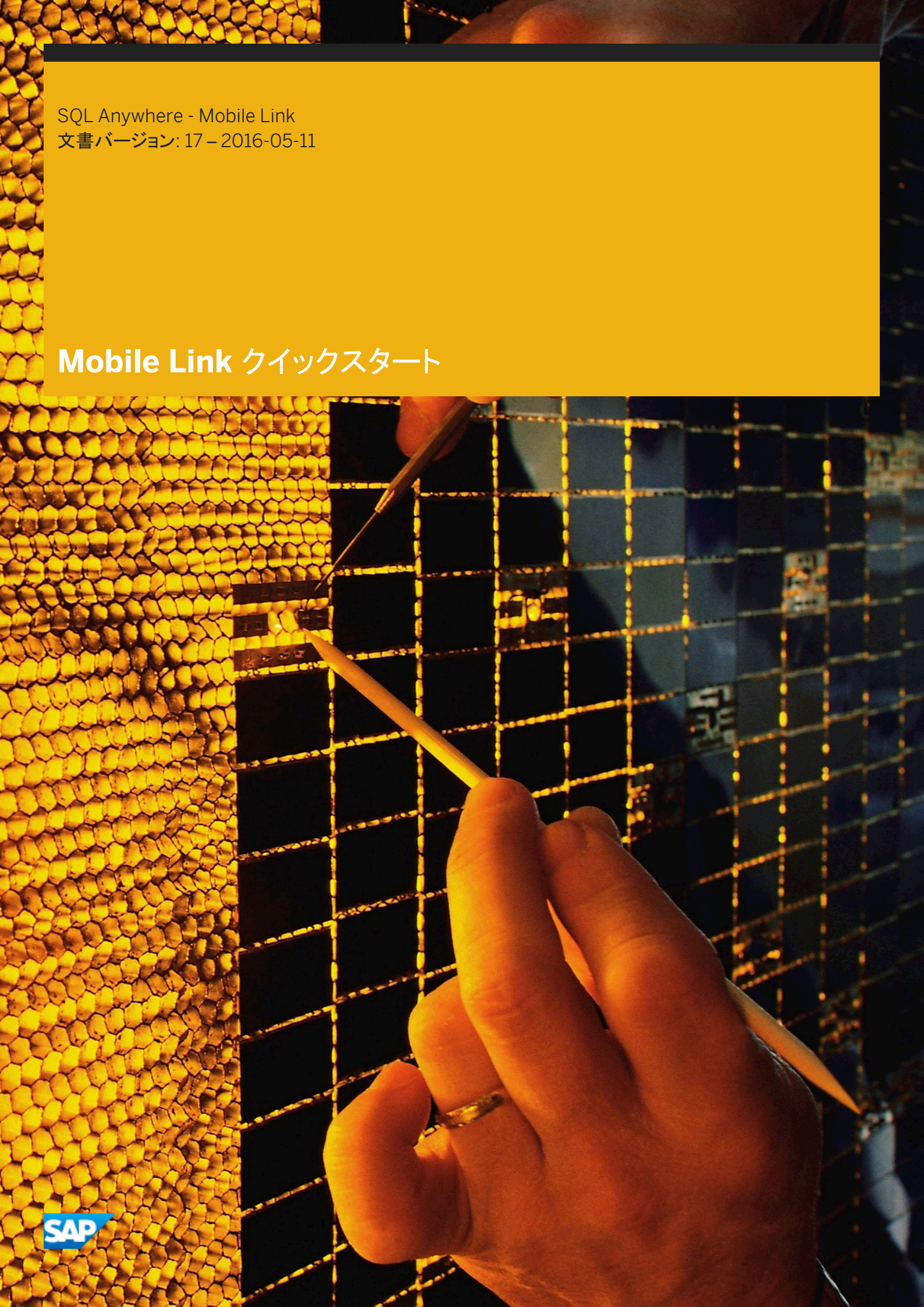
SQL Anywhere - Mobile Link
文書バージョン: 17 – 2016-05-11

Mobile Link クイックスタート



目次

1 Mobile Link クイックスタート

このマニュアルでは、セッションベースのリレーショナルデータベース同期システムである Mobile Link について説明します。Mobile Link テクノロジは、双方向レプリケーションを可能にし、モバイルコンピューティング環境に非常に適しています。

このセクションの内容:

1.1 Mobile Link 同期

Mobile Link は、Ultra Light と SQL Anywhere のリモートデータベースを統合データベースと同期させるために設計された、セッションベース同期テクノロジです。

このセクションの内容:

Mobile Link は、1 つまたは複数の中央のデータソースと断続的に接続する多数のリモートアプリケーション間でデータを同期するように設計されています。

Mobile Link アプリケーション開発 [11 ページ]
データベースアプリケーションには、次の 2 つの基本的なアーキテクチャがあります。オンラインアプリケーションと随時接続スマートクライアントアプリケーションです。

Mobile Link アプリケーションの開発オプション [14 ページ]
Mobile Link では、アプリケーション開発のためのさまざまな方法が用意されています。それぞれの方法を単独で使うことも、組み合わせて使うこともできます。

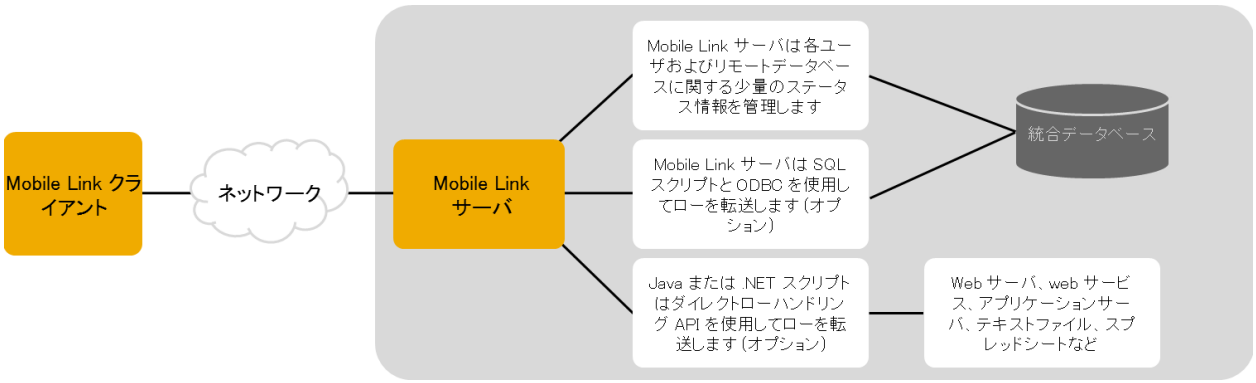
サーバ側の同期ロジックの作成オプション [15 ページ]
Mobile Link 同期スクリプトは、SQL で記述することも、Java (Java 用 Mobile Link サーバ API を使用) または .NET (.NET 用 Mobile Link サーバ API を使用) で記述することもできます。

同期処理 [17 ページ]
同期とは、Mobile Link クライアントと中央データソースの間で行われるデータ交換処理です。この処理の間、クライアントは Mobile Link サーバとのセッションを確立して維持します。同期に成功した場合、セッションによってリモートデータベースと統合データベースは互いに一貫した状態に保たれます。

Mobile Link セキュリティに関する考慮事項 [24 ページ]
Mobile Link のインストール環境のように、広範囲の分散システム全体のデータの安全を確保するには、いくつかの要素があります。

1.1.1 Mobile Link アプリケーションの各部分

Mobile Link 同期では、多くのクライアントが Mobile Link サーバを介して中央のデータソースと同期します。



Mobile Link クライアント

クライアントは、Android デバイス、サーバ、またはデスクトップコンピュータにインストールできます。2 種類のクライアントがサポートされています。Ultra Light および SQL Anywhere データベース。1 つの Mobile Link インストール環境では、これらのうちのどちらか 1 つまたは両方を使用できます。

ネットワーク

Mobile Link サーバと Mobile Link クライアント間の接続では、複数のプロトコルを使用できます。次を参照してください。

- Mobile Link サーバ: -x mlsrv17 オプション

- Ultra Light および SQL Anywhere クライアント: Mobile Link クライアントネットワークプロトコルオプション

Mobile Link サーバ

同期処理を管理し、すべての Mobile Link クライアントと統合データベースサーバ間のインタフェースを提供します。

統合データベース

統合データベースには通常、同期システムのアプリケーション情報の中核となるコピーが収められています。また、通常は、Mobile Link 同期に必要なシステムテーブルとプロシージャ、および同期するために必要なステータス情報も格納されます。

ステータス情報

Mobile Link サーバは通常、統合データベース内のシステムテーブルの同期情報を管理します。情報の管理は、ODBC 接続を介して行われます。

また、ステータス情報を別のデータベースに格納することもできます。

SQL ローハンドリング

Mobile Link サーバ用に SQL スクリプトを作成すると、サーバではこれらのスクリプトを使用し、ODBC 接続を介して、統合データベースとの間でローが転送されます。

ダイレクトローハンドリング

Mobile Link のダイレクトローハンドリングを使用して、統合データベース以外にオプションで他のデータソースと同期することもできます。

同期スクリプト

リモートデータベースの各テーブルに対して同期スクリプトを記述し、これらのスクリプトを統合データベースの Mobile Link システムテーブルに保存してください。これらのスクリプトは、アップロードデータに対して行う処理や、ダウンロードするデータを決定します。スクリプトには、テーブルスクリプトと接続レベルスクリプトの 2 種類があります。

関連情報

サーバ側の同期ロジックの作成オプション [15 ページ]

1.1.2 Mobile Link の機能とアーキテクチャ

Mobile Link 同期には高い適応性と柔軟性があります。

以下に、主な機能の一部を示します。

機能

使い始めるのが簡単

同期モデル作成ウィザードを使用すると、簡単に同期アプリケーションを作成できます。このウィザードによって、複雑な同期システムに伴う多くの難解な実装作業が処理されます。SQL Central を使用すると、同期モデルをオフラインで表示し、簡単なインタフェースで変更を行い、展開オプションを使用してモデルを統合データベースに展開できます。

モニタとレポート

Mobile Link には、同期をモニタする 3 つのメカニズムが用意されています。Mobile Link プロファイラ、Mobile Link 用 SQL Anywhere モニタ、統計スクリプトです。

パフォーマンスチューニング

Mobile Link のパフォーマンスをチューニングするための複数のメカニズムがあります。たとえば、競合レベル、アップロードのキャッシュサイズ、データベース接続数、ロギングの冗長性、または BLOB のキャッシュサイズを調整できます。

スケーラビリティ

Mobile Link はスケーラビリティに優れ、堅牢な同期プラットフォームです。Mobile Link の単一のサーバが数千の同期を同時に処理したり、負荷分散を使用して複数の Mobile Link サーバを同時に稼働したりできます。Mobile Link サーバはマルチスレッド化されており、統合データベースで接続プールを使用します。

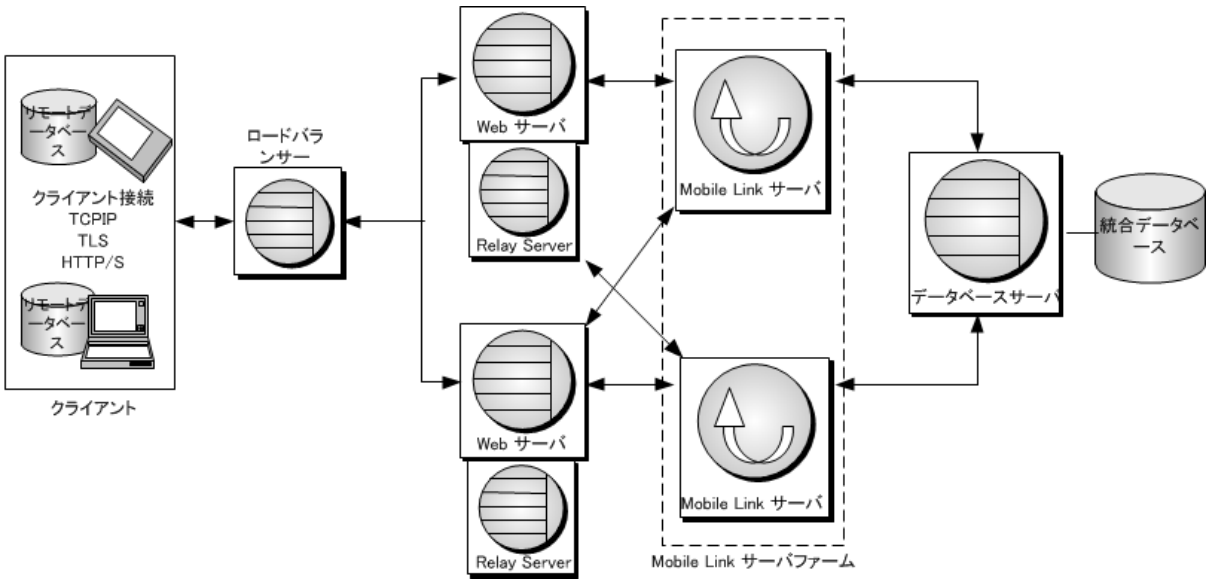
セキュリティ

Mobile Link には、既存の認証に統合できるユーザ認証、暗号化、安全な証明書の交換によって機能するトランスポートレイヤセキュリティなど、豊富なセキュリティオプションがあります。Mobile Link には、FIPS 認定のセキュリティオプションもあります。

