合データベースでデータベースワーカスレッドを使用すると、次のようになります。
   - さまざまなリモートユーザからのメッセージが並列で適用されます。
   - リモートユーザ 1 人からのメッセージは直列で適用されます。
     たとえば、リモートユーザ 1 人から 10 件のメッセージが送信されると、そのメッセージは 1 つのワーカスレッドで適切な順序で適用されます。

データベースワーカスレッドを使用する場合の利点

統合データベースでデータベースワーカスレッドを使用する場合、メッセージを直列でなく並列に適用できるようにするとスループットが向上します。分散されたドライブアレイがあるシステム上にデータベースサーバがあるときに、パフォーマンス上の利点が顕著に表れます。

データベースワーカスレッドを使用する場合の欠点

統合データベースでデータベースワーカスレッドを使用する場合、ワーカスレッドによってユーザ間に多数のロックが生じると、スループットが低下することがあります。

ロールバックされたトランザクションを後で再適用することで、デッドロックが処理されます。

データベースワーカスレッド数の設定

統合データベースで、-w オプションを使用してデータベースワーカスレッド数を設定します。たとえば、次のコマンドでは、ワーカスレッド数を 5 に設定します。

```
dbremote -c "DSN=SQL Anywhere 17 Demo;PWD=sql"  -w 5
```

関連情報

新しいメッセージをチェックするための間隔調整のポーリング [102 ページ]
受信メッセージのキャッシュによるスループットの調整 [103 ページ]
メッセージを再送するための要求調整 [104 ページ]
継続モードでの SQL Remote Message Agent の実行 (dbremote) [94 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]
メッセージを送信する場合のパフォーマンス [108 ページ]

1.4.4.3 メッセージを送信するタスク

SQL Remote Message Agent (dbremote) は、メッセージの送信時に一連のタスクを実行します。

パブリッシャのトランザクションログのスキャン

SQL Remote Message Agent (dbremote) は、パブリッシャデータベースのトランザクションログをスキャンして、トランザクションログのエントリをサブスクライバへのメッセージに変換します。-l オプションで定義されているメッセージの最大長は、システム内のすべてのデータベースで同じである必要があります。

大きなトランザクションの場合、SQL Remote Message Agent (dbremote) はマルチパートのメッセージを作成します。これらのメッセージには、トランザクション内でのそれぞれの位置を追跡するシーケンス番号が個別に付けられています。サブスクライバデータベースの SQL Remote Message Agent (dbremote) は、シーケンス番号を使用して、メッセージが正しい順序で適用され、失われることがないようにします。

リモートデータベースへのメッセージの送信

SQL Remote Message Agent (dbremote) は、各リモートユーザの送信頻度プロパティで指定された時刻にメッセージを送信します。

SQL Remote Message Agent (dbremote) は、キャッシュメモリが設定値を超えた場合、指定時刻より前にメッセージを送信します。SQL Remote Message Agent (dbremote) は、メッセージをキャッシュメモリに格納します。使用中のキャッシュメモリが指定された値を超えると、メッセージが送信されます。

リモートデータベースからの再送要求の処理

ユーザがメッセージの再送要求を発行すると、パブリッシャデータベースの SQL Remote Message Agent (dbremote) は通常のメッセージ送信処理を中断し、再送要求を処理します。

このような再送要求の緊急度は、-ru オプションを使用して制御します。

パブリッシャデータベースへの確認メッセージの送信

メッセージが受信されてサブスクライブされたデータベースに適用されると、パブリッシャには確認メッセージが返送されます。

関連情報

送信遅延調整 [109 ページ]
送信メッセージのキャッシュによるスループットの調整 [110 ページ]
再送要求処理速度 [110 ページ]
保証されたメッセージ配信システム [111 ページ]

1.4.4.4 メッセージを送信する場合のパフォーマンス

メッセージの送信におけるパフォーマンス上の主な問題は、あるサイトにデータが入力されてから他のサイトに表示されるまでのターンアラウンドタイムです。

このラグタイムを短縮するには、SQL Remote Message Agent (dbremote) でのメッセージの送信時に、次の変数を調整できます。

- リモートデータベースにメッセージを送信する頻度。
- メッセージのサイズ。
- 再送要求処理の緊急度。

このセクションの内容:

関連情報

メッセージ受信時のパフォーマンスのトラブルシューティング [102 ページ]

1.4.4.4.1 送信遅延調整

送信するメッセージを作成するため、SQL Remote Message Agent (dbremote) はトランザクションログから新しいデータをポーリングします。送信遅延は、ポーリングとポーリングの間に送信されるトランザクションログデータを待つ時間です。

ポーリングが終了してから次回のポーリングが開始されるまでのポーリング間隔のデフォルトは、1 分です。送信遅延は、-sd オプションを使用して設定できますが、通常はデフォルトで十分です。送信遅延は、リモートユーザの送信頻度より短いかまたは同じ時間にしてください。

送信遅延の短縮

秒単位の値を使用することによって、ポーリング頻度を上げることができます。たとえば、次のコマンドでは、30 秒ごとにポーリングが行われます。

```
dbremote -c "DSN=SQL Anywhere 17 Demo;PWD=sql"  -sd 30s ...
```

送信遅延の延長

ポーリングの頻度を下げることができます。次のコマンドでは、ポーリング間隔を 60 分に設定します。

```
dbremote -c "DSN=SQL Anywhere 17 Demo;PWD=sql"  -sd 60
```

一般的に、送信間隔が長いと、SQL Remote Message Agent (dbremote) によって、送信前にメッセージの作成処理の大部分が実行されます。メッセージ作成処理を分散するために、通常は短めの間隔をおすすめします。

関連情報

送信メッセージのキャッシュによるスループットの調整 [110 ページ]
再送要求処理速度 [110 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]

1.4.4.4.2 送信メッセージのキャッシュによるスループットの調整

SQL Remote Message Agent (dbremote) は、送信するメッセージをメモリの設定可能エリアにキャッシュします。

すべてのリモートデータベースが、レプリケートされるオペレーションのユニークサブセットを受信する場合、それぞれのリモートデータベースにメッセージが同時に作成されます。同じオペレーションを受信するリモートユーザのグループには、メッセージが 1 つだけ作成されます。メッセージは次の場合に送信されます。

- 送信頻度に達したとき
- 使用中のキャッシュメモリが -m の値を超えたとき
- メッセージのサイズがその最大サイズ (-l オプションで指定される) に達したとき

メッセージキャッシュサイズの指定

メッセージキャッシュのサイズは、-m オプションを使用して、SQL Remote Message Agent (dbremote) のコマンドで指定します。

-m オプションは、SQL Remote Message Agent (dbremote) がメッセージの構築に使用するメモリの最大容量を指定します。使用できるサイズは、n (バイト)、nK、nM で指定します。デフォルトは 2048 K (2 M) です。

-m オプションは、単一の統合データベースと多数のリモートデータベースを使用する場合に役立ちます。

例

次のコマンドは、12 MB のメモリをメッセージキャッシュとして使用して SQL Remote Message Agent (dbremote) を起動します。

```
dbremote -c "DSN=SQL Anywhere 17 Demo;PWD=sql" -m 12M
```

関連情報

送信遅延調整 [109 ページ]
再送要求処理速度 [110 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]

1.4.4.4.3 再送要求処理速度

メッセージを再送すると、通常のメッセージ送信処理が中断されるため、SQL Remote Message Agent (dbremote) は再送要求の処理を遅らせます。

デフォルトでは、SQL Remote Message Agent (dbremote) は、再送を要求したリモートユーザの送信頻度の半分の時間を待機します。

メッセージを再送する場合、SQL Remote Message Agent (dbremote) は次のタスクを実行します。

- トランザクションログのスキャンを停止し、新しいメッセージの作成を停止します。
- キャッシュに格納された送信待機中の現在のメッセージを削除します。トランザクションログの読み込み時とそれらのメッセージの作成時に SQL Remote Message Agent (dbremote) が実行したすべての作業が失われます。
- 再送要求で要求されたオフセットからトランザクションログを再読み込みします。SQL Remote Message Agent (dbremote) はメッセージを作成し、そのキャッシュに格納します。
- 次の送信頻度の時刻まで待機した後、メッセージを送信します。

メッセージの再送要求の緊急度と通常のメッセージ処理の優先度のバランスを取ってください。

-ru オプションでは、再送要求の緊急度を制御します。他のメッセージが到着するまで再送要求の処理を遅らせるには、このオプションの設定時間を長くします。たとえば、次のコマンドでは、再送要求を処理する前に 1 時間待機します。

```
dbremote -c "DSN=SQL Anywhere 17 Demo;PWD=sql" -ru 1h
```

関連情報

送信遅延調整 [109 ページ]
送信メッセージのキャッシュによるスループットの調整 [110 ページ]
メッセージを再送するための要求調整 [104 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]

1.4.5 保証されたメッセージ配信システム

保証されたメッセージ配信システムは、レプリケートされたオペレーションがすべて正しい順序で適用されるようにし、レプリケートされたオペレーションが欠落することや 2 回適用されることがないようにします。

保証されたメッセージ配信システムでは、次の情報が使用されます。

SYSREMOTEUSER システムテーブルで管理されているステータス情報

このテーブルには各サブスクライバに対応したローがあり、そこにはそのサブスクライバが送受信するメッセージのステータス情報が示されています。次に例を示します。

- 統合データベースでは、SYSREMOTEUSER システムテーブルに各リモートユーザに対応したローがあります。
- 各リモートデータベースでは、SYSREMOTEUSER システムテーブルに統合データベースの情報を含むローが 1 つあります。

SYSREMOTEUSER システムテーブルは、SQL Remote Message Agent (dbremote) が管理しています。

サブスクライバデータベースでは、SQL Remote Message Agent (dbremote) はパブリッシャデータベースに確認メッセージを送信し、サブスクリプションの最後で SYSREMOTEUSER システムテーブルが正しく管理されていることを確認します。

メッセージのヘッダ内の情報

SQL Remote Message Agent (dbremote) は、メッセージ内のヘッダ情報を読み込み、この情報を使用して SYSREMOTEUSER システムテーブルを更新します。各メッセージのヘッダには、次の情報が含まれています。

メッセージの **resend_count**

データベースがメッセージを失った受信の回数を追跡するカウンタ。

次の例では、resend_count は 1 です。

```
Current message's header: (1-0000942712-0001119170-0)
```

前のメッセージの最後の **COMMIT** のトランザクションログオフセット

次の例では、前のメッセージの最後のコミットのトランザクションログオフセットは、0000942712 です。

```
Previous message's header:(0-0000923357-0000942712-0)
Current message's header: (0-0000942712-0001119170-0)
```

現在のメッセージの最後の **COMMIT** のトランザクションログオフセット

次の例では、現在のメッセージの最後のコミットは、0001119170 です。

```
Current message's header: (0-0000942712-0001119170-0)
```

トランザクションがいくつかのメッセージにわたっている場合は、両方のトランザクションログオフセットは、最後のメッセージに COMMIT が含まれるまで同じになることがあります。

次の例では、4 番目のメッセージまで COMMIT は発生していません。

```
(0-0000942712-0000942712-0)
(0-0000942712-0000942712-1)
(0-0000942712-0000942712-2)
(0-0000942712-0001119170-3)
```

シーケンス番号

トランザクションがいくつかのメッセージにわたっている場合は、メッセージを正しい順序で並べるために、このシーケンス番号が使用されます。

シーケンス番号 0 は、次のことを示す場合があります。

- トランザクションログオフセットが異なる場合、メッセージはマルチパートのメッセージの一部ではありません。
  次の例では、メッセージはマルチパートのメッセージの一部ではありません。

  ```
  (0-0000923200-0000923357-0)
  (0-0000923357-0000942712-0)
  ```

- トランザクションログオフセットが同一の場合、メッセージはマルチパートのメッセージの最初のメッセージです。
  次の例では、最初のメッセージはマルチパートのメッセージの一部です。

  ```
  (0-0000942712-0000942712-0)
  (0-0000942712-0000942712-1)
  (0-0000942712-0000942712-2)
  (0-0000942712-0001119170-3)
  ```

このセクションの内容:

操作の順序 [113 ページ]
レプリケートされた文が正しい順序で適用されるようにするため、保証されたメッセージ配信システムは、パブリッシャデータベースとサブスクライバデータベースのトランザクションログオフセットを使用します。

消失または壊れたメッセージ [114 ページ]

SYSREMOTEUSER システムテーブルでは、メッセージの再送が管理されます。

メッセージは 1 回だけ適用 [115 ページ]
操作が複数回適用されないようにするために、サブスクライバの SQL Remote Message Agent (dbremote) は、メッセージヘッダ内の resend_count 値と、その SYSREMOTEUSER システムテーブル内の rereceive_count を比較します。resend_count 値が rereceive_count よりも小さい場合、メッセージは適用されることなく削除されます。

1.4.5.1 操作の順序

レプリケートされた文が正しい順序で適用されるようにするため、保証されたメッセージ配信システムは、パブリッシャデータベースとサブスクライバデータベースのトランザクションログオフセットを使用します。

トランザクションログにある各 COMMIT は、十分に定義されたオフセットでマーク付けされています。トランザクションの順序は、トランザクションログオフセット値を比較して決定されます。各メッセージには、次のトランザクションログオフセットが含まれています。

- 前のメッセージの最後の COMMIT のトランザクションログオフセットトランザクションがいくつかのメッセージにわたっている場合は、メッセージを正しい順序で並べるためのシーケンス番号がトランザクションにあります。
- メッセージの最後の COMMIT のトランザクションログオフセット。

メッセージの順序

メッセージを送信すると、前回のメッセージの最後の COMMIT のオフセットによって、メッセージが順に並べられます。トランザクションがいくつかのメッセージにわたっている場合は、メッセージを正しい順序で並べるために、トランザクション内のシーケンス番号が使用されます。

メッセージの送信

SYSREMOTEUSER システムテーブルの log_sent カラムには、サブスクライバに送信された最新メッセージのローカルトランザクションログのオフセットが入ります。

次の手順では、メッセージが送信されたときに SYSREMOTEUSER システムテーブルがどのように更新されるかを示します。

1. パブリッシャの SQL Remote Message Agent (dbremote) はサブスクライバにメッセージを送信すると、送信したメッセージの最後の COMMIT のトランザクションログオフセット値を log_sent 値に設定します。
   たとえば、パブリッシャが次のメッセージを user1 に送信します。

   ```
   (0-0000923200-0000923357-0)
   ```

   パブリッシャの SYSREMOTEUSER システムテーブル内で、パブリッシャは log_sent 値に user1 の 0000923357 を設定します。
2. メッセージが受信されてサブスクライバデータベースに適用されると、パブリッシャには確認メッセージが送信されます。確認メッセージには、サブスクライバデータベースによって適用された最新のトランザクションログオフセットが含まれています。

たとえば、確認メッセージでは、user1 がトランザクションログオフセット 0000923357 以前のすべてのトランザクションを適用したことが確認されます。

3. パブリッシャの SQL Remote Message Agent (dbremote) は確認メッセージを受信すると、SYSREMOTEUSER システムテーブルの confirm_sent カラムにユーザの確認メッセージのオフセット値を設定します。
   たとえば、パブリッシャは、パブリッシャの SYSREMOTEUSER システムテーブルの confirm_sent カラムに user1 の 0000923357 を設定します。

log_sent と confirm_sent の両方の値には、パブリッシャのトランザクションログのトランザクションログオフセットが含まれています。confirm_sent 値は、log_sent 値より後のオフセットにはなりません。

メッセージの受信

次の手順では、メッセージが受信されたときに SYSREMOTEUSER システムテーブルがどのように更新されるかを示します。

1. サブスクライバデータベースの SQL Remote Message Agent (dbremote) がレプリケーションのアップデートを受信して適用すると、そのメッセージの最後の COMMIT のオフセットによって SYSREMOTEUSER システムテーブルの log_received カラムが更新されます。
   たとえば、サブスクライバが次のメッセージを受信して適用すると、SYSREMOTEUSER システムテーブルの log_received 値に 0000923357 が設定されます。

   ```
   (0-0000923200-0000923357-0)
   ```

   すべてのサブスクライバデータベースの log_received カラムには、パブリッシャデータベースのトランザクションログでの、トランザクションログオフセットが入ります。
2. オペレーションが受信され適用されると、サブスクライバの SQL Remote Message Agent (dbremote) は、その SYSREMOTEUSER システムテーブルの confirm_received 値を設定し、パブリッシャデータベースに確認メッセージを送信します。

関連情報

消失または壊れたメッセージ [114 ページ]
メッセージは 1 回だけ適用 [115 ページ]

1.4.5.2 消失または壊れたメッセージ

SYSREMOTEUSER システムテーブルでは、メッセージの再送が管理されます。

SYSREMOTEUSER システムテーブルには、メッセージの再送を管理する 2 つのカラムが含まれています。

resend_count カラム

サブスクライバデータベースがメッセージを失った回数を追跡するカウンタ。

rereceive_count カラム

SQL Remote Message Agent (dbremote) がパブリッシャユーザからのメッセージが失われたと判断した回数を追跡するカウンタ。

サブスクライバデータベースでメッセージが正しい順序で受信されると、次の処理が実行されます。

1. サブスクライバの SQL Remote Message Agent (dbremote) は、メッセージを正しい順序で適用し、その SYSREMOTEUSER システムテーブルを更新します。
2. サブスクライバの SQL Remote Message Agent (dbremote) は、パブリッシャに確認メッセージを送信します。
3. パブリッシャが確認メッセージを受信すると、パブリッシャの SQL Remote Message Agent (dbremote) はその SYSREMOTEUSER システムテーブルを更新します。

メッセージが正しい順序で受信されなかった場合は、次の処理が実行されます。

1. サブスクライバの SQL Remote Message Agent (dbremote) は、再送要求を送信し、その SYSREMOTEUSER システムテーブルの rereceive_count 値を増分します。
2. パブリッシャは再送要求を受信すると、その SYSREMOTEUSER システムテーブルでサブスクライバの resend_count 値を増分します。
3. パブリッシャの SYSREMOTEUSER システムテーブル内で、log_sent 値に confirm_sent カラムの値が設定されます。log_sent 値を設定し直すと、オペレーションが再送されます。

関連情報

操作の順序 [113 ページ]
メッセージは 1 回だけ適用 [115 ページ]

1.4.5.3 メッセージは 1 回だけ適用

操作が複数回適用されないようにするために、サブスクライバの SQL Remote Message Agent (dbremote) は、メッセージヘッダ内の resend_count 値と、その SYSREMOTEUSER システムテーブル内の rereceive_count を比較します。resend_count 値が rereceive_count よりも小さい場合、メッセージは適用されることなく削除されます。

関連情報

操作の順序 [113 ページ]
消失または壊れたメッセージ [114 ページ]

1.4.6 メッセージサイズ

SQL Remote Message Agent (dbremote) には、多数のエンコードと圧縮の機能があります。

互換性

システムは、SQL Anywhere の古いバージョンと互換性を持つように設定されています。

圧縮

メッセージの圧縮レベルを選択できます。

メッセージのサイズは、メッセージがシステムを介して渡されるときの効率に影響を与えます。圧縮したメッセージは、圧縮していないメッセージよりも効率的にメッセージシステムで処理されます。ただし、圧縮の実行にかなりの時間がかかります。

エンコード

SQL Remote ではメッセージをエンコードして、メッセージが破壊されずにメッセージシステムを介して確実に渡されるようにします。エンコードスキームをカスタマイズすると、特別な機能も使用できるようになります。

このセクションの内容:

エンコードでのメッセージ破壊の防止 [116 ページ]
SQL Remote ではメッセージをエンコードして、メッセージが破壊されずにメッセージシステムを介して確実に渡されるようにします。

関連情報

SQL Remote Message Agent (dbremote) の稼働条件 [94 ページ]

1.4.6.1 エンコードでのメッセージ破壊の防止

SQL Remote ではメッセージをエンコードして、メッセージが破壊されずにメッセージシステムを介して確実に渡されるようにします。

SQL Remote のメッセージのエンコード機能は、デフォルトで次のように動作します。

- メッセージシステムでバイナリ形式のメッセージが使用できる場合、メッセージはエンコードされません。
- SMTP のように、メッセージシステムでテキストベースのメッセージ形式が必要とされる場合は、エンコード DLL (`dbencod`*17*`.dll`) によって、メッセージがテキストフォーマットに変換されてから送信されます。このメッセージ形式は、同じ DLL を使用する受信側ではエンコードされません。
  エンコードスキームをカスタマイズすると、特別な機能も使用できるようになります。
- データベースオプション compression に -1 を設定すると、すべてのメッセージシステムでバージョン 5 と互換性のあるエンコードが使用されます。

このセクションの内容:

カスタムエンコードスキーム [117 ページ]
カスタムエンコードスキームを実装するには、カスタムエンコード DLL を構築します。このカスタム DLL を使用して、特定のメッセージシステムで必要な特殊機能を適用したり、各ユーザに送信されたメッセージの数などの統計を取ったりすることができます。

1.4.6.1.1 カスタムエンコードスキーム

カスタムエンコードスキームを実装するには、カスタムエンコード DLL を構築します。このカスタム DLL を使用して、特定のメッセージシステムで必要な特殊機能を適用したり、各ユーザに送信されたメッセージの数などの統計を取ったりすることができます。

ヘッダファイル *%SQLANY17%*¥SDK¥Include¥dbrmt.h には、カスタムエンコードスキームの構築に使用できるアプリケーションプログラミングインタフェースが含まれています。

自分のカスタム DLL を使用するには、そのカスタム DLL へのフルパスの値をメッセージ制御パラメータ encode_dll に設定します。次に例を示します。

```
SET REMOTE FTP OPTION "Public"."encode_dll" = 'c:¥¥sqlany17¥¥Bin32¥¥custom.dll';
```

> **注記**
>
> エンコードとデコードは互換性がある必要があります。カスタムエンコードを実装する場合、その DLL が受信側にもあり、自分のメッセージを正確に復号化できるようになっていることを必ず確認してください。

関連情報

SET REMOTE OPTION 文 [SQL Remote] [242 ページ]

1.4.7 SQL Remote メッセージシステム

SQL Remote のレプリケーションでは、メッセージシステムとは、統合データベースとリモートデータベースの間でのメッセージのやりとりに使用するプロトコルのことです。SQL Remote は、基本となるメッセージシステムを 1 つ以上使用して、データベース間のデータ交換を行います。

SQL Remote では、次のメッセージシステムをサポートしています。

ファイル共有

他のソフトウェアを必要としない簡単なシステム。

FTP

インターネットファイル転送プロトコル。

HTTP

ハイパーテキスト転送プロトコル。

SMTP/POP

インターネット電子メール転送プロトコル。

REMOTE または CONSOLIDATE 権限をユーザに割り当てる場合は、メッセージシステムを選択します。

SQL Remote システムで使用される各メッセージシステムでは、制御パラメータやその他の設定についてセットアップします。

すべてのメッセージシステムがすべてのオペレーティングシステムでサポートされるわけではありません。

このセクションの内容:

関連情報

REMOTE 権限の付与 (SQL Central) [29 ページ]
CONSOLIDATE 権限の付与 (SQL Central) [32 ページ]

1.4.7.1 メッセージシステムの設定

メッセージシステムを使用する前に、必ずパブリッシャのアドレスを設定してください。

各メッセージタイプの定義には、メッセージシステムのタイプ名 (FILE、FTP、HTTP、または SMTP) とそのメッセージタイプにおけるパブリッシャのアドレスが含まれます。

メッセージタイプの定義に入力されるアドレスには、データベースのパブリッシャ ID と密接なつながりがあります。

抽出ユーティリティ (dbxtract)

リモートデータベースの作成時に抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードが、統合データベースでのパブリッシャアドレスを返信アドレスとして使用します。また SQL Remote Message Agent (dbremote) でも、FILE システムの受信メッセージの場所を識別するためにパブリッシャアドレスを使用します。

関連情報

FILE メッセージシステム [123 ページ]
FTP メッセージシステム [124 ページ]
SMTP メッセージシステム [131 ページ]
HTTP メッセージシステム [127 ページ]

1.4.7.2 メッセージタイプの作成 (SQL Central)

SQL Remote ユーザのメッセージタイプを追加します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. *SQL Remote* ユーザをダブルクリックします。
3. メッセージタイプタブをクリックします。
4. ファイル > 新規 > メッセージタイプ をクリックします。
5. メッセージタイプの名前を入力します。入力する名前は、お使いの SQL Anywhere インストールディレクトリにすでにインストールされているメッセージタイプ DLL と一致させてください。次へをクリックします。
6. パブリッシャアドレスを指定してください。フィールドにパブリッシャアドレスを入力します。完了をクリックします。

結果

新しいメッセージタイプが作成されます。

関連情報

メッセージタイプの変更 (SQL Central) [120 ページ]
メッセージタイプの削除 (SQL Central) [121 ページ]

1.4.7.3 メッセージタイプの変更 (SQL Central)

パブリッシャのアドレスを変更するには、メッセージタイプを変更します。既存のメッセージタイプの名前を変更することはできないので、メッセージタイプを削除してから、新しい名前で新しいメッセージタイプを作成してください。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. *SQL Remote* ユーザをダブルクリックします。
3. メッセージタイプタブをクリックします。
4. 変更するメッセージタイプを右クリックし、プロパティをクリックします。
5. メッセージタイプのプロパティを更新し、*OK* をクリックします。

結果

メッセージのプロパティが更新されます。

関連情報

メッセージタイプの作成 (SQL Central) [119 ページ]
メッセージタイプの削除 (SQL Central) [121 ページ]

1.4.7.4 メッセージタイプの削除 (SQL Central)

メッセージタイプを削除すると、パブリッシャアドレスがメッセージ定義から削除されます。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. *SQL Remote* ユーザをダブルクリックします。
3. メッセージタイプタブをクリックします。
4. 削除するメッセージタイプを右クリックし、削除をクリックします。
5. はいをクリックします。

結果

メッセージタイプが削除されます。

関連情報

メッセージタイプの作成 (SQL Central) [119 ページ]
メッセージタイプの変更 (SQL Central) [120 ページ]

1.4.7.5 リモートメッセージタイプ制御パラメータ

メッセージは、データベース内に保存されます。制御パラメータを設定する場合は、SET REMOTE OPTION 文を実行します。

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

たとえば、次の文は、ユーザ myuser 用の FTP リンクに対し、FTP ホストを `ftp.mycompany.com` に設定します。

```
SET REMOTE FTP OPTION myuser.host = 'ftp.mycompany.com';
```

SQL を使用してメッセージリンクパラメータを表示する場合は、SYSREMOTEOPTION システムビューのクエリを実行します。

```
SELECT * from SYSREMOTEOPTION;
```

ディスクに格納されるメッセージリンクパラメータ

SQL Remote の古いバージョンでは、メッセージリンクパラメータはデータベースの外に保存されていました。メッセージリンクパラメータの外部保存はお奨めしません。

メッセージリンク制御パラメータは、次の場所に格納されます。

Windows

レジストリ内の次の場所に格納されます。

```
HKEY_CURRENT_USER¥Software¥SAP¥SQL Remote
```

各メッセージリンクのパラメータは、メッセージリンクの名前 (4、SMTP など) を付けられて、SQL Remote キーの下のキーに格納されます。

UNIX

FILE システムディレクトリ設定は、SQLREMOTE 環境変数に格納されます。

SQLREMOTE 環境変数には、FILE メッセージシステムの制御パラメータのうち、その代替の 1 つとして使用されるパスが格納されます。

SQL Remote Message Agent (dbremote) がメッセージリンクをロードすると、リンクは現在のパブリッシャの設定を使用します。設定が指定されていない場合は、そのパブリッシャが属するグループの設定を使用します。

Windows では、メッセージリンクパラメータのデータベースへの保存をサポートする SQL Remote Message Agent (dbremote) を初めて起動すると、リンクのオプションがレジストリからデータベースにコピーされます。

関連情報

SET REMOTE OPTION 文 [SQL Remote] [242 ページ]

1.4.7.6 FILE メッセージシステム

FILE メッセージシステムを使用すれば、電子メールシステムまたは FTP システムが適切な場所になくても SQL Remote を使用できます。

FILE メッセージシステムのアドレス

FILE メッセージシステムは、単純な FILE 共有システムです。リモートユーザの FILE アドレスは、すべてのメッセージが書き込まれるサブフォルダです。メッセージを取得するには、アプリケーションがユーザのファイルを格納しているディレクトリからメッセージを読み込みます。返信メッセージは、統合データベースのアドレスに送信 (ディレクトリに書き込み) されます。

SQL Remote Message Agent (dbremote) がサービスとして起動中の場合は、稼働しているアカウントに、必要なすべてのディレクトリに対する読み込みと書き込みのパーミッションが必要です。正しいパーミッションが割り当てられていない場合、SQL Remote Message Agent はネットワークドライブにアクセスできません。

アドレスのルートディレクトリ

一般的に、FILE メッセージシステムのアドレスは、共有ディレクトリのサブフォルダです。このサブフォルダは、モデムまたはローカルエリアネットワークからアクセスする SQL Remote ユーザ全員が使用できます。各ユーザには、レジストリのエントリ、初期化ファイルのエントリ、または共有ディレクトリを示す SQLREMOTE 環境変数が必要です。

FILE システムを使用して、統合コンピュータまたはリモートコンピュータのディレクトリにメッセージを入れることもできます。簡易ファイル転送メカニズムを使用してファイルを交換し、レプリケーションを実行できます。

FILE メッセージ制御パラメータ

FILE メッセージシステムでは、SET REMOTE OPTION 文で設定される次の制御パラメータを使用します。

directory

メッセージが格納されるディレクトリです。このパラメータは、SQLREMOTE 環境変数の代替となります。

debug

このパラメータの設定は YES または NO です。デフォルトは NO です。YES を設定すると、FILE リンクが行った FILE システム呼び出しがすべて出力ログに表示されます。

encode_dll

カスタムエンコードスキームを使用している場合は、作成したカスタムエンコード DLL のフルパスをこのパラメータに設定する必要があります。

invalid_extensions

メッセージングシステムでのファイルの生成時に SQL Remote Message Agent (dbremote) で使用されないようにするファイル拡張子の、カンマで区切られたリスト。

max_retries

デフォルトでは、SQL Remote が継続モードで実行されていて、メッセージシステムにアクセスするときにエラーが発生する場合、送受信フェーズの後に停止します。このパラメータを使用して、停止前に SQL Remote が再試行する送受信フェーズの回数を指定します。

pause_after_failure

このパラメータが使えるのは、max_retries がゼロ以外の値に指定され、SQL Remote が継続モードで実行されている場合です。メッセージシステムでエラーが発生すると、このパラメータは、SQL Remote が送受信フェーズを再試行する間に待機する時間を秒で指定します。