Relay Server のリバースプロキシ

Relay Server は、Web サーバを通じて通信するモバイルデバイスとバックエンドサーバの間で安全な負荷分散通信を実現するリバースプロキシです。

次の図は、Mobile Link 環境への Relay Server の組み込み方を示しています。



アーキテクチャ

データ調整

Mobile Link によって、データの特定部分を同期対象として選択できます。また、Mobile Link 同期では、異なるデータベースで行われた変更内容の競合を解決できます。同期処理は、SQL、Java または .NET アプリケーションとして作成でき

る同期ロジックによって制御されます。この論理の各部は、スクリプトと呼ばれます。スクリプトを使用すると、たとえば、アップロードされたデータを統合データベースに適用する方法の指定やダウンロード内容を取得するデータベースの指定を行ったり、統合データベースとリモートデータベースとで異なるスキーマや名前を処理したりできます。イベントベースのスクリプト機能により、競合解決、エラーレポート、ユーザ認証などの機能を含め、同期処理の設計がきわめて柔軟になります。

双方向の同期

すべてのロケーションでデータベースを変更できます。

アップロード専用の同期またはダウンロード専用の同期

デフォルトでは、同期は双方向で、アップロードとダウンロードの両方が行われます。ただし、アップロード専用の同期またはダウンロード専用の同期を選択することもできます。

ファイルベースのダウンロード

ダウンロード内容はファイルとして配布することが可能であり、同期の変更をオフラインで配布できます。これには、適切なデータの適用を保証する機能が含まれます。

サーバ起動同期

Mobile Link 同期は、統合データベースから開始できます。これは、データの更新をリモートデータベースにプッシュし、リモートデータベースによってデータが統合データベースにアップロードされるようにできることを意味しています。

サーバ起動同期への代替機能としてサーバ起動リモートタスク (SIRT) を使用できます。

複数のネットワークプロトコル

同期は TCP/IP、HTTP、HTTPS 経由で実行できます。

セッションベース

すべての変更内容は、単一のトランザクションでアップロードし、単一のトランザクションでダウンロードできます。同期が成功するたびに、統合データベースとリモートデータベースが一貫した状態になります (トランザクションの順序を保持する場合は、リモートデータベースの各トランザクションを別個のトランザクションとしてアップロードすることもできます)。

トランザクション全体が同期されるか、トランザクション全体がまったく同期されないかのどちらかになります。これにより、各データベースでトランザクション単位の整合性が確保されます。

データの一貫性

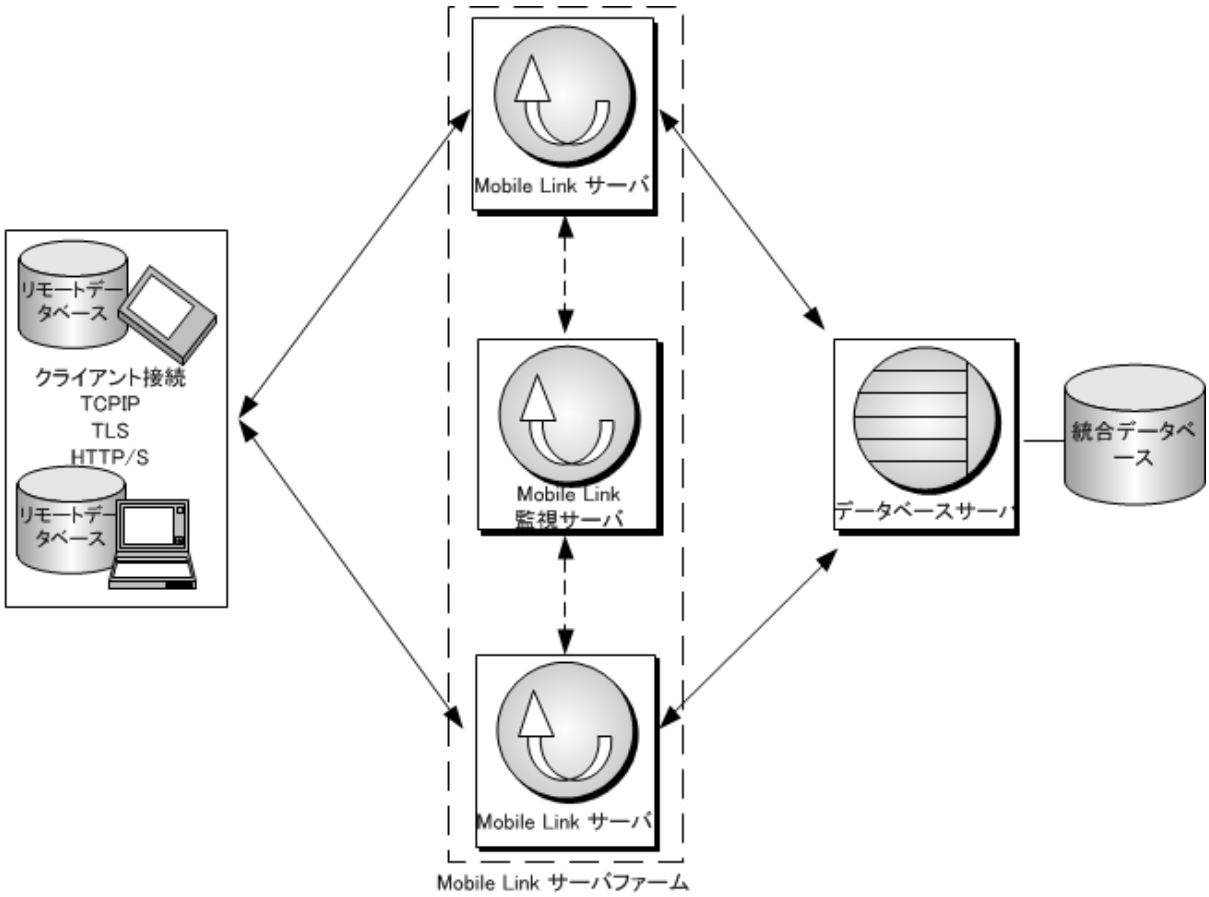
Mobile Link は、緩やかな一貫性方式を使用しています。つまり、変更内容はすべて一貫性が保たれるように各サイトで同期されますが、時間的にはわずかなズレがあるため、ある瞬間だけを見ると、各サイトに存在するデータのコピーが異なる場合もあります。

多様なハードウェアとソフトウェアのプラットフォーム

Mobile Link の統合データベースとして、各種の一般的なデータベース管理システムを使用できます。また、Mobile Link サーバ API を使用して任意のデータソースへの同期を定義することもできます。リモートデータベースには、SQL Anywhere または Ultra Light を使用できます。Mobile Link サーバは、Windows、UNIX、Linux、Mac OS X 上で動作します。SQL Anywhere は、Windows、UNIX、Linux、Mac OS X 上で動作します。Ultra Light は、Android、iOS、Windows Phone、Windows Mobile 上で動作します。

Mobile Link 監視サーバ

Mobile Link 監視サーバは、サーバファーム内の 1 台の Mobile Link サーバのみがプライマリサーバとして動作するようにします。これにより、サーバ起動同期環境での冗長な通知が回避されます。次の図は、サーバファーム環境の Mobile Link 監視サーバを示しています。



1.1.3 Mobile Link のクイックスタート

Mobile Link は、1 つまたは複数の中央のデータソースと断続的に接続する多数のリモートアプリケーション間でデータを同期するように設計されています。

基本的な Mobile Link アプリケーションでは、リモートクライアントは SQL Anywhere または Ultra Light のデータベースで、中央のデータソースはサポートされている ODBC 準拠のリレーショナルデータベースのいずれかです。Mobile Link サーバ API を使用してこのアーキテクチャを拡張すると、サーバ側の同期先の制限を事実上なくすことができます。

すべての Mobile Link アプリケーションで、Mobile Link サーバが同期処理の主要要素です。通常は、Mobile Link リモートサイトで Mobile Link サーバへの接続を開くと、同期が開始されます。同期中に、リモートサイト側の Mobile Link クライアントは、前回の同期後にリモートデータベースに対して行われたデータベースの変更をアップロードできます。Mobile Link サーバは、このデータを受信すると、統合データベースを更新し、変更内容を統合データベースからリモートデータベースにダウンロードできます。

Mobile Link アプリケーションの開発を始める最も簡単な方法は、同期モデル作成ウィザードを使用することです。このウィザードを使用すると、以下で説明している手順の大部分がウィザードにより処理されます。

ただし、Mobile Link モデルを使用する場合でも、Mobile Link 同期の処理とコンポーネントは理解しておく必要があります。

Mobile Link アプリケーションの概要

- 統合データベースを設定
  データベースに対して設定スクリプトを実行し、Mobile Link 同期に必要なシステムオブジェクトを追加します。または、これらのオブジェクトを格納するためのシステムデータベースを別途作成します。
- リモートデータベースを設定
  - リモートデータベースとしては SQL Anywhere と Ultra Light のどちらも使用できます。また、両方を組み合わせて使用することもできます。
  - リモートデータベースで Mobile Link ユーザを作成します。
  - SQL Anywhere のリモートデータベースでアップロードを特定するには、パブリケーションとサブスクリプションを作成します。Ultra Light のリモートデータベースでアップロードを特定するには、パブリケーションを作成します。
- サーバ同期ロジックを作成して、アップロードの適用形式を特定します。
- タイムスタンプベースの同期を設定して、前回のダウンロード以降に変更されたデータをダウンロードします。
- Mobile Link サーバを起動します。
- クライアントの同期を開始します。

入門情報

Mobile Link の入門情報については、Mobile Link - クイックスタートを参照してください。

Mobile Link - サーバ管理ガイドには、Mobile Link のクイックスタートに役立つ情報をはじめとして、同期方法やスクリプトの作成に関する情報も記載されています。

チュートリアル

以下のチュートリアルは、新しいユーザのための入門用チュートリアルから、高度な機能の使用方法の説明にまで及びます。

- Mobile Link の CustDB サンプル
- Mobile Link Contact サンプル
- チュートリアル: Mobile Link の概要
- チュートリアル: SQL Anywhere 統合データベースでの Mobile Link の使用
- チュートリアル: Ultra Light の CustDB サンプルアプリケーションの構築
- チュートリアル: Oracle Database 11g での Mobile Link の使用
- チュートリアル: Adaptive Server Enterprise 統合データベースと Mobile Link の使用
- チュートリアル: カスタムユーザ認証用の Java と .NET の使用
- チュートリアル: ダイレクトローハンドリングの使用
- チュートリアル: Microsoft Excel との同期
- チュートリアル: XML との同期
- チュートリアル: リモートデータベースの集中管理の使用
- チュートリアル: スクリプトバージョン句を使用したスキーマの変更
- チュートリアル: ScriptVersion 拡張オプションを使用したスキーマの変更

- チュートリアル: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート

クイックスタートのためのその他の資料

- Mobile Link には、Mobile Link 機能を確かめるために調べたり実行したりできるサンプルが数多く用意されています。Mobile Link のサンプルは、製品とともに *%SQLANYSAMP17%*¥MobiLink にインストールされます。
- SAP SQL Anywhere コミュニティには、開発者や技術部門エグゼクティブによるブログへのリンク (英語) があり、SQL Anywhere、Mobile Link、および関連テクノロジーの使用に関して意見やアイデアを交換できます。

関連情報

同期モデル [33 ページ]
Mobile Link の CustDB サンプル [65 ページ]
Mobile Link Contact サンプル [82 ページ]
チュートリアル: Mobile Link の概要 [97 ページ]
チュートリアル: SQL Anywhere 統合データベースでの Mobile Link の使用 [117 ページ]
チュートリアル: Oracle Database 11g での Mobile Link の使用 [131 ページ]
チュートリアル: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]
チュートリアル: カスタムユーザ認証用の Java と .NET の使用 [174 ページ]
チュートリアル: ダイレクトローハンドリングの使用 [184 ページ]
チュートリアル: Microsoft Excel との同期 [210 ページ]
チュートリアル: XML との同期 [231 ページ]
チュートリアル: リモートデータベースの集中管理の使用 [253 ページ]
チュートリアル: スクリプトバージョン句を使用したスキーマの変更 [283 ページ]
チュートリアル: ScriptVersion 拡張オプションを使用したスキーマの変更 [294 ページ]
チュートリアル: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート [302 ページ]
SAP コミュニティネットワーク

1.1.4 Mobile Link アプリケーション開発

データベースアプリケーションには、次の 2 つの基本的なアーキテクチャがあります。オンラインアプリケーションと随時接続スマートクライアントアプリケーションです。

オンラインアプリケーション

オンラインアプリケーションユーザが統合データベースに直接接続してデータを更新します。接続できない場合、ユーザによる作業はできません。

随時接続スマートクライアントアプリケーション

各ユーザがローカルデータベースを保有しています。各ユーザは接続状態にかかわらず常に自分のデータベースアプリケーションを使用できます。また、このデータベースアプリケーションはシステム内の他のデータベースと同期されます。

Mobile Link では随時接続スマートクライアントアプリケーションを作成できます。スマートクライアントアプリケーションにより、アプリケーションの利便性、効率性、スケーラビリティが大幅に向上します。しかし、アプリケーションの開発に関する新しい問題も発生します。スマートクライアントアプリケーションの開発に関する主な問題と、Mobile Link 同期環境でソリューションを実装する方法について、次に説明します。

必要な同期のみを実行

大部分のアプリケーションでは、リモートデバイスのデータを一部だけでも更新する場合に毎回統合データベース全体をダウンロードすると、大変なことになります。必要な時間と帯域が膨大なものとなり、システム全体の動作が停止してしまいます。各ユーザにとって必要なアップロードとダウンロードのみを行うようにするための方法は複数あります。

まず、各リモートデータベースには統合データベースのテーブルとカラムのサブセットのみが格納されるようにします。たとえば、地域 A の販売担当者が必要とするテーブルやカラムは地域 B の販売担当者や管理職とは異なる場合があります。

リモートデータベースで作成したテーブルやカラムのうち、同期が必要なものだけを同期対象として指定します。Mobile Link アプリケーションでは、データ型が一致していれば、名前が異なっていてもテーブルやカラムをマッピングすることができます。デフォルトではデータのアップロードとダウンロードの両方が可能ですが、Mobile Link では特定のカラムをアップロード専用またはダウンロード専用にすることもできます。

同期時には、各ユーザに関連するリモートデータベースにのみローをダウンロードするようにします。ダウンロードをリモートデータベース別、ユーザ別、またはその他の基準別に分割して行うこともできます。たとえば、地域 A の販売担当者が地域 A のデータのみを更新する必要がある場合を考えてみます。

更新する必要があるのは変更があるデータのみです。Mobile Link アプリケーションでは、アップロードはトランザクションログに基づいて行われるため、デフォルトではリモートデータベースで変更されているデータのみがアップロードされます。ダウンロードも同様に行うには、同期形式をタイムスタンプベースに指定して、データが正常にダウンロードされた日時をシステムが記録するようにします。これにより、データのダウンロードはその日時以降に変更があった場合に限られます。

また、高優先度同期方式の実装が必要になる場合もあります。たとえば、緊急のデータは更新頻度が高くなるようスケジュールし、緊急でないデータは夜間やデバイスがクレードルにある間に更新されるようスケジュールします。高優先度同期を実装するには、それぞれ異なる時刻に実行するようスケジュールされた複数のパブリケーションを作成します。

この他に、プッシュ同期により、必要に応じてデータを効果的にリモートデバイスにプッシュダウンすることもできます。たとえば、トラック運送会社の配送係が交通の混乱を知らされた場合に、該当地域に向かっているトラック運転手の更新情報をダウンロードできます。Mobile Link では、これをサーバ起動同期と呼んでいます。

アップロードの競合の処理

倉庫を例にとって考えてみます。各従業員はモバイルデバイスを保有しており、箱の搬入出時に在庫情報を更新します。最初に 100 個の箱が搬入されています。そのため各従業員のリモートデータベースと統合データベースには 100 と登録されています。デービッドが 20 個の箱を搬出しました。彼は自分のデータベースを更新して同期します。これで彼のデータベースと統合データベースの両方に 80 と登録されます。次にスーザンが 10 個の箱を搬出します。ここでスーザンは自分のデータベースを更新して同期しようとしますが、スーザンのアプリケーションは統合データベースに登録されている箱の個数が 80 ではなく100 であると想定しています。このため、アップロードの競合が発生します。

この倉庫アプリケーションの例では、この問題を解決するには、"デービッドによる更新値 - (最初の値 - スーザンによる更新値) = 正しい値" となるように、次のような競合解決論理を作成する必要があります。

```
80 - (100 - 90) = 70
```

この競合解決論理は倉庫などの在庫ベースのアプリケーションでは有効ですが、すべてのビジネスアプリケーションで適しているわけではありません。Mobile Link では、次の競合解決論理を定義できます。

在庫モデル

ローを更新してユニット数を修正します。

日付

最新の更新が適用されます (値がデータベースで変更された日付が基準です。値が同期された日付ではありません)。

操作者

たとえば、管理者を常に優先したり、レコードの所有者を常に優先したりします。

カスタム

その他実装が必要なビジネス用論理です。

場合によっては、アップロードの競合が発生しないようにシステムを設計することができます。重複を避けるためにデータがリモートで分割されている場合は、競合を回避できる可能性があります。それでも競合が発生する場合は、競合の検出と解決を行うためのプログラムを作成する必要があります。

ユニークなプライマリキー

データをアップロードし、アップロードの競合を検出して、削除されたローを統合データベースで同期するには、データベースシステム内で同期されているすべてのテーブルにユニークなプライマリキーが必要です。各ローには、データベース内だけでなく、データベースシステム全体でユニークなプライマリキーが必要です。プライマリキーは更新できないようにする必要があります。

Mobile Link では、ユニークなプライマリキーを設定するための方法が複数あります。方法の 1 つとして、プライマリキーのデータ型を GUID に設定することがあげられます。GUID (グローバルユニーク識別子) は 16 バイトの 16 進数です。Mobile Link の NEWID 関数を使用すると、新しいローに対して自動的に GUID を作成できます。

また、別の解決方法として、複合キーを使用することがあげられます。Mobile Link では各リモートデータベースにリモート ID と呼ばれるユニークな値が設定されています。リモート ID と通常のプライマリキー (序数など) を組み合わせたものをプライマリキーとして使用することができます。

SQL Anywhere ではグローバルオートインクリメント方式による解決方法もあります。あるカラムを GLOBAL AUTOINCREMENT として宣言すると、ローの追加時に、これまでのプライマリキーの最後の値を増分することにより、プライマリキーが自動的に作成されます。この解決方法は、統合データベースが SQL Anywhere の場合に最適です。

最後に、リモートデータベースに配布されるプライマリキー値のプールを作成することもできます。

同期ソリューションの開発時における各種の決定と同様に、どのプライマリキー方式を選択するかは、統合データベースとリモートデータベースに対する制御のレベルによって異なります。多くの場合、リモートデータベースは管理なしで動作できる必要があります。また、統合データベースのスキーマを変更することが困難な場合もあります。さらに、統合データベースとして RDBMS を選択した場合、すべての RDBMS ですべての機能がサポートされているわけではないため、使用できるオプションが限られることがあります。

削除の処理

同期方式に関する別の問題として、統合データベースから削除されたローの処理があげられます。たとえば、統合データベースからあるローを削除したとします。次にデービッドが自分のリモートデータベースを同期すると、この削除データがダウンロードされ、デービッドのデータベースからローが削除されます。しかし統合データベースでこの後どうすればよいでしょうか。実際にはスーザンに対しても削除データのダウンロードが必要になるため、ローを削除できません。

ダウンロード削除を処理する方法は 2 通りあります。1 つ目の方法は、ローが削除されているかどうかを示すステータスカラムを各テーブルに追加することです。この場合、ローは実際には削除されず、削除するようマークが付けられるだけです。削除するようマーク付けされたローは、後ですべてのリモートデータベースが更新されていることを確認できた時点で消去できます。また、各テーブルのシャドウテーブルを作成することもできます。シャドウテーブルには、削除されたローのプライマリキーの値が格納されています。ローを削除すると、トリガによりシャドウテーブルに値が入力されます。シャドウテーブルの値により、リモートデータベースで削除するローが決定されます。

トランザクション

同期データベースシステムでは、コミットされたデータベーストランザクションのみが同期されます。さらに、同期するデータが関わっている、コミットされたすべてのトランザクションを同期する必要があります。この処理が正しく行われないと、エラーが発生します。これは Mobile Link でのデフォルトの動作です。

また、統合データベースへの接続の独立性レベルも考慮する必要があります。データの一貫性を確保できる範囲で最高のパフォーマンスを実現できる独立性レベルを使用する必要があります。通常、独立性レベル 0 (READ UNCOMMITTED) は同期には不適切で、データの不整合を引き起こす可能性があります。

デフォルトでは、Mobile Link はアップロードには独立性レベル SQL_TXN_READ_COMMITTED を使用し、可能な場合には、ダウンロードにはスナップショットアイソレーションを使用します (不可能な場合には、SQL_TXN_READ_COMMITTED を使用します)。スナップショットアイソレーションは、トランザクションが統合データベースで閉じられるまでダウンロードがブロックされる問題を解消します。ただし、すべての RDBMS がこの機能をサポートしているわけではありません。

夏時間

毎年、夏時間から通常時間への移行時にデータベースの同期に関する問題が発生する可能性があります。秋に時刻が 1 時間戻ると、午前 2 時が午前 1 時になります。午前 1 時と午前 2 時の間に同期を実行しようとすると、同期のタイムスタンプが、たとえば最初の午前 1 時 15 分なのか次の 1 時 15 分なのかわからなくなります。

この問題を解決するには、秋になり時間が移行するときに、1 時間の間シャットダウンするか、統合データベースサーバを協定世界時 (UTC) にします。

1.1.5 Mobile Link アプリケーションの開発オプション

Mobile Link では、アプリケーション開発のためのさまざまな方法が用意されています。それぞれの方法を単独で使うことも、組み合わせて使うこともできます。

- 同期モデル作成ウィザード

このウィザードを使用すると、アプリケーションの開発作業を順を追って簡単に実行できます。まずスキーマがある中央データベースを作成し、その後でリモートデータベースと同期に必要なスクリプトを作成できます。また、このウィザードではダウンロード削除などの処理を行うためのシャドウテーブルを統合データベース上に作成できます。ウィザードが完了したら、モデルをさらにカスタマイズできます。同期モデル展開ウィザードでは、データベースとテーブルの作成、Mobile Link システムテーブルの更新、Mobile Link ユーティリティを実行するためのスクリプトの作成ができます。

展開した Mobile Link モデルがさらにカスタマイズを必要とする場合は、次のいずれかの方法で変更を加えることができます。

SQL Central

SQL Central の 17 プラグインを使用して、Mobile Link アプリケーションのすべての要素を更新できます。

システムプロシージャ

中央データベースが統合データベースとして動作するよう設定すると、Mobile Link 同期で使用するシステムオブジェクトが作成されます。この中には Mobile Link システムテーブルも含まれます。ここにはサーバ側の Mobile Link アプリケーションの大部分が格納されます。また、この中には、Mobile Link スクリプトの Mobile Link システムテーブルへの挿入やリモートユーザの登録などの作業を行うためのシステムプロシージャやユーティリティも含まれます。

Mobile Link システムテーブルの直接操作

上級ユーザの場合は、Mobile Link システムテーブルのデータの追加、削除、更新を直接行うこともできます。この操作を行うには、Mobile Link の仕組みを詳細に理解している必要があります。

関連情報

SQL Central の Mobile Link プラグイン [24 ページ]
同期モデルタスク [37 ページ]

1.1.6 サーバ側の同期ロジックの作成オプション

Mobile Link 同期スクリプトは、SQL で記述することも、Java (Java 用 Mobile Link サーバ API を使用) または .NET (.NET 用 Mobile Link サーバ API を使用) で記述することもできます。

サポートされている統合データベースと同期する場合は、通常は SQL 同期ロジックが最適です。

サポート対象外の統合データベースと同期する場合は、Java や .NET が便利です。また、SQL 言語の制限事項やデータベース管理システムの機能によって設計が制限されている場合や、異なる RDBMS タイプ間での移植性が必要な場合も、Java や .NET が便利です。