Unlink_delay

ファイル削除に失敗した後、次にファイルの削除を試みるまでの秒数。unlink_delay に対して値が定義されていない場合、デフォルト動作は、最初の試行失敗の後に 1 秒間、2 回目の試行失敗の後に 2 秒間、3 回目の試行失敗の後に 3 秒間、4 回目の試行失敗の後に 4 秒間一時停止するように設定されています。

関連情報

FTP メッセージシステム [124 ページ]
SMTP メッセージシステム [131 ページ]
HTTP メッセージシステム [127 ページ]
メッセージサイズ [115 ページ]
SET REMOTE OPTION 文 [SQL Remote] [242 ページ]

1.4.7.7 FTP メッセージシステム

FTP メッセージシステムでは、メッセージは FTP ホストのルートディレクトリの下位ディレクトリに保存されます。ホストとルートディレクトリは、レジストリまたは初期化ファイル内のメッセージシステム制御パラメータによって指定されます。またメッセージが保存されるサブフォルダが各ユーザのアドレスになります。

FTP メッセージ制御パラメータ

FTP メッセージシステムでは、SET REMOTE OPTION 文で設定される次の制御パラメータを使用します。

host

FTP サーバを実行しているコンピュータのホスト名。このパラメータには、ホスト名 (ftp.sap.com など) または IP アドレス (192.138.151.66 など) を使用できます。

user

FTP ホストにアクセスするためのユーザ名。

password

FTP ホストにアクセスするためのパスワード。

root_directory

メッセージが保存される、FTP ホストサイトのルートディレクトリ。

port

FTP の接続に使用される IP ポート番号。このパラーメータは通常は不要です。

debug

このパラメータは、YES または NO のいずれかに設定されます。デフォルトは NO です。YES を設定すると、デバッグ出力が出力ログに表示されます。

active_mode

このパラメータは、SQL Remote がクライアント/サーバ接続を確立する方法を制御します。このパラメータは、YES または NO のいずれかに設定されます。デフォルトは NO (受動モード) です。受動モードは推奨の転送モードで、FTP メッセージリンクにデフォルトとして設定されています。受動モードでは、すべてのデータ転送接続はクライアント (この場合はメッセージリンク) から開始されます。アクティブモードでは、FTP サーバがすべてのデータ接続を開始します。

reconnect_retries

失敗になる前に、リンクがサーバでソケットを開こうと試行する回数。デフォルト値は 4 です。このパラメータを設定すると、再接続のみに影響します。FTP リンクによる最初の接続には影響しません。

reconnect_pause

接続の試行失敗後に次の接続まで待機する秒数。デフォルトは 30 秒です。このパラメータを設定すると、再接続のみに影響します。FTP リンクによる最初の接続には影響しません。

suppress_dialogs

このパラメータは、ON または OFF に設定されます。ON に設定されている場合は、FTP サーバへの接続の試行失敗後に、接続ウィンドウは表示されません。代わりに、エラーが発生します。

invalid_extensions

メッセージングシステムでのファイルの生成時に dbremote で使用されないようにするファイル拡張子の、カンマで区切られたリスト。

encode_dll

カスタムエンコードスキームを実装している場合は、作成したカスタムエンコード DLL のフルパスをこのパラメータに設定する必要があります。

max_retries

デフォルトでは、SQL Remote が継続モードで実行されていて、メッセージシステムにアクセスするときにエラーが発生する場合、送受信フェーズの後に停止します。このパラメータを使用して、停止前に SQL Remote が再試行する送受信フェーズの回数を指定します。

pause_after_failure

このパラメータが使えるのは、max_retries がゼロ以外の値に指定され、SQL Remote が継続モードで実行されている場合です。メッセージシステムでエラーが発生すると、このパラメータは、SQL Remote が送受信フェーズを再試行する間に待機する時間を秒で指定します。

このセクションの内容:

FTP 問題のトラブルシューティング [126 ページ]
FTP メッセージリンクにおける問題のほとんどは、ネットワークシステムの問題によって発生します。

関連情報

メッセージサイズ [115 ページ]
FILE メッセージシステム [123 ページ]
SMTP メッセージシステム [131 ページ]
SET REMOTE OPTION 文 [SQL Remote] [242 ページ]

1.4.7.7.1 FTP 問題のトラブルシューティング

FTP メッセージリンクにおける問題のほとんどは、ネットワークシステムの問題によって発生します。

次の一覧では、問題に対処するためのテストについて説明します。

デバッグメッセージ制御パラメータの設定

デバッグ出力を確認して、FTP サーバに接続しているかどうかを判断します。接続している場合は、どの FTP コマンドに問題があるかがデバッグ出力に示されます。

FTP サーバの ping

FTP リンクが FTP サーバに接続できない場合は、システムネットワーク設定をテストします。たとえば、次のコマンドを実行します。

```
ping ftp-server-name
```

FTP サーバの IP アドレスと、FTP サーバへの ping (往復) 時間が返されます。FTP サーバの ping が実行できない場合は、ネットワーク設定に問題があります。ネットワーク管理者に問い合わせてください。

受動モードの動作確認

FTP リンクが FTP サーバに接続しているのに、データ接続を開けない場合は、FTP クライアントが受動モードを使用してサーバからデータを転送できることを確認します。

受動モードは推奨の転送モードで、FTP メッセージリンクにデフォルトとして設定されています。受動モードでは、すべてのデータ転送接続はクライアント (この場合はメッセージリンク) から開始されます。アクティブモードでは、FTP サーバがすべてのデータ接続を開始します。FTP サーバが正しく設定されていないファイアウォールの保護を受けていると、ファイアウォールが FTP 制御ポート以外のポート上の FTP サーバへのソケット接続をブロックするため、デフォルトの受動転送モードを使用できない場合があります。

転送モードを **active** か **passive** に設定できる FTP ユーザプログラムを使用する場合は、転送モードを受動に設定して、ファイルのアップロードやダウンロードを行ってください。使用しているクライアントが、アクティブモードを使用しないとファイルを転送できない場合は、受動モードでの転送ができるようにファイアウォールと FTP サーバを設定し直すか、active_mode メッセージ制御パラメータに YES を設定してください。すべてのネットワーク設定でアクティブモード転送が動作するとはかぎりません。次に例を示します。クライアントが IP マスカレードが機能しているゲートウェイの後方にある場合、ゲートウェイソフトウェアによっては受信接続で障害が発生する可能性があります。

パーミッションとディレクトリ構造の確認

FTP サーバが接続中で、ディレクトリのリストの取得やファイルの操作に問題がある場合は、パーミッションが正しく設定されていることと、必要なディレクトリが存在していることを確認します。

FTP プログラムを使用して FTP サーバにログインします。ディレクトリを、root_directory パラメータに保存されている場所に変更します。必要なディレクトリが表示されない場合は、root_directory 制御パラメータが間違って設定されているか、ディレクトリが存在しない可能性があります。

メッセージディレクトリのファイルを取り出してパーミッションをテストし、統合データベースディレクトリにファイルをアップロードします。エラーが返される場合は、FTP サーバのパーミッションが正しく設定されていません。

1.4.7.8 HTTP メッセージシステム

HTTP メッセージシステムを使用して、SQL Remote はインターネット経由で Hypertext Transfer Protocol (HTTP) を使用してメッセージを送信します。メッセージはテキストフォーマットにエンコードされ、HTTP 経由でターゲットデータベースに送信されます。

メッセージは、HTTP サーバとして動作する SQL Anywhere データベースを使用して送受信されます。

HTTP メッセージサーバの設定

SQL Remote メッセージをリモートデータベースとの間で転送する HTTP サーバとして動作する SQL Anywhere データベースサーバを使用します。デフォルトでは、新しく初期化された SQL Anywhere データベースには、メッセージサーバとして動作できるように定義された Web サービスはありません。sr_add_message_server、sr_drop_message_server、sr_update_message_server という 3 つのシステムストアドプロシージャが、新しく作成された SQL Anywhere データベースに定義されています。これにより、データベースが SQL Remote メッセージを転送するための HTTP サーバとして動作できるよう必要なデータベースオブジェクトを定義できます。

> **注記**
>
> データベースは SQL Anywhere17.0.4 で初期化され、3336 以上のビルド番号のデータベースサーバで初期化されている必要があります。SYS.SYSHISTORY システムテーブルに問い合わせ、データベースの初期化に使用されたデータベースサーバのバージョンとビルドを判断します。データベースが 17.0.4 と 3336 未満のビルド番号で初期化されている場合、`ALTER DATABASE UPGRADE PROCEDURE ON` を実行してデータベースを更新します。

別のデータベースサーバを実行するか、それとも既存の統合データベースをメッセージサーバとして使用するのかを決める必要があります。この決定を行う場合は次のことを考慮します。

1. リモートデータベースが、メッセージサーバで認証する場合、そのリモートデータベースのパブリッシャと認証用に提供されたパスワードが使用されます。統合データベースにユーザが存在していても、そのユーザは HTTP メッセージシステムの要件である定義済みパスワードを持っていないかもしれません (リモートユーザは接続の権限を持っていないかもしれません)。統合データベース内でリモートユーザに CONNECT 権限を付与することがセキュリティについての考慮事項である場合は、個別のデータベースを作成して、メッセージサーバとして機能させます。
2. 統合データベースの負荷が大きい場合、ここにメッセージサーバ機能を追加すると、統合データベースを実行したときにコンピュータ上のリソースに過剰な負荷がかかるかもしれません。

メッセージサーバとして動作するデータベースに必要なデータベースオブジェクトを設定するには、データベースの SQL Remote 定義を問い合わせる sr_add_message_server ストアドプロシージャを呼び出します。

メッセージサーバを別のデータベースとして作成する場合、統合データベースの定義と一致する SQL Remote 定義で 2 番目のデータベースを定義する必要があります。dbunload ユーティリティを使用して統合データベースのコピーを作成し、-n オプションを指定して、データではなく統合データベースのスキーマのみをアンロードします。

```
dbunload -n -an -c "ServerName=cons.DBN=cons;UID=DBA;PWD=passwd"
```

メッセージサーバとして別のデータベースを使用している場合、統合データベースの SQL Anywhere 定義に変更が行われた場合には、メッセージサーバデータベースにも対応する変更が行われる必要があります。

メッセージサーバを設定するには、SQL Remote メッセージが格納されているディレクトリにデータベースサーバがアクセスできる必要があります。メッセージが格納されているディレクトリを定義するには、SET REMOTE OPTION コマンドを使用して、*root_directory* HTTP メッセージパラメータを SQL Remote メッセージが格納されているディレクトリに設定します。次に、作成される新しいオブジェクトを所有しているデータベースユーザを選択して、ユーザがロールであることを確認します。最後に、sr_add_message_server ストアドプロシージャを実行して、そのオブジェクトを所有するユーザの名前を渡します。

(リモートユーザの追加や削除など) メッセージサーバの SQL Remote 定義に変更が行われるたびに、sr_update_message_system ストアドプロシージャを実行して、メッセージサーバのサポートに必要なオブジェクトの定義を更新します。ストアドプロシージャが稼働し、オブジェクトが削除されて再作成される間、短期間メッセージサーバはレプリケーションに利用できなくなります。

このデータベースをもうメッセージサーバとして使用していない場合、sr_drop_message_system ストアドプロシージャを実行して、メッセージサーバをサポートするために作成されたオブジェクトを削除できます。

メッセージサーバデータベースサーバの起動

メッセージサーバをサポートするのに必要なオブジェクトを作成した後で、メッセージサーバデータベースサーバを起動するときに、-xs オプションを使用してデータベースサーバに対する HTTP (および HTTPS) サポートを有効にする必要があります。

メッセージサーバに必要なオブジェクトを定義した人々に関係する HTTP サーバ側プロトコルオプションには次のものがあります。

ServerPort | PORT

デフォルトポートである 80 と 443 がすでにコンピュータ上で使用されている場合に、HTTP または HTTPS 要求を受信するためにデータベースサーバが使用するポート番号を指定します。

MaxRequestSize | MAXSIZE

1 つの HTTP 要求の最大サイズを指定します。デフォルト値は 100 KB です。SQL Remote メッセージサイズ (dbremote コマンドライン上の -l オプション) を 100 KB を超える値に定義した場合、データベースサーバが受信できる最大 HTTP 要求のサイズも増やす必要があります。デフォルトの SQL Remote メッセージサイズは 50 KB です。

Identity (HTTP のみ)

HTTPS を使用している場合、ID ファイルには、パブリック証明書とプライベートキーが含まれており、自己署名されない証明書の場合は、さらに署名を行うすべての証明書も含まれています。これには暗号化証明書も含まれます。この証明書のパスワードは、*Identity_Password* パラメータで指定してください。

Identity_Password (HTTP のみ)

トランスポートレイヤセキュリティを使用している場合、このオプションにより、Identity プロトコルオプションで指定した暗号化証明書に対応するパスワードを指定します。

HTTP アドレスとユーザ ID

SQL Remote と HTTP を使用するには、システムに参加する各データベースに HTTP アドレス、ユーザ ID、パスワードが必要です。これらは別々の識別子です。HTTP アドレスは各メッセージの送信先で、ユーザ ID とパスワードは、サーバに認証を行うときにユーザが入力する名前とパスワードです。

HTTP メッセージ制御パラメータ

SQL Remote Message Agent (dbremote) がメッセージシステムに接続してメッセージを送受信する前に、ユーザは制御パラメータのセットを自分のコンピュータにあらかじめ設定しておく必要があります。設定していない場合は、ユーザに必要な情報の指定を要求するプロンプトが表示されます。この情報が必要となるのは、初回接続時のみです。この情報は保存され、以後の接続でデフォルトとして使用されます。

HTTP メッセージシステムでは、SET REMOTE OPTION 文で設定される次の制御パラメータを使用します。

certificate

安全な (HTTPS) 要求を行うには、HTTPS サーバで使用される証明書にクライアントがアクセスできる必要があります。必要な情報は、セミコロンで区切られたキーワード/値のペアの文字列で指定されます。ファイルキーワードを使用して証明書のファイル名を指定できます。ファイルキーワードと証明書キーワードを一緒に指定することはできません。次のキーワードを使用できます。

キーワード	省略形	説明
file		証明書のファイル名
certificate	cert	証明書自体
company	co	証明書で指定された会社
unit		証明書で指定された会社の部署
name		証明書で指定された通称

証明書は、HTTPS サーバに対する要求、または安全でないサーバから安全なサーバにリダイレクトされる可能性がある要求に対してのみ必要です。PEM でフォーマットされた証明書のみがサポートされています。
certificate='file=filename'

client_port

SQL Remote が HTTP を使用して通信するポート番号を識別します。これは "送信" TCP/IP 接続をフィルタするファイアウォール経由で接続するためのもので、この用途にかぎり推奨されています。単一のポート番号、ポート番号の範囲、または両方の組み合わせを指定できます。指定したクライアントポートの数が少ないと、SQL Remote が前回の実行でポートを閉じた後すぐにオペレーティングシステムがポートを解放しなかった場合に、SQL Remote でメッセージの送受信ができなくなる可能性があります。*client_port*=nnnnn[-mmmmm]'

debug

YES に設定すると、すべての HTTP コマンドと応答が出力ログに表示されます。この情報は、HTTP のサポート問題のトラブルシューティングに使用できます。デフォルトは NO です。

https

HTTPS (*https=yes*) または HTTP (*https=no*) を使用するかどうかを指定します。

password

メッセージサーバのデータベースパスワード。RFC 2617 の基本認証を使用してサードパーティの HTTP サーバとゲートウェイに対する認証を行います。*password='password'*

proxy_host

プロキシサーバの URI を指定します。SQL Remote がプロキシサーバを介してネットワークにアクセスする場合に使用します。SQL Remote がプロキシサーバに接続し、そのプロキシサーバを介してメッセージサーバに要求を送信することを示します。*proxy_host='http://proxy-server[:port-number]'*

reconnect_retries

失敗になる前に、リンクがサーバでソケットを開こうと試行する回数。デフォルト値は 4。このパラメータを設定すると、再接続のみに影響します。FTP リンクによる最初の接続には影響しません。

reconnect_pause

接続の試行失敗後に次の接続まで待機する秒数。デフォルトは 30 秒です。このパラメータを設定すると、再接続のみに影響します。FTP リンクによる最初の接続には影響しません。

root_directory

この HTTP 制御パラメータは、クライアント側で指定された場合には無視されます。sr_add_message_server または sr_update_message_server ストアドプロシージャを呼び出す前に、メッセージサーバでこの制御パラメータを定義します。メッセージサーバがアクセスできる SQL Remote メッセージの格納されているディレクトリを指定します。HTTP メッセージシステムを使用する場合、リモートユーザまたはパブリッシャに指定するアドレスには、1 つのサブディレクトリのみを含めることができます。複数のサブディレクトリを含めることはできません。*root_directory='c:¥msgs'*

url

サーバ名または IP アドレスを指定し、任意で、使用する HTTP サーバのポート番号をセミコロンで区切って指定します。要求が Relay Server を経由する場合は、URL 拡張子を任意で追加して、要求の渡し先のサーバファームを示すことができます。*url ='server-name*[:*port-number*][*url-extension*]'

user

メッセージサーバのデータベースユーザ ID。RFC 2617 の基本認証を使用してサードパーティの HTTP サーバとゲートウェイに対する認証を行います。*user='userid'*

関連情報

FILE メッセージシステム [123 ページ]
SMTP メッセージシステム [131 ページ]
FTP メッセージシステム [124 ページ]
チュートリアル: 別のメッセージサーバで HTTP メッセージシステムを使用するレプリケーションシステムの設定 [185 ページ]
チュートリアル: メッセージサーバとして HTTP メッセージシステムと統合データベースを使用するレプリケーションシステムの設定 [174 ページ]
チュートリアル: Relay Server 経由で HTTP メッセージシステムを使用するレプリケーションシステムの設定 [196 ページ]
SET REMOTE OPTION 文 [SQL Remote] [242 ページ]
sr_add_message_server システムプロシージャ [230 ページ]
sr_update_message_server システムプロシージャ [232 ページ]
sr_drop_message_server システムプロシージャ [231 ページ]

SET REMOTE OPTION 文 [SQL Remote] [242 ページ]
sr_add_message_server システムプロシージャ [230 ページ]

1.4.7.9 SMTP メッセージシステム

SQL Remote では、SMTP システムを使用してインターネットメールでメッセージを送信します。

メッセージはテキストフォーマットにエンコードされ、電子メールメッセージとしてターゲットデータベースに送信されます。メッセージは、SMTP サーバを使用して送信され、POP サーバから受信されます。

SMTP アドレスとユーザ ID

SQL Remote と SMTP メッセージシステムを使用するには、システムに参加する各データベースで、SMTP アドレス、POP3 ユーザ ID、パスワードが必要です。これらは別々の識別子です。SMTP アドレスは各メッセージの送信先で、POP3 ユーザ ID とパスワードは、ユーザが自分の電子メールサーバに接続するときに入力する名前とパスワードです。

> **注記**
>
> SQL Remote メッセージを送受信するには、POP 電子メールアカウントを個別に設定してください。

SMTP メッセージ制御パラメータ

SQL Remote Message Agent (dbremote) がメッセージシステムに接続してメッセージを送受信する前に、ユーザは制御パラメータのセットを自分のコンピュータにあらかじめ設定しておく必要があります。設定していない場合は、ユーザに必要な情報の指定を要求するプロンプトが表示されます。この情報が必要となるのは、初回接続時のみです。この情報は保存され、以後の接続でデフォルトのエントリとして使用されます。

SMTP メッセージシステムでは、SET REMOTE OPTION 文で設定される次の制御パラメータを使用します。

local_host

ローカルコンピュータの名前。SQL Remote がローカルホスト名を決定できないコンピュータで設定するのに便利です。ローカルホスト名は、任意の SMTP サーバとのセッションを開始するのに必要です。ほとんどのネットワーク環境では、ローカルホスト名が自動的に決定されるため、このエントリは不要です。

top_supported

受信メッセージを列挙するときに、SQL Remote は TOP という POP3 コマンドを使用します。TOP コマンドは、すべての POP サーバでサポートされているわけではありません。top_supported パラメータを NO に設定すると、SQL Remote は RETR コマンドを使用します。このコマンドは TOP よりも効率は落ちますが、すべての POP サーバで動作します。デフォルトは YES です。

smtp_authenticate

SMTP リンクがユーザを認証するかどうかを決定します。デフォルト値は YES です。SMTP 認証を無効にする場合は、このパラメータを NO に設定します。

smtp_userid

SMTP 認証のユーザ ID。デフォルトでは、このパラメータは pop3_userid パラメータと同じ値を取ります。smtp_userid は、ユーザ ID が POP サーバ上のユーザ ID と異なる場合にのみ設定する必要があります。

smtp_password

SMTP 認証のパスワード。デフォルトでは、このパラメータは pop3_password パラメータと同じ値を取ります。smtp_password は、ユーザ ID が POP サーバ上のユーザ ID と異なる場合にのみ設定する必要があります。

smtp_host

SMTP サーバが動作しているコンピュータの名前。SMTP/POP3 ログインウィンドウの SMTP ホストフィールドに対応しています。

smtp_port

SMTP サーバの受信対象ポートの番号。デフォルトは 25 です。

pop3_host

POP ホストを実行しているコンピュータの名前。一般的には、SMTP ホストと同じ名前です。SMTP/POP3 ログインウィンドウの POP3 ホストフィールドに対応しています。

pop3_userid

メールの受信に使用するユーザ ID。POP ユーザ ID は、SMTP/POP3 ログインウィンドウのユーザ ID フィールドに対応しています。必ず POP ホスト管理者からユーザ ID を取得してください。

pop3_password

メールの受信に使用するパスワード。SMTP/POP3 ログインウィンドウのパスワードフィールドに対応しています。

pop3_port

POP サーバの受信対象ポートの番号。デフォルトは 110 です。

debug

YES に設定すると、SMTP と POP3 のすべてのコマンドと応答が出力ログに表示されます。この情報は、SMTP/POP のサポート問題のトラブルシューティングに使用できます。デフォルトは NO です。

suppress_dialogs

このパラメータは、ON または OFF に設定されます。このパラメータが OFF に設定されている場合は、メールサーバへの接続の試行失敗後に、接続ウィンドウは表示されません。代わりに、エラーが発生します。

encode_dll

カスタムエンコードスキームを実装している場合は、作成したカスタムエンコード DLL のフルパスをこの値に設定する必要があります。

max_retries

デフォルトでは、SQL Remote が継続モードで実行されていて、メッセージシステムにアクセスするときにエラーが発生する場合、送受信フェーズの後に停止します。このパラメータを使用して、停止前に SQL Remote が再試行する送受信フェーズの回数を指定します。

pause_after_failure

このパラメータが使えるのは、max_retries がゼロ以外の値に指定され、SQL Remote が継続モードで実行されている場合です。メッセージシステムでエラーが発生すると、このパラメータは、SQL Remote が送受信フェーズを再試行する間に待機する時間を秒で指定します。

このセクションの内容:

SMTP/POP アドレス共有 [133 ページ]
SQL Remote メッセージには専用の電子メールアカウントを使用してください。個人的な電子メールメッセージまたは業務上の電子メールメッセージに使用する同じ電子メールアカウントで SQL Remote メッセージを送受信することはお奨めしません。

SMTP リンクのトラブルシューティング [133 ページ]
SMTP リンクが動作しない場合は、SQL Remote Message Agent (dbremote) が稼働している同じコンピュータから、同じアカウントとパスワードを使用して SMTP/POP3 サーバに接続します。

関連情報

FILE メッセージシステム [123 ページ]
FTP メッセージシステム [124 ページ]
メッセージサイズ [115 ページ]
SET REMOTE OPTION 文 [SQL Remote] [242 ページ]

1.4.7.9.1 SMTP/POP アドレス共有

SQL Remote メッセージには専用の電子メールアカウントを使用してください。個人的な電子メールメッセージまたは業務上の電子メールメッセージに使用する同じ電子メールアカウントで SQL Remote メッセージを送受信することはお奨めしません。

SQL Remote メッセージと通常の電子メールメッセージで同じ電子メールアカウントを共有する必要がある場合は、電子メールプログラムがメールサーバからのすべてのメッセージ (SQL Remote の電子メールメッセージと個人的なメッセージを含む) をダウンロードして削除しないようにします。通常の電子メールメッセージのダウンロード時に、SQL Remote メッセージを変更したり、削除したりしないように電子メールプログラムを設定してください。SQL Remote メッセージの件名には、---SQL Remote-- という文字が含まれています。

1.4.7.9.2 SMTP リンクのトラブルシューティング

SMTP リンクが動作しない場合は、SQL Remote Message Agent (dbremote) が稼働している同じコンピュータから、同じアカウントとパスワードを使用して SMTP/POP3 サーバに接続します。

SMTP/POP3 をサポートするインターネット電子メールプログラムを使用します。SMTP メッセージリンクが動作することがわかったら、このプログラムを無効にします。

電子メールが正しく動作していることの確認

SQL Remote メッセージが適切に送受信されない場合、電子メールメッセージシステムを使用しているときは、2 台のコンピュータの間で電子メールが正しく動作していることを確認してください。

1.4.8 SQL Remote システムバックアップ

バックアップアクティビティを実行する前に、多くの考慮事項を認識している必要があります。

SQL Remote レプリケーションは、トランザクションログ内のオペレーションへのアクセスと、古いトランザクションログへのアクセスに依存して行われます。実装するバックアップ方式には、SQL Remote のトランザクションログの管理を組み込んでください。

バックアップアクティビティを実行する場合は、BACKUP DATABASE システム権限が必要です。

現在のトランザクションログと古いトランザクションログが不要になるまでは、SQL Remote Message Agent (dbremote) がこれらのログにアクセスできるようにしてください。

統合データベースがそのトランザクションログを必要としなくなるのは、すべてのリモートデータベースが受信を完了し、トランザクションログに含まれるメッセージが正常に適用されたことを確認したときです。

リモートデータベースがそのトランザクションログを必要としなくなるのは、統合データベースが受信を完了し、トランザクションログに含まれるメッセージを正常に適用したことを確認したときです。

リモートデータベースのバックアップ

リモートデータベースの場合、次のいずれかを選択する必要があります。

バックアップ方法として統合データベースへのレプリケーションに頼る

リモートデータベースでは、バックアップ手順は統合データベースの場合ほど重要ではありません。データのバックアップを、統合データベースへのレプリケーションに頼る方法もあります。

この方法を選択する場合は、リモートデータベースのトランザクションログを管理する方式を作成してください。

リモートデータベースのバックアップ方式を作成する

リモートデータベースに行われた変更が重要な場合は、トランザクションログの管理を含むリモートデータベースのバックアップ方式を作成する必要があります。

統合データベースのバックアップ

トランザクションログの管理を含む統合データベースのバックアップ方式が必要です。

バックアップユーティリティ (dbbackup) と SQL Remote Message Agent (dbremote) の -x オプション

データベースでは、-x オプションを指定した SQL Remote Message Agent (dbremote) とバックアップユーティリティ (dbbackup) の両方を実行しないでください。

-x オプションは、レプリケーションのためのトランザクションログの管理に使用されます。-x オプションを使用すると、SQL Remote Message Agent は古いトランザクションログにアクセスし、これらのトランザクションログが不要になると削除します。-x オプションでは、トランザクションログはバックアップされません。

バックアップユーティリティ (dbbackup) は、現在のトランザクションログのバックアップに使用されます。バックアップユーティリティ (dbbackup) が -r オプションと -n オプションを使用して実行される場合、バックアップユーティリティは、現在のトランザクションログをバックアップディレクトリにバックアップし、現在のトランザクションログの名前を変更して再起動します。バックアップユーティリティ (dbbackup) は、前回のバックアップ後に名前を変更して再起動したトランザクションログと現在のトランザクションログが同じであると想定します。

-x オプションを指定した SQL Remote Message Agent とバックアップユーティリティ (dbbackup) を同じデータベースで実行しようとすると、相互に干渉します。両方が実行されている場合、トランザクションログが失われる可能性があります。

バックアップされていないリモートデータベースでは、-x オプションを指定した SQL Remote Message Agent (dbremote) のみを実行してください。

このセクションの内容:

関連情報

リモートデータベースのバックアップ (コマンドラインの場合) [137 ページ]

1.4.8.1 リモートデータベースのトランザクションログの管理 (コマンドラインの場合)

統合データベースのレプリケーションによるリモートデータベースのバックアップを行う場合は、リモートデータベーストランザクションログを管理します。つまり、リモートデータベースとトランザクションログではバックアップユーティリティ (dbbackup) を実行しません。

コンテキスト

> ⚠ 警告
>
> バックアップ中のデータベースに対して、-x オプションを指定して SQL Remote Message Agent (dbremote) を実行しないでください。

手順

1. リモートデータベースで、-x オプションを指定して SQL Remote Message Agent (dbremote) を実行し、トランザクションログのサイズを指定します。このオプションにより、トランザクションログが指定したサイズを超えると、SQL Remote Message Agent (dbremote) はトランザクションログの名前を変更して再起動します。

   次のコマンドでは、トランザクションログが 1 MB より大きくなると、削除されます。

   ```
   dbremote -x 1M  -c "UID=ManagerSteve;PWD=passwd;DBF=c:¥mydata.db"
   ```

2. リモートデータベースで、delete_old_logs オプションを On に設定します。delete_old_logs オプションの設定により、古いトランザクションログファイルは、レプリケーションで不要になると、dbremote によって自動的に削除されます。

   トランザクションログが不要になるのは、そのトランザクションログファイルに記録されている変更をすべて受信して正常に適用したという確認を、すべてのサブスクライバから受け取った時点です。delete_old_logs オプションは、PUBLIC ロール用、または dbremote の接続文字列に含まれるユーザ用のいずれかに設定できます。

   次の文では、PUBLIC グループに delete_old_logs を設定して、作成されてから 10 日以上過ぎたトランザクションログを削除します。

   ```
   SET OPTION PUBLIC.delete_old_logs = '10 days';
   ```

結果

データベーストランザクションログは、指定したルールに従って削除されます。

このセクションの内容:

リモートデータベースのバックアップ (コマンドラインの場合) [137 ページ]
レポートデータベースをバックアップします。

1.4.8.1.1 リモートデータベースのバックアップ (コマンドラインの場合)

レポートデータベースをバックアップします。

前提条件

DATABASE BACKUP システム権限が必要です。この手順には、SQL Remote がトランザクションログを使用する場合の管理方式が含まれています。この手順を使用して、-x オプションを指定した SQL Remote Message Agent (dbremote) を実行しないでください。

手順

1. リモートデータベースのフルバックアップを作成します。
   a. データベースに接続します。
   b. -r オプションと -n オプションを指定して dbbackup を実行します。

      たとえば、バックアップディレクトリを `e:¥backup_dir` と想定すると、データベースファイルは `c:¥live` ディレクトリにあり、これに対応するトランザクションログファイルは `d:¥live` フォルダにあります。

      ```
      dbbackup -r -n -c "UID=DBA;PWD=passwd;DBF=c:¥live¥remotedatabase.db" e:
      ¥backup_dir
      ```