Java と .NET の同期ロジックは、SQL 論理と同様に機能します。Mobile Link サーバは、Mobile Link イベントの発生時に SQL スクリプトにアクセスするのと同様に、Mobile Link イベントの発生時に Java メソッドや .NET メソッドを呼び出すことができます。Java または .NET を使用している場合は、一部の追加処理を実行するイベントを使用できます。ただし、アップロードローやダウンロードローを直接処理するイベントのスクリプトを処理するときは、Java または .NET の実装が SQL 文字列を返す必要があります。ダイレクトローハンドリングで使用される 2 つのイベントを除き、Java または .NET の同期ロジックでは、アップロードとダウンロードに直接アクセスできません。Java または .NET から SQL 文字列で返された内容を Mobile Link が実行します。

ダイレクトローハンドリングでは、handle_UploadData イベントと handle_DownloadData イベントを使用して、データソースと同期します。この操作により、アップロードローとダウンロードローが直接操作されます。

Java または .NET でのスクリプトの作成を検討する場合のシナリオを以下に示します。

ダイレクトローハンドリング

Java および .NET の同期ロジックでは、Mobile Link を使用して、統合データベース以外のデータソース (アプリケーションサーバ、Web サーバ、ファイルなど) にアクセスできます。

認証

ユーザ認証プロシージャを Java または .NET で記述し、Mobile Link 認証を企業のセキュリティポリシーに組み込みます。

ストアドプロシージャ

RDBMS でユーザ定義のストアドプロシージャを使用できない場合は、Java または .NET でメソッドを作成できます。

外部呼び出し

プログラムが同期イベント中に外部サーバへの接続を必要とする場合は、Java または .NET 同期ロジックを使用して、同期イベントによりトリガされるアクションを実行できます。Java および .NET 同期ロジックは、複数の接続間で共有できます。

変数

データベースに変数を処理する機能がない場合は、接続または同期の間持続する変数を Java または .NET で作成できます。また、SQL スクリプトでユーザ定義の名前付きパラメータを使用することもできます。この方法は、すべてのタイプの統合データベースに使用できます。

Mobile Link サーバ API

Java および .NET 同期ロジックは、Mobile Link サーバ API を介して使用できます。Mobile Link サーバ API は、Mobile Link 同期用のクラスとインタフェースのセットです。

Java 用 Mobile Link サーバ API には、次の利点があります。

- 統合データベースへの既存の ODBC 接続に JDBC 接続としてアクセスできます。
- JDBC、Web サービス、JNI などのインタフェースを使用して、別のデータソースにアクセスできます。
- 統合データベースへの新規 JDBC 接続を作成し、現在の同期接続の外部でデータベースを変更できます。たとえば、同期接続でロールバックを行う場合でも、これをエラーログや監査に使用できます。
- 統合データベースと同期する場合は、Java コードを作成してデバッグしてから Mobile Link サーバで実行できます。多くのデータベース管理システムの SQL 開発環境は、Java アプリケーションが使用可能な環境に比べると初歩的です。
- SQL ローハンドリングとダイレクトローハンドリングの両方を使用できます。
- 高度で豊かな Java 言語が提供する多数の既存のコードやライブラリを使用できます。

.NET 用 Mobile Link サーバ API には、次の利点があります。

- .NET から ODBC を呼び出す Sap クラスを使用して、統合データベースへの既存の ODBC 接続にアクセスできます。
- ADO.NET、Web サービス、OLE DB などのインタフェースを使用して、別のデータソースにアクセスできます。
- 統合データベースと同期する場合は、.NET コードを作成してデバッグしてから、Mobile Link サーバで実行できます。多くのデータベース管理システムの SQL 開発環境は、.NET アプリケーションが使用可能な環境に比べると初歩的です。
- SQL ローハンドリングとダイレクトローハンドリングの両方を使用できます。

- .NET Common Language Runtime (CLR) 内でコードが実行されるため、すべての .NET ライブラリ (SQL ローハンドリングとダイレクトローハンドリングの両方を含む) にアクセスできます。

1.1.7 同期処理

同期とは、Mobile Link クライアントと中央データソースの間で行われるデータ交換処理です。この処理の間、クライアントは Mobile Link サーバとのセッションを確立して維持します。同期に成功した場合、セッションによってリモートデータベースと統合データベースは互いに一貫した状態に保たれます。

クライアントは同期処理を正常に開始します。この処理は、Mobile Link サーバとの接続を確立することから始まります。

アップロードとダウンロード

ローをアップロードするために、Mobile Link クライアントがアップロードを準備し送信します。このアップロードは、リモートデータベース上で前回の同期以後に更新、挿入、または削除されたすべてのローのリストを含みます。同様に、ローをダウンロードするために、挿入、更新、削除のリストを含むダウンロードを Mobile Link サーバが準備し送信します。

アップロード

デフォルトでは、Mobile Link クライアントは、前回成功した同期以後にリモートデータベースで挿入、更新、または削除されたローを自動的に追跡します。接続が確立すると、Mobile Link クライアントはこれらのすべての変更を記載したリストを Mobile Link サーバにアップロードします。

アップロードは、リモートデータベースで変更されたローに対する新旧のロー値のセットで構成されます (更新には新旧のロー値があります。削除には古い値のみ、挿入には新しい値のみがあります)。ローが更新されたり削除されたりしていれば、前回成功した同期直後に存在していた値が古い値になります。ローが挿入または更新されていれば、現在のローの値が新しい値です。現在の状態に至るまでローが複数回変更されていても、その途中の値は送信されません。

Mobile Link サーバは、アップロードを受信して、定義されたアップロードスクリプトを実行します。デフォルトでは、1 回のトランザクションですべての変更が適用されます。処理が完了すると、Mobile Link サーバはトランザクションをコミットします。

ダウンロード

Mobile Link サーバは、ユーザが作成した同期ロジックを使用して、Mobile Link クライアント側で挿入、更新、または削除されるローのリストを収集します。これらのローを Mobile Link クライアントにダウンロードします。このリストを収集するために、Mobile Link サーバは統合データベースで新しいトランザクションを開きます。

Mobile Link クライアントは、ダウンロードを受信します。Mobile Link クライアントは、ダウンロードの着信を、アップロードしたすべての変更内容が統合データベースで正常に適用されたことの確認とみなします。確認後、Mobile Link クライアントはこれらの変更内容が統合データベースに再送されないようにします。

次に、Mobile Link クライアントは、ダウンロードを自動的に処理して、古いローの削除、新しいローの挿入、変更されたローの更新を行います。これらの変更はすべて、リモートデータベース内の 1 つのトランザクションで適用されます。終了すると、トランザクションをコミットします。

同期中に情報が明確に交換されることはほとんどありません。クライアントは完全なアップロードを構築してアップロードします。これに応答して、Mobile Link サーバは完全なダウンロードを構築してダウンロードします。電話回線または公共無線ネットワークを使用している場合など、通信が低速で遅延時間が長い場合は、プロトコルの冗長性の制限が重要になります。

> **注記**
>
> Mobile Link は、統合データベースのデフォルトの独立性レベルとして ODBC 独立性レベル SQL_TXN_READ_COMMITTED を使用して動作します。統合データベースで使用される RDBMS がスナップショットアイソレーションをサポートし、スナップショットがデータベースに対して有効である場合、Mobile Link はデフォルトで、スナップショットアイソレーションをダウンロードに使用します。

このセクションの内容:

1.1.7.1 Mobile Link イベント

Mobile Link クライアントが同期を開始すると、複数の同期イベントが発生します。これらのイベントが発生すると、Mobile Link はそのイベントに対応するスクリプトを探します。このスクリプトには、実行する作業の詳細を示す指示が含まれています。

イベント用のスクリプトが定義され、Mobile Link システムテーブルに格納されている場合は、そのスクリプトが呼び出されます。

Mobile Link スクリプト

イベントに関連するスクリプトが作成されている場合、イベントの発生時に Mobile Link サーバがそのスクリプトを実行します。スクリプトが存在しなければ、次の順位のイベントが発生します。

> **注記**
>
> 同期モデル作成ウィザードを使用して Mobile Link アプリケーションを作成すると、必要な Mobile Link のスクリプトがすべて自動的に作成されます。ただし、デフォルトのスクリプトをカスタマイズしたり、新しいスクリプトを作成することもできます。

テーブルに対する一般的なアップロードスクリプトを以下に示します。最初のイベント upload_insert は、upload_insert スクリプトの実行をトリガします。このスクリプトにより、emp_id カラムと emp_name カラムのすべての変更が emp テーブルに挿入されます。upload_delete スクリプトと upload_update スクリプトは、emp テーブルでの削除と更新アクションに対して同様の機能を実行します。

イベント	スクリプトの内容例
upload_insert	`INSERT INTO emp (emp_id,emp_name) VALUES ({ml r.emp_id}, {ml r.emp_name})`
upload_delete	`DELETE FROM emp WHERE emp_id = {ml r.emp_id}`
upload_update	`UPDATE emp SET emp_name = {ml r.emp_name} WHERE emp_id = {ml r.emp_id}`

ダウンロードスクリプトはカーソルを使用します。次に、download_cursor スクリプトの例を示します。

```
SELECT order_id, cust_id
FROM ULOrder
WHERE last_modified >= {ml s.last_table_download}
AND emp_name = {ml r.emp_id}
```

SQL、Java、または .NET でスクリプトを作成可能

統合データベースのネイティブ SQL ダイアレクトを使用するか、Java または .NET の同期ロジックを使用して、スクリプトを記述できます。Java および .NET の同期ロジックを使用すると、Mobile Link サーバによって呼び出されるコードを記述して、データベースへの接続、変数の操作、アップロードされたローハンドリングの直接操作、またはダウンロードへのローハンドリングの追加が可能です。同期の要件に適したクラスとメソッドを持つ Java 用 Mobile Link サーバ API と .NET 用 Mobile Link サーバ API があります。

スクリプトの格納

SQL スクリプトは統合データベースの Mobile Link システムテーブルに格納されます。Mobile Link サーバ API を使用して記述されたスクリプトの場合は、完全に修飾されたメソッド名をスクリプトとして格納します。スクリプトを統合データベースに追加する方法は複数あります。

- 同期モデル作成ウィザードを使用する場合は、プロジェクトを展開するときにスクリプトが Mobile Link システムテーブルに格納されます。
- 統合データベースを設定するときにインストールされたストアドプロシージャを使用して、スクリプトを手動でシステムテーブルに追加できます。
- SQL Central を使用して、スクリプトをシステムテーブルに手動で追加できます。

関連情報

サーバ側の同期ロジックの作成オプション [15 ページ]

1.1.7.2 同期処理のトランザクション

Mobile Link サーバは、各 Mobile Link クライアントからアップロードされた変更を、1 回のトランザクションで統合データベースに組み込みます。Mobile Link サーバは、新しいローの挿入、古いローの削除、更新の実行、競合の解決が完了した後で、これらの変更をコミットします。

警告

SQL 同期スクリプト、または SQL 同期スクリプトから呼び出されるプロシージャやトリガで、暗黙的または明示的なコミットまたはロールバックを実行しないでください。SQL スクリプト内に COMMIT 文または ROLLBACK 文があると、同期手順のトランザクションの性質が変化してしまいます。これらの文を使用すると、Mobile Link では、障害が発生した場合にデータの整合性を保証できません。

ダウンロードした情報の追跡

Mobile Link では、リモートデータベースに格納されている最終ダウンロードタイムスタンプを使用して、ダウンロードの作成方法が簡素化されます。

ダウンロードトランザクションの主な役割は、統合データベースのローを選択することです。ダウンロードに失敗しても、リモートデータベースが同じ最終ダウンロードタイムスタンプを繰り返しアップロードするため、データが失われることはありません。

開始時と終了時のトランザクション

Mobile Link クライアントはダウンロードの情報を 1 回のトランザクションで処理します。ローを挿入、更新、削除して、リモートデータベースを統合データを持った最新の状態にします。

Mobile Link サーバは、他にトランザクションを 2 つ使用します。1 つは同期の開始時に、もう 1 つは同期の終了時に使用します。これらのトランザクションは、各同期とその処理時間に関する情報を記録します。したがって、試行された同期、成功した同期、同期にかかった時間についての統計を記録できます。データは処理のさまざまな時点でコミットされるので、これらのトランザクションによって、データを失敗した同期の分析に役立てられるようにコミットできます。

1.1.7.3 同期の障害処理の方法

Mobile Link 同期は、フォールトトレラントになっています。たとえば、同期中に通信リンクに障害が起きた場合は、リモートデータベースと統合データベースの両方が同じ状態のままになります。

クライアントでは、障害はリターンコードで示されます。

同期障害の処理方法は、発生したタイミングによって異なります。次に示すケースは、それぞれ異なる方法で処理されます。

アップロード中の障害

アップロードの構築中や適用中に障害が起きた場合は、リモートデータベースは同期の起動時とまったく同じ状態のままになります。サーバ側では、適用されたアップロードのすべての部分がロールバックされます。

アップロードとダウンロード間の障害

アップロードの完了後、Mobile Link クライアントがダウンロードを受信する前に障害が発生した場合、クライアントはアップロードした変更が統合データベースに適切に適用されたかどうかを確認できません。アップロードが完全に適用されコミットされているか、サーバがアップロード全体を適用する前に障害が起きています。Mobile Link サーバは、統合データベースにある不完全なトランザクションを自動的にロールバックします。

Mobile Link クライアントは、アップロードされたすべての変更を記録します。Mobile Link クライアントは、次に同期したときに前回のアップロードの状態を要求してから、新しいアップロードを構築します。前回のアップロードがコミットされていない場合は、新しいアップロードに前回のアップロードからの変更がすべて含まれます。

ダウンロード中の障害

ダウンロードの適用中にリモートデバイスで障害が起きた場合は、適用されたダウンロードはすべての部分がロールバックされ、リモートデータベースはダウンロード前と同じ状態のままになります。

非ブロッキングダウンロード確認を使用している場合、ダウンロードトランザクションはすでにコミットされていますが、nonblocking_download_ack スクリプトと publication_nonblocking_download_ack スクリプトは呼び出されません。

ダウンロード確認を使用していない場合、ダウンロード中に障害が発生しても、サーバ側には影響はありません。

> **注記**
>
> Mobile Link の再起動可能なダウンロード機能は、ダウンロードの失敗からのリカバリを支援する機能を備えています。この機能を使用すると、ダウンロード全体の再送を防ぐこともできます。この機能は、SQL Anywhere と Ultra Light リモートデータベースでそれぞれ別に実装されています。

障害が発生しても、データは失われません。Mobile Link サーバと Mobile Link クライアントが障害時のデータ管理を行います。開発者やユーザは、アプリケーション内のデータが一貫性を保持しているかどうか心配する必要はありません。

1.1.7.4 アップロードの処理方法

Mobile Link サーバが Mobile Link クライアントからアップロードを受信すると、同期が完了するまでアップロード全体が格納されます。

このような処理が行われるのは、次の理由によります。

ダウンロードローのフィルタ

ダウンロードするローを決定する方法で最も一般的なのは、前回のダウンロード以後に修正されたローをダウンロードすることです。同期中は、ダウンロードよりアップロードが優先されます。アップロード中に挿入または更新されたローが、前回のダウンロード以後に修正されたローになります。

アップロードの一部として送られたダウンロードローから除外する download_cursor スクリプトを記述するのは困難です。このため、Mobile Link サーバがダウンロードからこのようなローを自動的に取り除きます。

挿入と更新の処理

デフォルトでは、アップロード内のテーブルは、参照整合性に違反しない順序で統合データベースに適用されます。アップロード内のテーブルは、外部キー関係に基づいて並べられます。たとえば、テーブル A とテーブル C の両方がテーブル B のプライマリキーカラムを参照する外部キーを持っている場合は、テーブル B のローの挿入や更新が先にアップロードされます。

挿入と更新後の削除の処理

削除は、すべての挿入と更新が適用された後で統合データベースに適用されます。削除が適用されると、テーブルはアップロードの処理とは逆の順序で処理されます。削除されるローが、削除される別のテーブルのローを参照している場合、この操作の順序では、参照元ローが参照先ローより先に削除されることになります。

デッドロック

アップロードを統合データベースに適用すると、他のトランザクションとの同時実行性が原因でデッドロックが発生することがあります。そのトランザクションは、他の Mobile Link サーバのデータベース接続からのアップロードトランザクションの場合や、統合データベースを使用している他のアプリケーションからのトランザクションの場合があります。アップロードトランザクションがデッドロックされている場合は、そのトランザクションはロールバックされ、Mobile Link サーバが自動的にアップロードをもう一度最初から適用し始めます。

> **注記**
>
> パフォーマンスに関するヒント
>
> 競合をできるだけ避けるように同期スクリプトを書くことが重要です。複数のユーザが同時に同期しているときに競合が起きると、パフォーマンスに大きな影響があります。

1.1.7.5 参照整合性と同期

Ultra Light Java Edition を除くすべての Mobile Link クライアントは、ダウンロードをリモートデータベースに組み込むときに参照整合性を確保します。

参照整合性に違反するローがあった場合、デフォルトでは、Mobile Link クライアントはダウンロードトランザクションを失敗させずに、参照整合性に違反するすべてのローを自動的に削除します。

この機能には次のような利点があります。

- 同期スクリプトの間違いから保護します。スクリプトに柔軟性があると、リモートデータベースの整合性をそこなうローを誤ってダウンロードしてしまうことがあります。Mobile Link クライアントは、介入を要求せずに参照整合性を自動的に管理します。
- この参照整合性のメカニズムを使用して、リモートデータベースから情報を効率的に削除できます。親レコードに削除データを送信するだけで、Mobile Link クライアントはすべての子レコードを自動的に削除します。これにより、Mobile Link がリモートデータベースに送信するトラフィックの量を大幅に減らすことができます。

Mobile Link クライアントは、参照整合性を維持するためにローを明示的に削除する必要がある場合、次のような通知を行います。

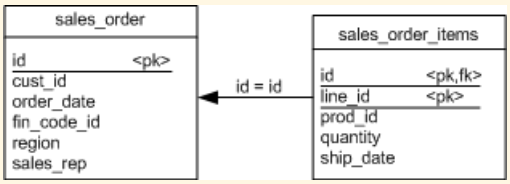
- SQL Anywhere クライアントの場合は、dbmlsync によってログにエントリが書き込まれます。dbmlsync イベントフックも使用できます。
- Ultra Light クライアントの場合は、SQLE_ROW_DELETED_TO_MAINTAIN_REFERENTIAL_INTEGRITY 警告が発生します。この警告には、テーブル名のパラメータが含まれます。参照整合性を維持するため、削除されるすべてのローで警告が発生します。同期をそのまま進める場合は、警告を無視してかまいません。警告を明示的に処理する場合は、エラーコールバック関数を使用して警告をトラップすることで、たとえば、削除されたローの数を取得することができます。警告が発生したときに同期を失敗させるには、同期 observer を実装し、observer に (グローバル変数などを使用して) エラーコールバック関数から信号を送信する必要があります。この場合、同期は observer への次回の呼び出しで失敗します。

トランザクション終了時にチェックされる参照整合性

Mobile Link クライアントは、1 つのトランザクション内のダウンロードから変更を組み込みます。柔軟性をより向上させるために、参照整合性のチェックはこのトランザクションの終了時に行われます。チェックが遅いため、参照整合性に違反する状態を一時的にパスすることがあります。しかし、参照整合性に違反するローは、ダウンロードがコミットされる前に自動的に削除されます。

例

Ultra Light 販売アプリケーションに、次の 2 つのテーブルが含まれているとします。1 つのテーブルには、販売注文が含まれています。もう 1 つのテーブルには、各注文で販売された商品情報が含まれています。この 2 つのテーブルは、次のような関係です。



1 つの注文を削除するために sales_order テーブルに download_delete_cursor を使用すると、削除された受注を示す sales_order_items テーブルのすべてのローが、デフォルトの参照整合性メカニズムによって自動的に削除されます。

この方法には、次のような利点があります。

- sales_order_items テーブルからのローは自動的に削除されるので、sales_order_items テーブルスクリプトは必要ありません。
- 同期の効率が向上します。sales_order_items テーブルから削除するローをダウンロードする必要がありません。各販売注文に項目がたくさんある場合は、ダウンロードが小さくなるのでパフォーマンスが向上します。この方法は、速度の遅い通信方法を使用しているときに特に役立ちます。

デフォルトの動作の変更

SQL Anywhere クライアントの場合は、sp_hook_dbmlsync_download_ri_violation クライアントイベントフックを使用して、参照整合性違反を処理できます。Dbmlsync も、ログにエントリを書き込みます。

1.1.8 Mobile Link セキュリティに関する考慮事項

Mobile Link のインストール環境のように、広範囲の分散システム全体のデータの安全を確保するには、いくつかの要素があります。

統合データベースのデータ保護

統合データベース内のデータは、データベースのユーザ認証システムとその他のセキュリティ機能で保護できます。

詳細については、使用しているデータベースのマニュアルを参照してください。

リモートデータベースのデータ保護

SQL Anywhere リモートデータベースを使用している場合、SQL Anywhere のセキュリティ機能を使用してデータを保護できます。このセキュリティ機能は、デフォルトではクライアント/サーバ通信での不正なアクセスを防止するように設計されていますが、完全には保護されず、データベースファイルから直接情報を抽出するという悪質な攻撃を回避しないこともあります。

クライアント上のファイルは、クライアントのオペレーティングシステムのセキュリティ機能によって保護されます。

同期中のデータ保護

Mobile Link クライアントから Mobile Link サーバへの通信は、Mobile Link トランスポートレイヤセキュリティ機能で保護できます。Mobile Link サーバから統合データベースへの通信は、データベース固有の方法で保護できます。

権限のないユーザからの同期システムの保護

Mobile Link 同期は、パスワードによるユーザ認証システムで保護できます。このメカニズムにより、権限のないユーザはデータを同期できなくなります。

不要な権限を持つユーザからのデータの保護

ロールと権限の付与および取り消しをすることで、ユーザが同期を実行するための適切な権限を持ち、必要以上の権限を持たないようにすることができます。

1.2 SQL Central の Mobile Link プラグイン

SQL Central の Mobile Link プラグインはバージョン 12 で再設計されています。以前のバージョンでは、プラグインに 2 つのモードがありました。モデルモードと管理モードです。Mobile Link の機能は 2 つのモードに分割されていたため、どちらのモードを使用しているのかを常に認識する必要がありました。

バージョン 12 以降では、これらのモードはなくなりました。

Mobile Link プラグインのフォルダウィンドウ枠から使用できる最上位レベルの機能は、次のとおりです。

- Mobile Link プロジェクトの操作。
- 統合データベースの操作。
- Mobile Link サーバのコマンドラインの操作。
- リモートスキーマ名の操作。
- グループの操作。
- リモートタスクの操作。
- 同期モデルの操作。

リモートスキーマ名、グループ、リモートタスクはすべて、リモートデータベース機能の集中管理の一部です。

SQL Central で Mobile Link を操作するには、まず Mobile Link プロジェクトを定義する必要があります。

Mobile Link プロジェクトは、モバイルアプリケーションに関連する同期モデル、統合データベース、リモートタスクで構成されるフレームワークです。

Mobile Link プロジェクトは、次の要素から成る名前付きコレクションです。

- 同期モデルのリスト
- 設計済みで展開されていないリモートタスクのリスト
- 統合データベースへの接続のリスト
- Mobile Link ユーザのユーザ定義グループのリスト

このセクションの内容:

Mobile Link プロジェクトの作成 [25 ページ]
SQL Central の Mobile Link を操作する前に、Mobile Link プロジェクトを作成します。

統合データベースの追加 [27 ページ]
SQL Central で、Mobile Link プロジェクトに 1 つまたは複数の統合データベースを追加します。

同期モデル [33 ページ]
同期モデルは、Mobile Link アプリケーションを簡単に作成できるツールです。同期モデルは、SQL Central のプロジェクト作成ウィザードまたは同期モデル作成ウィザードによって作成されるファイルです。

関連情報

Mobile Link サーバのコマンドラインを作成 [30 ページ]

1.2.1 Mobile Link プロジェクトの作成

SQL Central の Mobile Link を操作する前に、Mobile Link プロジェクトを作成します。

コンテキスト

サンプルの Mobile Link プロジェクトは *%SQLANYSAMP17%*¥MobiLink¥CustDB¥project.mlp にあります。

手順

1. SQL Central で、ツール > *Mobile Link17* > 新しいプロジェクト をクリックします。
2. 新しいプロジェクトの名前を指定してください。フィールドに、プロジェクトの名前を入力します。

3. 新しいプロジェクトの保存場所を指定してください。フィールドにプロジェクトフォルダのロケーションを入力するか、参照をクリックしてプロジェクトファイルのフォルダを選択します。
4. 次へをクリックします。
5. 統合データベースを指定ページで、次のタスクを実行します。
   a. データベースの表示名フィールドに、統合データベースに使用する表示名を入力します。この名前は、プロジェクトの統合データベースリストに表示されます。
   b. 接続文字列フィールドに、統合データベースへの接続に使用するデータベース接続パラメータを入力するか、編集をクリックして ODBC データソースに接続するためのウィンドウを開きます。
   c. パスワードを記憶を選択し、データベースへの接続に使用するパスワードを保存します。