      `d:¥live` ディレクトリ内のトランザクションログは、フルバックアップによって変更されません。
   c. `e:¥backup_dir` ディレクトリにあるバックアップファイルを、外部のドライブまたは DVD にコピーします。
   d. 現在のトランザクションログファイルにアクセスしながら SQL Remote Message Agent (dbremote) を実行するには、次のコマンドを使用します。

      ```
      dbremote -c "UID=DBA;PWD=passwd;DBF=c:¥live¥remotedatabase.db" d:¥live
      ```

   > ⚠ 警告
   >
   > バックアップ中のデータベースに対して、-x オプションを指定して SQL Remote Message Agent (dbremote) を実行しないでください。

2. バックアップユーティリティ (dbbackup) を設定し、リモートデータベースのトランザクションログのインクリメンタルバックアップを作成します。
   a. データベースに接続します。
   b. -r オプション、-n オプション、-t オプションを指定して dbbackup を実行します。

次に例を示します。

```
dbbackup -r -n -t -c "UID=DBA;PWD=passwd;DBF=c:¥live¥remotedatabase.db" e:
¥backup_dir
```

c. 現在のトランザクションログファイルにアクセスしながら SQL Remote Message Agent (dbremote) を実行するには、次のコマンドを使用します。

```
dbremote -c "UID=DBA;PWD=passwd;DBF=c:¥live¥remotedatabase.db" d:¥live
```

結果

リモートデータベースがバックアップされます。

1.4.8.2 メディア障害からの統合データベースの保護

SQL Remote では、メディア障害によるデータ損失のリスクを軽減するための多くの方法がサポートされています。

メディア障害から SQL Remote レプリケーションシステムを保護するには、次の手順に従います。

バックアップしたトランザクションのみをレプリケートする

バックアップしたトランザクションのみを含むメッセージを送信します。バックアップしたトランザクションのみを送信することで、トランザクションログ上のメディア障害からレプリケーションシステムを保護できます。これは、次の方法で実現できます。

- -u オプションを指定して SQL Remote Message Agent (dbremote) を実行します。SQL Remote Message Agent (dbremote) が -u オプションを使用して稼働しているときは、バックアップ済みのコミットされたトランザクションのみがメッセージにパッケージされて送信されます。
  さらに、他のサイトでもバックアップを実行すれば、-u オプションによってサイト全体の障害を防ぐこともできます。

トランザクションログミラーを使用する

トランザクションログミラーの使用は、トランザクションログデバイス上のメディア障害から保護します。

統合データベースで **-x** オプションを指定して **SQL Remote Message Agent (dbremote)** を実行しない

バックアップ中のデータベースでは、-x オプションを指定して SQL Remote Message Agent (dbremote) を実行しないでください。-x オプションでは、バックアップやリカバリのためでなく、レプリケーションのためにトランザクションログが管理されます。

関連情報

統合データベースのバックアップ (コマンドラインの場合) [139 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]

1.4.8.3 統合データベースのバックアップ (コマンドラインの場合)

統合データベースとトランザクションログのフルバックアップを作成して統合データベースをバックアップした後、トランザクションログのインクリメンタルバックアップを作成します。

前提条件

BACKUP DATABASE システム権限が必要です。

コンテキスト

> ⚠ 警告
>
> バックアップ中のデータベースに対して、-x オプションを指定して SQL Remote Message Agent (dbremote) を実行しないでください。

手順

1. 統合データベースとそのトランザクションログのフルバックアップを作成します。
   a. データベースに接続します。
   b. -r オプションと -n オプションを指定して dbbackup を実行します。

      次に例を示します。

      ```
      dbbackup -r -n -c "UID=DBA;PWD=passwd;DBF=c:¥live¥database.db" e:¥archive
      ```
2. 統合データベースのトランザクションログのインクリメンタルバックアップを作成します。トランザクションログのバックアップ時に、トランザクションログの名前変更と再起動を選択します。
   a. データベースに接続します。
   b. -r オプション、-n オプション、-t オプションを指定して dbbackup を実行します。

      次に例を示します。

      ```
      dbbackup -r -n -t -c "UID=DBA;PWD=passwd;DBF=c:¥live¥database.db" e:¥archive
      ```
3. 現在のトランザクションログファイルにアクセスしながら SQL Remote Message Agent (dbremote) を実行します。

   次に例を示します。

   ```
   dbremote -c "UID=DBA;PWD=passwd;DBF=c:¥live¥database.db" d:¥live
   ```

結果

統合データベースとトランザクションログがバックアップされます。

例

`c:¥live` ディレクトリ内のデータベース `database.db` と `d:¥live` ディレクトリ内のトランザクションログ `database.log` を検討します。

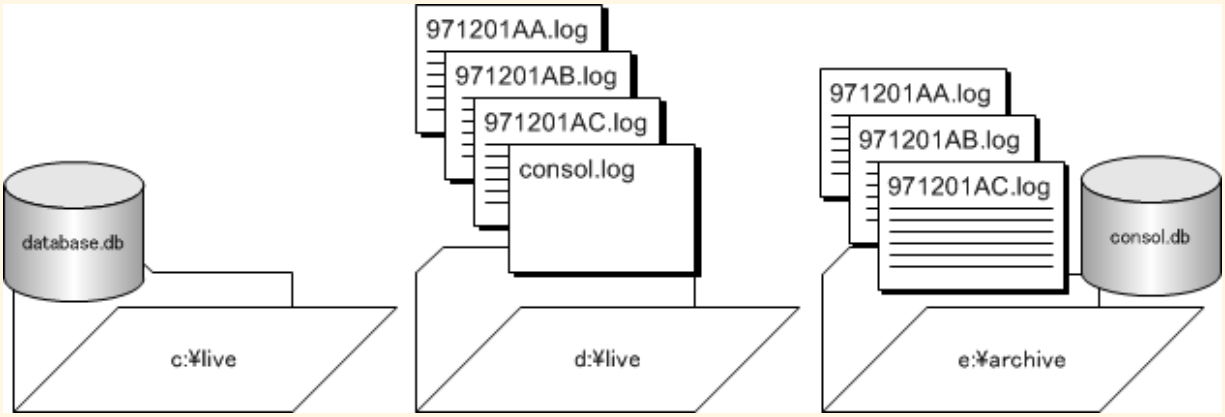
トランザクションログの名前を変更して再起動する -r オプションと -n オプションを使用して、バックアップディレクトリ `e:¥archive` にトランザクションログをバックアップすると、バックアップユーティリティ (dbbackup) が次のタスクを実行します。

1. 現在のトランザクションログファイルの名前を `971201xx.log` (`xx` は `AA` ～ `ZZ` のアルファベット順の英字) に変更します。
2. トランザクションログファイルをバックアップディレクトリにバックアップし、バックアップファイル `971201xx.log` を作成します。

   #### 注記

   リリース 8.0.1 より前の SQL Anywhere では、古いトランザクションログファイルの名前が、`yymmdd01.log`、`yymmdd02.log` のようになっています。名前を変更したのは、古いトランザクションログをより多く保存できるようにするためです。SQL Remote Message Agent (dbremote) は、指定されたフォルダ内で、ファイル名に関係なく全ファイルをスキャンするので、ログファイル名が変わっても既存のアプリケーションに影響はありません。

3. `database.log` という名前で新しいトランザクションログを作成します。
   バックアップを何回か行うと、live フォルダと archive フォルダに連続した名前の一連のトランザクションログができます。



関連情報

メディア障害からの統合データベースの保護 [138 ページ]

1.4.9 統合データベースの手動リカバリ (コマンドラインの場合)

各トランザクションログをデータベースに適用して、統合データベースをリカバリします。

前提条件

BACKUP DATABASE システム権限が必要です。

データベースとトランザクションログファイルのコピーを作成します。この手順では、データベースファイルが事前にバックアップされており、利用可能であることを想定しています。

手順

1. テンポラリディレクトリを作成します。
2. データベース (.db) ファイルの最新のバックアップをリストアします。トランザクションログファイルはリストアしません。

   テンポラリディレクトリで、次の処理を実行します。
   a. データベースのバックアップコピーを開始します。
   b. -a オプションを使用して、古いトランザクションログを適用します。
   c. データベースを停止します。
   d. 現在のトランザクションログと -a オプションを使用してデータベースを起動し、トランザクションを適用してデータベースファイルを最新の状態に更新します。
   e. データベースを停止します。
   f. データベースをバックアップします。
3. データベースを運用ディレクトリにコピーします。
4. データベースを起動します。

   新しいアクティビティは、すべて現行のトランザクションログに追加されます。

結果

統合データベースがリカバリされます。

例

c:¥dbdir¥cons.db という名前の統合データベースファイル、トランザクションログファイル c:¥dbdir¥cons.log、トランザクションログミラーファイル d:¥mirdir¥cons.mlg があると想定します。

毎週実行するフルバックアップに加え、次のコマンドを使用してインクリメンタルバックアップを毎日実行しているものとします。

```
dbbackup -c "UID=DBA;PWD=passwd" -r -n -t e:¥backdir
```

このコマンドは、トランザクションログ cons.log をディレクトリ e:¥backdir にバックアップします。トランザクションログの名前は datexx.log (date はそのときの日付、xx はアルファベット順の次の英字) に変更され、新しいトランザクションログが作成されます。その後、ディレクトリ e:¥backdir がサードパーティユーティリティを使用してバックアップされます。

ここでは、名前を変更したトランザクションログファイルを示すディレクトリを任意で指定して SQL Remote Message Agent (dbremote) を実行します。次に例を示します。

```
dbremote -c "UID=DBA;PWD=passwd" c:¥dbdir
```

毎週実行しているバックアップの 3 日後に、ディスクブロック破損のためにデータベースファイルが破壊されてしまったと想定します。次の手順を実行します。

1. トランザクションログミラーファイル d:¥mirdir¥cons.mlg をバックアップします。
2. リカバリを実行するテンポラリディレクトリを作成します。この例では c:¥recover というディレクトリを作成します。
3. データベースファイル cons.db の最新のバックアップを c:¥recover¥cons.db にリストアします。

   ```
   dbeng17 -a c:¥dbdir¥dateAA.log c:¥recover¥cons.db
   dbeng17 -a c:¥dbdir¥dateAB.log c:¥recover¥cons.db
   ```
4. 名前を変更したトランザクションログを、次の順序で適用します。
5. 現在のトランザクションログ d:¥mirdir¥cons.log を、リカバリディレクトリにコピーし、c:¥recover¥cons.log を作成します。

   ```
   dbeng17 c:¥recover¥cons.db
   ```
6. 次のコマンドを使用してデータベースを起動します。
7. データベースサーバを停止します。
8. リカバリされたデータベースとトランザクションログを c:¥recover からバックアップします。
9. c:¥recover から実際に使用する適切なディレクトリに、ファイルをコピーします。
   - c:¥recover¥cons.db を c:¥dbdir¥cons.db にコピーします。
   - c:¥recover¥cons.log を c:¥dbdir¥cons.log と d:¥mirdir¥cons.mlg にコピーします。

関連情報

統合データベースの自動リカバリ (コマンドラインの場合) [143 ページ]

1.4.10 統合データベースの自動リカバリ (コマンドラインの場合)

統合データベースを自動でリカバリします。

前提条件

BACKUP DATABASE システム権限が必要です。

データベースとトランザクションログファイルのコピーを作成します。この手順では、データベースファイルが事前にバックアップされており、利用可能であることを想定しています。

手順

1. データベース (`.db`) ファイルの最新のバックアップコピーをテンポラリディレクトリにリストアします。トランザクションログファイルはリストアしません。
2. テンポラリディレクトリで、次の処理を実行します。
   a. データベースを起動し、-ad オプションを使用してトランザクションログを適用します。

      -ad オプションを指定すると、データベースサーバは指定されたディレクトリでデータベースのトランザクションログを検索します。次にトランザクションログの正しい順序を特定し、トランザクションログオフセットに基づいてトランザクションログを適用します。
   b. 現在のトランザクションログをテンポラリディレクトリにコピーします。
   c. データベースを起動し、現在のトランザクションログを適用します。
   d. データベースサーバを停止します。
   e. データベースとトランザクションログをバックアップします。
3. データベースとトランザクションログファイルを適切な運用ディレクトリにコピーします。
4. システムを通常どおり再起動します。

   新しいアクティビティは、すべて現行のトランザクションログに追加されます。

結果

統合データベースがリカバリされます。

例

`c:¥dbdir¥cons.db` という名前の統合データベースファイル、トランザクションログファイル `c:¥dbdir¥cons.log`、トランザクションログミラーファイル `d:¥mirdir¥cons.mlg` があると想定します。

次のコマンドを使用して、フルバックアップを毎週実行するとします。

```
dbbackup -c "UID=DBA;PWD=passwd" -r -n e:¥backdir
```

また、次のコマンドを使用して、インクリメンタルバックアップを毎日実行するとします。

```
dbbackup -c "UID=DBA;PWD=passwd" -r -n -t e:¥backdir
```

このコマンドは、トランザクションログ cons.log をディレクトリ e:¥backdir にバックアップします。トランザクションログの名前は datexx.log (date はそのときの日付、xx はアルファベット順の次の英字) に変更され、新しいトランザクションログが作成されます。その後、ディレクトリ e:¥backdir がサードパーティユーティリティを使用してバックアップされます。

ここでは、名前を変更したトランザクションログファイルを示すディレクトリを任意で指定して SQL Remote Message Agent (dbremote) を実行するものとします。次に例を示します。

```
dbremote -c "UID=DBA;PWD=passwd" c:¥dbdir
```

毎週実行しているバックアップの 3 日後に、ディスクブロック破損のためにデータベースファイルが破壊されてしまったと想定します。次の手順を実行します。

1. c:¥ ドライブを交換します。
2. トランザクションログミラーファイル d:¥mirdir¥cons.mlg をバックアップします。
3. リカバリを実行するテンポラリディレクトリを作成します。この例では c:¥recover というディレクトリを作成します。
4. データベースファイル cons.db の最新のバックアップを c:¥recover¥cons.db にリストアします。
5. バックアップ済みのトランザクションログを c:¥dbdir にコピーします。
6. 名前を変更したトランザクションログを適用します。

   ```
   dbeng17 c:¥recover¥cons.db -ad c:¥dbdir
   ```
7. 現在のトランザクションログ d:¥mirdir¥cons.log を、リカバリディレクトリにコピーし、c:¥recover¥cons.log を作成します。
8. 次のコマンドを使用してデータベースを起動します。

   ```
   dbeng17 c:¥recover¥cons.db
   ```
9. データベースサーバを停止します。
10. リカバリされたデータベースとトランザクションログを c:¥recover からバックアップします。
11. c:¥recover から実際に使用する適切なディレクトリに、ファイルをコピーします。
    - c:¥recover¥cons.db を c:¥dbdir¥cons.db にコピーします。
    - c:¥recover¥cons.log を c:¥dbdir¥cons.log と d:¥mirdir¥cons.mlg にコピーします。

関連情報

統合データベースの手動リカバリ (コマンドラインの場合) [141 ページ]

1.4.11 レプリケーションエラーのレポートと処理

SQL Remote には、発生する可能性のあるエラーを処理するための方法が多数用意されています。

SQL Remote システムでは、次のエラーが発生する可能性があります。

- ローが見つからないエラー
- 参照整合性エラー
- 重複プライマリキーエラー

デフォルトでは、エラーが発生すると、SQL Remote Message Agent (dbremote) はエラーをログ出力ウィンドウに出力します。SQL Remote Message Agent (dbremote) は、メッセージウィンドウよりも出力メッセージファイルに多くの情報を出力できます。

SQL Remote Message Agent (dbremote) のメッセージログファイルには、次の情報が含まれます。

- 適用されたメッセージ
- 失敗した SQL 文
- その他のエラー

出力ログファイルにエラーを出力するには、-o オプションを指定して SQL Remote Message Agent (dbremote) を実行します。

エラーが発生したときに、次の処理を実行するように SQL Remote を設定できます。

エラー処理プロシージャの実行

デフォルトではプロシージャを呼び出しません。ただし、replication_error データベースオプションを使用すると、エラーが発生したときに SQL Remote Message Agent (dbremote) で呼び出すストアドプロシージャを指定できます。

たとえば、次の処理を実行するように SQL Remote を設定できます。

- リモートデータベースの出力ログの一部を統合データベースに送信し、ファイルに書き込みます。
- リモートデータベースでエラーが発生したときに電子メールによる通知を送信します。

エラーの無視

SQL Remote Message Agent (dbremote) のエラーをレポートしないようにする場合があります。たとえば、エラーが発生する状況を理解しており、エラーによってデータの不一致が発生しないことが確実である場合は、エラーの無視を選択できます。

このセクションの内容:

エラー処理プロシージャのレプリケーション [146 ページ]
replication_error オプションを設定して、SQL エラーが発生したときにプロシージャを呼び出します。

レプリケーションエラーを無視する方法 [150 ページ]
既知のエラーの原因となる動作に BEFORE トリガを作成します。このトリガは、エラーを通知するものです。

関連情報

リモートデータベースからのエラーの収集 (SQL) [146 ページ]

電子メールによるリモートデータベースのエラーに関する通知の受信 (SQL) [148 ページ]
ローが見つからないエラー [54 ページ]
参照整合性エラー [55 ページ]
重複プライマリキーエラー [57 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]

1.4.11.1 エラー処理プロシージャのレプリケーション

replication_error オプションを設定して、SQL エラーが発生したときにプロシージャを呼び出します。

デフォルトでは、SQL エラーの発生時にプロシージャを呼び出しません。

呼び出すプロシージャでは、データ型が CHAR、VARCHAR、または LONG VARCHAR の引数を 1 つ指定してください。プロシージャは、SQL エラーメッセージで 1 回、エラーの原因となった SQL 文でもう 1 回呼び出されます。

eplication_error オプションを設定するには、次の文を実行します。`remote-user` は SQL Remote Message Agent (dbremote) コマンドのパブリッシャ名、`procedure-name` は SQL エラーが検出されたときに呼び出されるプロシージャです。

```
SET OPTION
remote-user.replication_error
= 'procedure-name';
```

このセクションの内容:

リモートデータベースからのエラーの収集 (SQL) [146 ページ]
リモートデータベースの出力ログの部分を統合データベースに送信します。情報はファイルに書き込まれます。このファイルには、システム内の一部またはすべてのリモートデータベースからの出力ロギング情報を格納できます。

電子メールによるリモートデータベースのエラーに関する通知の受信 (SQL) [148 ページ]
リモートデータベースでエラーが発生したときに電子メールによる通知を送信します。電子メールまたはページングシステムを使用して、通知を受信できます。

1.4.11.1.1 リモートデータベースからのエラーの収集 (SQL)

リモートデータベースの出力ログの部分を統合データベースに送信します。情報はファイルに書き込まれます。このファイルには、システム内の一部またはすべてのリモートデータベースからの出力ロギング情報を格納できます。

手順

1. 統合データベースに出力ログ情報を送信するようにリモートデータベースを設定します。
   a. SET REMOTE 文と output_log_send_on_error オプションを使用して、エラーの発生時にログ情報を送信します。

リモートデータベースに対して、次の文を実行します。

```
SET REMOTE link-name OPTION
PUBLIC.output_log_send_on_error = 'Yes';
```

SQL Remote Message Agent (dbremote) は、エラーを示す E で始まるメッセージを読み込むと、統合データベースに出力ログ情報を送信します。

b. この手順は省略可能です。統合データベースに送信される情報の量を制限する場合は、SET REMOTE 文に output_log_send_limit オプションを設定します。output_log_send_limit オプションには、統合データベースに送信されるバイト数を指定します。これは、出力ログの最後に表示されます (つまり、最後のエントリです)。デフォルトは 5K です。

output_log_send_limit に最大メッセージサイズを超える値を指定すると、SQL Remote は output_log_send_limit の値を上書きし、最大メッセージサイズ以下のメッセージだけを送信します。

リモートデータベースに対して、次の文を実行します。

```
SET REMOTE link-name OPTION
PUBLIC.output_log_send_limit = '7K';
```

2. ログ情報を受信するように統合データベースを設定します。

統合データベースで、-ro オプションまたは -rt オプションを指定して SQL Remote Message Agent (dbremote) を実行します。

3. この手順は省略可能です。設定をテストする場合は、出力ログ情報を統合データベースに送信する output_log_send_now オプションを設定します。

リモートデータベースで、output_log_send_now オプションを YES に設定します。

次のポーリング時にリモートデータベースは出力ログ情報を送信し、その後、output_log_send_now オプションが NO にリセットされます。

結果

出力ログ情報は、統合データベースに送信されます。

関連情報

電子メールによるリモートデータベースのエラーに関する通知の受信 (SQL) [148 ページ]
SET REMOTE OPTION 文 [SQL Remote] [242 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]

1.4.11.1.2 電子メールによるリモートデータベースのエラーに関する通知の受信 (SQL)

リモートデータベースでエラーが発生したときに電子メールによる通知を送信します。電子メールまたはページングシステムを使用して、通知を受信できます。

手順

1. ユーザ Cons として統合データベースに接続します。
2. エラーが発生したことを電子メールで DBA ユーザに通知するストアドプロシージャを作成します。

   たとえば、次の文を実行して sp_LogReplicationError プロシージャを作成します。

```
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. 電子メールの送信を管理するストアドプロシージャを作成します。

   たとえば、次の文を実行して sp_notifiy_DBA プロシージャを作成します。

```
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. エラーの発生を電子メールで DBA に通知するプロシージャを呼び出す replication_error データベースオプションを設定します。

たとえば、次の文を実行して、エラーの発生時に sp_LogReplicationError プロシージャを呼び出します。

```
SET OPTION PUBLIC.replication_error =
 'cons.sp_LogReplicationError';
```

5. 監査テーブルを作成します。

   たとえば、次の文を実行して replication_audit テーブルを作成します。

```
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)
);
```

   次の表は、replication_audit テーブルのカラムを示します。

カラム	説明
pub	データベースの現在のパブリッシャ (パブリッシャが挿入されたデータベースを識別する)。
remoteuser	メッセージを適用しているリモートユーザ (リモートユーザのデータベースを識別する)。
errormsg	replication_error プロシージャに渡されたエラーメッセージ。

6. プロシージャをテストします。

   たとえば、リモートデータベースのローと同じプライマリキーを使用するローを統合データベースに挿入します。このローが統合データベースからリモートデータベースにレプリケートされると、プライマリキーの競合エラーが発生し、次の処理が実行されます。

   - リモートデータベースの SQL Remote Message Agent (dbremote) が、その出力ログに次のメッセージを出力します。

```
"cons" (0-0000000000-0) メッセージを受信しました。
SQL 文が失敗しました: (-193) テーブル 'reptable' のプライマリキーがユニークではありません。
INSERT INTO cons.reptable( id,text,last_contact )
VALUES (2,'dave','1997/apr/21 16:02:38.325')
COMMIT WORK
```

   - 統合データベースに次の INSERT 文が送信されます。

```
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;
```

- 次のメッセージを含む電子メールが John Doe と Elton Smith に送信されます。

```
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')
```

結果

リモートデータベースでエラーが発生すると、電子メール通知が送信されます。

関連情報

リモートデータベースからのエラーの収集 (SQL) [146 ページ]

1.4.11.2 レプリケーションエラーを無視する方法

既知のエラーの原因となる動作に BEFORE トリガを作成します。このトリガは、エラーを通知するものです。

たとえば、テーブルで参照先カラムが見つからないときに発生する INSERT 文のエラーを無視する場合は、参照先カラムが存在しないときに SQLE_REMOTE_STATEMENT_FAILED SQLSTATE を通知する BEFORE INSERT トリガを作成します。INSERT 文は失敗しますが、この失敗は SQL Remote Message Agent (dbremote) の出力ログにはレポートされません。

1.4.12 SQL Remote セキュリティに関する考慮事項

SQL Remote には、データを保護するための多数の機能が用意されています。

SQL Remote には、次のようなセキュリティ機能があります。

SYS_RUN_REPLICATION_ROLE システムロール

SYS_RUN_REPLICATION_ROLE システムロールを持つユーザを使用して SQL Remote Message Agent (dbremote) に接続してください。

データベースの暗号化

-ek オプションを使用してデータベースを暗号化できます。

メッセージの暗号化

SQL Remote Message Agent (dbremote) は、単純難読化アルゴリズムを使用して、不意なスヌーピングからメッセージを保護します。しかし、この暗号化スキームでは、強力な暗号解読方法からメッセージを完全に保護することはできません。ネットワーク上のデータのセキュリティが問題である場合は、独自のカスタムエンコードスキームを実装できます。

関連情報

抽出ユーティリティ (dbxtract) [219 ページ]

1.4.13 アップグレードと再同期

さまざまな方法で SQL Remote データベースをアップグレードおよび再同期できます。

SYS_RUN_REPLICATION_ROLE システムロールが必要です。

SQL Remote システムをアップグレードするときは、次のことに注意してください。SQL Remote システムは、次のいずれかの方法でアップグレードできます。

ソフトウェアのアップグレード

ソフトウェアをアップグレードします。
データベーススキーマの変更

データベーススキーマを変更する場合は、次のことを実行できます。

パススルーモードの使用

パススルーモードによって、スキーマの変更を SQL Remote システム内の一部またはすべてのデータベースに送信することが可能です。しかし、パススルーモードの使用には慎重な計画と実行が必要です。
サブスクリプションの再同期

再同期させるには、データの新しいコピーをリモートデータベースにコピーする必要があります。リモートデータベースが多数ある場合は、再同期に時間がかかり、作業の中断やデータの消失が発生する恐れがあります。

このセクションの内容:

実行中のシステムに行ってはならない変更 [151 ページ]
配備済で実行中の SQL Remote システムに変更を加える場合は、認識しておくべき制限事項がいくつかあります。

関連情報

SQL Remote のパススルーモード [152 ページ]
サブスクリプションの再同期 [155 ページ]

1.4.13.1 実行中のシステムに行ってはならない変更

配備済で実行中の SQL Remote システムに変更を加える場合は、認識しておくべき制限事項がいくつかあります。

次の変更は、配備済みで実行中の SQL Remote システムには行わないでください。ただし、条件が記述されている場合は除きます。

パブリッシャの変更

配備済みで実行中の SQL Remote システムの統合データベースでパブリッシャのユーザ名を変更すると、問題が発生する可能性があります。統合データベースのパブリッシャのユーザ名を変更する必要がある場合は、SQL Remote システムを停止し、すべてのリモートユーザを再同期してください。

リモートデータベースでパブリッシャのユーザ名を変更すると、情報が失われるなど、そのリモートデータベースが関連するサブスクリプションに問題が発生します。リモートデータベースのパブリッシャのユーザ名を変更する必要がある場合は、リモートデータベースを停止し、そのリモートユーザを再同期します。

テーブルに対して行う制限のある変更

テーブルに対して制限のある変更を行うことはできません。たとえば、カラムを削除したり、カラムを変更して NULL 値を使用不可にしたりしないでください。これらのカラムを参照するメッセージがシステム内に存在する可能性があるためです。

パブリケーションの変更

パブリケーションの定義は、統合データベースとリモートデータベースの両方で管理されます。SQL Remote システムの実行中にパブリケーションを変更すると、レプリケーションエラーが発生し、レプリケーションシステムのデータが失われる可能性があります。

サブスクリプションの削除

サブスクリプションは削除できますが、パススルーモードを使用してリモートデータベースのデータを削除する必要があります。

データベースのアンロードと再ロード

トランザクションログが正しく管理されていることを確認してください。

多層階層で行う変更

多層階層に変更を加えるには、dbxtract またはデータベース抽出ウィザードを使用します。

関連情報

SQL Remote のパススルーモード [152 ページ]
パススルーモードの制限事項 [153 ページ]
サブスクリプションの再同期 [155 ページ]
多層階層システムのデータベースの抽出 [89 ページ]

1.4.14 SQL Remote のパススルーモード

パススルーモードを使用して、標準 SQL 文をそれらの実行が可能なリモートデータベースに渡します。

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

パススルーモードを使用すると、SQL Remote システムの実行時に、次のタスクを完了できます。

- 新しいユーザを追加します。
- ユーザを再同期します。

- システムからユーザを削除します。
- リモートユーザのアドレス、メッセージタイプ、または頻度を変更します。
- テーブルにカラムを追加します。

> ⚠ 警告
>
> - SQL Remote は、システム内の各データベースが同じオブジェクトを持っているものとして動作します。あるテーブルが一部のサイトのみで変更されたときに、データの変更をレプリケートしようとしても失敗します。SQL Remote システムの実行時にスキーマの追加変更を実行すると、問題が発生する場合があります。
> - パススルーオペレーションは、サブスクライブされたリモートデータベースのコピーがある統合データベースのコピーに対してテストしてください。テストを行っていないパススルースクリプトを、運用データベースで実行しないでください。
> - 所有者名でオブジェクト名を修飾します。PASSTHROUGH 文は、同じユーザ名のリモートデータベースでは実行されません。所有者名の修飾子を持たないオブジェクト名は正しく解析されないことがあります。

このセクションの内容:

関連情報

実行中のシステムに行ってはならない変更 [151 ページ]

1.4.14.1 パススルーモードの制限事項

パススルーモードには、認識しておくべき多くの制限事項があります。

階層の 1 つのレベルのみで動作するパススルー

多層 SQL Remote システムでは、現在のレベルのすぐ下でパススルー文が動作することが重要になります。多層システムでは、統合データベースで、その下のレベルを動作の対象としてパススルー文を入力してください。

プロシージャの呼び出し

パススルーモードで CALL 文または EXEC 文を使用してストアドプロシージャを呼び出すと、次のようになります。

- プロシージャが統合データベースで実行されない場合でも、プロシージャはパススルーコマンドを呼び出す統合データベースに存在します。
- プロシージャは、リモートデータベースにも存在します。CALL 文または EXEC 文はレプリケートされますが、プロシージャ内の文はレプリケートされません。レプリケートされるデータベースのプロシージャは適切な作用をしていると仮定しています。

制御文

IF や LOOP などの制御文と、すべてのカーソル処理は、パススルーモードではレプリケートされません。ループ構造または制御構造内の文は、レプリケートされます。

カーソル処理

カーソルの処理はレプリケートされません。

SQL SET OPTION 文

静的な Embedded SQL の SET OPTION 文はレプリケートされません。しかし、動的 SQL 文はレプリケートされます。

たとえば、次の文はパススルーモードではレプリケートされません。

```
EXEC SQL SET OPTION ...
```

しかし、次の動的 SQL 文はレプリケートされます。

```
EXEC SQL EXECUTE IMMEDIATE "SET OPTION ... "
```

バッチ文

バッチ文 (BEGIN と END に囲まれた一連の文) は、パススルーモードではレプリケートされません。パススルーモードでバッチ文を使おうとすると、エラーが発生します。

関連情報

サブスクリプションの再同期 [155 ページ]