      このオプションを選択すると、パスワードは難読化した形式でプロジェクトファイルに保存されます。
   d. 次へをクリックします。
6. 新しいリモートデータベーススキーマページで、リモートデータベースに表示する統合テーブルとカラムを選択します。次へをクリックします。
7. リモートスキーマ名をプロジェクトに追加を選択し、スキーマが同じであるリモートデータベースのグループを識別します。または後から追加することもできます。リモートスキーマ名は、リモートタスクを作成する場合にのみ役に立ちます。

   リモートスキーマ名を追加する場合は、次のように指定します。

   新しいリモートスキーマ名を指定してください。

   同じスキーマ名を共有するリモートデータベースのグループを特定するために使用する名前を入力します。バージョン番号を含めることをお奨めします。

   次へをクリックします。
8. リモートデータベースタイプを指定ページで、使用するリモートデータベースのタイプとして、*SQL Anywhere* または *Ultra Light* のいずれかを選択します。この設定は、後からプロジェクトのプロパティページで変更できます。
9. 完了をクリックして新しいプロジェクトを保存します。
10. Mobile Link システム設定のインストールを求められたら、はいをクリックしてから *OK* をクリックします。

結果

Mobile Link プロジェクトが作成され、指定したデータベースへの接続が確立され、同期モデルが作成されます。

次のステップ

これで、同期モデルを編集または展開したり、Mobile Link プロジェクトの他のオブジェクト (リモートタスクなど) を操作したりできます。

1.2.2 統合データベースの追加

SQL Central で、Mobile Link プロジェクトに 1 つまたは複数の統合データベースを追加します。

前提条件

Mobile Link プロジェクトは、定義されている必要があります。

コンテキスト

リモートタスクを展開するためには、少なくとも 1 つの統合データベースを割り当てておく必要があります。

手順

1. Mobile Link プロジェクトを選択します。
2. プロジェクト名をダブルクリックし、ファイル > 新規 > 統合データベース をクリックします。
3. 必要なデータベース接続パラメータを入力して、次へをクリックします。
4. 表示名フィールドに、プロジェクトでこのデータベースに使用する名前を入力します。デフォルトの表示名は ODBC データソース名です。データベースの説明を指定するには、説明フィールドに入力します。
5. パスワードを記憶を選択し、データベースへの接続に使用するパスワードを保存します。

   このオプションを選択すると、パスワードは難読化した形式でプロジェクトファイルに保存されます。
6. 完了をクリックして、統合データベースをプロジェクトに追加します。
7. Mobile Link システム設定のインストールを求められたら、はいをクリックしてから *OK* をクリックします。

結果

統合データベースが、Mobile Link プロジェクトに追加されます。

このセクションの内容:

Mobile Link システム設定 [28 ページ]
テーブル、カラム、トリガなどの同期に必要なオブジェクトを追加してから、データベースを Mobile Link 統合データベースとして使用する必要があります。これらのオブジェクトの追加は、データベースに対して設定スクリプトを実行することによって行われます。

Mobile Link サーバのコマンドラインを作成 [30 ページ]
SQL Central の *Mobile Link* サーバのコマンドラインのプロパティウィンドウを使用して、Mobile Link プロジェクトに取り込む Mobile Link サーバのコマンドラインを保存します。

LDAP サーバの追加 [31 ページ]
SQL Central を使用して、信頼できる LDAP サーバを追加します。Mobile Link サーバでは LDAP サーバを使用してユーザを認証します。

ユーザ認証ポリシーの追加 [32 ページ]
SQL Central でユーザ認証ポリシーを作成し、LDAP サーバに対して Mobile Link ユーザを認証します。

複数のユーザへのユーザ認証ポリシーの割当 [32 ページ]
SQL Central で、複数のユーザに対してユーザ認証ポリシーを割り当てます。

1.2.2.1 Mobile Link システム設定

テーブル、カラム、トリガなどの同期に必要なオブジェクトを追加してから、データベースを Mobile Link 統合データベースとして使用する必要があります。これらのオブジェクトの追加は、データベースに対して設定スクリプトを実行することによって行われます。

サポートされている各 RDBMS 用に個別の設定スクリプトがあります。これらのスクリプトは、すべて *%SQLANY17%* `¥MobiLink¥setup` フォルダにあります。スクリプトによる処理の内容を詳細に確認するには、テキストエディタでスクリプトを開きます。

統合データベースを Mobile Link プロジェクトに追加すると、Mobile Link システム設定がチェックされます。Mobile Link システム設定が存在しない場合は、インストールを求めるプロンプトが表示されます。古いバージョンが検出された場合にも、アップグレードを求めるプロンプトが表示されます。また、同期モデルを展開する場合 (まだ行っていない場合) にも、Mobile Link システム設定のインストールを求めるプロンプトが表示されます。

このセクションの内容:

Mobile Link システム設定のチェック [29 ページ]
SQL Central を使用して、同期に必要なオブジェクトをチェックします。

Mobile Link システム設定の削除 [29 ページ]
SQL Central を使用して、Mobile Link システム設定によってインストールされている、同期に必要なオブジェクトを削除します。

関連情報

同期モデルの展開 [54 ページ]

1.2.2.1.1 Mobile Link システム設定のチェック

SQL Central を使用して、同期に必要なオブジェクトをチェックします。

前提条件

Mobile Link プロジェクトが定義されており、そのプロジェクトに少なくとも 1 つの統合データベースが存在する必要があります。

手順

1. Mobile Link プロジェクトをダブルクリックします。
2. 統合データベースをダブルクリックし、チェックする統合データベースを選択し、*Mobile Link* システム設定のチェックをクリックします。
3. 設定がインストール済みでないか、または最新でない場合は、はいをクリックしてインストールまたは更新し、次に、*OK* をクリックします。

結果

システム設定は、指定されたとおり、インストールまたは更新されます。

1.2.2.1.2 Mobile Link システム設定の削除

SQL Central を使用して、Mobile Link システム設定によってインストールされている、同期に必要なオブジェクトを削除します。

前提条件

Mobile Link プロジェクトが定義されており、その統合データベースに Mobile Link システム設定がインストールされている必要があります。

手順

1. Mobile Link オブジェクトをダブルクリックします。
2. 統合データベースをダブルクリックし、チェックする統合データベースを選択し、*Mobile Link* システム設定を削除をクリックします。
3. すべての同期モデルがデータベースから削除され、同期状態が失われ、データベースがプロジェクトから削除されることを示す警告が表示されます。はいをクリックします。

結果

Mobile Link システム設定がデータベースから削除され、データベースが Mobile Link プロジェクトから削除されます。

1.2.2.2 Mobile Link サーバのコマンドラインを作成

SQL Central の *Mobile Link* サーバのコマンドラインのプロパティウィンドウを使用して、Mobile Link プロジェクトに取り込む Mobile Link サーバのコマンドラインを保存します。

前提条件

Mobile Link プロジェクトは、定義されている必要があります。

手順

1. Mobile Link プロジェクト名をダブルクリックします。
2. *Mobile Link* サーバのコマンドラインをダブルクリックします。
3. ファイル ▶ 新規 ▶ *Mobile Link* サーバのコマンドライン をクリックします。
4. コマンドラインに対してわかりやすい名前を入力します。
5. 統合データベースドロップダウンリストから統合データベースを選択します。
6. 追加をクリックして、同期要求を Mobile Link サーバで受信する方法を決定するネットワークオプションを指定します。オプションが指定されていない場合、Mobile Link サーバはポート 2439 で TCP/IP 接続を受信します。
7. 冗長性ドロップダウンリストから冗長性レベルを選択します。カスタムオプションを選択するとウィンドウが開き、使用できる冗長性オプションの中から必要なものを選択できるようになります。
8. その他の必要なオプションは、詳細ページで設定します。必要なオプションがリストにない場合は、それをその他のオプションフィールドに入力できます。
9. コマンドラインフィールドに表示されたコマンドラインを確認します。必要に応じて上記オプションを編集するか、または *OK* をクリックしてコマンドラインを保存します。

結果

指定したオプションと一緒にコマンドラインが保存されます。

1.2.2.3 LDAP サーバの追加

SQL Central を使用して、信頼できる LDAP サーバを追加します。Mobile Link サーバでは LDAP サーバを使用してユーザを認証します。

前提条件

Mobile Link プロジェクトが定義されており、そのプロジェクトに少なくとも 1 つの統合データベースが存在する必要があります。

手順

1. Mobile Link プロジェクト名をダブルクリックしてから、使用する統合データベースをダブルクリックします。
2. *LDAP* サーバをダブルクリックします。
3. ファイル > 新規 > *LDAP* サーバ をクリックします。
4. *LDAP* サーバ作成ウィザードの手順に従います。
5. 終了をクリックします。

結果

LDAP サーバが作成され、*LDAP* サーバタブに表示されます。

次のステップ

ユーザ認証ポリシーを設定することで、LDAP サーバによる Mobile Link ユーザの認証の方法を定義できます。

関連情報

ユーザ認証ポリシーの追加 [32 ページ]

1.2.2.4 ユーザ認証ポリシーの追加

SQL Central でユーザ認証ポリシーを作成し、LDAP サーバに対して Mobile Link ユーザを認証します。

前提条件

統合データベースに少なくとも 1 つの LDAP サーバが定義されている必要があります。

手順

1. Mobile Link プロジェクト名をダブルクリックしてから、使用する統合データベースをダブルクリックします。
2. ユーザ認証ポリシーをダブルクリックします。
3. ファイル > 新規 > ユーザ認証ポリシー をクリックします。
4. ユーザ認証ポリシー作成ウィザードの手順に従います。
5. 終了をクリックします。

結果

ユーザ認証ポリシーが作成され、ユーザ認証ポリシータブに表示されます。

次のステップ

新しい Mobile Link ユーザを作成するか、または認証ポリシーを使用するように既存の Mobile Link ユーザを変更することができます。

1.2.2.5 複数のユーザへのユーザ認証ポリシーの割当

SQL Central で、複数のユーザに対してユーザ認証ポリシーを割り当てます。

前提条件

統合データベースに少なくとも 1 つの LDAP サーバと 1 人以上のユーザが定義されている必要があります。

手順

1. Mobile Link プロジェクト名をダブルクリックしてから、使用する統合データベースをダブルクリックします。
2. ユーザを選択します。
3. 右側の詳細ウィンドウ枠で、認証ポリシーを割り当てるユーザを選択します。
4. 右クリックし、認証ポリシーの設定を選択します。現在定義されている認証ポリシーのリストから認証ポリシーを選択します。

結果

選択したユーザにユーザ認証ポリシーが割り当てられます。

次のステップ

新しい Mobile Link ユーザを作成するか、または認証ポリシーを使用するように既存の Mobile Link ユーザを変更することができます。

1.2.3 同期モデル

同期モデルは、Mobile Link アプリケーションを簡単に作成できるツールです。同期モデルは、SQL Central のプロジェクト作成ウィザードまたは同期モデル作成ウィザードによって作成されるファイルです。

同期モデルが作成されたら、引き続きモデルのカスタマイズを行うことができます。統合データベースまたはリモートデータベースは、モデルを展開するまで変更されません。モデルは拡張子 `.mlsm` のモデルファイルに保存され、このファイルの参照は Mobile Link プロジェクトファイルに保存されます。

モデルが完成したら、同期モデル展開ウィザードを使用してモデルを展開します。同期モデル展開ウィザードでは、選択した展開オプションを使用して、Mobile Link サーバとクライアントを実行するスクリプトファイルを作成できます。展開時に既存のデータベースを変更するか、ウィザードを使用して後で実行する SQL スクリプトを作成するかを選択できます。リモートデータベース用に作成されたファイルは、リモートタスクで使用できます。

展開後に、同期モデルまたはデータベースを引き続きカスタマイズし、再展開することができます。必要に応じて、Mobile Link のマニュアル全体に記載されている技術を使用して、展開した同期システムを SQL Central 外で変更することもできます。

このセクションの内容:

同期モデル作成ウィザードで同期モデルを作成したら、それを使用していくつかのタスクを実行できます。変更内容は、同期モデルファイルにのみ保存されます。同期モデルを展開するまで、変更内容は統合データベースまたはリモートデータベースには保存されません。

スキーマの更新 [52 ページ]
スキーマ更新ウィザードを使用して、同期モデル内の統合データベースとリモートデータベースのスキーマを更新します。

同期モデルの展開 [54 ページ]
同期モデルは、同期モデル展開ウィザードを使用して展開します。

同期モデルの制限事項 [63 ページ]
同期モデルにはいくつかの制限事項があります。

1.2.3.1 同期モデル作成ウィザードを使用した Mobile Link アプリケーションの設定

SQL Central の同期モデル作成ウィザードを使用して、Mobile Link アプリケーションの同期ロジックを設定します。

前提条件

Mobile Link プロジェクトは、定義されている必要があります。

手順

1. Mobile Link プロジェクト名をダブルクリックし、同期モデルをダブルクリックします。
2. ファイル > 新規 > 同期モデル をクリックし、同期モデル作成ウィザードを起動します。
3. ようこそページで、同期モデルの名前を選択します。モデルは、プロジェクトディレクトリに `.mlsm` ファイルとして保存されます。次へをクリックします。
4. プライマリキー要件ページで、システムがプライマリキー要件を満たしていることを確認し、3 つのチェックボックスをオンにし、次へをクリックします。
5. 統合データベーススキーマページで、統合データベーススキーマを取得するための統合データベースを選択し、次へをクリックします。

   Oracle データベースを選択した場合、全所有者のスキーマをロードするには時間がかかるため、所有者のサブセットの選択を求めるプロンプトが表示されることがあります。
6. リモートデータベーススキーマは、統合データベーススキーマまたは既存のリモートデータベースに基づいて作成できます。既存のリモートデータベースには、SQL Anywhere または Ultra Light を使用できます。
7. 同期モデル作成ウィザードの残りの指示に従います。可能な場合はベストプラクティスに基づいたデフォルトの推奨事項が使用されます。
8. 完了をクリックします。

結果

完了をクリックすると、SQL Central の右ウィンドウ枠に同期モデルが表示され、モデルの表示、編集、または展開が行えるようになります。

次のステップ

同期モデルを作成したら、次のいずれかを実行します。

- テストウィンドウを使用して同期モデルをテストします。
- マッピング、イベント、認証情報を編集します。
- 同期モデル展開ウィザードを使用して、完了したモデルを展開します。

このセクションの内容:

関連情報

同期モデルタスク [37 ページ]
同期モデルの展開 [54 ページ]
テーブルとカラムのマッピング [38 ページ]

1.2.3.1.1 同期モデルの削除

SQL Central で Mobile Link プラグインを使用して同期モデルを追加してある場合は、Mobile Link プラグインを使用して統合データベースから同期モデルを削除することができます。

前提条件

Mobile Link プラグインを使用して同期モデルが追加されている必要があります。

コンテキスト

この機能は、DB2 データベースでは使用できません。

手順

1. 削除する同期モデルが含まれているプロジェクトをダブルクリックします。
2. 同期モデルをダブルクリックし、削除する同期モデルを右クリックし、統合データベースから削除を選択します。
3. 該当する同期モデルが複数の統合データベースにある場合は、その中でも同期モデルを削除する統合データベースを選択します。
4. *OK* をクリックします。

結果

選択した統合データベースから同期モデルが削除されます。

1.2.3.1.2 リモートスキーマ

同期モデルには、リモートデータベースのスキーマが含まれています。このスキーマは、既存のリモートデータベースまたは統合データベースから取得できます。

既存のリモートデータベースは、次のような場合に使用します。

- すでにリモートデータベースがある場合、特にスキーマが統合データベーススキーマのサブセットではない場合。
- 統合カラムとリモートカラムのタイプが異なっている必要がある場合。
- リモートテーブルと統合データベースのテーブルの所有者が異なっている必要がある場合。統合データベースから作成された新しい SQL Anywhere リモートスキーマでは、リモートテーブルの所有者は、統合データベース内の対応するテーブルの所有者と同じになります。別の所有者にするには、設定するテーブル所有者により所有される既存の SQL Anywhere リモートデータベースを使用します。

> **注記**
>
> 既存のデータベーススキーマを手動で変更し、スキーマ更新ウィザードを実行すると、Mobile Link プロジェクトで同期モデルを更新できます。

モデルの展開時は、モデルでのリモートスキーマの作成方法にかかわらず、リモートデータベースは 3 つのオプションから選択して作成できます。展開時のリモートデータベースのオプションは次のとおりです。

新しいリモートデータベースを作成します。

展開時に、同期モデルのスキーマが使用され、新しいリモートデータベースが作成されます。データベースはデフォルトオプションで作成されます。

ユーザテーブルがない既存のリモートデータベースを更新します。

展開中に、同期するユーザテーブルを作成するか、または再作成するかを選択できます。このオプションは、特定の照合など、デフォルトでないデータベース作成オプションを使用する場合に便利です。

SQL Anywhere データベースの場合、一部のオプションはデータベース作成後には設定できません。

Ultra Light データベースの場合は、データベース作成後にデータベースのプロパティを変更することはできません。

モデルと同じスキーマを持つ既存のリモートデータベースを更新します。

このオプションは、同期する既存のリモートデータベースがある場合に便利です。既存のリモートデータベースに直接展開する場合は、同期するテーブルが作成または再作成されないように選択できます。既存のテーブルとその内容は変更されません。

SQL Anywhere リモートデータベースでは、テーブルと元のデータベースの所有者は同じです。Ultra Light データベーステーブルには、所有者はありません。

関連情報

同期モデルの展開 [54 ページ]
スキーマの更新 [52 ページ]

1.2.3.2 同期モデルタスク

同期モデル作成ウィザードで同期モデルを作成したら、それを使用していくつかのタスクを実行できます。変更内容は、同期モデルファイルにのみ保存されます。同期モデルを展開するまで、変更内容は統合データベースまたはリモートデータベースには保存されません。

SQL Central 外で同期モデルを変更することはできますが、変更内容をリバースエンジニアリングでモデルに戻すことはできません。たとえば、システムプロシージャを使用して Mobile Link スクリプトの追加や変更を行うことができます。

このセクションの内容:

テーブルとカラムのマッピング [38 ページ]
テーブルマッピングは、どのテーブルを同期するか、テーブルをどのように同期するか、および統合データベースとリモートデータベースとの間で同期データをどのようにマッピングするかを指定します。

ダウンロード方式の変更 [43 ページ]
詳細ウィンドウ枠のダウンロード方式タブでダウンロードタイプを変更します。テーブルマッピングのダウンロードタイプには、タイムスタンプ、スナップショット、カスタムのいずれかを設定できます。

削除の記録方法の変更 [45 ページ]
統合データベースでの削除をリモートにダウンロードするかどうか、およびこのような削除に関する情報を統合データベースにどのように保存するかを制御する場合には、削除のダウンロードタブを使用します。

サブセットのダウンロード [46 ページ]
各 Mobile Link リモートデータベースでは、統合データベースに格納されているデータのサブセットを受信できます。テーブルごとにダウンロードサブセットをカスタマイズできます。

ダウンロードサブセットの変更 [46 ページ]

テーブルごとにダウンロードサブセットをカスタマイズします。各 Mobile Link リモートデータベースでは、統合データベースに格納されているデータのサブセットを同期できます。

ダウンロード削除サブセットの変更 [49 ページ]
ダウンロードサブセットを使用して統合データベースでデータのサブセットを同期している場合は、デフォルトによりダウンロード削除サブセットがダウンロード済みに設定され、ダウンロードサブセットとまったく同じになります。この設定は、すべてまたはカスタムに変更できます。

競合の検出と解決 [49 ページ]
リモートデータベースと統合データベースの両方でローが更新された場合は、次にデータベースを同期するときに競合が発生します。

同期モデル内のスクリプトの変更 [51 ページ]
同期モデル内のスクリプトは、イベントタブを使用して変更します。

1.2.3.2.1 テーブルとカラムのマッピング

テーブルマッピングは、どのテーブルを同期するか、テーブルをどのように同期するか、および統合データベースとリモートデータベースとの間で同期データをどのようにマッピングするかを指定します。

アップロード専用、ダウンロード専用、非同期テーブルまたはカラム

デフォルトでは、Mobile Link は完全に双方向の同期を行います。各テーブルは、アップロード専用またはダウンロード専用に変更できます。また、テーブルを同期しないことを選択することもでき、この場合は、テーブルマッピングは削除されます。

同期モデルでは、テーブルをダウンロード専用として指定することのみができ、ダウンロード専用パブリケーションを作成することはできません。

このセクションの内容:

テーブルマッピングの方向の変更 [39 ページ]
SQL Central のマッピングタブからテーブルマッピングの方向を変更して、テーブルを同期する方法を示すか、またはテーブルを同期しないことを示します。

テーブルマッピングの削除 [40 ページ]
テーブルマッピングを削除すると、テーブルは同期しなくなります。SQL Central のマッピングタブから、テーブルマッピングを削除できます。

テーブルマッピングの変更 [40 ページ]
テーブルとカラムのマッピングを確認しカスタマイズします。既存のリモートデータベースに基づいてモデルを作成すると、テーブルとカラムのマッピングは推測に基づいて設定されます。

マッピングされていないテーブルをマッピングに追加 [41 ページ]
統合データベース内のテーブルのうち、同期できるようにマッピングされていないテーブルを同期モデルに追加します。

テーブルマッピングのカラムマッピングの変更 [42 ページ]
必要に応じてテーブルとカラムのマッピングを確認しカスタマイズします。既存のリモートデータベースに基づいてモデルを作成すると、テーブルとカラムのマッピングは推測に基づいて設定されます。

1.2.3.2.1.1 テーブルマッピングの方向の変更

SQL Central のマッピングタブからテーブルマッピングの方向を変更して、テーブルを同期する方法を示すか、またはテーブルを同期しないことを示します。

前提条件

同期モデルが必要です。

手順

1. Mobile Link プロジェクトをダブルクリックします。
2. 同期モデルをダブルクリックし、同期モデル名をダブルクリックします。
3. マッピングタブをクリックします。
4. テーブルマッピングウィンドウ枠で、統合テーブルを選択します。
5. 方向ドロップダウンリストから、次のいずれかを選択します。

   同期しない

   このオプションを選択することは、テーブルマッピングを削除することと同じです。

   双方向

   データベース操作は、リモートデータベースから統合データベースの方向へ、および統合データベースからリモートデータベースの方向へ同期されます。

   リモートにのみダウンロード

   変更は統合データベースからリモートデータベースの方向へのみ同期されます。

   統合にのみアップロード

   変更はリモートデータベースから統合データベースの方向へのみ同期されます。

結果

テーブルマッピングが更新されます。

1.2.3.2.1.2 テーブルマッピングの削除

テーブルマッピングを削除すると、テーブルは同期しなくなります。SQL Central のマッピングタブから、テーブルマッピングを削除できます。

前提条件

同期モデルが必要です。

手順

1. Mobile Link プロジェクト名をダブルクリックします。
2. 同期モデルをダブルクリックし、同期モデル名をダブルクリックします。
3. マッピングタブをクリックします。
4. テーブルマッピングウィンドウ枠で、テーブルマッピングを選択します。
5. 方向ドロップダウンリストから、同期しないをクリックします。

結果

マッピングは、次に同期モデルを保存するときに削除されます。

1.2.3.2.1.3 テーブルマッピングの変更

テーブルとカラムのマッピングを確認しカスタマイズします。既存のリモートデータベースに基づいてモデルを作成すると、テーブルとカラムのマッピングは推測に基づいて設定されます。

前提条件

同期モデルが必要です。

手順

1. Mobile Link プロジェクト名をダブルクリックします。
2. 同期モデルをダブルクリックし、同期モデル名をダブルクリックします。
3. マッピングタブをクリックします。
4. テーブルマッピングウィンドウ枠で、変更するマッピングのリモートテーブルをクリックします。
5. リモートテーブル名の横にある省略記号 (ピリオド 3 つ) ボタンをクリックし、同期されていないリモートテーブルのリストから別のテーブルを選択します。
   - 統合テーブルにまだマッピングされていないリモートテーブルのみを選択できます。
   - リモートテーブルにテーブルを追加するには、スキーマの更新に関するトピックを参照してください。

結果

テーブルマッピングが更新されます。

関連情報

スキーマの更新 [52 ページ]

1.2.3.2.1.4 マッピングされていないテーブルをマッピングに追加

統合データベース内のテーブルのうち、同期できるようにマッピングされていないテーブルを同期モデルに追加します。

前提条件

同期モデルが必要です。

手順

1. Mobile Link プロジェクト名をダブルクリックします。
2. 同期モデルをダブルクリックし、同期モデル名をダブルクリックします。
3. マッピングタブをクリックします。
4. ファイル > 新規 > テーブルマッピング をクリックして、新しいテーブルマッピングの作成ウィンドウを開きます。このウィンドウでは、追加する統合テーブルを選択します。

5. 1つまたは複数の統合テーブルを選択し、リモートスキーマに追加します。新しいリモートテーブルには、対応する統合テーブルと同じ名前で、同じカラムのセットができます。また、リモートテーブルと統合テーブル間のマッピングが自動的に作成されます。