1.4.14.2 パススルーモードの開始と停止

パススルーモードの開始には PASSTHROUGH 文を使用し、停止には PASSTHROUGH STOP 文を使用します。パススルーセッションは、これらの PASSTHROUGH 文の間に入力された文のことを指します。

パススルーセッションに入力された文は、次のように処理されます。

- 構文エラーがチェックされます。
- ONLY キーワードを指定しない場合は、統合データベースで実行されます。ONLY が指定されている場合は、文は統合データベースで実行されずに、リモートデータベースに送信されます。
  次の文を使用すると、現在のデータベースで実行されずに、指定した 2 つのサブスクライバのリストに文を渡すパススルーセッションが開始されます。

  ```
  PASSTHROUGH ONLY
  FOR userid_1, userid_2;
  ```
- 識別されたサブスクライバのデータベースに渡されます。パススルー文は、通常のレプリケーションメッセージと一緒に、文がトランザクションログに記録された順序で連続してレプリケートされます。
- サブスクライバのデータベースで実行されます。

パススルー文の指示

次の文は、pubname パブリケーションに対してサブスクライブされたすべてのユーザに文を渡すパススルーセッションを開始します。

```
PASSTHROUGH ONLY
FOR SUBSCRIPTION TO [owner].pubname statement1;
```

パススルーモードは、加算的です。次の例では、statement_1 は user_1 に送られ、statement_2 は user_1 と user_2 の両方に送られます。

```
PASSTHROUGH ONLY FOR user_1 ;
statement_1;
PASSTHROUGH ONLY FOR user_2 ;
statement_2;
```

次の文は、すべてのリモートユーザのパススルーセッションを停止します。

```
PASSTHROUGH STOP;
```

データ操作言語 (DML)

パススルーモードは、データ操作言語文を送信するために使用します。この場合、レプリケートされた DML 文は、パススルーの前に *before* スキーマを使用し、パススルーの後に *after* スキーマを使用します。

次の例は、リモートデータベースおよび統合データベースでテーブルを削除します。

```
-- Drop a table on the remote database
-- and at the consolidated database
PASSTHROUGH FOR Joe_Remote;
DROP TABLE CrucialData;
PASSTHROUGH STOP;
```

次の例は、PASSTHROUGH ONLY を使用して、リモートデータベースでのみテーブルを削除します。

```
-- Drop a table on the remote database only
PASSTHROUGH ONLY FOR Joe_Remote;
DROP TABLE CrucialData;
PASSTHROUGH STOP;
```

1.4.15 サブスクリプションの再同期

リモートデータベースを作成する場合は、統合データベースからスキーマとデータの両方を抽出し、リモートデータベースの構築に使用します。この処理により、各データベースにデータの初期コピーが確実に格納されます。

配備の後、次の状況ではサブスクリプションの再同期を検討することがあります。

統合データベースに重要な管理作業を実行した後

たとえば、統合データベースに変更を行い、これによりデータベース内のすべてのローが更新されます。デフォルトでは、SQL Remote は更新メッセージを作成し、サブスクライブされている各リモートに送信します。これらの更新メッセージには、ローごとに UPDATE 文、DELETE 文、INSERT 文が含まれる場合があります。

SYNCHRONIZE SUBSCRIPTION 文を使用してサブスクリプションを同期させるよう選択した場合は、サブスクライブされたテーブル内のすべてのローを削除するために必要な文と、すべての新しいローを挿入する INSERT 文の送信のみを行います。

リモートデータベースが統合データベースと同調していない場合

リモートデータベースが統合データベースと同調していない場合は、パススルーモードの使用を試みることができます。

パススルーモードが機能しない場合は、サブスクリプションを同期させることができます。サブスクリプションを同期させる場合は、リモートデータベースを統合データベースと強制的に同調させます。SYNCHRONIZE SUBSCRIPTION 文には、リモートデータベース内のサブスクライブされたテーブルの内容を削除する文と、サブスクリプションのローを統合データベースからリモートデータベースに挿入する文が含まれています。

制限事項

同期はサブスクリプション全体に適用される

1 つのテーブルを同期させることはできません。

同期でのデータの消失

サブスクリプションの一部を成し、統合データベースにレプリケートされなかったリモートデータベースのデータは、失われます。

SQL Central のデータベースアンロードウィザードまたはアンロードユーティリティ (dbunload) を使用してリモートデータベースをアンロードまたはバックアップしてから、データベースを同期させます。

このセクションの内容:

関連情報

SQL Remote のパススルーモード [152 ページ]

1.4.15.1 サブスクリプションの同期 (SQL Central)

サブスクライブされたユーザのパブリケーションを手動で同期します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

抽出ユーティリティ (dbxtract) またはデータベース抽出ウィザードのいずれかを使用して、指定したリモートデータベース用のデータを抽出してから、そのデータをリモートデータベースに手動でロードします。

> ⚠ 警告
>
> 抽出ユーティリティ (dbxtract) またはデータベース抽出ウィザードの実行中は、SQL Remote Message Agent (dbremote) を実行しないでください。

手順

1. リモートデータベースと統合データベースで SQL Remote Message Agent を停止します。
2. 統合データベースに接続します。
3. パブリケーションをダブルクリックします。
4. パブリケーションをダブルクリックします。
5. *SQL Remote* サブスクリプションタブをクリックします。
6. サブスクリプションを手動で同期します。
   a. サブスクライバリストでユーザを右クリックして、プロパティをクリックします。
   b. 詳細タブをクリックします。
   c. すぐに同期化をクリックします。

      すぐに同期化ボタンをクリックすると、サブスクリプションが処理されます。プロパティウィンドウでキャンセルを続けてクリックしても、同期アクションはキャンセルされません。
7. *OK* をクリックします。

結果

サブスクリプションが同期されます。

関連情報

抽出ユーティリティ (dbxtract) [219 ページ]

1.4.15.2 SQL Remote Message Agent (dbremote) を使用した同期

サブスクライブされたテーブルの現在の内容を新しいコピーに置き換えます。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

抽出ユーティリティ (dbxtract) または データベース抽出ウィザード を使用して、サブスクリプションを同期します。

多数のサブスクリプションを抽出したり、サブスクリプションを大規模で使用頻度の高いテーブルに同期させたりすると、データベースにアクセスするときの処理速度が低下します。SEND AT 句を使用すると、時間を指定して統合データベースへのアクセスが少ないときに同期させることができます。

手順

1. 統合データベースに接続します。
2. SYNCHRONIZE SUBSCRIPTION 文を実行します。

   統合データベースの SQL Remote Message Agent (dbremote) は、サブスクリプション内のすべてのローのコピーをサブスクライバに送信します。SQL Remote Message Agent (dbremote) は、適切なデータベーススキーマがリモートデータベースに設定されていると想定します。

   サブスクライバデータベースの SQL Remote Message Agent (dbremote) は、同期メッセージを受信し、サブスクライブされているテーブルの現在の内容を新しいコピーに置き換えます。

結果

サブスクライブされたテーブルの現在の内容が、新しいコピーに置き換えられます。

関連情報

送信頻度 [95 ページ]
サブスクリプションの同期 (SQL Central) [157 ページ]

1.4.15.3 サブスクリプションの開始 (SQL Central)

サブスクライブされたユーザのパブリケーションに対するサブスクリプションを開始します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

1 つのトランザクション内でいくつかのサブスクリプションを開始するには、REMOTE RESET 文を使用します。

手順

1. SQL Central で、*SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. パブリケーションをダブルクリックします。
3. パブリケーションをダブルクリックします。
4. *SQL Remote* サブスクリプションタブをクリックします。
5. サブスクリプションを手動で同期します。
   a. サブスクライバリストでユーザを右クリックして、プロパティをクリックします。
   b. 詳細タブをクリックします。
   c. すぐに同期化をクリックします。

      すぐに同期化ボタンをクリックすると、サブスクリプションが処理されます。プロパティウィンドウでキャンセルを続けてクリックしても、同期アクションはキャンセルされません。

結果

サブスクライブされたユーザのパブリケーションに対するサブスクリプションが開始されます。

1.4.15.4 サブスクリプションの停止 (SQL Central)

ユーザのサブスクリプションをキャンセルします。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. *SQL Anywhere17* プラグインを使用してデータベースに接続します。
2. パブリケーションをダブルクリックします。
3. 希望するパブリケーションをダブルクリックします。
4. *SQL Remote* サブスクリプションタブをクリックします。
5. サブスクリプションを手動で同期する場合は、サブスクライバリストでユーザを右クリックして、プロパティをクリックします。

   詳細タブをクリックします。このタブで、すぐに停止をクリックしてサブスクリプションを停止します。

   すぐに停止ボタンをクリックすると、サブスクリプションが処理されます。プロパティウィンドウでキャンセルを続けてクリックしても、停止アクションはキャンセルされません。

結果

サブスクリプションが停止します。

1.5 チュートリアル: SQL Remote システムの作成

このチュートリアルでは、SQL Remote レプリケーションシステムの設定方法について学びます。

前提条件

このチュートリアルはユーザ、DBA を対象としています。このユーザは、SYS_REPLICATION_ADMIN_ROLE システムロールを持っている必要があります。

コンテキスト

このチュートリアルでは、次の手順を実行します。

1. レッスン 1: 統合データベースの作成 [161 ページ]
   チュートリアル用の統合データベースとディレクトリを作成します。
2. レッスン 2: 統合データベースでの PUBLISH 権限と REMOTE 権限の付与 [164 ページ]
   Interactive SQL を使用して統合データベースのパブリッシャを作成します。
3. レッスン 3: パブリケーションとサブスクリプションの作成 [165 ページ]
   Interactive SQL を使用して、統合データベースでパブリケーションを作成します。
4. レッスン 4: SQL Remote のメッセージタイプの作成 [167 ページ]
   レプリケーションのためのデータやメッセージを送信するときに使用するメッセージタイプを定義します。
5. レッスン 5: リモートデータベースの抽出 [168 ページ]
   統合データベース (hq) からリモートデータベースを抽出して、リモートユーザ用のデータベースを作成します。
6. レッスン 6: 統合データベースからリモートデータベースへのデータの送信 [170 ページ]
   Interactive SQL を使用して、統合データベース (hq) からリモートデータベース (field) にデータをレプリケートします。
7. レッスン 7: リモートデータベースでのデータの受信 [171 ページ]
   統合データベース (hq) から送信されたデータを、リモートデータベース (field) で受信します。
8. レッスン 8: リモートデータベースから統合データベースへのデータの送信 [173 ページ]
   Interactive SQL を使用して、リモートデータベース (field) から統合データベース (hq) にデータをレプリケートします。

1.5.1 レッスン 1: 統合データベースの作成

チュートリアル用の統合データベースとディレクトリを作成します。

前提条件

DBA ユーザには、SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. ディレクトリ `c:¥tutorial`、`c:¥tutorial¥hq`、および `c:¥tutorial¥field` を作成します。
2. `c:¥tutorial` ディレクトリから次のコマンドを実行して、統合データベース (hq) を作成します。

```
dbinit -dba DBA,passwd hq.db
```

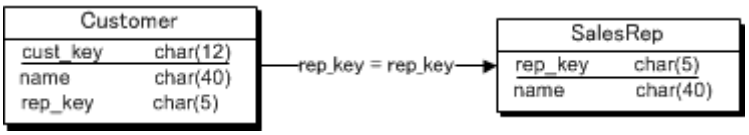
3. Interactive SQL から統合データベース (hq) に接続します。

```
dbisql -c "UID=DBA;PWD=passwd;SERVER=server_hq;DBF=c:¥tutorial¥hq.db"
```

4. 次の文を実行して、統合データベース (hq) 内に 2 つのテーブルを作成します。

```
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)
);
```

次の数字は、チュートリアル用の統合データベース (hq) スキーマを示します。



- 各営業担当は SalesReps テーブルの 1 つのローで表されています。
- 各顧客は Customers テーブルの 1 つのローで表されています。
- 各顧客は 1 人の営業担当に割り当てられており、この割り当ては Customers テーブルから SalesReps テーブルへの外部キーとしてデータベースに組み込まれています。Customers テーブルと SalesReps テーブルとの関係は、多対 1 です。

テーブル名	説明
SalesRep	SalesReps テーブルには、会社の営業担当それぞれに 1 つのローがあります。SalesReps テーブルには、次のカラムがあります。 **rep_key** 各営業担当の識別子。これがプライマリキーです。 **name** 各営業担当の名前。

テーブル名	説明
Customers	Customers テーブルには、会社と取引がある顧客それぞれに 1 つのローがあります。Customers テーブルには、次のカラムがあります。 **cust_key** 各顧客の識別子。これがプライマリキーです。 **name** 各顧客の名前。 **rep_key** 取引を行う営業担当者の識別子。これが SalesReps テーブルへの外部キーです。

5. 次の文を実行して、SalesReps および Customers テーブルにサンプルデータを追加します。

```
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. 次の文を実行して、テーブルが作成されたことを確認します。

```
SELECT * FROM SalesReps;
```

上記のクエリは、SalesReps テーブルから次のデータを返します。

rep_key	name
rep1	Field User
rep2	Another User

```
SELECT * FROM Customers;
```

上記のクエリは、Customers テーブルから次のデータを返します。

cust_key	name	rep_key
cust1	Ocean Sports	rep1
cust2	Sports Plus	rep2

結果

テーブルが作成され、データが埋め込まれます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: SQL Remote システムの作成 [160 ページ]

次のタスク: レッスン 2: 統合データベースでの PUBLISH 権限と REMOTE 権限の付与 [164 ページ]

1.5.2 レッスン 2: 統合データベースでの PUBLISH 権限と REMOTE 権限の付与

Interactive SQL を使用して統合データベースのパブリッシャを作成します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

SQL Remote システムのすべてのデータベースにパブリッシャが必要です。パブリッシャは、PUBLISH 権限を持つユニークなユーザです。パブリケーションの更新と受信確認を含む、SQL Remote のすべての出力メッセージは、パブリッシャによって識別されます。SQL Remote システムのすべてのデータベースは、受信確認を送信します。

手順

1. 現在、統合データベース (hq) に接続されていない場合は、次のコマンドを実行します。

   ```
   dbisql -c "UID=DBA;PWD=passwd;SERVER=server_hq;DBF=c:¥tutorial¥hq.db"
   ```

2. 次の文を実行して、CONNECT 権限と PUBLISH 権限を持つユーザ hq_user を作成します。

   ```
   CREATE USER hq_user IDENTIFIED BY hq_pwd;
   GRANT CONNECT TO hq_user IDENTIFIED BY hq_pwd;
   GRANT PUBLISH TO hq_user;
   ```

3. 次の文を実行して、データベースのパブリッシュユーザ ID (hq_user) を確認します。

   ```
   SELECT CURRENT PUBLISHER;
   ```

4. 統合データベースなど、他のデータベースにメッセージを送信するデータベースでは、メッセージの送信先となるリモートデータベースを指定する必要があります。統合データベースでこれらのリモートデータベースを指定するには、リモートデ

ータベースのパブリッシャに REMOTE 権限を付与します。REMOTE 権限によって、現在のデータベースからメッセージを受信するデータベースが識別されます。次の文を実行して、CONNECT 権限と REMOTE 権限を持ち、パスワード field_pwd を使用するリモートユーザ field_user を作成します。

```
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';
```

結果

統合データベースおよびリモートデータベースのパブリッシャを識別するユーザが作成されます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: SQL Remote システムの作成 [160 ページ]

前のタスク: レッスン 1: 統合データベースの作成 [161 ページ]

次のタスク: レッスン 3: パブリケーションとサブスクリプションの作成 [165 ページ]

1.5.3 レッスン 3: パブリケーションとサブスクリプションの作成

Interactive SQL を使用して、統合データベースでパブリケーションを作成します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

パブリケーションでは、レプリケートされるデータセットを説明します。このレッスンでは、SalesRepData というパブリケーションを作成します。これは、SalesReps テーブルのすべてのローと、Customers テーブルのいくつかのローをレプリケートします。パブリケーションに対してユーザをサブスクライブするには、サブスクリプションを作成します。

手順

1. 現在、統合データベース (hq) に接続されていない場合は、次のコマンドを実行します。

   ```
   dbisql -c "UID=DBA;PWD=passwd;SERVER=server_hq;DBF=c:¥tutorial¥hq.db"
   ```

2. 次の文を実行して、SalesRepData というパブリケーションを作成します。

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

   SalesRepData パブリケーションは以下をパブリッシュします。

   - SalesReps テーブル全体
   - 指定された rep_key 値と一致するローを除くすべての Customers テーブルのカラム

3. 次の文を実行して、field_user に SalesRepData へのサブスクリプションを作成します。

   ```
   CREATE SUBSCRIPTION
   TO SalesRepData ('rep1')
   FOR field_user;
   ```

   値 rep1 は、SalesReps テーブルの field_user に対する rep_key の値です。

   > **注記**
   >
   > このチュートリアルでは、プライマリキーの値が重複したエントリを防ぐようにはなっていません。

結果

SalesRepData パブリケーションが作成され、field_user に SalesReps テーブルのすべてのローと、Customers テーブルの一部のローがレプリケートされます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: SQL Remote システムの作成 [160 ページ]

前のタスク: レッスン 2: 統合データベースでの PUBLISH 権限と REMOTE 権限の付与 [164 ページ]

次のタスク: レッスン 4: SQL Remote のメッセージタイプの作成 [167 ページ]

関連情報

SQL Remote システムの作成 [11 ページ]

1.5.4 レッスン 4: SQL Remote のメッセージタイプの作成

レプリケーションのためのデータやメッセージを送信するときに使用するメッセージタイプを定義します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

メッセージタイプの記述は、次の 2 つの部分で構成されます。

SQL Remote でサポートされているメッセージシステム

このチュートリアルでは、FILE メッセージシステムを使用します。FILE メッセージシステムは、単純なファイル共有システムです。

FILE アドレス

ユーザの FILE アドレスは、すべての着信メッセージが送信されるサブディレクトリです。アプリケーションはこのディレクトリからメッセージを検索します。このチュートリアルでは、統合データベースの FILE アドレスは hq で、`c:¥tutorial` のサブディレクトリです。

手順

1. 現在、統合データベース (hq) に接続されていない場合は、次のコマンドを実行します。

```
dbisql -c "UID=DBA;PWD=passwd;SERVER=server_hq;DBF=c:¥tutorial¥hq.db"
```

2. 次の文を実行して、FILE メッセージタイプを作成します。

```
CREATE REMOTE MESSAGE
TYPE file
ADDRESS 'hq';
```

結果

FILE メッセージタイプが作成されます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: SQL Remote システムの作成 [160 ページ]

前のタスク: レッスン 3: パブリケーションとサブスクリプションの作成 [165 ページ]

次のタスク: レッスン 5: リモートデータベースの抽出 [168 ページ]

1.5.5 レッスン 5: リモートデータベースの抽出

統合データベース (hq) からリモートデータベースを抽出して、リモートユーザ用のデータベースを作成します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

メッセージの送受信を行い SQL Remote システムに組み込まれるように、リモートデータベースを設定する必要があります。統合データベース (hq) と同様に、出力メッセージのソースを識別するためにリモートデータベースには CURRENT PUBLISHER が必要です。また、サブスクライバとして識別された統合データベース (hq) も必要です。

dbxtract ユーティリティを実行して、以下を含むリモートデータベースを作成します。

- 統合データベースに対するサブスクリプション
- パブリケーション
- データの現在のコピー

手順

1. c:¥tutorial ディレクトリから次のコマンドを実行して、統合データベース (hq) からユーザ field_user のリモートデータベースのスキーマを抽出します。

   ```
   dbxtract -v -c "UID=DBA;PWD=passwd;SERVER=server_hq;DBF=C:¥tutorial¥hq.db" -an c:¥tutorial¥field.db field_user
   ```

   このコマンドは以下を実行します。

   - 統合データベースのリモートユーザに対してサブスクリプションを開始します。
   - c:¥tutorial¥field.db and performs the following tasks: に新しい空のデータベースを作成します。
     - リモートデータベース (field) でメッセージタイプを作成します。
     - リモートデータベース (field) に PUBLISH 権限を付与します。
     - リモートデータベース (field) に SalesReps テーブルと Customers テーブルを作成します。これらのテーブルは、統合データベース (hq) 中のデータと同じデータを含みます。
     - リモートデータベース (field) でレプリケートするデータを示すパブリケーションを作成します。
     - 統合データベース (hq) 用のサブスクリプションを作成し、開始します。

   > ⚠ 警告
   >
   > 運用環境では、同じディレクトリに 2 つのレプリケートデータベースを格納しないでください。

2. 2.次のコマンドを使用して、新しく作成されたリモートデータベースに接続します。

   ```
   dbisql -c "UID=DBA;PWD=passwd;SERVER=field_db;DBF=c:¥tutorial¥field.db"
   ```

3. 次の文を実行して、テーブルが作成されたことを確認します。

   ```
   SELECT * FROM SalesReps;
   ```

   上記のクエリは、SalesReps テーブルから次のデータを返します。

rep_key	name
rep1	Field User
rep2	Another User

   ```
   SELECT * FROM Customers;
   ```

   上記のクエリは、Customers テーブルから次のデータを返します。

cust_key	name	rep_key
cust1	Ocean Sports	rep1

結果

リモートユーザ用のリモートデータベースが作成されます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: SQL Remote システムの作成 [160 ページ]

前のタスク: レッスン 4: SQL Remote のメッセージタイプの作成 [167 ページ]

次のタスク: レッスン 6: 統合データベースからリモートデータベースへのデータの送信 [170 ページ]

1.5.6 レッスン 6: 統合データベースからリモートデータベースへのデータの送信

Interactive SQL を使用して、統合データベース (hq) からリモートデータベース (field) にデータをレプリケートします。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

次のとおりです。

手順

1. 現在、統合データベース (hq) に接続されていない場合は、次のコマンドを実行します。

```
dbisql -c "UID=DBA;PWD=passwd;SERVER=server_hq;DBF=c:¥tutorial¥hq.db"
```

2. 次の文を実行して、SalesReps および Customers テーブルにサンプルデータを追加します。

```
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. 次の文を実行して、データが入力されたことを確認します。

```
SELECT * FROM SalesReps;
SELECT * FROM Customers;
```

4. リモートデータベース (field) にローを送信するには、`c:¥tutorial` ディレクトリの統合データベースで Message Agent を実行します。

```
dbremote -c "UID=DBA;PWD=passwd;SERVER=server_hq;DBF=c:¥tutorial¥hq.db"
```

5. [Message Agent] ウィンドウに `Execution Completed` と表示されたら、シャットダウンをクリックします。
6. `c:¥tutorial¥field` を参照します。

   ファイル名 `hq.0` がディレクトリにリストされています。このファイルは、統合データベース (hq) から送信された変更を含みます。

結果

SalesReps テーブルと Customers テーブルにサンプルデータが追加され、統合データベースからリモートデータベースに送信されます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: SQL Remote システムの作成 [160 ページ]

前のタスク: レッスン 5: リモートデータベースの抽出 [168 ページ]

次のタスク: レッスン 7: リモートデータベースでのデータの受信 [171 ページ]

1.5.7 レッスン 7: リモートデータベースでのデータの受信

統合データベース (hq) から送信されたデータを、リモートデータベース (field) で受信します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. 現在、リモートデータベース (field) に接続されていない場合は、次のコマンドを実行します。

   ```
   dbisql -c "UID=DBA;PWD=passwd;SERVER=field_db;DBF=c:¥tutorial¥field.db"
   ```

2. リモートデータベース (field) で `c:¥tutorial` ディレクトリから Message Agent を実行します。

   ```
   dbremote -c "UID=DBA;PWD=passwd;SERVER=field_db;DBF=c:¥tutorial¥field.db;"
   ```

3. [Message Agent] ウィンドウに `Execution Completed` と表示されたら、シャットダウンをクリックします。

   `c:¥tutorial¥field¥hq.0` ファイルは `c:¥tutorial¥hq¥field.0` というファイルに置き換えられました。`field.0` ファイルは受信確認を含みます。

4. 次の方法でリモートデータベース (field) がデータを含んでいることを確認してください。

   a. 次の文を実行して、SalesReps テーブルをの内容を表示します。

      ```
      SELECT * FROM SalesReps;
      ```

      統合データベース (hq) で入力したローが両方とも SalesReps テーブルにあります。これは、SalesRepsData パブリケーションに SalesReps テーブルからのすべてのデータが含まれていたからです。

rep_key	name
rep1	Field User
rep2	Another User
rep3	Example User

   b. 次の文を実行して、Customers テーブルをの内容を表示します。

      ```
      SELECT * FROM Customers;
      ```

      Customers テーブルには現在、統合データベース (hq) で入力された Land Sports の顧客データを持つローも含まれています。

cust_key	name	rep_key
cust1	Ocean Sports	rep1
cust3	Land Sports	rep1

5. 統合データベース (hq) で `c:¥tutorial` ディレクトリから Message Agent を実行します。

   ```
   dbremote -c "UID=DBA;PWD=passwd;SERVER=server_hq;DBF=c:¥tutorial¥hq.db"
   ```

   `c:¥tutorial¥hq` ディレクトリにファイル `field.0` が表示されなくなります。

結果

統合データベースから送信されたデータが、リモートデータベースで受信されます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: SQL Remote システムの作成 [160 ページ]

前のタスク: レッスン 6: 統合データベースからリモートデータベースへのデータの送信 [170 ページ]

次のタスク: レッスン 8: リモートデータベースから統合データベースへのデータの送信 [173 ページ]

1.5.8 レッスン 8: リモートデータベースから統合データベースへのデータの送信

Interactive SQL を使用して、リモートデータベース (field) から統合データベース (hq) にデータをレプリケートします。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. 現在、リモートデータベース (field) に接続されていない場合は、次のコマンドを実行します。

   ```
   dbisql -c "UID=DBA;PWD=passwd;SERVER=field_db;DBF=c:¥tutorial¥field.db"
   ```

2. 次の文を実行し、リモートデータベース (field) でローを挿入します。

   ```
   INSERT INTO Customers ( cust_key, name, rep_key )
   VALUES ( 'cust5', 'North Land Trading', 'rep1' );
   COMMIT;
   ```

3. `c:¥tutorial` ディレクトリから、リモートデータベース (field) に対して dbremote ユーティリティを実行します。

   ```
   dbremote -c "UID=DBA;PWD=passwd;SERVER=field_db;DBF=c:¥tutorial¥field.db"
   ```

   `c:¥tutorial¥hq` ディレクトリにファイル `field.1` が表示されます。

4. 現在、統合データベース (hq) に接続されていない場合は、次のコマンドを実行します。

   ```
   dbisql -c "UID=DBA;PWD=passwd;SERVER=server_hq;DBF=c:¥tutorial¥hq.db"
   ```

5. 統合データベース (hq) で `c:¥tutorial` ディレクトリから Message Agent を実行します。

   ```
   dbremote -c "UID=DBA;PWD=passwd;SERVER=server_hq;DBF=c:¥tutorial¥hq.db"
   ```

6. [Message Agent] ウィンドウに `Execution Completed` と表示されたら、シャットダウンをクリックします。
7. `c:¥tutorial¥field` を参照します。

   `hq.1` ファイルは `hq.2` というファイルに置き換えられました。`hq.2` ファイルは受信確認を含みます。
8. 次の文を実行して、統合データベース (hq) 内の Customers テーブルのデータを表示します。

   ```
   SELECT * FROM Customers;
   ```

   このクエリは、次の結果を返します。

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

結果

リモートデータベースから統合データベースにデータがレプリケートされます。

タスクの概要: チュートリアル: SQL Remote システムの作成 [160 ページ]

前のタスク: レッスン 7: リモートデータベースでのデータの受信 [171 ページ]

1.6 チュートリアル: メッセージサーバとして HTTP メッセージシステムと統合データベースを使用するレプリケーションシステムの設定

このチュートリアルでは、HTTP メッセージシステムを使用する SQL Remote レプリケーションシステムの設定方法について学びます。

前提条件

このチュートリアルはユーザ、DBA を対象としています。このユーザは、SYS_REPLICATION_ADMIN_ROLE システムロールを持っている必要があります。

コンテキスト

統合データベースは、FILE メッセージシステムを使用して変更をレプリケートし、リモートデータベースは、HTTP メッセージシステムを使用して変更をレプリケートします。

このチュートリアルでは、次のことを学習します。

- SQL Anywhere 統合データベースと、統合データベース内のすべてのデータを含む SQL Anywhere リモートデータベースを作成します。
- SQL Remote によって生成されたメッセージを格納するディレクトリ構造を作成します。統合データベースは、FILE メッセージシステムを使用してファイルにアクセスし、リモートデータベースは HTTP メッセージシステムを使用します。
- HTTP メッセージシステム用メッセージサーバとして動作する統合データベースを設定します。
- HTTP メッセージシステムを使用してメッセージを送信するリモートデータベースを作成します。
- 統合データベースとリモートデータベース間でデータをレプリケートします。

1. レッスン 1: 統合データベースの作成 [175 ページ]
   データベースとそのトランザクションログの保存に必要なディレクトリ、およびメッセージのディレクトリ構造を作成します。また、リモートユーザおよびデータのレプリケートに必要なパブリケーションとサブスクリプションなど、統合データベースのスキーマを定義します。
2. レッスン 2: メッセージサーバとして動作する統合データベースの設定 [178 ページ]
   HTTP メッセージシステム用メッセージサーバとして動作する統合データベースを設定します。
3. レッスン 3: リモートデータベースの作成 [179 ページ]
   リモートデータベースを抽出してから、リモートデータベースの FILE メッセージシステムを HTTP メッセージシステムと置き換えます。
4. レッスン 4: 統合データベースとリモートデータベースにおけるデータの追加とレプリケート [181 ページ]
   統合データベースとリモートデータベースにデータを追加し、SQL Remote を実行して変更をレプリケートし、両方のデータベースのデータが一致していることを確認します。
5. レッスン 5: クリーンアップ [184 ページ]
   リモートデータベースと統合データベースを停止します。

1.6.1 レッスン 1: 統合データベースの作成

データベースとそのトランザクションログの保存に必要なディレクトリ、およびメッセージのディレクトリ構造を作成します。また、リモートユーザおよびデータのレプリケートに必要なパブリケーションとサブスクリプションなど、統合データベースのスキーマを定義します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

SQL Remote が統合データベースに対して稼働する場合、FILE メッセージシステムを使用してメッセージを送受信しますが、リモートデータベースは HTTP メッセージシステムを使用します。

手順

1. 統合データベースとリモートデータベースを保持するため次のディレクトリを作成します。
   - `c:¥tutorial`
   - `c:¥tutorial¥cons`
   - `c:¥tutorial¥rem`
2. 統合データベースとリモートデータベースが生成したメッセージファイルを格納するために、次のディレクトリを作成します。
   - `c:¥tutorial¥messages`
   - `c:¥tutorial¥messages¥cons`
   - `c:¥tutorial¥messages¥rem`
3. `c:¥tutorial¥cons` ディレクトリから次のコマンドを実行して、統合データベース (cons) を作成します。

   ```
   dbinit -dba DBA,passwd cons.db
   ```
4. 統合データベースを起動します。

   ```
   dbeng17 -n cons c:¥tutorial¥cons¥cons.db -xs http(port=8033)
   ```

   これは、リモートデータベースからの HTTP 要求を受信し、`c:¥tutorial¥messages` ディレクトリに存在するメッセージファイルにアクセスするデータベースサーバであるため、コマンドラインには *-xs http(8033)* が必要です。この時点では、Web サービスは定義されていません。このレッスンでは、パーソナルデータベースサーバのみを起動します。したがって、このコンピュータの SQL Remote プロセスだけが、HTTP を使用してメッセージサーバと通信できます。生産環境では通常、他のコンピュータ上の SQL Remote プロセスも Web サービスにアクセスできるよう、ネットワークサーバを使用するはずです。
5. Interactive SQL を使用して、SYS_REPLICATION_ADMIN_ROLE システムロールを持つユーザとして統合データベース (cons) に接続します。

   ```
   dbisql -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd"
   ```
6. 統合データベース (cons) のグローバルデータベース ID を設定するには、次の文を実行します (GLOBAL AUTOINCREMENT デフォルトを使用する場合には、すべてのデータベースに排他的なプライマリキーが選択されるようグローバルデータベース ID が必要です)。

   ```
   SET OPTION public.global_database_id=0;
   ```
7. このチュートリアルのデータベースのスキーマは、1 つのテーブルから構成され、テーブルのすべてのカラムとローはすべてのリモートユーザにレプリケートされます。統合データベース (cons) 用に次の文を実行し、データベースに 1 つのテーブルを作成します。

   ```
   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. 統合データベース (cons) で次の文を実行し、サンプルデータを従業員テーブルに追加します。

```
INSERT INTO employees (first_name, last_name) VALUES ('Kelly', 'Meloy');
INSERT INTO employees (first_name, last_name) VALUES ('Melisa', 'Boysen');
COMMIT;
```

9. 統合データベース (cons) で次の文を実行し、そのテーブルが作成され、データが挿入されていることを確認します。

```
SELECT * FROM employees;
```

   クエリは、従業員テーブルから次のデータを返します。ただし、hire_date カラムには、次のテーブルにある値ではなく、そのローを挿入した時間が記載されます。

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	4/28/2015 11:17 AM
2	Melisa	Boysen	4/28/2015 11:17 AM

10. このチュートリアルでは、パブリッシャとリモートユーザにはパスワードが割り当てられます。これは、統合データベースが HTTP メッセージシステム用のメッセージサーバとして動作するためです。次の文を実行して、CONNECT と PUBLISH の権限を持つユーザ cons を作成します。

```
GRANT CONNECT TO cons;
GRANT PUBLISH TO cons;
```

11. パフォーマンス上の理由から、HTTP メッセージシステムはリモートデータベースのみで使用することができ、統合データベースでは使用できません。次の文は、統合データベースで FILE ベースのメッセージシステムを使用する設定を行うものです。

```
CREATE REMOTE MESSAGE TYPE FILE ADDRESS 'cons';
SET REMOTE FILE OPTION public.directory='c:¥¥tutorial¥¥messages';
SET REMOTE FILE OPTION public.debug='yes';
```

12. 次の文を実行して、リモートユーザ rem をパスワードなしで作成します。次に、REMOTE 権限を付与し、FILE メッセージシステムにユーザのアドレスを定義します。

```
GRANT CONNECT TO rem IDENTIFIED BY passwd;
GRANT REMOTE TO rem TYPE FILE ADDRESS 'rem';
```

13. パブリケーションでは、レプリケートされるデータセットを説明します。従業員テーブルのすべてのローをレプリケートする pub_employees という名前のパブリケーションを作成します。パブリケーションに対してユーザをサブスクライブするには、サブスクリプションを作成します。

```
CREATE PUBLICATION pub_employees ( TABLE employees );
CREATE SUBSCRIPTION TO pub_employees FOR rem;
```

14. Interactive SQL との接続を切断します。

結果

データベースとそのトランザクションログの保存に必要なディレクトリ、およびメッセージのディレクトリ構造が作成されます。リモートユーザおよびデータのレプリケートに必要なパブリケーションとサブスクリプションの作成など、統合データベースのスキーマが定義されます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: メッセージサーバとして HTTP メッセージシステムと統合データベースを使用するレプリケーションシステムの設定 [174 ページ]

次のタスク: レッスン 2: メッセージサーバとして動作する統合データベースの設定 [178 ページ]

1.6.2 レッスン 2: メッセージサーバとして動作する統合データベースの設定

HTTP メッセージシステム用メッセージサーバとして動作する統合データベースを設定します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

メッセージサーバとして動作する別のデータベースとデータベースサーバを設定することも可能です。

手順

1. Interactive SQL を使用して、SYS_REPLICATION_ADMIN_ROLE システムロールを持つユーザとして統合データベースに接続します。

   ```
   dbisql -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd"
   ```

2. データベースが最初に初期化されたとき、リモートユーザからの HTTP 要求を受け付けるのに必要な Web サービスはどれも定義されておらず、メッセージファイルが格納されているディレクトリにデータベースサーバがアクセスするのを許可する定義はありません。これらのオブジェクトの作成は、誰がすべてのオブジェクトを所有するのかを指定するオプションのパラメータを持つ sr_add_message_server ストアドプロシージャの使用により自動化されています。統合データベース (cons) に次の文を実行して、メッセージサーバに必要なすべてのオブジェクトを定義して、すべてのオブジェクトが cons ユーザによって所有されていることを指定します。

```
CREATE ROLE FOR USER cons;
SET REMOTE http OPTION cons.root_directory='c:¥¥tutorial¥¥messages';
CALL sr_add_message_server( 'cons' );
COMMIT;
```

3. Interactive SQL との接続を切断します。

結果

統合データベースが HTTP メッセージシステム用メッセージサーバとして動作するように設定されます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: メッセージサーバとして HTTP メッセージシステムと統合データベースを使用するレプリケーションシステムの設定 [174 ページ]

前のタスク: レッスン 1: 統合データベースの作成 [175 ページ]

次のタスク: レッスン 3: リモートデータベースの作成 [179 ページ]

1.6.3 レッスン 3: リモートデータベースの作成

リモートデータベースを抽出してから、リモートデータベースの FILE メッセージシステムを HTTP メッセージシステムと置き換えます。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. c:¥tutorial¥rem ディレクトリから次のコマンドを実行して、リモートデータベース (rem) を作成します。

   ```
   dbinit -dba DBA,passwd rem.db
   ```

2. 統合データベースを起動します。

   ```
   dbeng17 -n rem c:¥tutorial¥rem¥rem.db
   ```

3. このレッスンでは、dbxtract を使用してリモートデータベースを作成します。次のコマンドを実行して統合データベースから rem ユーザのデータベースを抽出し、抽出後もそのリモートデータベースのデータベースサーバを稼働状態にしておきます。

   ```
   dbxtract -xx -ac "SERVER=rem;DBN=rem;DBF=c:¥tutorial¥rem
   ¥rem.db;UID=DBA;PWD=passwd" -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd" rem
   ```

   現在、リモートデータベース (rem) に接続されていない場合は、次のコマンドを実行します。

   ```
   dbisql -c "SERVER=rem;DBN=rem;UID=DBA;PWD=passwd"
   ```

4. 統合データベースは、FILE メッセージシステムを使用しているため、dbxtract が稼働すると、統合データベースは、rem リモートデータベースも FILE メッセージシステムを使用していると仮定して SQL Remote 定義を作成します。リモートデータベースが HTTP メッセージシステムを使用するよう設定するには、リモートデータベース (rem) で次の文を実行して、このリモートデータベースの FILE メッセージシステムを削除します。

   ```
   CREATE REMOTE TYPE "FILE" ADDRESS '';
   SET REMOTE FILE OPTION public.directory='';
   SET REMOTE FILE OPTION public.debug='';
   ```

5. リモートデータベース (rem) で次の文を実行して、このリモートデータベースの HTTP メッセージシステムを設定します。

   ```
   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='passwd';
   SET REMOTE HTTP OPTION public.debug='yes';
   SET REMOTE HTTP OPTION public.https='no';
   SET REMOTE HTTP OPTION public.url='localhost:8033';
   COMMIT;
   ```

6. リモートデータベース (rem) の従業員テーブルに、抽出後の統合データベースに存在したこの 2 ローのデータが含まれることを確認します。次の文を実行して、従業員テーブルの内容を表示します。

   ```
   SELECT * FROM employees;
   ```

   クエリは、従業員テーブルから次のデータを返します。ただし、hire_date カラムには、次のテーブルにあるデータではなく、そのローを挿入した時間が記載されます。

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	4/28/2015 11:17 AM
2	Melisa	Boysen	4/28/2015 11:17 AM

7. Interactive SQL との接続を切断します。

結果

リモートデータベースが抽出され、リモートデータベースの FILE メッセージシステムが HTTP メッセージシステムに置き換わります。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: メッセージサーバとして HTTP メッセージシステムと統合データベースを使用するレプリケーションシステムの設定 [174 ページ]

前のタスク: レッスン 2: メッセージサーバとして動作する統合データベースの設定 [178 ページ]

次のタスク: レッスン 4: 統合データベースとリモートデータベースにおけるデータの追加とレプリケート [181 ページ]

1.6.4 レッスン 4: 統合データベースとリモートデータベースにおけるデータの追加とレプリケート

統合データベースとリモートデータベースにデータを追加し、SQL Remote を実行して変更をレプリケートし、両方のデータベースのデータが一致していることを確認します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. 現在、統合データベース (cons) に接続されていない場合は、次のコマンドを実行します。

   ```
   dbisql -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd"
   ```

2. 統合データベース (cons) で次の文を実行し、追加サンプルデータを従業員テーブルに追加します。

   ```
   INSERT INTO employees (first_name, last_name) VALUES ('Javier', 'Spoor');
   COMMIT;
   ```

3. 現在、リモートデータベース (rem) に接続されていない場合は、次のコマンドを実行します。

```
dbisql -c "SERVER=rem;DBN=rem;UID=DBA;PWD=passwd"
```

4. リモートデータベース (rem) で次の文を実行し、追加サンプルデータを従業員テーブルに追加します。

```
INSERT INTO employees (first_name, last_name) VALUES ('Nelson', 'Kreitzer');
COMMIT;
```

5. 統合データベース (cons) で、Message Agent を実行します。

```
dbremote -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd" -qc -v -o c:¥tutorial
¥cons1.txt
```

このコマンドにより、統合データベース (cons) のトランザクションログのスキャンが開始され、FILE メッセージシステムを使用してリモートデータベース (rem) のメッセージが生成されます。デバッグメッセージシステムパラメータは、統合データベースの FILE メッセージシステム用に設定されているため、`c:¥tutorial¥cons1.txt` ファイルを見て、メッセージが `c:¥tutorial¥messages¥rem` ディレクトリに書き込まれていることを示すデバッグメッセージがあることを確認できます。次に例を示します。

```
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"
```

6. リモートデータベース (rem) で、Message Agent を実行します。

```
dbremote -c "SERVER=rem;DBN=rem;UID=DBA;PWD=passwd" -qc -v -o c:¥tutorial¥rem.txt
```

このコマンドは、HTTP メッセージシステムを使用して、統合データベースで生成されたばかりのメッセージを受信して適用します。次に、トランザクションログをスキャンし、メッセージをリモートデータベースに追加された新しいローとともに統合データベースに送り返します。デバッグメッセージシステムパラメータは、リモートデータベースの HTTP メッセージシステム用に設定されているため、`c:¥tutorial¥rem.txt` ファイルを見て、HTTP メッセージシステムが使用されていることを示すデバッグメッセージがあることを確認できます。次に例を示します。

```
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
```

7. 統合データベース (cons) で、Message Agent を実行します。

```
dbremote -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd" -qc -v -o c:¥tutorial
¥cons2.txt
```

このコマンドは、FILE ベースのメッセージシステムを使用してリモートデータベースによって生成されたばかりのメッセージを受信して適用します。

8. 統合データベースに 4 ローのデータがすべて含まれていることを確認するには、次の文を実行して従業員テーブルの内容を表示します。

```
SELECT * FROM employees
ORDER BY employee_id;
```

クエリは、従業員テーブルから次のデータを返します。ただし、hire_date カラムには、次のテーブルにある値ではなく、そのローを挿入した時間が記載されます。

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	4/28/2015 11:17 AM
2	Melisa	Boysen	4/28/2015 11:17 AM
3	Javier	Spoor	4/28/2015 11:17 AM
102000001	Nelson	Kreitzer	4/28/2015 11:17 AM

9. Interactive SQL と統合データベースの接続を切断します。
10. 次の文を実行して従業員テーブルの内容を表示することにより、リモートデータベース (rem) に 4 ローのデータがすべて含まれていることを確認します。

```
SELECT * FROM employees
ORDER BY employee_id;
```

クエリは、従業員テーブルから次のデータを返します。ただし、hire_date カラムには、次のテーブル内のデータではなく、そのローを挿入した時間が記載されます。

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	4/28/2015 11:17 AM
2	Melisa	Boysen	4/28/2015 11:17 AM
3	Javier	Spoor	4/28/2015 11:17 AM
102000001	Nelson	Kreitzer	4/28/2015 11:17 AM

11. Interactive SQL とリモートデータベースの接続を切断します。

結果

統合データベースとリモートデータベースにデータが追加され、変更がレプリケートされて、データの一致が確認されました。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: メッセージサーバとして HTTP メッセージシステムと統合データベースを使用するレプリケーションシステムの設定 [174 ページ]

前のタスク: レッスン 3: リモートデータベースの作成 [179 ページ]

次のタスク: レッスン 5: クリーンアップ [184 ページ]

1.6.5 レッスン 5: クリーンアップ

リモートデータベースと統合データベースを停止します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. 次のコマンドを実行して、リモートデータベースを停止します。

   ```
   dbstop -y -c "SERVER=rem;DBN=rem;UID=DBA;PWD=passwd"
   ```

2. 次のコマンドを実行して、統合データベースを停止します。

   ```
   dbstop -y -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd"
   ```

3. 作成した任意のディレクトリを削除します。

結果

リモートデータベースと統合データベースが停止します。

タスクの概要: チュートリアル: メッセージサーバとして HTTP メッセージシステムと統合データベースを使用するレプリケーションシステムの設定 [174 ページ]

前のタスク: レッスン 4: 統合データベースとリモートデータベースにおけるデータの追加とレプリケート [181 ページ]

1.7 チュートリアル: 別のメッセージサーバで HTTP メッセージシステムを使用するレプリケーションシステムの設定

このチュートリアルでは、別のメッセージサーバで HTTP メッセージシステムを使用する SQL Remote レプリケーションシステムの設定方法について学びます。

前提条件

このチュートリアルはユーザ、DBA を対象としています。このユーザは、SYS_REPLICATION_ADMIN_ROLE システムロールを持っている必要があります。

コンテキスト

統合データベースは、FILE メッセージシステムを使用して変更をレプリケートし、リモートデータベースは、HTTP メッセージシステムを使用して変更をレプリケートします。

このチュートリアルでは、次のことを学習します。

- SQL Anywhere 統合データベースと、統合データベース内のすべてのデータを含む SQL Anywhere リモートデータベースを作成します。
- SQL Remote によって生成されたメッセージを格納するディレクトリ構造を作成します。統合データベースは、FILE メッセージシステムを使用してファイルにアクセスし、リモートデータベースは HTTP メッセージシステムを使用します。
- HTTP プロトコルを使用してリモートデータベースからメッセージを受信する Web サーバとして動作するメッセージサーバ SQL Anywhere データベースを作成します。
- HTTP メッセージシステムを使用してメッセージを送信するリモートデータベースを作成します。
- 統合データベースとリモートデータベース間でデータをレプリケートします。

1. レッスン 1: 統合データベースの作成 [186 ページ]
   データベースとそのトランザクションログの保存に必要なディレクトリ、およびメッセージのディレクトリ構造を作成します。また、リモートユーザおよびデータのレプリケートに必要なパブリケーションとサブスクリプションなど、統合データベースのスキーマを定義します。
2. レッスン 2: メッセージサーバの作成 [188 ページ]
   メッセージサーバをホストするには、別のデータベースサーバを使用します。こうすることにより、この 2 つのデータベースサーバ間でメッセージを処理するために実行される作業の量が分散されます。また、統合データベースへの HTTP アクセスを開放していないため、セキュリティのレベルが上がります。
3. レッスン 3: リモートデータベースの作成 [191 ページ]
   リモートデータベースを抽出してから、リモートデータベースの FILE メッセージシステムを HTTP メッセージシステムと置き換えます。
4. レッスン 4: 統合データベースとリモートデータベースにおけるデータの追加とレプリケート [192 ページ]
   統合データベースとリモートデータベースにデータを追加し、SQL Remote を実行して変更をレプリケートし、両方のデータベースのデータが一致していることを確認します。

5. レッスン 5: クリーンアップ [195 ページ]
   このチュートリアルで起動した 3 つのデータベースサーバを停止します。

1.7.1 レッスン 1: 統合データベースの作成

データベースとそのトランザクションログの保存に必要なディレクトリ、およびメッセージのディレクトリ構造を作成します。また、リモートユーザおよびデータのレプリケートに必要なパブリケーションとサブスクリプションなど、統合データベースのスキーマを定義します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

SQL Remote が統合データベースに対して稼働する場合、FILE メッセージシステムを使用してメッセージを送受信しますが、リモートデータベースは HTTP メッセージシステムを使用します。

手順

1. 統合データベース、リモートデータベース、メッセージサーバデータベースを格納するために、次のディレクトリを作成します。
   - `c:¥tutorial`
   - `c:¥tutorial¥cons`
   - `c:¥tutorial¥rem`
   - `c:¥tutorial¥msgsrv`
2. 統合データベースとリモートデータベースが生成したメッセージファイルを格納するために、次のディレクトリを作成します。
   - `c:¥tutorial¥messages`
   - `c:¥tutorial¥messages¥cons`
   - `c:¥tutorial¥messages¥rem`
3. `c:¥tutorial¥cons` ディレクトリから次のコマンドを実行して、統合データベース (cons) を作成します。

   ```
   dbinit -dba DBA,passwd cons.db
   ```

4. Interactive SQL を使用して、SYS_REPLICATION_ADMIN_ROLE システムロールを持つユーザとして統合データベース (cons) に接続し、AutoStop (ASTOP) 接続パラメータに対して AutoStop=NO を指定して切断を行ってもデータベースは稼働状態となるようにします。

```
dbisql -c "UID=DBA;PWD=passwd;SERVER=cons;DBF=c:¥tutorial¥cons
¥cons.db;autostop=no"
```

5. 統合データベース (cons) のグローバルデータベース ID を設定するには、次の文を実行します (GLOBAL AUTOINCREMENT デフォルトを使用する場合には、すべてのデータベースに排他的なプライマリキーが選択されるようグローバルデータベース ID が必要です)。

```
SET OPTION public.global_database_id=0;
```

6. このチュートリアルのデータベースのスキーマは、レプリケートする 1 つのテーブルから構成され、テーブルのすべてのカラムとローはすべてのリモートユーザにレプリケートされます。統合データベース (cons) で次の文を実行し、データベースに 1 つのテーブルを作成します。

```
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. 統合データベース (cons) で次の文を実行し、サンプルデータを従業員テーブルに追加します。

```
INSERT INTO employees (first_name, last_name) VALUES ('Kelly', 'Meloy');
INSERT INTO employees (first_name, last_name) VALUES ('Melisa', 'Boysen');
COMMIT;
```

8. 統合データベース (cons) で次の文を実行し、そのテーブルが作成され、データが挿入されていることを確認します。

```
SELECT * FROM employees;
```

クエリは、従業員テーブルから次のデータを返します。ただし、hire_date カラムには、次のテーブルにある値ではなく、そのローを挿入した時間が記載されます。

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	4/28/2015 11:17 AM
2	Melisa	Boysen	4/28/2015 11:17 AM

9. このチュートリアルでは、パブリッシャとリモートユーザにはパスワードは割り当てられません。したがって、ユーザはデータベースに存在しますが、それらのユーザではデータベースに接続できません。次の文を実行して、CONNECT と PUBLISH の権限を持つユーザ cons を作成します。

```
GRANT CONNECT TO cons;
GRANT PUBLISH TO cons;
```

10. パフォーマンス上の理由から、HTTP メッセージシステムはリモートデータベースのみで使用することができ、統合データベースでは使用できません。次の文は、統合データベースで FILE ベースのメッセージシステムを使用する設定を行うものです。

```
CREATE REMOTE MESSAGE TYPE FILE ADDRESS 'cons';
SET REMOTE FILE OPTION public.directory='c:¥¥tutorial¥¥messages';
```

```
SET REMOTE FILE OPTION public.debug='yes';
```

11. 次の文を実行して、リモートユーザ rem をパスワードなしで作成します。次に、REMOTE 権限を付与し、FILE メッセージシステムにユーザのアドレスを定義します。

```
GRANT CONNECT TO rem;
GRANT REMOTE TO rem TYPE FILE ADDRESS 'rem';
```

12. パブリケーションでは、レプリケートされるデータセットを説明します。従業員テーブルのすべてのローをレプリケートする pub_employees という名前のパブリケーションを作成します。パブリケーションに対してユーザをサブスクライブするには、サブスクリプションを作成します。

```
CREATE PUBLICATION pub_employees ( TABLE employees );
CREATE SUBSCRIPTION TO pub_employees FOR rem;
```

13. Interactive SQL との接続を切断します。

結果

データベースとそのトランザクションログの保存に必要なディレクトリ、およびメッセージのディレクトリ構造が作成されます。リモートユーザおよびデータのレプリケートに必要なパブリケーションとサブスクリプションの作成など、統合データベースのスキーマが定義されます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: 別のメッセージサーバで HTTP メッセージシステムを使用するレプリケーションシステムの設定 [185 ページ]

次のタスク: レッスン 2: メッセージサーバの作成 [188 ページ]

1.7.2 レッスン 2: メッセージサーバの作成

メッセージサーバをホストするには、別のデータベースサーバを使用します。こうすることにより、この 2 つのデータベースサーバ間でメッセージを処理するために実行される作業の量が分散されます。また、統合データベースへの HTTP アクセスを開放していないため、セキュリティのレベルが上がります。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. c:¥tutorial¥msgsrv ディレクトリから次のコマンドを実行して、メッセージサーバデータベース (msgsrv) を作成します。

   ```
   dbinit -dba DBA,passwd msgsrv.db
   ```

2. メッセージサーバを起動します。

   ```
   dbeng17 -n msgsrv c:¥tutorial¥msgsrv¥msgsrv.db -xs http(port=8033)
   ```

   これは、リモートデータベースからの HTTP 要求を受信し、c:¥tutorial¥messages ディレクトリに存在するメッセージファイルにアクセスするデータベースサーバであるため、コマンドラインには *-xs http(8033)* が必要です。データベースサーバが起動した時点では Web サービスは定義されていませんが、このレッスンで作成されます。その上、パーソナルデータベースサーバだけが起動しているため、このコンピュータ上の SQL Remote プロセスだけが HTTP を使用してメッセージサーバと通信できます。生産環境では通常、他のコンピュータ上の SQL Remote プロセスも Web サービスにアクセスできるよう、ネットワークサーバを使用するはずです。

3. 別のメッセージサーバを作成する場合、統合データベースのスキーマの多くをそのメッセージサーバにコピーする必要があります。とりわけ定義されたリモートユーザとアドレスについての情報などのコピーが必要です。これは手動で行うことができますが、このタスクを行うもっとも簡単な方法は、dbunload ユーティリティを使用して、統合データベースと同じスキーマで新しいデータベースを作成することです。

   ```
   dbunload -n -xx -ac "SERVER=msgsrv;DBN=msgsrv;UID=DBA;PWD=passwd" -c
   "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd"
   ```

   dbunload コマンドで使用されるオプションは、次のことを行います。

   -n

   スキーマだけがアンロードされ、統合データベースのデータはまったくメッセージサーバに追加されないことを示します。

   -xx

   外部アンロードと再ロードを実行します。これは、両方の関連するデータベースがすでに稼働している場合に必要です。

   -ac "SERVER=msgsrv;DBN=msgsrv;UID=DBA;PWD=passwd"

   アンロードの送信先接続を定義します。このレッスンでは、メッセージサーバです。

   -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd"

   アンロードの送信元接続を定義します。このレッスンでは、統合データベースです。

4. Interactive SQL を使用して、SYS_REPLICATION_ADMIN_ROLE システムロールを持つユーザとしてメッセージサーバデータベース (msgsrv) に接続します。

   ```
   dbisql -c "SERVER=msgsrv;DBN=msgsrv;UID=DBA;PWD=passwd"
   ```

   パブリッシャ (cons) とリモートユーザ (rem) にまだパスワードがないため、これらのユーザはいずれも統合データベースに接続できません。メッセージサーバではこれらのユーザにパスワードが必要です。なぜなら、リモートユーザからの HTTP 要求では、リモートデータベースのパブリッシャおよびメッセージサーバでの認証用に提供されたパスワードが使用されるためです。メッセージサーバデータベース (msgsrv) で次の文を実行して、パブリッシャとリモートユーザのパスワードを定義します。

   ```
   GRANT CONNECT TO cons IDENTIFIED BY passwd;
   ```

```
GRANT CONNECT TO rem IDENTIFIED BY passwd;
```

5. データベースが最初に初期化されたとき、リモートユーザからの HTTP 要求を受け付けるのに必要な Web サービスはどれも定義されておらず、メッセージファイルが格納されているディレクトリにデータベースサーバがアクセスするのを許可する定義はありません。これらのオブジェクトの作成は、誰がすべてのオブジェクトを所有するのかを指定するオプションのパラメータを持つ sr_add_message_server ストアドプロシージャの使用により自動化されています。メッセージサーバデータベース (msgsrv) に次の文を実行して、メッセージサーバに必要なすべてのオブジェクトを定義し、すべてのオブジェクトが cons ユーザによって所有されていることを指定します。

```
CREATE ROLE FOR USER cons;
SET REMOTE http OPTION cons.root_directory='c:¥¥tutorial¥¥messages';
CALL sr_add_message_server( 'cons' );
COMMIT;
```

6. Interactive SQL との接続を切断します。

結果

別のデータベースサーバによってメッセージサーバがホストされます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: 別のメッセージサーバで HTTP メッセージシステムを使用するレプリケーションシステムの設定 [185 ページ]

前のタスク: レッスン 1: 統合データベースの作成 [186 ページ]

次のタスク: レッスン 3: リモートデータベースの作成 [191 ページ]

関連情報

sr_add_message_server システムプロシージャ [230 ページ]

1.7.3 レッスン 3: リモートデータベースの作成

リモートデータベースを抽出してから、リモートデータベースの FILE メッセージシステムを HTTP メッセージシステムと置き換えます。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. c:¥tutorial¥rem ディレクトリから次のコマンドを実行して、リモートデータベース (rem) を作成します。

   ```
   dbinit -dba DBA,passwd rem.db
   ```

2. このレッスンでは、dbxtract を使用してリモートデータベースを作成します。次のコマンドを実行して統合データベースから rem ユーザのデータベースを抽出し、抽出後もそのリモートデータベースのデータベースサーバを稼働状態にしておきます。

   ```
   dbxtract -xx -ac "SERVER=rem;DBN=rem;dbf=c:¥tutorial¥rem
   ¥rem.db;UID=DBA;PWD=passwd;autostop=no" -c
   "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd" rem
   ```

3. 現在、リモートデータベース (rem) に接続されていない場合は、次のコマンドを実行します。

   ```
    dbisql -c "SERVER=rem;DBN=rem;UID=DBA;PWD=passwd"
   ```

4. 統合データベースは、FILE メッセージシステムを使用しているため、dbxtract が稼働すると、rem リモートデータベースも FILE メッセージシステムを使用していると仮定して SQL Remote 定義を作成します。リモートデータベースが HTTP メッセージシステムを使用するよう設定するには、リモートデータベース (rem) で次の文を実行して、このリモートデータベースの FILE メッセージシステムを削除します。

   ```
   CREATE REMOTE TYPE "FILE" ADDRESS '';
   SET REMOTE FILE OPTION public.directory='';
   SET REMOTE FILE OPTION public.debug='';
   ```

5. リモートデータベース (rem) で次の文を実行して、このリモートデータベースの HTTP メッセージシステムを設定します。

   ```
   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. リモートデータベース (rem) に、抽出後の統合データベースに存在したこの 2 ローのデータが含まれることを確認します。次の文を実行して、従業員テーブルの内容を表示します。

```
SELECT * FROM employees;
```

クエリは、従業員テーブルから次のデータを返します。ただし、hire_date カラムには、次のテーブルにある値ではなく、そのローを挿入した時間が記載されます。

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	4/28/2015 11:17 AM
2	Melisa	Boysen	4/28/2015 11:17 AM

7. Interactive SQL との接続を切断します。

結果

リモートデータベースが抽出され、リモートデータベースの FILE メッセージシステムが HTTP メッセージシステムに置き換わります。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: 別のメッセージサーバで HTTP メッセージシステムを使用するレプリケーションシステムの設定 [185 ページ]

前のタスク: レッスン 2: メッセージサーバの作成 [188 ページ]

次のタスク: レッスン 4: 統合データベースとリモートデータベースにおけるデータの追加とレプリケート [192 ページ]

1.7.4 レッスン 4: 統合データベースとリモートデータベースにおけるデータの追加とレプリケート

統合データベースとリモートデータベースにデータを追加し、SQL Remote を実行して変更をレプリケートし、両方のデータベースのデータが一致していることを確認します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. 現在、統合データベース (cons) に接続されていない場合は、次のコマンドを実行します。

```
dbisql -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd"
```

2. 統合データベース (cons) で次の文を実行し、追加サンプルデータを従業員テーブルに追加します。

```
INSERT INTO employees (first_name, last_name) VALUES ('Javier', 'Spoor');
COMMIT;
```

3. 4.現在、リモートデータベース (rem) に接続されていない場合は、次のコマンドを実行します。

```
dbisql -c "SERVER=rem;DBN=rem;UID=DBA;PWD=passwd"
```

4. リモートデータベース (rem) で次の文を実行し、追加サンプルデータを従業員テーブルに追加します。

```
INSERT INTO employees (first_name, last_name) VALUES ('Nelson', 'Kreitzer');
COMMIT;
```

5. 統合データベース (cons) で、Message Agent を実行します。

```
dbremote -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd" -qc -v -o c:¥tutorial
¥cons1.txt
```

これにより、統合データベース (cons) のトランザクションログがスキャンされ、FILE メッセージシステムを使用してリモートデータベース (rem) のメッセージが生成されます。デバッグメッセージシステムパラメータは、統合データベースの FILE メッセージシステム用に設定されているため、`c:¥tutorial¥cons1.txt` ファイルを見て、メッセージが `c:¥tutorial¥messages¥rem` ディレクトリに書き込まれていることを示すデバッグメッセージがあることを確認できます。次に例を示します。

```
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"
```

6. リモートデータベース (rem) で、Message Agent を実行します。

```
dbremote -c "SERVER=rem;DBN=rem;UID=DBA;PWD=passwd" -qc -v -o c:¥tutorial¥rem.txt
```

このコマンドは、HTTP メッセージシステムを使用して、統合データベースで生成されたばかりのメッセージを受信して適用します。次に、トランザクションログをスキャンし、メッセージをリモートデータベースに追加された新しいローとともに統合データベースに送り返します。デバッグメッセージシステムパラメータは、リモートデータベースの HTTP メッセージシステム用に設定されているため、`c:¥tutorial¥rem.txt` ファイルを見て、HTTP メッセージシステムが使用されていることを示すデバッグメッセージがあることを確認できます。次に例を示します。

```
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
```

7. 統合データベース (cons) で、Message Agent を実行します。

```
dbremote -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd" -qc -v -o c:¥tutorial
¥cons2.txt
```

このコマンドは、FILE ベースのメッセージシステムを使用してリモートデータベースによって生成されたばかりのメッセージを受信して適用します。

8. 統合データベースに 4 ローのデータがすべて含まれていることを確認するには、次の文を実行して従業員テーブルの内容を表示します。

```
SELECT * FROM employees
ORDER BY employee_id;
```

クエリは、従業員テーブルから次のデータを返します。ただし、hire_date カラムには、次のテーブルにある値ではなく、そのローを挿入した時間が記載されます。

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	4/28/2015 11:17 AM
2	Melisa	Boysen	4/28/2015 11:17 AM
3	Javier	Spoor	4/28/2015 11:17 AM
102000001	Nelson	Kreitzer	4/28/2015 11:17 AM

9. Interactive SQL と統合データベースの接続を切断します。
10. 次の文を実行して従業員テーブルの内容を表示することにより、リモートデータベース (rem) に 4 ローのデータがすべて含まれていることを確認します。

```
SELECT * FROM employees
ORDER BY employee_id;
```

クエリは、従業員テーブルから次のデータを返します。ただし、hire_date カラムには、次のテーブルにある値ではなく、そのローを挿入した時間が記載されます。

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	4/28/2015 11:17 AM
2	Melisa	Boysen	4/28/2015 11:17 AM
3	Javier	Spoor	4/28/2015 11:17 AM
102000001	Nelson	Kreitzer	4/28/2015 11:17 AM

11. Interactive SQL とリモートデータベースの接続を切断します。

結果

統合データベースとリモートデータベースにデータが追加され、変更がレプリケートされて、データの一致が確認されました。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: 別のメッセージサーバで HTTP メッセージシステムを使用するレプリケーションシステムの設定 [185 ページ]

前のタスク: レッスン 3: リモートデータベースの作成 [191 ページ]

次のタスク: レッスン 5: クリーンアップ [195 ページ]

1.7.5 レッスン 5: クリーンアップ

このチュートリアルで起動した 3 つのデータベースサーバを停止します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. 次のコマンドを実行して、リモートデータベースを停止します。

   ```
   dbstop -y -c "SERVER=rem;DBN=rem;UID=DBA;PWD=passwd"
   ```

2. 次のコマンドを実行して、メッセージサーバデータベースを停止します。

   ```
   dbstop -y -c "SERVER=msgsrv;DBN=msgsrv;UID=DBA;PWD=passwd"
   ```

3. 次のコマンドを実行して、統合データベースを停止します。

   ```
   dbstop -y -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd"
   ```

結果

3 つのデータベースサーバが停止します。

タスクの概要: チュートリアル: 別のメッセージサーバで HTTP メッセージシステムを使用するレプリケーションシステムの設定 [185 ページ]

前のタスク: レッスン 4: 統合データベースとリモートデータベースにおけるデータの追加とレプリケート [192 ページ]

1.8 チュートリアル: Relay Server 経由で HTTP メッセージシステムを使用するレプリケーションシステムの設定

このチュートリアルでは、HTTP トラフィックを統合データベースに転送するために Relay Server を使用する SQL Remote レプリケーションシステムの設定方法について学びます。

前提条件

このチュートリアルはユーザ、DBA を対象としています。このユーザは、SYS_REPLICATION_ADMIN_ROLE システムロールを持っている必要があります。

コンテキスト

統合データベースは、FILE メッセージシステムを使用して変更をレプリケートし、リモートデータベースは、HTTP メッセージシステムを使用して変更をレプリケートします。

このチュートリアルでは、次のことを学習します。

- SQL Anywhere 統合データベースと、統合データベース内のすべてのデータを含む SQL Anywhere リモートデータベースを作成します。
- SQL Remote によって生成されたメッセージを格納するディレクトリ構造を作成します。統合データベースは、FILE メッセージシステムを使用してファイルにアクセスし、リモートデータベースは HTTP メッセージシステムを使用します。
- HTTP トラフィックを統合データベースに転送するために既存の Relay Server を設定します。
- HTTP メッセージシステムのメッセージサーバとして動作し、転送された HTTP トラフィックを Relay Server から受信する統合データベースを設定します。
- HTTP メッセージシステムを使用してメッセージを送信するリモートデータベースを作成します。
- 統合データベースとリモートデータベース間でデータをレプリケートします。

1. レッスン 1: 統合データベースの作成 [197 ページ]
   データベースとそのトランザクションログの保存に必要なディレクトリ、およびメッセージのディレクトリ構造を作成します。また、リモートユーザおよびデータのレプリケートに必要なパブリケーションとサブスクリプションなど、統合データベースのスキーマを定義します。
2. レッスン 2: Relay Server の設定 [200 ページ]
   Relay Server の設定を変更して、HTTP 要求をバックエンドの SQL Anywhere データベースに転送します。
3. レッスン 3: メッセージサーバとして動作する統合データベースの設定 [201 ページ]
   HTTP メッセージシステム用メッセージサーバとして動作する統合データベースを設定します。
4. レッスン 4: リモートデータベースの作成 [203 ページ]
   リモートデータベースを抽出してから、リモートデータベースの FILE メッセージシステムを HTTP メッセージシステムと置き換えます。
5. レッスン 5: 統合データベースとリモートデータベースにおけるデータの追加とレプリケート [205 ページ]

統合データベースとリモートデータベースにデータを追加し、SQL Remote を実行して変更をレプリケートし、両方のデータベースのデータが一致していることを確認します。

6. レッスン 6: クリーンアップ [207 ページ]
   Relay Server Outbound Enabler および統合データベースとリモートデータベースを停止します。

1.8.1 レッスン 1: 統合データベースの作成

データベースとそのトランザクションログの保存に必要なディレクトリ、およびメッセージのディレクトリ構造を作成します。また、リモートユーザおよびデータのレプリケートに必要なパブリケーションとサブスクリプションなど、統合データベースのスキーマを定義します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

SQL Remote が統合データベースに対して稼働する場合、FILE メッセージシステムを使用してメッセージを送受信しますが、リモートデータベースは HTTP メッセージシステムを使用します。

このチュートリアルのため、統合データベース (およびメッセージサーバ) が稼働しているコンピュータの名前を machine_cons とします。

手順

1. 統合データベースとリモートデータベースを保持するため次のディレクトリを作成します。
   - `c:¥tutorial`
   - `c:¥tutorial¥cons`
   - `c:¥tutorial¥rem`
2. 統合データベースとリモートデータベースが生成したメッセージファイルを格納するために、次のディレクトリを作成します。
   - `c:¥tutorial¥messages`
   - `c:¥tutorial¥messages¥cons`
   - `c:¥tutorial¥messages¥rem`
3. `c:¥tutorial¥cons` ディレクトリから次のコマンドを実行して、統合データベース (cons) を作成します。

```
dbinit -dba DBA,passwd cons.db
```

4. 統合データベースを起動します。

```
dbsrv17 -n cons c:¥tutorial¥cons¥cons.db -xs http(port=8033)
```

これは、リモートデータベースからの HTTP 要求を受信し、`c:¥tutorial¥messages` ディレクトリに存在するメッセージファイルにアクセスするデータベースサーバであるため、コマンドラインには *-xs http(8033)* が必要です。この時点では、Web サービスは定義されていません。このレッスンでは、パーソナルデータベースサーバのみを起動します。したがって、このコンピュータの SQL Remote プロセスだけが、HTTP を使用してメッセージサーバと通信できます。生産環境では通常、他のコンピュータ上の SQL Remote プロセスも Web サービスにアクセスできるよう、ネットワークサーバを使用するはずです。このレッスンではネットワークサーバを起動して、これを cons と命名しました。ネットワークにこの名前ですでに稼働している別のデータベースサーバがある場合は、このネットワークサーバに別の名前を選択して、この代替名を使用するために、このチュートリアルの残りの部分で接続ストリングを変更する必要があります。

5. Interactive SQL を使用して、SYS_REPLICATION_ADMIN_ROLE システムロールを持つユーザとして統合データベース (cons) に接続します。

```
dbisql -c "UID=DBA;PWD=passwd;SERVER=cons;DBN=cons"
```

6. 統合データベース (cons) のグローバルデータベース ID を設定するには、次の文を実行します (GLOBAL AUTOINCREMENT デフォルトを使用する場合には、すべてのデータベースに排他的なプライマリキーが選択されるようグローバルデータベース ID が必要です)。

```
SET OPTION public.global_database_id=0;
```

7. このチュートリアルのデータベースのスキーマは、1 つのテーブルから構成され、テーブルのすべてのカラムとローはすべてのリモートユーザにレプリケートされます。統合データベース (cons) で次の文を実行し、データベースに 1 つのテーブルを作成します。

```
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. 統合データベース (cons) で次の文を実行し、サンプルデータを従業員テーブルに追加します。

```
INSERT INTO employees (first_name, last_name) VALUES ('Kelly', 'Meloy');
INSERT INTO employees (first_name, last_name) VALUES ('Melisa', 'Boysen');
COMMIT;
```

9. 統合データベース (cons) で次の文を実行し、そのテーブルが作成され、データが挿入されていることを確認します。

```
SELECT * FROM employees;
```

クエリは、従業員テーブルから次のデータを返します。ただし、hire_date カラムには、次のテーブルにある値ではなく、そのローを挿入した時間が記載されます。

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	4/28/2015 11:17 AM
2	Melisa	Boysen	4/28/2015 11:17 AM

10. このチュートリアルでは、リモートユーザにはパスワードが割り当てられます。これは、統合データベースが HTTP メッセージシステム用のメッセージサーバとして動作するためです。次の文を実行して、CONNECT と PUBLISH の権限を持つユーザ cons を作成します。

```
GRANT CONNECT TO cons;
GRANT PUBLISH TO cons;
```

11. パフォーマンス上の理由から、HTTP メッセージシステムはリモートデータベースのみで使用され、統合データベースでは使用できません。次の文は、統合データベースで FILE ベースのメッセージシステムを使用する設定を行うものです。

```
CREATE REMOTE MESSAGE TYPE FILE ADDRESS 'cons';
SET REMOTE FILE OPTION public.directory='c:¥¥tutorial¥¥messages';
SET REMOTE FILE OPTION public.debug='yes';
```

12. 次の文を実行して、リモートユーザ rem をパスワードなしで作成します。次に、REMOTE 権限を付与し、FILE メッセージシステムにユーザのアドレスを定義します。

```
GRANT CONNECT TO rem IDENTIFIED BY passwd;
GRANT REMOTE TO rem TYPE FILE ADDRESS 'rem';
```

13. パブリケーションでは、レプリケートされるデータセットを説明します。従業員テーブルのすべてのローをレプリケートする pub_employees という名前のパブリケーションを作成します。パブリケーションに対してユーザをサブスクライブするには、サブスクリプションを作成します。

```
CREATE PUBLICATION pub_employees ( TABLE employees );
CREATE SUBSCRIPTION TO pub_employees FOR rem;
```

14. Interactive SQL との接続を切断します。

結果

データベースとそのトランザクションログの保存に必要なディレクトリ、およびメッセージのディレクトリ構造が作成されます。リモートユーザおよびデータのレプリケートに必要なパブリケーションとサブスクリプションの作成など、統合データベースのスキーマが定義されます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: Relay Server 経由で HTTP メッセージシステムを使用するレプリケーションシステムの設定 [196 ページ]

次のタスク: レッスン 2: Relay Server の設定 [200 ページ]

1.8.2 レッスン 2: Relay Server の設定

Relay Server の設定を変更して、HTTP 要求をバックエンドの SQL Anywhere データベースに転送します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

このチュートリアルは次の条件に基づいています。

1. Relay Server が稼働するコンピュータの名前は machine_iis であり、IIS 7.5 を実行する Microsoft Windows 2008 Server R2 です。
2. machine_iis にインストールされている Relay Server のバージョンは、SQL Anywhere バージョン 17 です。
3. Relay Server の設定指示は、マニュアルに記載されたとおり正確に従っています。
4. Relay Server コンポーネントは、Windows Server 2008/Windows Server 2008 R2 上の Microsoft IIS 7.5 に配備されています。

コンテキスト

Relay Server の設定は、Relay Server が使用する `rs.config` ファイルの変更に依存します。

手順

1. machine_iis の `rs.config` ファイルを変更して、メッセージサーバとして動作するバックエンドの SQL Anywhere データベースサーバのエントリを追加します。

   ```
   [backend_farm]
   id=srhttp_tutorial_farm
   description=SQL Anywhere Web Services farm for tutorial
   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. `%WINDIR%¥system32¥` ディレクトリから、次のコマンドを実行して設定の更新を適用します。

   ```
   iisreset.exe
   ```

結果

Relay Server の設定を変更して、HTTP 要求をバックエンドの SQL Anywhere データベースに転送します。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: Relay Server 経由で HTTP メッセージシステムを使用するレプリケーションシステムの設定 [196 ページ]

前のタスク: レッスン 1: 統合データベースの作成 [197 ページ]

次のタスク: レッスン 3: メッセージサーバとして動作する統合データベースの設定 [201 ページ]

1.8.3 レッスン 3: メッセージサーバとして動作する統合データベースの設定

HTTP メッセージシステム用メッセージサーバとして動作する統合データベースを設定します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

コンテキスト

メッセージサーバとして動作する別のデータベースとデータベースサーバを設定することも可能です。

手順

1. Interactive SQL を使用して、SYS_REPLICATION_ADMIN_ROLE システムロールを持つユーザとして接続します。

   ```
   dbisql -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd"
   ```

2. データベースが最初に初期化されたときに、リモートユーザからの HTTP 要求を受け付け Web サービスはどれも定義されていません。また、メッセージファイルが格納されているディレクトリにデータベースサーバがアクセスするのを許可する

定義はありません。誰がすべてのオブジェクトを所有するのかを指定するオプションのパラメータを持つ sr_add_message_server ストアドプロシージャを使用して、これらのオブジェクトを作成します。統合データベース (cons) に次の文を実行して、メッセージサーバに必要なすべてのオブジェクトを定義して、すべてのオブジェクトが cons ユーザによって所有されていることを指定します。

```
CREATE ROLE FOR USER cons;
SET REMOTE http OPTION cons.root_directory='c:¥¥tutorial¥¥messages';
CALL sr_add_message_server( 'cons' );
COMMIT;
```

3. Relay Server が HTTP 要求をバックエンドの SQL Anywhere サーバに転送する場合には追加設定が必要となります。いくつかのノードが読み込み専用、いくつかのノードが読み込み/書き込みノードとして定義されているバックエンドの SQL Anywhere データベースサーバのために、高可用性環境を設定することが可能です。このチュートリアルでは、システムに 1 つのデータベースサーバしかないため、このデータベースは読み込み/書き込みノードとして定義します。統合データベース (cons) で次の文を実行し、Relay Server がこのデータベースサーバを読み込み/書き込みノードとして認識するよう必要なオブジェクトをすべて定義します。

```
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. Interactive SQL との接続を切断します。
5. Outbound Enabler は、Relay Server とバックエンドの SQL Anywhere データベースの間のチャネルとして動作します。machine_cons コンピュータ上で次のコマンドを実行して、Relay Server Outbound Enabler (RSOE) を起動します。

```
rsoe2 -cr "host=machine_iis;port=80;url_suffix=/rs/server/rs.dll"
-cs "host=localhost;port=8033"
-f srhttp_tutorial_farm -id srhttp_tutorial_server -v 5 -o rsoe.log
```

結果

統合データベースが HTTP メッセージシステム用メッセージサーバとして動作するように設定されます。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: Relay Server 経由で HTTP メッセージシステムを使用するレプリケーションシステムの設定 [196 ページ]

前のタスク: レッスン 2: Relay Server の設定 [200 ページ]

次のタスク: レッスン 4: リモートデータベースの作成 [203 ページ]

1.8.4 レッスン 4: リモートデータベースの作成

リモートデータベースを抽出してから、リモートデータベースの FILE メッセージシステムを HTTP メッセージシステムと置き換えます。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. c:¥tutorial¥rem ディレクトリから次のコマンドを実行して、リモートデータベース (rem) を作成します。

   ```
   dbinit -dba DBA,passwd rem.db
   ```

2. このレッスンでは、dbxtract を使用してリモートデータベースを作成します。次のコマンドを実行して統合データベースから rem ユーザのデータベースを抽出し、抽出後もそのリモートデータベースのデータベースサーバを稼働状態にしておきます。

   ```
   dbxtract -xx -ac "SERVER=rem;DBN=rem;DBF=c:¥tutorial¥rem
   ¥rem.db;UID=DBA;PWD=passwd;autostop=no" -c
   "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd" rem
   ```

3. 現在、Interactive SQL からリモートデータベース (rem) に接続されていない場合は、次のコマンドを実行します。

   ```
   dbisql -c "SERVER=rem;DBN=rem;UID=DBA;PWD=passwd"
   ```

4. 統合データベースは、FILE メッセージシステムを使用しているため、dbxtract が稼働すると、rem リモートデータベースも FILE メッセージシステムを使用していると仮定して SQL Remote 定義を作成します。リモートデータベースが HTTP メッセージシステムを使用するよう設定するには、リモートデータベース (rem) で次の文を実行して、このリモートデータベースの FILE メッセージシステムを削除します。

   ```
   CREATE REMOTE TYPE "FILE" ADDRESS '';
   SET REMOTE FILE OPTION public.directory='';
   SET REMOTE FILE OPTION public.debug='';
   ```

5. リモートデータベース (rem) で次の文を実行して、このリモートデータベースの HTTP メッセージシステムを設定します。

```
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='passwd';
SET REMOTE HTTP OPTION public.debug='yes';
SET REMOTE HTTP OPTION public.https='no';
SET REMOTE HTTP OPTION public.url='machine_iis:80/rs17/client/rs.dll/
srhttp_tutorial_farm';
COMMIT;
```

6. リモートデータベース (rem) に、抽出後の統合データベースに存在したこの 2 ローのデータが含まれることを確認します。次の文を実行して、従業員テーブルの内容を表示します。

```
SELECT * FROM employees;
```

クエリは、従業員テーブルから次のデータを返します。ただし、hire_date カラムには、次のテーブルにあるデータではなく、そのローを挿入した時間が記載されます。

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	4/28/2015 11:17 AM
2	Melisa	Boysen	4/28/2015 11:17 AM

7. Interactive SQL との接続を切断します。

結果

リモートデータベースが抽出され、リモートデータベースの FILE メッセージシステムが HTTP メッセージシステムに置き換わります。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: Relay Server 経由で HTTP メッセージシステムを使用するレプリケーションシステムの設定 [196 ページ]

前のタスク: レッスン 3: メッセージサーバとして動作する統合データベースの設定 [201 ページ]

次のタスク: レッスン 5: 統合データベースとリモートデータベースにおけるデータの追加とレプリケート [205 ページ]

1.8.5 レッスン 5: 統合データベースとリモートデータベースにおけるデータの追加とレプリケート

統合データベースとリモートデータベースにデータを追加し、SQL Remote を実行して変更をレプリケートし、両方のデータベースのデータが一致していることを確認します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. 現在、統合データベース (cons) に接続されていない場合は、次のコマンドを実行します。

   ```
   dbisql -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd"
   ```

2. 統合データベース (cons) で次の文を実行し、追加サンプルデータを従業員テーブルに追加します。

   ```
   INSERT INTO employees (first_name, last_name) VALUES ('Javier', 'Spoor');
   COMMIT;
   ```

3. 現在、リモートデータベース (cons) に接続されていない場合は、次のコマンドを実行します。

   ```
   dbisql -c "SERVER=rem;DBN=rem;UID=DBA;PWD=passwd"
   ```

4. リモートデータベース (rem) で次の文を実行し、追加サンプルデータを従業員テーブルに追加します。

   ```
   INSERT INTO employees (first_name, last_name) VALUES ('Nelson', 'Kreitzer');
   COMMIT;
   ```

5. 統合データベース (cons) で、Message Agent を実行します。

   ```
   dbremote -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd" -qc -v -o c:¥tutorial
   ¥cons1.txt
   ```

   このコマンドにより、統合データベース (cons) のトランザクションログがスキャンされ、FILE メッセージシステムを使用してリモートデータベース (rem) のメッセージが生成されます。デバッグメッセージシステムパラメータは、統合データベースの FILE メッセージシステム用に設定されているため、`c:¥tutorial¥cons1.txt` ファイルを見て、メッセージが `c:¥tutorial¥messages¥rem` ディレクトリに書き込まれていることを示すデバッグメッセージがあることを確認できます。次に例を示します。

   ```
   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"
   ```

6. リモートデータベース (rem) で、Message Agent を実行します。

   ```
   dbremote -c "SERVER=rem;DBN=rem;UID=DBA;PWD=passwd" -qc -v -o c:¥tutorial¥rem.txt
   ```

このコマンドは、HTTP メッセージシステムを使用して、統合データベースで生成されたばかりのメッセージを受信して適用します。次に、トランザクションログをスキャンし、メッセージをリモートデータベースに追加された新しいローとともに統合データベースに送り返します。デバッグメッセージシステムパラメータは、リモートデータベースの HTTP メッセージシステム用に設定されているため、c:¥tutorial¥rem.txt ファイルを見て、HTTP メッセージシステムが使用されていることを示すデバッグメッセージがあることを確認できます。次に例を示します。

```
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
```

7. 統合データベース (cons) で、Message Agent を実行します。

```
dbremote -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd" -qc -v -o c:¥tutorial
¥cons2.txt
```

このコマンドは、FILE ベースのメッセージシステムを使用してリモートデータベースによって生成されたばかりのメッセージを受信して適用します。

8. 統合データベースに 4 ローのデータがすべて含まれていることを確認するには、次の文を実行して従業員テーブルの内容を表示します。

```
SELECT * FROM employees
ORDER BY employee_id;
```

クエリは、従業員テーブルから次のデータを返します。ただし、hire_date カラムには、次のテーブルにある値ではなく、そのローを挿入した時間が記載されます。

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	4/28/2015 11:17 AM
2	Melisa	Boysen	4/28/2015 11:17 AM
3	Javier	Spoor	4/28/2015 11:17 AM
102000001	Nelson	Kreitzer	4/28/2015 11:17 AM

9. 次の文を実行して従業員テーブルの内容を表示することにより、リモートデータベース (rem) に 4 ローのデータがすべて含まれていることを確認します。

```
SELECT * FROM employees;
```

クエリは、従業員テーブルから次のデータを返します。ただし、hire_date カラムには、次のテーブル内のデータではなく、そのローを挿入した時間が記載されます。

employee_id	first_name	last_name	hire_date
1	Kelly	Meloy	4/28/2015 11:17 AM
2	Melisa	Boysen	4/28/2015 11:17 AM
3	Javier	Spoor	4/28/2015 11:17 AM
102000001	Nelson	Kreitzer	4/28/2015 11:17 AM

10. Interactive SQL との接続を切断します。

結果

統合データベースとリモートデータベースにデータが追加され、変更がレプリケートされて、データの一致が確認されました。

次のステップ

次のレッスンに進みます。

タスクの概要: チュートリアル: Relay Server 経由で HTTP メッセージシステムを使用するレプリケーションシステムの設定 [196 ページ]

前のタスク: レッスン 4: リモートデータベースの作成 [203 ページ]

次のタスク: レッスン 6: クリーンアップ [207 ページ]

1.8.6 レッスン 6: クリーンアップ

Relay Server Outbound Enabler および統合データベースとリモートデータベースを停止します。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. 次のコマンドを実行して、RSOE を停止します。

   ```
   rsoe2 -s -cr "host=machine_iis;port=80;url_suffix=/rs17/rs.dll"
   -cs "host=localhost;port=8033"
   -f srhttp_tutorial_farm -id srhttp_tutorial_server
   ```

2. 次のコマンドを実行して、統合データベースを停止します。

   ```
   dbstop -y -c "SERVER=cons;DBN=cons;UID=DBA;PWD=passwd"
   ```

   次のコマンドを実行して、リモートデータベースを停止します。

   ```
   dbstop -y -c "SERVER=rem;DBN=rem;UID=DBA;PWD=passwd"
   ```

結果

RSOE データベース、統合データベース、リモートデータベースが停止します。

タスクの概要: チュートリアル: Relay Server 経由で HTTP メッセージシステムを使用するレプリケーションシステムの設定 [196 ページ]

前のタスク: レッスン 5: 統合データベースとリモートデータベースにおけるデータの追加とレプリケート [205 ページ]

1.9 SQL Remote のリファレンス

SQL Remote は、ソフトウェアで機能するユーティリティ、オプション、プロシージャ、文を備えています。

このセクションの内容:

SQL Remote ユーティリティとオプションのリファレンス [208 ページ]
この章では、SQL Remote に用意されているユーティリティ、オプション、プロシージャについて説明します。

SQL Remote システムテーブル [241 ページ]
SQL Remote のシステム情報は、SQL Anywhere カタログに保持されます。この情報をより分かりやすい形にしたものが、一連のシステムビューに保持されます。

SQL Remote の SQL 文 [241 ページ]
次の SQL 文を使用して、SQL Remote コマンドを実行します。

1.9.1 SQL Remote ユーティリティとオプションのリファレンス

この章では、SQL Remote に用意されているユーティリティ、オプション、プロシージャについて説明します。

このセクションの内容:

SQL Remote Message Agent ユーティリティ (dbremote) [209 ページ]
SQL Remote Message Agent ユーティリティ (dbremote) は、SQL Remote メッセージの送信と適用、およびメッセージの配信を確認するメッセージトラッキングシステムの管理を行います。

抽出ユーティリティ (dbxtract) [219 ページ]
抽出ユーティリティは、SQL Anywhere 統合データベースからリモートデータベースを抽出できません。

SQL Remote オプション [228 ページ]
レプリケーションオプションは、レプリケーション動作を制御するためのデータベースオプションです。

SQL Remote ストアドプロシージャ [230 ページ]
HTTP メッセージングシステムを管理するには、次のストアドプロシージャを使用します。

SQL Remote システムプロシージャ [232 ページ]

このストアドプロシージャを使用して、SQL Remote データベースでレプリケーションをカスタマイズするためのインタフェースに名前と引数を提供します。

1.9.1.1 SQL Remote Message Agent ユーティリティ (dbremote)

SQL Remote Message Agent ユーティリティ (dbremote) は、SQL Remote メッセージの送信と適用、およびメッセージの配信を確認するメッセージトラッキングシステムの管理を行います。

SQL Remote を実行するには、SYS_RUN_REPLICATION_ROLE システムロールが必要です。

構文

```
dbremote [ options ] [ transaction-logs-directory ]
```

オプション	説明

@data	指定された環境変数または設定ファイルからオプションを読み込みます。同じ名前の環境変数と設定ファイルが両方存在する場合は、環境変数が使用されます。
	
	設定ファイル内の情報を保護する場合は、ファイル非表示ユーティリティ (dbfhide) を使用して、設定ファイルの内容をエンコードします。
	
	環境変数には、あらゆるオプションのセットを格納できます。たとえば、次の文の組み合わせは、4 MB のキャッシュサイズで起動し、メッセージだけを受信し、*myserver* というデータベースサーバの *field* というデータベースに接続する SQL Remote プロセス用に、オプションのセットを格納する環境変数を設定します。SET 文は 1 行で入力してください。
	
	`SET envvar=-m 4096 -r -c "Server=myserver;DBN=field;UID=sa;PWD=sysadmin" dbremote @envvar`
	
	設定ファイルには、改行を含めたり、あらゆるオプションの設定を格納したりできます。たとえば、次のコマンドファイルには、4 MB のキャッシュサイズで起動し、メッセージだけを送信し、*field* というデータベースサーバの *myserver* というデータベースに接続する SQL Remote Message Agent 用のオプションのセットが格納されています。
	
	`-m 4096 -s -c "Server=myserver;DBN=field;UID=sa;PWD=sysadmin"`
	
	この設定ファイルを `c:¥config.txt` として保存すると、コマンドで次のように使用できます。
	
	`dbremote @c:¥config.txt`

-a	受信したメッセージ (受信ボックス内にあるもの) を、データベースに適用しないで処理します。このオプションを使用するときに、-v (冗長出力) と -p (メッセージがパージされない) の両方を指定すると、入力メッセージの問題を検出しやすくなります。このオプションを使用するときに -p を指定しないと、メッセージを適用しないで受信ボックスがパージされるため、サブスクリプションを再開する場合に便利です。
-b	バッチモードで実行します。このモードでは、SQL Remote Message Agent は受信メッセージを処理し、トランザクションログを 1 回スキャンし、出力メッセージを処理して停止します。

オプション	説明
-c"keyword=value;..."	接続パラメータを指定します。このオプションを指定しない場合、環境変数 SQLCONNECT が使用されます。 たとえば、次の文では c:¥mydata.db のデータベースファイルに対して dbremote を実行します。接続にはユーザ ID *DBA* とパスワード *passwd* を使用しています (ユーザ DBA は SYS_RUN_REPLICATION_ROLE システムロールを持っている必要があります)。 `dbremote -c "UID=DBA;PWD=passwd;DBF=c:¥mydata.db"` SQL Remote Message Agent は、SYS_RUN_REPLICATION_ROLE システムロールを持つユーザが実行する必要があります。 SQL Remote Message Agent は、SQL Anywhere 接続パラメータの全種類をサポートします。
-dl	SQL Remote Message Agent ウィンドウまたはコマンドプロンプトにメッセージを表示します。指定されている場合はトランザクションログファイルに出力します。
*-ek*key	コマンドプロンプトで、強力に暗号化されたデータベースの暗号化キーの入力を求めるプロンプトを表示するよう指定します。強力に暗号化されたデータベースを扱う場合には、データベースやトランザクションログ (オフライントランザクションログなど) を使用するのに、常に暗号化キーを使用する必要があります。強力な暗号化が適用されたデータベースの場合、-ek または -ep のどちらかを指定します。両方同時には指定できません。強力に暗号化されたデータベースでは、キーを指定しないとコマンドが失敗します。
-ep	暗号化キーの入力を求めるプロンプトを表示するよう指定します。このオプションを指定すると、暗号化キーを入力するためのウィンドウが表示されます。クリアテキストでは暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。強力な暗号化が適用されたデータベースの場合、-ek または -ep のどちらかを指定します。両方同時には指定できません。強力に暗号化されたデータベースでは、キーを指定しないとコマンドが失敗します。
*-g*n	n 個未満のオペレーションのトランザクションを、後に続くトランザクションとまとめるように SQL Remote Message Agent に指示します。デフォルトのオペレーション数は 20 です。n の値を大きくするとコミットが少なくなるため、入力メッセージの処理速度が向上します。ただし、トランザクションのサイズが大きくなると、デッドロックやブロックの原因になることもあります。

オプション	説明
-*l*length	送信する各メッセージの最大長を指定します。*size* には、メモリ容量をバイト単位で指定します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ k、m、g のいずれかを使用してください。長いトランザクションは複数のメッセージに分割されます。デフォルトは 50000 バイトで、最小長は 10000 バイトです。 ⚠ 警告 メッセージの最大長は、同一のインストール環境内のすべてのサイトで必ず同じ長さにしてください。 メモリの割り付けに制限のあるプラットフォームの場合、この値はオペレーティングシステムのメモリ割り付けの最大値より小さい値にしてください。
-*m*size	メッセージ作成と入力メッセージのキャッシュに、SQL Remote Message Agent が使用する最大メモリサイズを指定します。*size* には、メモリ容量をバイト単位で指定します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ k、m、g のいずれかを使用してください。デフォルトは 2048 キロバイトです (2 MB)。 すべてのリモートデータベースが、レプリケートされるオペレーションのユニークサブセットを受信する場合、それぞれのリモートデータベースにメッセージが同時に作成されます。同じオペレーションを受信するリモートユーザのグループには、メッセージが 1 つだけ作成されます。使用されるメモリが -m 値を超えると、メッセージの送信後に最大サイズ (-l オプションで指定したサイズ) に達します。 メッセージが届くと、適用されるまで SQL Remote Message Agent によってメモリ内に格納されます。このようなメッセージのキャッシュによって、適切でないメッセージをメッセージシステムが再読み込みしないようにできますが、大規模なインストール環境ではパフォーマンスが低下することもあります。-m オプションで指定したメモリ使用量を超過すると、最低使用頻度 (LRU) 方式でメッセージが削除されます。

オプション	説明
-ml`directory`	オフライントランザクションログミラーファイルのロケーションを指定します。このオプションを指定すると、次のどちらかの状況になった場合に、dbremote は古いトランザクションログミラーファイルを削除できます。 • オフライントランザクションログミラーが、トランザクションログミラーとは異なるディレクトリに置かれています • dbremote がリモートデータベースサーバとは異なるコンピュータで実行されています 通常の設定では、アクティブなトランザクションログミラーと名前が変更されたトランザクションログミラーは同じディレクトリ内に存在し、dbremote はリモートデータベースと同じコンピュータ上で実行されるため、このオプションを指定しなくても、古いトランザクションログミラーファイルは自動的に削除されます。このディレクトリ内のトランザクションログが影響を受けるのは、delete_old_logs データベースオプションが Off 以外の値に設定されている場合だけです。
-o`filename`	指定したファイルにメッセージを表示します。デフォルトでは、画面に出力を表示します。
-os`size`	出力メッセージをロギングするファイルの最大サイズを指定します。*size* には、メモリ容量をバイト単位で指定します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ k、m、g のいずれかを使用してください。デフォルトによる制限はなく、最小制限は 10000 バイトです。 SQL Remote では、現在のファイルサイズをチェックしてから、出力メッセージを出力ログファイルに記録します。ログメッセージによって、ファイルサイズが指定したサイズより大きくなると、SQL Remote は出力ファイルの名前をそれぞれ `yymmddxx.dbs` に変更します。`xx` は 00 で始まる番号で、1 ずつ増えていきます (xx は 2 桁以上の場合があります)。また、`yymmdd` は現在の年月日を表します。 SQL Remote Message Agent を継続モードで長時間実行する場合、このオプションを使用して、手動で古い出力ログファイルを削除し、ディスク領域を解放できます。
-ot`file`	出力ログファイルをトランケートし、このファイルに出力メッセージを追加します。デフォルトでは、画面に出力を送信します。
-p	メッセージをパージしません。
-q	SQL Remote Message Agent を最小化ウィンドウで起動します。このオプションは、Windows オペレーティングシステムの場合にのみ適用されます。
-qc	完了後、SQL Remote のウィンドウを閉じます。

オプション	説明
-r	メッセージを受信します。-r と -s のいずれも指定しない場合、SQL Remote Message Agent は両方のフェーズを実行します。それ以外の場合、指定されたフェーズのみ実行されます。 -r オプションで起動された場合、SQL Remote Message Agent は継続モードで動作します。メッセージ受信後に SQL Remote Message Agent を停止するには、-r オプションと -b オプションを使用します。
-rd`minutes`	受信メッセージのポーリング頻度を指定します。デフォルトでは、SQL Remote Message Agent は入力メッセージを 1 分ごとにポーリングします。このオプション (rd は "受信遅延 (*receive delay*)" の意味) によってポーリング頻度を設定できます。これは、ポーリングが高負荷である場合に便利です。デフォルト値は 1 分です。 頻繁にポーリングする場合は、秒数を表す数の後にサフィックス s を付けると便利です。たとえば、次のコマンドでは、30 秒ごとにポーリングが行われます。 `dbremote -rd 30s` -rd オプションは、欠落しているメッセージの再送を要求するまでに SQL Remote Message Agent が待機するポーリングの回数を設定する -rp オプションとともに使用されることが多くあります。
-ro`filename`	ファイルにリモート出力を記録します。このオプションは統合サイトで使用します。統合データベースに出力ログ情報を送信するようにリモートデータベースを設定する場合、このオプションを使用すると情報がファイルに書き込まれます。このオプションを指定すると、管理者がリモートサイトでエラーのトラブルシューティングを行う場合に役立ちます。
-rp`number`	メッセージを消失したと判断するまでのポーリング受信回数を指定します。SQL Remote Message Agent を継続モードで実行すると、メッセージが一定の間隔でポーリングされます。設定回数 (デフォルトでは 1 回) のポーリングが行われた後にメッセージが欠落していると、SQL Remote Message Agent はそのメッセージが失われたと判断して、メッセージの再送を要求します。このため、メッセージシステムの処理速度が遅い場合、この動作によって必要のない再送要求が多く出されることがあります。このオプションを使用して、再送要求が出される前に行うポーリング回数を指定すると、再送要求の数を最小限に抑えることができます。 -rp オプションは、受信メッセージのポーリング頻度を設定する -rd オプションとともに使用されることが多くあります。

オプション	説明
*-rt*filename	起動時に出力ログファイルをトランケートし、このファイルにリモートデータベースのログ出力を追加します。このオプションは統合サイトで使用します。ファイルが起動時にトランケートされることを除いて、-ro オプションと同じです。
*-ru*time	再送の宛先のログを再スキャンする待ち時間を指定します。 このオプションは *resend urgency* を制御します。再送信要求の検出から、SQL Remote Message Agent がこの要求の処理を開始するまでの時間を指定します。このオプションを使用すると、SQL Remote Message Agent が複数のユーザの再送信要求を集めてからログの再スキャンを行うことができます。指定できる時間の単位は s (秒)、m (分)、h (時間)、または d (日数) です。
-s	メッセージを送信します。-r と -s のいずれも指定しない場合、SQL Remote Message Agent は両方のフェーズを実行します。それ以外の場合、指定されたフェーズのみ実行されます。
*-sd*time	データベーストランザクションログのポーリングとポーリングの間の遅延を制御します。-sd オプションは、継続モードで実行する場合にのみ使用されます。デフォルト値は 1 分です。 "送信遅延 (*send delay*)" を制御します。これは、ポーリングとポーリングの間に送信されるトランザクションログデータを待つ時間です。
-t	すべてのトリガをレプリケートします。このオプションを使用する場合は、トリガの動作がリモートデータベースで 2 回 (リモートサイトでトリガを起動するとき、および統合データベースからレプリケートされた動作を明示的に適用するとき) 実行されないようにする必要があります。 トリガアクションが 2 回実行されないようにするには、トリガの本文を IF CURRENT REMOTE USER IS NULL...END IF 文で囲みます。
*-ts*session-name(session-option=[option-value;...])	SQL Remote のトレースセッションを設定します。セッション名は *logging* にする必要があります。 オプションの *-ts logging* 部分の後で指定するすべての情報は、スペースなしで指定する必要があります。
-u	オフライントランザクションログにあるトランザクションのみを処理します。このオプションを指定すると、最後にバックアップされた以降のトランザクションは処理されません。このオプションを使用すると、出力トランザクションと入力トランザクションの確認が、オフライントランザクションログから削除されてから送信されるようになります。 名前が変更されたログのトランザクションのみが処理されます。

オプション	説明
-ud	UNIX プラットフォームで、SQL Remote Message Agent をデーモンとして実行します。SQL Remote Message Agent をデーモンとして実行する場合は、-o または -ot オプションも指定して、出力情報のログを取ってください。 SQL Remote Message Agent をデーモンとして実行し、FTP または SMTP メッセージリンクを使用する場合は、データベースにメッセージリンクパラメータを格納してください。これは、SQL Remote Message Agent をデーモンとして実行していると、これらのオプションの入力をユーザに要求しないためです。 SQL Remote Message Agent をデーモンとして起動すると、現在のユーザの umask 設定によってパーミッションが制御されます。SQL Remote Message Agent に適切なパーミッションを付与するため、SQL Remote Message Agent を起動する前に umask 値を設定してください。
-ui	X Window サーバがサポートされている Linux で、使用可能な表示がない場合にシェルモードで SQL Remote Message Agent を起動します。
-ux	Solaris と Linux で SQL Remote Message Agent のウィンドウを開きます。 -ux が指定されている場合、dbremote は使用可能な表示を見つけます。たとえば、DISPLAY 環境変数が設定されていなかったり、X-Window Server が実行されていなかったりしたために、使用可能な表示が見つからなかった場合、dbremote は起動できません。Windows では、SQL Remote のメッセージウィンドウが自動的に表示されます。
-v	冗長出力を表示します。このオプションによって、メッセージに含まれる SQL 文がメッセージウィンドウに表示されます。-o または -ot オプションを指定するとメッセージログファイルに出力されます。

オプション	説明
-*w*n	受信メッセージを適用するデータベースワーカスレッド数を指定します。このオプションは、Windows Mobile ではサポートされていません。 デフォルトは 0 です。この場合、すべてのメッセージがメインの (1 つだけの) スレッドによって適用されます。1 の値を指定すると、1 つのスレッドがメッセージシステムからメッセージを受信し、1 つのスレッドがメッセージをデータベースに適用します。データベースワーカスレッドの最大数は 50 です。 -w オプションを指定すると、ハードウェアのアップグレードによって、受信メッセージのスループットを増加できます。多くのオペレーションを同時に実行できるデバイスに統合データベースを配置すると、受信メッセージのスループットが向上します。このような統合データベースの配置方法をストライプ論理ドライブの RAID アレイといいます。また、SQL Remote Message Agent を実行するコンピュータにマルチプロセッサを搭載しても、入力メッセージのスループットは向上します。 多くのオペレーションを同時に実行できないハードウェアでは、-w オプションを使用してもパフォーマンスはそれほど向上しません。 単一のリモートデータベースからの受信メッセージを複数のスレッドに適用できません。単一のリモートデータベースからのメッセージは、常に適切な順序で逐次適用されます。
-*x* [size]	出力メッセージがスキャンされた後、トランザクションログの名前を変更し、再起動します。場合によっては、リモートデータベースのバックアップが実行されたり、データベースサーバをシャットダウンするときにトランザクションログの名前を変更する代わりに、統合データベースにデータがレプリケートされます。 オプションの size 修飾子が指定された場合、トランザクションログは、指定されたサイズよりも大きい場合にのみ名前を変更されます。*size* には、メモリ容量をバイト単位で指定します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ k、m、g のいずれかを使用してください。デフォルトは 0 です。

オプション	説明
`transaction-logs-directory`	SQL Anywhere リモートデータベースまたは SQL Anywhere 統合データベースのトランザクションログが格納されているディレクトリ。アクティブなトランザクションログファイルと 0 個以上のトランザクションログアーカイブファイルがあります。dbremote がレプリケートするデータを判別するには、このすべてのファイルが必要です。トランザクションログアーカイブファイルがアーカイブトランザクションログとは別のディレクトリにある場合、このパラメータを指定します。 この機能を使用すると、データベースサーバがトランザクションログを取得するために実行する必要のある処理が増えるため、パフォーマンスが若干低下します。

備考

DBTools ライブラリに呼び出すと、自分のアプリケーションから SQL Remote Message Agent を実行することができます。詳細については、*%SQLANY17%*`¥SDK¥Include¥` ディレクトリの `dbrmt.h` ファイルを参照してください。

SQL Remote Message Agent コマンドのユーザ ID には、SYS_RUN_REPLICATION_ROLE システムロールが必要です。

SQL Remote Message Agent はいくつかのデータベース接続を使用します。

メッセージシステム制御パラメータ

SQL Remote は、複数のレジストリ設定を使用してメッセージリンク動作を制御します。

Windows では、メッセージリンク制御パラメータは、レジストリ内の次の場所に格納されます。

```
HKEY_CURRENT_USER¥Software¥SAP¥SQL Remote
```

関連情報

SQL Remote Message Agent (dbremote) [92 ページ]
SQL Remote メッセージシステム [117 ページ]
CURRENT REMOTE USER 特別値 [51 ページ]
リモートデータベースからのエラーの収集 (SQL) [146 ページ]
メッセージ受信時のパフォーマンスのトラブルシューティング [102 ページ]
メッセージ受信時のパフォーマンスのトラブルシューティング [102 ページ]
リモートメッセージタイプ制御パラメータ [121 ページ]

1.9.1.2 抽出ユーティリティ (dbxtract)

抽出ユーティリティは、SQL Anywhere 統合データベースからリモートデータベースを抽出できません。

ユーザには SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

構文

```
dbxtract [ options ] [ directory ] subscriber
```

オプション	説明
@data	指定された環境変数または設定ファイルからオプションを読み込みます。同じ名前の環境変数と設定ファイルが両方存在する場合は、環境変数が使用されます。 設定ファイル内の情報を保護する場合は、ファイル非表示ユーティリティ (dbfhide) を使用して、設定ファイルの内容をエンコードします。
-ac"keyword=value;..."	接続文字列で指定したデータベースに接続して、再ロードします。 このオプションを使用すると、データベースのアンロード処理と、既存データベースへの結果の再ロード処理を組み合わせることができます。 たとえば、次のコマンド (1 行で入力) は、field_user サブスクライバのデータのコピーを既存のデータベースファイル field.db にロードします。 `dbxtract -c "UID=DBA;PWD=passwd;DBF=cons.db" -ac "UID=DBA;PWD=passwd;DBF=field.db" field_user` このオプションを指定した場合、データのコピーはディスク上に作成されないため、コマンドでアンロード用ディレクトリを指定する必要はありません。これによりデータのセキュリティは高まりますが、パフォーマンスは多少低下します。 このオプションを指定した場合、パスワード値は生成されたデータベースにアップロードおよび再ロードされます。
-alfilename	-an オプションを使用している場合は、新しいデータベースのトランザクションログファイル名を指定します。

オプション	説明
-*an*database	抽出するデータベースと同じ設定でデータベースファイルを作成し、それを自動的に再ロードします。 このオプションを使用すると、データベースのアンロード、新規データベースの作成、データのロードを組み合わせて実行できます。 たとえば、次のコマンド (1 行で入力) は、新規のデータベースファイル field.db を作成し、そこに cons.db の field_user サブスクライバのスキーマとデータをコピーします。 `dbxtract -c "UID=DBA;PWD=passwd;DBF=cons.db" -an field.db field_user` このオプションを指定した場合、データのコピーはディスク上に作成されないため、コマンドでアンロード用ディレクトリを指定する必要はありません。これによりデータのセキュリティは高まりますが、パフォーマンスは多少低下します。 このオプションを指定した場合、パスワード値は生成されたデータベースにアップロードおよび再ロードされます。
-*ap*size [k]	新しいデータベースのページサイズを設定します。-an が使用されていない場合、このオプションは無視されます。データベースのページサイズには、2048 (非推奨)、4096、8192、16384、32768 バイトのいずれかを指定できます。デフォルトは元のデータベースのページサイズです。*k* を使用して、キロバイトの単位を指定します (-ap 4k など)。データベースサーバですでにデータベースが実行中の場合、サーバのページサイズ (-gp オプションで設定) は新規ページサイズを処理するのに十分な容量でなければなりません。
-*b*	サブスクリプションを開始しません。このオプションを指定した場合は、統合データベース (リモートデータベース用) とリモートデータベース (統合データベース用) で、レプリケーションを開始するために START SUBSCRIPTION 文を使用して、サブスクリプションを明示的に開始しなければいけません。

オプション	説明
-c"keyword=value;..."	データベース接続パラメータを文字列として指定します。 データベース内のすべてのテーブルでユーザが権限を持っていることを確認するには、ユーザ ID に SYS_RUN_REPLICATION_ROLE システムロールが必要です。 たとえば、次の文 (1 行で入力) は、ユーザ ID が DBA、パスワードが passwd で接続している sample_server データベースサーバ上で実行しているサンプルデータベースから、リモートユーザ ID joe_remote のデータベースを抽出します。データは ¥temp ¥extract ディレクトリにアンロードされます。 `dbxtract -c "Server=sample_server;DBN=demo;UID=DBA;PWD=sql" ¥temp¥extract joe_remote` 接続パラメータを指定しない場合、SQLCONNECT 環境変数が設定されていると、SQLCONNECT 環境変数からの接続パラメータを使用します。
-d	データのみを抽出します。このオプションを指定すると、スキーマ定義はアンロードされません。また、リモートデータベースに対するパブリケーションとサブスクリプションも作成されません。このオプションは、適切なスキーマのあるリモートデータベースがすでに存在していて、データを格納するためだけに使用します。

オプション	説明
-ea*alg*	新しいデータベースで使用する暗号化アルゴリズムを指定します。このオプションにより、新しいデータベースの暗号化に強力な暗号化アルゴリズムを選択できます。AES (デフォルト) または AES_FIPS (FIPS 認定のアルゴリズム) のどちらかを選択できます。AES_FIPS は個別のライブラリを使用するため、AES との互換性はありません。 セキュリティを強化するには、128 ビットの場合は AES、256 ビットの場合は AES256 の強力な暗号化を指定します。FIPS 認定の暗号化を使用するには、128 ビットの場合は AES_FIPS、256 ビットの場合は AES256_FIPS をそれぞれ指定してください。強力な暗号化を使用するためには、-ek または -ep オプションも指定する必要があります。 暗号化されていないデータベースを作成するには、-ea none を指定するか、-ea オプションを指定しません (-e、-et、-ep、-ek オプションも指定しません)。 -ea オプションを指定しない場合、デフォルトの動作は次のようになります。 • -ea NONE (-ek、-ep、-et が指定されない場合) • -ea AES (-ek または -ep が指定される場合) (-et の指定とは無関係) • -ea SIMPLE (-ek または -ep を指定せずに -et を指定する場合) アルゴリズム名の大文字と小文字は区別されません。 FIPS 認定の暗号化には別途ライセンスが必要です。
-ek*key*	新しいデータベースで使用する暗号化キーを指定します。このオプションを使用すると、コマンドに暗号化キーを直接指定することで、強力に暗号化されたデータベースを作成できます。データベースの暗号化に使用されるアルゴリズムは、-ea オプションで指定した AES または AES_FIPS です。-ek オプションを指定して、-ea オプションを指定しないと、AES アルゴリズムが使用されます。 ⚠ 警告 強力な暗号化が適用されたデータベースの場合、キーのコピーは必ず安全な場所に保管してください。暗号化キーがわからなくなった場合は、テクニカルサポートに依頼してもデータにはアクセスできません。アクセスできなくなったデータベースは、廃棄して、新しくデータベースを作成する必要があります。

オプション	説明
-ep	新しいデータベースで使用する暗号化キーの入力を要求します。このオプションを使用すると、ウィンドウに暗号化キーを入力することで、強力に暗号化されたデータベースを作成するように指定できます。クリアテキストでは暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。 暗号化キーは、正確に入力されたことを確認するために 2 回入力してください。キーが一致しない場合は、初期化は失敗します。
-er	アンロード中に暗号化されたテーブルの暗号化を解除します。 テーブル暗号化が有効であるデータベースから抽出するとき、-er または -et を指定して、新しいデータベースのテーブルで暗号化を有効にするかどうかを指定する必要があります。このオプションを指定しないと、データを新しいデータベースにロードしようとしたときにエラーが発生します。 次のコマンド (すべて 1 行で入力) は、テーブルの暗号化を解除してデータベース (`cons.db`) を抽出し、テーブル暗号化が有効になっていない新しいデータベース (`field.db`) に再ロードします。 `dbxtract -an field.db -er -c "UID=DBA;PWD=passwd;DBF=cons.db;DBKEY=29bN8cj1z" field_user`
-et	新しいデータベースのテーブル暗号化を有効にします (-an または -ar も一緒に指定する必要があります)。-ea オプションを指定せずに -et オプションを指定すると、AES アルゴリズムが使用されます。-et オプションを指定する場合、-ep または -ek も指定しないと操作は失敗します。新しいデータベースのテーブル暗号化設定を変更して、アンロードしているデータベースのテーブル暗号化設定とは異なるものに設定できます。 テーブル暗号化が有効であるデータベースを再構築するとき、-er または -et を指定して、新しいデータベースのテーブルで暗号化を有効にするかどうかを指定する必要があります。このオプションを指定しないと、データを新しいデータベースにロードしようとしたときにエラーが発生します。 次の例 (すべて 1 行で入力) は、データベース (`cons.db`) を、テーブル暗号化が有効になっている新しいデータベース (`field.db`) にアンロードし、キーを 34jh として AES_FIPS アルゴリズムを使用します。 `dbxtract -an field.db -et -ea AES_FIPS -ek 34jh -c "UID=DBA;PWD=passwd;DBF=cons.db" field_user`

オプション	説明
-f	完全に修飾されたパブリケーションを抽出します。ほとんどの場合、完全に修飾されたパブリケーション定義をリモートデータベース用に抽出する必要はありません。通常、すべてのローはレプリケートされ、統合データベースに戻されます。 しかし、多層の設定や、統合データベースにないローがリモートデータベースにある設定では、完全に修飾されたパブリケーションが必要な場合もあります。
-g	マテリアライズドビュー MANUAL REFRESH と定義されているマテリアライズドビューは、デフォルトでは再ロード後に初期化されません。このようなマテリアライズドビューを再ロードプロセス中に初期化するには、-g オプションを指定してください。-g を指定すると、データベースサーバによって sa_refresh_materialized_views システムプロシージャが実行されます。 -g オプションを指定するかどうかを決定するときには、すべてのマテリアライズドビューを初期化すると再ロードプロセスの実行時間が大幅に長くなる可能性があることを考慮に入れてください。しかし、-g オプションを指定しない場合、初期化されていないマテリアライズドビューを最初に使用しようとしたクエリはビューの初期化が完了するまで待つことになり、予想外の遅延が生じることがあります。-g オプションを指定しない場合は、再ロードの完了後に、マテリアライズドビューを手動で初期化することもできます。 テキストインデックス MANUAL REFRESH と定義されているテキストインデックスは、デフォルトでは再ロード後に初期化されません。このようなテキストインデックスを再ロードプロセス中に初期化するには、-g オプションを指定してください。-g を指定すると、データベースサーバによって sa_refresh_text_indexes システムプロシージャが実行されます。
-ii	内部アンロードと内部再ロードを実行します。このオプションを使用すると、再ロードスクリプトは、データのアンロードとロードそれぞれに対して、Interactive SQL の OUTPUT 文と INPUT 文ではなく、内部の UNLOAD 文と LOAD TABLE 文を強制的に使用します。このオペレーションの組み合わせがデフォルトの動作です。 データファイルのパスには、外部オペレーションでは dbxtract の現在の作業ディレクトリからの相対パスを使用し、内部の文ではデータベースサーバからの相対パスを使用します。

オプション	説明
-ix	内部アンロードと外部再ロードを実行します。このオプションを使用すると、再ロードスクリプトは、データのアンロードに対して内部の UNLOAD 文を強制的に使用し、新しいデータベースへのデータのロードに対して Interactive SQL の INPUT 文を強制的に使用します。 データファイルのパスには、外部オペレーションでは dbxtract の現在の作業ディレクトリからの相対パスを使用し、内部の文ではデータベースサーバからの相対パスを使用します。
-l`level`	指定した独立性レベルですべての抽出オペレーションを実行します。デフォルト設定では、独立性レベルは 0 です。アクティブなデータベースサーバからデータベースを抽出する場合は、独立性レベル 3 で実行し、抽出されたデータベース内のデータがデータベースサーバ上のデータと一致するようにします。独立性レベルを大きくすると、抽出ユーティリティ (dbxtract) が多数のロックを使用することになり、他のユーザによるデータベースの使用が制限される可能性があります。
-n	スキーマ定義のみ抽出します。この定義を指定すると、データはアンロードされません。再ロードファイルには、データベーススキーマだけを構築する SQL 文が記述されています。SYNCHRONIZE SUBSCRIPTION 文を使用すると、メッセージシステム全体のデータをロードできます。パブリケーション、サブスクリプション、PUBLISH 権限、SUBSCRIBE 権限は、スキーマの一部です。 `dbxtract -c "UID=DBA;PWD=passwd;DBF=¥cons¥cons.db" -n -r "¥remote ¥reload.sql" "¥remote" field_user`
-nl	構造体を抽出します (-n オプションと同じ動作)。ただし、生成される `reload.sql` ファイルには、各テーブルに対する LOAD TABLE 文または INPUT 文も含まれます。このオプションを使用した場合、ユーザデータは抽出されません。-nl を指定するときには、LOAD/INPUT 文を生成できるようにデータディレクトリも指定する必要があります。ただし、このディレクトリにファイルは書き込まれません。このオプションを指定すると、データをアンロードしない再ロードスクリプトを生成できます。データを抽出するには、-d を指定します。データベースにデータのアンロードが不要なテーブルがある場合は、`dbxtract -d -e table-name` と指定することによって、そのテーブルのデータのアンロードを回避できます。
-o`filename`	指定したログファイルにメッセージを出力します。
-p`character`	エスケープ文字を指定します。このオプションを使用して、デフォルトのエスケープ文字 (¥) を別の文字に置き換えることができます。

オプション	説明
-q	クワイエットモードで処理を実行し、メッセージまたはウィンドウの表示を行いません。このオプションは -y オプションと一緒に指定してください。そうしないと操作は失敗します。 このオプションは、コマンドラインユーティリティに対してのみ使用できます。
-r`file`	生成された再ロード Interactive SQL スクリプトファイルの名前を指定します。 再ロードスクリプトファイルのデフォルト名は、現在のディレクトリの `reload.sql` です。このオプションを使用して異なるファイル名を指定できます。
-u	アンロード中にデータの順序を変更しません。デフォルトでは、各テーブルのデータはプライマリキーを基準に順序付けられます。-u オプションを使用するとアンロード処理は高速になりますが、リモートデータベースへのデータのロード処理は遅くなります。
-up	ユーザパスワードをアップロードします。デフォルトでは、-an または -ac が指定され (再ロードを伴う抽出)、パスワードは抽出されません。パスワードを抽出するには、-up オプションを使用します。
-v	冗長メッセージを表示します。アンロードされているテーブル名、アンロードされたロー数、使用された SELECT 文が表示されます。
-xf	外部キーを除外します。リモートデータベースに統合データベーススキーマのサブセットがあり、いくつかの外部キー参照がない場合に、このオプションを使用できます。
-xh	プロシージャフックを除外します。
-xi	外部アンロードと内部再ロードを実行します。データベースのアンロードでは、デフォルトの動作として UNLOAD 文を使用します。これはデータベースサーバが実行します。外部アンロードを選択すると、dbxtract は UNLOAD 文の代わりに OUTPUT 文を使用します。OUTPUT 文はクライアントで実行されます。 データファイルのパスには、外部オペレーションでは dbxtract の現在の作業ディレクトリからの相対パスを使用し、内部の文ではデータベースサーバからの相対パスを使用します。
-xp	データベースからストアドプロシージャを抽出しません。
-xt	データベースからトリガを抽出しません。
-xv	データベースからビューを抽出しません。

オプション	説明
-xx	外部アンロードと外部ロードを実行します。データをアンロードする場合には、OUTPUT 文が使用されます。また、新規データベースにデータをロードする場合には、INPUT 文が使用されます。 アンロードのデフォルト動作では UNLOAD 文が使用され、ロードのデフォルト動作では LOAD TABLE 文が使用されます。内部の UNLOAD 文と LOAD TABLE 文を使用すると、OUTPUT 文と INPUT 文よりも高速で処理します。 データファイルのパスには、外部オペレーションでは dbxtract の現在の作業ディレクトリからの相対パスを使用し、内部の文ではデータベースサーバからの相対パスを使用します。
y	確認なしで、既存の SQL スクリプトファイルを置き換えます。
`directory`	ファイルが書き込まれるディレクトリを指定します。-an または -ac を指定する場合は不要です。
`subscriber`	データベースを抽出するサブスクライバを指定します。

備考

デフォルトでは、抽出ユーティリティ (dbxtract) は独立性レベル 0 で実行されます。アクティブなデータベースサーバからデータベースを抽出する場合は、独立性レベル 3 で実行し、抽出されたデータベース内のデータがデータベースサーバ上のデータと一致するようにします。独立性レベル 3 で実行すると、多数のロックが必要になるため、データベースサーバ上の他のユーザのターンアラウンドタイムに影響が出ることがあります。データベースサーバへのアクセスが少ないときに抽出ユーティリティ (dbxtract) を実行するか、データベースのコピーに対して抽出ユーティリティ (dbxtract) を実行してください。

抽出ユーティリティ (dbxtract) は、スクリプトファイルと、一連の関連データファイルを作成します。新しく初期化したデータベースに対してスクリプトファイルを実行し、データベースオブジェクトを作成してリモートデータベース用のデータをロードできます。

デフォルトの SQL スクリプトファイル名は `reload.sql` です。

リモートユーザがグループの場合、グループのメンバーのユーザ ID すべてを抽出します。これにより、リモートデータベースの複数のユーザに対して異なるユーザ ID を使用できます。カスタム抽出処理は必要ありません。

バージョン 10.0.0 以降のデータベースの抽出ユーティリティ (dbxtract) またはデータベース抽出ウィザードを使用する場合は、使用する dbxtract のバージョンが、データベースへのアクセスに使用するデータベースサーバのバージョンと一致する必要があります。dbxtract のバージョンがデータベースサーバのバージョンより古いまたは新しい場合は、エラーがレポートされます。

抽出ユーティリティ (dbxtract) とデータベース抽出ウィザードでは、データベース作成時に dbo ユーザ ID 用に作成されたオブジェクトをアンロードしません。データをアンロードするとき、システムプロシージャの再定義など、これらのオブジェクトに加えられた変更は失われます。抽出ユーティリティ (dbxtract) でアンロードされるため、データベースの初期化以降に dbo ユーザ ID によって作成されたオブジェクトは保存されます。

権限

-ac、-an、-up オプションを指定するには、SELECT ANY TABLE と ACCESS USER PASSWORD が必要です。

例

リモートデータベースを自動的に抽出するには、次の手順に従います。

1. 統合データベースへの接続に使用するユーザに SYS_RUN_REPLICATION_ROLE システムロールがあることを確認します。
2. dbxtract を実行します。-ac オプションを指定して既存のデータベースに抽出するか、-an オプションを指定して新しいデータベースに抽出します。
   -ac オプションを指定する場合は、空のデータベースを作成してから dbxtract を実行してください。たとえば、次のコマンドは mydata.db という空のデータベースを作成し、dbxtract を実行して新たに作成されたデータベースを移植します。新たに抽出されたデータベースは、field_user というリモートユーザ用です。

   ```
   dbinit -dba DBA,passwd ¥remote¥mydata.db
   dbxtract -c "UID=DBA;PWD=passwd;DBF=¥consolidated¥data.db" -ac
   "UID=DBA;PWD=passwd;DBF=¥remote¥mydata.db" field_user
   ```

   または、-an を使用して同じプロセスを 1 つの手順で実行します。

   ```
   dbxtract -c "UID=DBA;PWD=passwd;DBF=¥consolidated¥data.db" -an ¥remote
   ¥mydata.db field_user
   ```

   適切なスキーマ、リモートユーザ、パブリケーション、サブスクリプション、トリガを含む新しいリモートデータベース mydata.db が作成されます。デフォルトでは、統合データベースのデータがリモートデータベースに抽出され、サブスクリプションが開始されます。ただし、抽出ユーティリティ (dbxtract) では、SQL Remote Message Agent が開始されないため、メッセージが交換されません。

関連情報

リモートデータベースの抽出 [83 ページ]

1.9.1.3 SQL Remote オプション

レプリケーションオプションは、レプリケーション動作を制御するためのデータベースオプションです。

構文

```
SET [ TEMPORARY ] OPTION
[ userid. | PUBLIC. ]option-name = [ option-value ]
```

パラメータ	説明
option-name	変更されるオプションの名前。

パラメータ	説明
option-value	オプションの設定が含まれる文字列。

備考

これらのオプションは SQL Remote Message Agent が使用します。SQL Remote Message Agent のコマンドで指定されたユーザ ID 用に設定してください。一般的な public の使用にも設定できます。

オプション	値	デフォルト
blob_threshold オプション [SQL Remote]	整数 (バイト)	256
compression オプション [SQL Remote]	-1 ～ 9 の整数	6
delete_old_logs オプション [SQL Remote]	On、Off、Delay、n days (日数)	Off
external_remote_options オプション [SQL Remote]	On、Off	Off
qualify_owners オプション [SQL Remote]	On、Off	On
quote_all_identifiers オプション [SQL Remote]	On、Off	Off
replication_error オプション [SQL Remote]	ストアドプロシージャ名	(プロシージャなし)
replication_error_piece オプション [SQL Remote]	ストアドプロシージャ名	(プロシージャなし)
save_remote_passwords オプション [SQL Remote]	On、Off	On
sr_date_format オプション [SQL Remote]	date-string	yyyy/mm/dd
sr_time_format オプション [SQL Remote]	time-string	hh:nn:ss.Ssssss
sr_timestamp_format オプション [SQL Remote]	timestamp-string	yyyy/mm/dd hh:nn:ss.Ssssss
sr_timestamp_with_time_zone_format オプション [SQL Remote]	timestamp-with-time-zone-string	yyyy/mm/dd hh:nn:ss.Ssssss +hh:nn
subscribe_by_remote オプション [SQL Remote]	On、Off	On
verify_all_columns オプション [SQL Remote]	On、Off	Off
verify_threshold オプション [SQL Remote]	整数 (バイト)	1000

1.9.1.4 SQL Remote ストアドプロシージャ

HTTP メッセージングシステムを管理するには、次のストアドプロシージャを使用します。

このセクションの内容:

sr_add_message_server システムプロシージャ [230 ページ]
このプロシージャは、リモートユーザからの HTTP 要求を受信するのに必要な Web サービスを定義します。また、メッセージファイルが格納されているディレクトリにデータベースサーバがアクセスするのを許可するための定義を行います。

sr_drop_message_server システムプロシージャ [231 ページ]
このプロシージャは、sr_add_message_server によって作成されたすべてのオブジェクトを削除します。

sr_update_message_server システムプロシージャ [232 ページ]
このプロシージャは、メッセージサーバの SQL Remote 定義が変更されるたびに呼び出す必要があります。

1.9.1.4.1 sr_add_message_server システムプロシージャ

このプロシージャは、リモートユーザからの HTTP 要求を受信するのに必要な Web サービスを定義します。また、メッセージファイルが格納されているディレクトリにデータベースサーバがアクセスするのを許可するための定義を行います。

構文

```
CALL sr_add_message_server( 'owner' );
```

戻り値

なし。メッセージサーバの定義に必要なオブジェクトの作成時に問題が生じると、エラーが返されます。

備考

データベースが最初に初期化されたとき、リモートユーザからの HTTP 要求を受け付けるのに必要な Web サービスはどれも定義されておらず、メッセージファイルが格納されているディレクトリにデータベースサーバがアクセスするのを許可する定義はありません。これらのオブジェクトの作成は、誰がすべてのオブジェクトを所有するのかを指定するオプションのパラメータを持つ sr_add_message_server ストアドプロシージャの使用により自動化されています。オブジェクト名は重複することができません。

例

次の文により、メッセージサーバデータベース (msgsrv) はメッセージサーバに必要なすべてのオブジェクトを定義して、すべてのオブジェクトが cons ユーザ (この例では統合データベース) によって所有されることを指定します。

```
CREATE ROLE FOR USER cons;
SET REMOTE http OPTION cons.root_directory='c:¥¥tutorial¥¥messages';
CALL sr_add_message_server( 'cons' );
COMMIT;
```

関連情報

sr_drop_message_server システムプロシージャ [231 ページ]
sr_update_message_server システムプロシージャ [232 ページ]

1.9.1.4.2 sr_drop_message_server システムプロシージャ

このプロシージャは、sr_add_message_server によって作成されたすべてのオブジェクトを削除します。

構文

```
CALL sr_drop_message_server;
```

戻り値

なし。メッセージサーバの定義に必要なオブジェクトの作成時に問題が生じると、エラーが返されます。

備考

ストアドプロシージャは、sr_add_message_server によって作成されたすべてのオブジェクトを削除するのに使用されます。

関連情報

sr_add_message_server システムプロシージャ [230 ページ]
sr_update_message_server システムプロシージャ [232 ページ]

1.9.1.4.3 sr_update_message_server システムプロシージャ

このプロシージャは、メッセージサーバの SQL Remote 定義が変更されるたびに呼び出す必要があります。

構文

```
CALL sr_update_message_server( 'owner' );
```

戻り値

なし。メッセージサーバの定義に必要なオブジェクトの作成時に問題が生じると、エラーが返されます。

備考

このプロシージャは、ストアドプロシージャで作成されたオブジェクトを所有するユーザというオプションのパラメータを持ちます。

関連情報

sr_add_message_server システムプロシージャ [230 ページ]
sr_drop_message_server システムプロシージャ [231 ページ]

1.9.1.5 SQL Remote システムプロシージャ

このストアドプロシージャを使用して、SQL Remote データベースでレプリケーションをカスタマイズするためのインタフェースに名前と引数を提供します。

注記

特に明記しないかぎり、イベントフックプロシージャには次の条件が適用されます。

- ストアドプロシージャには、SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。
- プロシージャでは、オペレーションのコミットとロールバックはできません。また、暗黙的なコミットを実行するアクションも実行できません。プロシージャのアクションは、呼び出し側のアプリケーションによって自動的にコミットされます。
- SQL Remote Message Agent の冗長モードをオンにすると、フックのトラブルシューティングができます。

このセクションの内容:

1.9.1.5.1 #Hook_dict テーブル

SQL Remote Message Agent は #hook_dict テーブルを使用して、値をフック関数に渡します。フック関数は #hook_dict テーブルを使用して、値を SQL Remote Message Agent に返します。

#hook_dict テーブルは、次の CREATE 文を使用して、フックが呼び出される直前に作成されます。

```
CREATE TABLE #hook_dict(
NAME VARCHAR(128) NOT NULL UNIQUE,
value VARCHAR(255) NOT NULL );
```

1.9.1.5.2 sp_hook_dbremote_begin システムプロシージャ

このシステムプロシージャを使用すると、レプリケーション処理の開始時にカスタムアクションを追加できます。

#hook_dict テーブルのロー

名前	値	説明
send	*true* または *false*	処理がレプリケーションの送信フェーズを実行中かどうかを示します。
receive	*true* または *false*	処理がレプリケーションの受信フェーズを実行しているかどうかを示します。

備考

この名前のプロシージャが存在する場合、プロシージャは、SQL Remote Message Agent が起動するときに呼び出されます。

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

1.9.1.5.3 sp_hook_dbremote_end システムプロシージャ

このシステムプロシージャを使用すると、SQL Remote Message Agent の終了直前にカスタムアクションを追加できます。

#hook_dict テーブルのロー

名前	値	説明
send	*true* または *false*	処理がレプリケーションの送信フェーズを実行中かどうかを示します。
receive	*true* または *false*	処理がレプリケーションの受信フェーズを実行しているかどうかを示します。
exitcode	*integer*	0 以外の終了コードはエラーを示します。

備考

この名前のプロシージャが存在する場合、プロシージャは、SQL Remote Message Agent が停止する前の最後のイベントとして呼び出されます。

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

1.9.1.5.4 sp_hook_dbremote_shutdown システムプロシージャ

このシステムプロシージャを使用すると、SQL Remote Message Agent のシャットダウンを開始できます。

#hook_dict テーブルのロー

名前	値	説明
send	*true* または *false*	処理がレプリケーションの送信フェーズを実行中かどうかを示します。
receive	*true* または *false*	処理がレプリケーションの受信フェーズを実行しているかどうかを示します。
shutdown	*true* または *false*	プロシージャが呼び出されるときにはこのローは *false* です。プロシージャがこのローを *true* に更新すると、SQL Remote Message Agent がシャットダウンされます。

備考

この名前のプロシージャが存在する場合、プロシージャは、SQL Remote Message Agent がメッセージを送受信していないときに呼び出され、SQL Remote Message Agent のフックで開始されるシャットダウンを可能にします。

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

1.9.1.5.5 sp_hook_dbremote_receive_begin システムプロシージャ

このシステムプロシージャを使用すると、レプリケーションの受信フェーズの開始前にアクションを実行できます。

#hook_dict テーブルのロー

なし

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

1.9.1.5.6 sp_hook_dbremote_receive_end システムプロシージャ

このシステムプロシージャを使用すると、レプリケーションの受信フェーズの終了後にアクションを実行できます。

#hook_dict テーブルのロー

なし

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

1.9.1.5.7 sp_hook_dbremote_send_begin

このストアドプロシージャを使用すると、レプリケーションの送信フェーズの開始前にアクションを実行できます。

#hook_dict テーブルのロー

なし

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

1.9.1.5.8 sp_hook_dbremote_send_end

このストアドプロシージャを使用すると、レプリケーションの送信フェーズの終了後にアクションを実行できます。

#hook_dict テーブルのロー

なし

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

1.9.1.5.9 sp_hook_dbremote_message_sent

このストアドプロシージャを使用すると、任意のメッセージが送信された後にアクションを実行できます。

#hook_dict テーブルのロー

名前	値
remoteuser	メッセージの送信先

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

1.9.1.5.10 sp_hook_dbremote_message_missing

このストアドプロシージャを使用すると、SQL Remote Message Agent がリモートユーザからの 1 つ以上のメッセージが見つからないと判断した場合にアクションを実行できます。

#hook_dict テーブルのロー

名前	値
remoteuser	メッセージを再送する必要があるリモートユーザの名前

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。

- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

1.9.1.5.11 sp_hook_dbremote_message_apply_begin

このストアドプロシージャを使用すると、SQL Remote Message Agent がユーザからの一連のメッセージを適用する直前にアクションを実行できます。

#hook_dict テーブルのロー

名前	値
remote user	適用されるメッセージを送信したリモートユーザの名前

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

1.9.1.5.12 sp_hook_dbremote_message_apply_end

このストアドプロシージャを使用すると、SQL Remote Message Agent がユーザからの一連のメッセージを適用した直後にアクションを実行できます。

#hook_dict テーブルのロー

名前	値
remoteuser	適用されたメッセージを送信したリモートユーザの名前

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

1.9.2 SQL Remote システムテーブル

SQL Remote のシステム情報は、SQL Anywhere カタログに保持されます。この情報をより分かりやすい形にしたものが、一連のシステムビューに保持されます。

次のビューを使用して SQL Remote データにアクセスできます。

- SYSARTICLE システムビュー
- SYSARTICLECOL システムビュー
- SYSPUBLICATION システムビュー
- SYSREMOTEOPTION システムビュー
- SYSREMOTEOPTIONTYPE システムビュー
- SYSREMOTETYPE システムビュー
- SYSREMOTEUSER システムビュー
- SYSSUBSCRIPTION システムビュー

1.9.3 SQL Remote の SQL 文

次の SQL 文を使用して、SQL Remote コマンドを実行します。

- ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]
- ALTER REMOTE MESSAGE TYPE 文 [SQL Remote]
- CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]
- CREATE REMOTE [MESSAGE] TYPE 文 [SQL Remote]
- CREATE SUBSCRIPTION 文 [SQL Remote]
- CREATE TRIGGER 文
- DROP PUBLICATION 文 [Mobile Link] [SQL Remote]
- DROP REMOTE MESSAGE TYPE 文 [SQL Remote]
- DROP SUBSCRIPTION 文 [SQL Remote]
- GRANT CONSOLIDATE 文 [SQL Remote]
- GRANT PUBLISH 文 [SQL Remote]
- GRANT REMOTE 文 [SQL Remote]
- GRANT ROLE 文

- PASSTHROUGH 文 [SQL Remote]
- REMOTE RESET 文 [SQL Remote]
- REVOKE CONSOLIDATE 文 [SQL Remote]
- REVOKE PUBLISH 文 [SQL Remote]
- REVOKE REMOTE 文 [SQL Remote]
- REVOKE ROLE 文
- SET REMOTE OPTION 文 [SQL Remote]
- START SUBSCRIPTION 文 [SQL Remote]
- STOP SUBSCRIPTION 文 [SQL Remote]
- SYNCHRONIZE SUBSCRIPTION 文 [SQL Remote]
- UPDATE 文 [SQL Remote]

このセクションの内容:

関連情報

SET REMOTE OPTION 文 [SQL Remote] [242 ページ]

1.9.3.1 SET REMOTE OPTION 文 [SQL Remote]

SQL Remote メッセージリンクのメッセージ制御パラメータを設定します。

構文

```
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
```

パラメータ

userid

`userid` を指定しない場合、現在のパブリッシャが使用されます。

common-options

FILE、FTP、HTTP、SMTP メッセージシステムに共通のオプションは、次のとおりです。

debug

このパラメータは、YES または NO のいずれかに設定されます。デフォルトは NO です。YES を設定すると、メッセージシステムに固有のデバッグ出力が表示されます。この情報は、メッセージシステムのトラブルシューティングに使用できます。

max_retries

デフォルトでは、SQL Remote が継続モードで実行されていて、メッセージシステムにアクセスするときにエラーが発生する場合、送受信フェーズの後に停止します。このパラメータを使用して、停止前に SQL Remote が再試行する送受信フェーズの回数を指定します。

output_log_send_on_error

エラーが発生すると、情報を送信します。

output_log_send_limit

統合データベースに送信される情報の量を制限します。output_log_send_limit オプションには、統合データベースに送信されるバイト数を指定します。これは、出力ログの最後に表示されます (つまり、最後のエントリです)。デフォルトは 5K です。

output_log_send_now

YES に設定されると、統合データベースに出力ログ情報を送信します。次のポーリング時にリモートデータベースは出力ログ情報を送信し、その後、output_log_send_now オプションが NO にリセットされます。

pause_after_failure

このパラメータが使えるのは、max_retries がゼロ以外の値に指定され、SQL Remote が継続モードで実行されている場合です。メッセージシステムでエラーが発生すると、このパラメータは、SQL Remote が送受信フェーズを再試行する間に待機する時間を秒で指定します。

encode_dll

カスタムエンコードスキームを実装している場合は、作成したカスタムエンコード DLL のフルパスをこの値に設定する必要があります。

file-options

これらのオプションは、ファイルメッセージシステムにのみ適用されます。

directory

メッセージが格納されるディレクトリです。このパラメータは、SQLREMOTE 環境変数の代替となります。

invalid_extensions

メッセージングシステムでのファイルの生成時に SQL Remote Message Agent (dbremote) で使用されないようにするファイル拡張子の、カンマで区切られたリスト。

Unlink_delay

ファイル削除に失敗した後、次にファイルの削除を試みるまでの秒数。unlink_delay に対して値が定義されていない場合、デフォルト動作は、最初の試行失敗の後に 1 秒間、2 回目の試行失敗の後に 2 秒間、3 回目の試行失敗の後に 3 秒間、4 回目の試行失敗の後に 4 秒間一時停止するように設定されています。

ftp-options

これらのオプションは、FTP メッセージシステムにのみ適用されます。

active_mode

このパラメータは、SQL Remote がクライアント/サーバ接続を確立する方法を制御します。このパラメータは、YES または NO のいずれかに設定されます。デフォルトは NO (受動モード) です。受動モードは推奨の転送モードで、FTP メッセージリンクにデフォルトとして設定されています。受動モードでは、すべてのデータ転送接続はクライアント (この場合はメッセージリンク) から開始されます。アクティブモードでは、FTP サーバがすべてのデータ接続を開始します。

host

FTP サーバを実行しているコンピュータのホスト名。このパラメータには、ホスト名 (ftp.sap.com など) または IP アドレス (192.138.151.66 など) を使用できます。

invalid_extensions

メッセージングシステムでのファイルの生成時に dbremote で使用されないようにするファイル拡張子の、カンマで区切られたリスト。

password

FTP ホストにアクセスするためのパスワード。

port

FTP の接続に使用される IP ポート番号。このパラーメータは通常は不要です。

reconnect_retries

失敗になる前に、リンクがサーバでソケットを開こうと試行する回数。デフォルト値は 4 です。このパラメータを設定すると、再接続のみに影響します。FTP リンクによる最初の接続には影響しません。

reconnect_pause

接続の試行失敗後に次の接続まで待機する秒数。デフォルトは 30 秒です。このパラメータを設定すると、再接続のみに影響します。FTP リンクによる最初の接続には影響しません。

root_directory

メッセージが保存される、FTP ホストサイトのルートディレクトリ。

suppress_dialogs

このパラメータは、TRUE または FALSE に設定されます。TRUE に設定されている場合は、FTP サーバへの接続の試行失敗後に、接続ウィンドウは表示されません。代わりに、エラーが発生します。

user

FTP ホストにアクセスするためのユーザ名。

http-options

これらのオプションは、HTTP メッセージシステムにのみ適用されます。

certificate

安全な (HTTPS) 要求を行うには、HTTPS サーバで使用される証明書にクライアントがアクセスできる必要があります。必要な情報は、セミコロンで区切られたキーワード/値のペアの文字列で指定されます。ファイルキーワードを使用して証明書のファイル名を指定できます。ファイルキーワードと証明書キーワードを一緒に指定することはできません。次のキーワードを使用できます。

キーワード	省略形	説明
file		証明書のファイル名
certificate	cert	証明書自体

キーワード	省略形	説明
company	co	証明書で指定された会社
unit		証明書で指定された会社の部署
name		証明書で指定された通称

証明書は、HTTPS サーバに対する要求、または安全でないサーバから安全なサーバにリダイレクトされる可能性がある要求に対してのみ必要です。PEM でフォーマットされた証明書のみがサポートされています (たとえば、certificate='file=filename'

SQL Anywhere データベース内で証明書名を作成するには:

```
CREATE OR REPLACE CERTIFICATE certificate_name FROM FILE 'certificate_file';
```

HTTPS メッセージタイプの証明書名を使用するには:

```
SET REMOTE HTTP OPTION user_name.certificate= 'cert_name=certificate_name';
```

client_port

SQL Remote が HTTP を使用して通信するポート番号を識別します。これは "送信" TCP/IP 接続をフィルタするファイアウォール経由で接続するためのもので、この用途にかぎり推奨されています。単一のポート番号、ポート番号の範囲、または両方の組み合わせを指定できます。指定したクライアントポートの数が少ないと、SQL Remote が前回の実行でポートを閉じた後すぐにオペレーティングシステムがポートを解放しなかった場合に、SQL Remote でメッセージの送受信ができなくなる可能性があります。

debug

YES に設定すると、すべての HTTP コマンドと応答が出力ログに表示されます。この情報は、HTTP のサポート問題のトラブルシューティングに使用できます。デフォルトは NO です。

https

HTTPS (**https=yes**) または HTTP (**https=no**) を使用するかどうかを指定します。

password

メッセージサーバのデータベースパスワード。パスワードは、RFC 2617 の基本認証を使用してサードパーティの HTTP サーバとゲートウェイに対する認証を行います。

proxy_host

プロキシサーバの URI を指定します。SQL Remote がプロキシサーバを介してネットワークにアクセスする場合に使用します。SQL Remote がプロキシサーバに接続し、そのプロキシサーバを介してメッセージサーバに要求を送信することを示します。

reconnect_retries

失敗になる前に、リンクがサーバでソケットを開こうと試行する回数。デフォルト値は 4 です。このパラメータを設定すると、再接続のみに影響します。FTP リンクによる最初の接続には影響しません。

reconnect_pause

接続の試行失敗後に次の接続まで待機する秒数。デフォルトは 30 秒です。このパラメータを設定すると、再接続のみに影響します。FTP リンクによる最初の接続には影響しません。

root_directory

この HTTP 制御パラメータは、クライアント側で指定された場合には無視されます。sr_add_message_server または sr_update_message_server ストアドプロシージャを呼び出す前に、メッセージサーバでこの制御パラメータを定義します。HTTP メッセージシステムを使用する場合、リモートユーザまたはパブリッシャに指定するアドレスには、1 つのサブディレクトリのみを含めることができます。複数のサブディレクトリを含めることはできません。

url

サーバ名または IP アドレスを指定し、任意で、使用する HTTP サーバのポート番号をセミコロンで区切って指定します。要求が Relay Server を経由する場合は、URL 拡張子を任意で追加して、要求の渡し先のサーバファームを示すことができます。

user

メッセージサーバのデータベースユーザ ID。RFC 2617 の基本認証を使用してサードパーティの HTTP サーバとゲートウェイに対する認証を行います。

smtp-options

これらのオプションは、SMTP メッセージシステムにのみ適用されます。

local_host

ローカルコンピュータの名前。SQL Remote がローカルホスト名を決定できないコンピュータで設定すると便利です。ローカルホスト名は、任意の SMTP サーバとのセッションを開始するのに必要です。ほとんどのネットワーク環境では、ローカルホスト名が自動的に決定されるため、このエントリは不要です。

pop3_host

POP ホストを実行しているコンピュータの名前。一般的には、SMTP ホストと同じ名前です。SMTP/POP3 ログインウィンドウの POP3 ホストフィールドに対応しています。

pop3_password

メールの受信に使用するパスワード。SMTP/POP3 ログインウィンドウのパスワードフィールドに対応しています。

pop3_port

POP サーバの受信対象ポートの番号。デフォルトは 110 です。SMTP/POP3 ログインウィンドウのポートフィールドに対応しています。

pop3_userid

メールの受信に使用するユーザ ID。POP ユーザ ID は、SMTP/POP3 ログインウィンドウのユーザ ID フィールドに対応しています。必ず POP ホスト管理者からユーザ ID を取得してください。

smtp_host

SMTP サーバが動作しているコンピュータの名前。SMTP/POP3 ログインウィンドウの SMTP ホストフィールドに対応しています。

top_supported

受信メッセージを列挙するときに、SQL Remote は TOP という POP3 コマンドを使用します。TOP コマンドは、すべての POP サーバでサポートされているわけではありません。top_supported パラメータを NO に設定すると、SQL Remote は RETR コマンドを使用します。このコマンドは TOP よりも効率は落ちますが、すべての POP サーバで動作します。デフォルトは YES です。

smtp_authenticate

SMTP リンクがユーザを認証するかどうかを決定します。デフォルト値は YES です。SMTP 認証を無効にする場合は、このパラメータを NO に設定します。

smtp_userid

SMTP 認証のユーザ ID。デフォルトでは、このパラメータは pop3_userid パラメータと同じ値を取ります。smtp_userid は、ユーザ ID が POP サーバ上のユーザ ID と異なる場合にのみ設定する必要があります。

smtp_password

SMTP 認証のパスワード。デフォルトでは、このパラメータは pop3_password パラメータと同じ値を取ります。smtp_password は、ユーザ ID が POP サーバ上のユーザ ID と異なる場合にのみ設定する必要があります。

smtp_port

SMTP サーバの現在の受信対象ポートの番号。デフォルトは 25 です。SMTP/POP3 ログインウィンドウのポートフィールドに対応しています。

suppress_dialogs

このパラメータが true に設定されている場合は、メールサーバへの接続の試行失敗後に、接続ウィンドウは表示されません。代わりに、エラーが発生します。

備考

メッセージリンクの初回使用時に、ユーザがメッセージリンクパラメータを [メッセージリンク] ウィンドウに入力すると、SQL Remote (dbremote) Message Agent はそのパラメータを保存します。その場合はこの文を明示的に使用する必要はありません。この文は、たくさんのデータベースから抽出を行うための統合データベースを作成する場合に非常に効果的です。

オプション名では、大文字と小文字が区別されます。オプションの大文字小文字の区別は、オプションによって異なります。Boolean 値では大文字と小文字が区別されず、パスワード、ディレクトリ、その他の文字列の大文字と小文字の区別は、ファイルシステム (ディレクトリ名の場合) またはデータベース (ユーザ ID とパスワードの場合) の大文字と小文字の区別に依存します。

権限

パブリッシャは自分自身のオプションを設定できます。それ以外の場合、SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

関連する動作

オートコミット。

標準

ANSI/ISO SQL 標準

標準になし。

例

次の文は、ユーザ Sam_Singer 用の FTP リンクに対し、FTP ホストを `ftp.mycompany.com` に設定します。

```
SET REMOTE FTP OPTION Sam_Singer.host = 'ftp.mycompany.com';
```

次の文は、生成されるメッセージの特定ファイル拡張子を SQL Remote が使用しないようにします。

```
SET REMOTE FTP OPTION "Public"."invalid_extensions"='exe,pif,dll,bat,cmd,vbs';
```

次の文は、ユーザ Sam_Singer の HTTP リンクのローカルホストを示す URL を設定します。

```
SET REMOTE HTTP OPTION Sam_Singer.url='localhost:8033';
```

次の文は、要求を srhttp ファームに転送する Relay Server を指すように HTTP URL を設定します。

```
SET REMOTE HTTP OPTION "public"."url"='iis7.company.com:80/rs/client/rs.dll/
srhttp';
```

関連情報

カスタムエンコードスキーム [117 ページ]
FTP メッセージシステム [124 ページ]
FILE メッセージシステム [123 ページ]
HTTP メッセージシステム [127 ページ]
SMTP メッセージシステム [131 ページ]
リモートデータベースからのエラーの収集 (SQL) [146 ページ]
チュートリアル: 別のメッセージサーバで HTTP メッセージシステムを使用するレプリケーションシステムの設定 [185 ページ]
リモートメッセージタイプ制御パラメータ [121 ページ]

1.10 このマニュアルの印刷、再生、および再配布

次の条件に従うかぎり、このマニュアルの全部または一部を使用、印刷、再生、配布することができます。

1. ここに示したものとそれ以外のすべての版権と商標の表示をすべてのコピーに含めること。
2. マニュアルに変更を加えないこと。
3. SAP 以外の人間がマニュアルの著者または情報源であるかのように示す一切の行為をしないこと。

ここに記載された情報は事前の通知なしに変更されることがあります。

重要免責事項および法的情報

コードサンプル

この文書に含まれるソフトウェアコード及び / 又はコードライン / 文字列 (「コード」) はすべてサンプルとしてのみ提供されるものであり、本稼動システム環境で使用することが目的ではありません。「コード」は、特定のコードの構文及び表現規則を分かりやすく説明及び視覚化することのみを目的としています。SAP は、この文書に記載される「コード」の正確性及び完全性の保証を行いません。更に、SAP は、「コード」の使用により発生したエラー又は損害が SAP の故意又は重大な過失が原因で発生させたものでない限り、そのエラー又は損害に対して一切責任を負いません。

アクセシビリティ

この SAP 文書に含まれる情報は、公開日現在のアクセシビリティ基準に関する SAP の最新の見解を表明するものであり、ソフトウェア製品のアクセシビリティ機能の確実な提供方法に関する拘束力のあるガイドラインとして意図されるものではありません。SAP は、この文書に関する一切の責任を明確に放棄するものです。ただし、この免責事項は、SAP の意図的な違法行為または重大な過失による場合は、適用されません。さらに、この文書により SAP の直接的または間接的な契約上の義務が発生することは一切ありません。

ジェンダーニュートラルな表現

SAP 文書では、可能な限りジェンダーニュートラルな表現を使用しています。文脈により、文書の読者は「あなた」と直接的な呼ばれ方をされたり、ジェンダーニュートラルな名詞 (例: 「販売員」又は「勤務日数」) で表現されます。ただし、男女両方を指すとき、三人称単数形の使用が避けられない又はジェンダーニュートラルな名詞が存在しない場合、SAP はその名詞又は代名詞の男性形を使用する権利を有します。これは、文書を分かりやすくするためです。

インターネットハイパーリンク

SAP 文書にはインターネットへのハイパーリンクが含まれる場合があります。これらのハイパーリンクは、関連情報を見いだすヒントを提供することが目的です。SAP は、この関連情報の可用性や正確性又はこの情報が特定の目的に役立つことの保証は行いません。SAP は、関連情報の使用により発生した損害が、SAP の重大な過失又は意図的な違法行為が原因で発生したものでない限り、その損害に対して一切責任を負いません。すべてのリンクは、透明性を目的に分類されています (http://help.sap.com/disclaimer を参照)。

go.sap.com/registration/contact.html