   次のオプションを使用できます。

   選択したテーブルが存在しない場合はリモートスキーマに追加

   リモートスキーマに変更を加えない場合は、このオプションを無効にします。
   新しいリモートテーブルのカラムを選択する

   リモートスキーマにテーブルを追加するが、すべてのカラムを追加するわけではない場合に、このオプションを選択します。このオプションを有効にすると、選択したテーブルごとにカラムを選択できます。
   同期モデルのシャドウテーブルのような名前のテーブルを非表示にする (そのようなテーブルは同期してはならないため)

   デフォルトで、同期モデルシャドウテーブル名のような名前の統合データベーステーブルは表示されません。これは、そのようなシャドウテーブルは同期してはならないためです。

結果

同期モデルのマッピングに新しいテーブルが追加されます。

1.2.3.2.1.5 テーブルマッピングのカラムマッピングの変更

必要に応じてテーブルとカラムのマッピングを確認しカスタマイズします。既存のリモートデータベースに基づいてモデルを作成すると、テーブルとカラムのマッピングは推測に基づいて設定されます。

前提条件

同期モデルが必要です。

コンテキスト

同期された統合テーブルのカラムをリモートテーブルカラムにマッピングできます。カラムを同期するか同期から除外すると、値が決まります。値にマッピングするとき、Mobile Link ユーザ名、リモートデータベース ID、または SQL 式 (Mobile Link 名前付きパラメータを含むことができる) を使用できます。プライマリキーカラムを値にマッピングし、テーブルマッピングが双方向の場合は、リモートデータベースにダウンロードするときにプライマリキーが重複しないようにしてください。

手順

テーブルマッピングタブで、変更するカラムマッピングを右クリックし、サブセットのダウンロードコンテキストメニューから次のいずれかのオプションを選択します。

- なし
- ユーザ
- リモート
- カスタム
- マッピングされていないリモートカラム

統合カラムをリモートカラムと統合するには、メニューの一番下のグループからマッピングされいないリモートカラムを選択します。マッピングされていないリモートカラムのみが表示されます。

統合カラムを同期から除外するには、なしをクリックします。方向アイコンに、統合カラムが同期されないことが示されます。

統合カラムを値にマッピングするには、ユーザ、リモート、またはカスタムを選択して、同期中にリモートテーブルの upload_insert、upload_update、upload_delete 同期スクリプトが実行されるときに評価される SQL 式を入力します。[方向] アイコンには値がアップロード専用であることが示されます。すなわち、統合カラムはリモートデータベースにダウンロードされません。

結果

カラムマッピングが更新されます。

1.2.3.2.2 ダウンロード方式の変更

詳細ウィンドウ枠のダウンロード方式タブでダウンロードタイプを変更します。テーブルマッピングのダウンロードタイプには、タイムスタンプ、スナップショット、カスタムのいずれかを設定できます。

前提条件

同期モデルが必要です。

コンテキスト

ダウンロードタイプは次のとおりです。

[タイムスタンプベースのダウンロード]

このオプションを選択すると、タイムスタンプベースのダウンロードがデフォルトとして使用されます。最後の同期後に変更されたローのみがダウンロードされます。

[スナップショットダウンロード]

このオプションを選択すると、スナップショットダウンロードがデフォルトとして使用されます。最後の同期後に変更されていない場合でも、同期のたびにすべてのローがダウンロードされます。

[カスタムダウンロードロジック]

download_cursor スクリプトと download_delete_cursor スクリプトを自動的に生成せず、独自に作成する場合は、このオプションを選択します。

手順

1. Mobile Link プロジェクト名をダブルクリックします。
2. 同期モデルをダブルクリックし、同期モデル名をダブルクリックします。
3. マッピングタブを開きます。
4. テーブルマッピングウィンドウ枠で、テーブルマッピングを選択します。
5. 詳細ウィンドウ枠で、ダウンロード方式タブを選択します。
6. 方式ドロップダウンリストで、タイムスタンプ、スナップショット、カスタムのいずれかを選択します。
7. カスタムを選択した場合は、上ウィンドウ枠のイベントタブをクリックして、download_cursor と download_delete_cursor の各スクリプトを入力します。
8. タイムスタンプを選択した場合は、次のタスクを実行します。
   a. タイムスタンプカラム名フィールドにカラム名を入力します。
   b. 次のうちの 1 つを選択してください。
      1. 次を使用してタイムスタンプを更新する
         - デフォルトカラム値
         - トリガ
      2. タイムスタンプカラムをシャドウテーブルに保存する

結果

ダウンロードタイプが変更されます。

1.2.3.2.3 削除の記録方法の変更

統合データベースでの削除をリモートにダウンロードするかどうか、およびこのような削除に関する情報を統合データベースにどのように保存するかを制御する場合には、削除のダウンロードタブを使用します。

前提条件

同期モデルが必要です。さらに、テーブルマッピングのダウンロードタイプがタイムスタンプに設定されている必要があります。

コンテキスト

スナップショットダウンロードを使用している場合は、リモートデータベースのすべてのローが削除されてからスナップショットがダウンロードされます。このため、削除操作を追跡する必要はありません。タイムスタンプベースのダウンロードを使用している場合は、統合データベースでの削除をリモートデータベースへのダウンロードに記録する方法を指定できます。

統合データベースからローを削除したときに、リモートデータベースからもローを削除する場合は、削除された各ローを記録しておく必要があります。この情報は、シャドウテーブルを使用するか、論理削除によって記録することができます。

手順

1. Mobile Link プロジェクト名をダブルクリックします。
2. 同期モデルをダブルクリックし、同期モデル名を選択します。
3. マッピングタブをクリックします。
4. テーブルマッピングウィンドウ枠で、テーブルマッピングを選択します。
5. 詳細ウィンドウ枠で、削除のダウンロードタブを開きます。
6. 統合データベースから削除のダウンロードを行う場合は、統合データベースからローを削除したときに、リモートデータベースからもローを削除するチェックボックスをオンにします。統合データベースから削除のダウンロードを行わない場合は、チェックボックスをオフにします。

   削除を記録するには、シャドウテーブルまたは論理削除の使用を設定します。

結果

削除の処理方法が更新されます。

1.2.3.2.4 サブセットのダウンロード

各 Mobile Link リモートデータベースでは、統合データベースに格納されているデータのサブセットを受信できます。テーブルごとにダウンロードサブセットをカスタマイズできます。

ダウンロードサブセットのオプションは、次のとおりです。

[ユーザ]

このオプションを選択すると、Mobile Link ユーザ名によってデータが分割され、登録済みの Mobile Link ユーザごとに異なるデータがダウンロードされます。

このオプションを使用するには、Mobile Link ユーザ名を保持するカラムが各ローに含まれている必要があります。Mobile Link ユーザ名は展開時に選択するため、統合データベースの既存の値と一致する名前を選択できます (Mobile Link ユーザ名用に使用するカラムは、ユーザ名として使用する値を格納できるタイプである必要があります)。サブセットのテーブルとは異なるテーブルに Mobile Link ユーザ名がある場合は、そのテーブルにジョインする必要があります。

[リモート]

このオプションを選択すると、リモート ID によってデータが分割され、リモートデータベースごとに異なるデータがダウンロードされます。

このオプションを使用するには、リモート ID を保持するカラムが各ローに含まれている必要があります。リモート ID はデフォルトでは GUID として作成されますが、統合データベースの既存の値に一致するリモート ID を設定できます (リモート ID 用に使用するカラムは、リモート ID として使用する値を格納できるタイプである必要があります)。サブセットのテーブルとは異なるテーブルにリモート ID がある場合は、そのテーブルにジョインする必要があります。

> **注記**
>
> リモート ID は、リモートコンピュータがリセットされるか、置き換えられたときに変更される場合があるため、通常、リモート ID 別ではなく、ユーザ別、または認証パラメータ別に分割することをお奨めします。

[カスタム]

ダウンロードするローを指定する SQL 式を使用するには、このオプションを選択します。各同期は、SQL 式が true のローのみをダウンロードします。この SQL 式は、生成された download_cursor スクリプトの WHERE 句に追加されます。Mobile Link 名前付きパラメータを式で使用できます。また、他のテーブルを参照することもできます。他のテーブルを参照する場合は、式の上にあるフィールドで他のテーブルをリストし、式にジョイン条件を含める必要があります。

1.2.3.2.5 ダウンロードサブセットの変更

テーブルごとにダウンロードサブセットをカスタマイズします。各 Mobile Link リモートデータベースでは、統合データベースに格納されているデータのサブセットを同期できます。

前提条件

同期モデルが必要です。さらに、テーブルマッピングのダウンロードタイプがカスタムに設定されていない必要があります。

手順

1. Mobile Link プロジェクト名をダブルクリックします。
2. 同期モデルをダブルクリックし、同期モデル名を選択します。
3. マッピングタブをクリックします。
4. テーブルマッピングウィンドウ枠で、リモートテーブルを選択します。
5. 詳細ウィンドウ枠で、サブセットのダウンロードタブを開きます。
6. サブセットのダウンロードドロップダウンリストから次のダウンロードサブセットのいずれかを選択します。なし、ユーザ、リモート、またはカスタム。
7. ユーザまたはリモートを選択した場合は、ユーザ名とリモート ID を保持するカラムが置かれている場所を識別します。

   同期されている統合テーブルにカラムがある場合は、テーブル `table name` を選択し、カラム名ドロップダウンリストからユーザ名またはリモート ID を保持するカラムを選択します。

   カラムが別のテーブルにある場合は、テーブル `table name` とジョインできる別のテーブルを選択します。別のテーブルドロップダウンリストからカラムが含まれているテーブルを選択します。*Mobile Link* ユーザを含むカラムドロップダウンリストで、ユーザ名またはリモート ID を保持しているカラムを選択します。ジョイン条件を使用して、同期テーブルをシャドウテーブルにジョインするためのジョイン条件を定義します。
8. カスタムを選択した場合は、2 つのテキストボックスが表示されます。このテキストボックスには download_cursor スクリプトの作成に使用する情報を追加できます。download_cursor をすべて自分で作成する必要はありません。ダウンロードサブセットのジョインやその他の制限を指定するために追加情報を指定する必要があるだけです。
   - 1 つ目のテキストボックス (*SQL* 式) に、ダウンロードサブセット条件とジョイン条件を指定する、生成された WHERE 句に追加する SQL 式を入力します。認証パラメータを含む Mobile Link 名前付きパラメータを式で使用できます。デフォルトでは、これと同じ式とジョインされたテーブルがダウンロード削除サブセット用に使用されます。削除の追跡用にシャドウテーブルを使用しており、同じ式を使用する場合、式にベーステーブル名は使用しないようにします。それができない場合は、カスタムダウンロード削除サブセットを使用します。
   - download_cursor にその他のテーブルへのジョインが必要な場合は、2 つ目のテキストボックス (別のテーブル) にテーブル名を入力します。ジョインで複数のテーブルが必要な場合は、カンマで区切ります。

結果

ダウンロードサブセットが変更されます。

例

ユーザの例

たとえば、CustDB の ULOrder テーブルは、ユーザ間で共有できます。デフォルトで、注文は作成した従業員に割り当てられていますが、別の従業員によって作成された注文を確認したい場合があります。たとえば、マネージャは、部署の従業員が作成したすべての注文を確認する必要があるかもしれません。CustDB データベースでは、この状況に備えるために、ULEmpCust テーブルを使用します。これにより、顧客を従業員に割り当てることができます。マネージャは、ある従業員と顧客の関係における注文をダウンロードします。

ダウンロードサブセットなしで、ULOrder の download_cursor スクリプトを表示します。次に、マッピングタブで ULEmpCust テーブルを選択します。ダウンロード方式でタイムスタンプ、サブセットのダウンロードでなしをそれぞれ選択します。テーブルを右クリックし、イベントに移動をクリックします。テーブルの download_cursor は、次のようになります。

```
SELECT "DBA"."ULOrder"."order_id",
    "DBA"."ULOrder"."cust_id",
    "DBA"."ULOrder"."prod_id",
    "DBA"."ULOrder"."emp_id",
    "DBA"."ULOrder"."disc",
    "DBA"."ULOrder"."quant",
    "DBA"."ULOrder"."notes",
    "DBA"."ULOrder"."status"
FROM "DBA"."ULOrder"
WHERE "DBA"."ULOrder"."last_modified" >= {ml s.last_table_download}
```

マッピングタブに戻ります。詳細ウィンドウ枠のサブセットのダウンロードタブで、ULOrder のサブセットのダウンロードドロップダウンリストをユーザに変更します。テーブル *DBA.ULOrder* とジョインできる別のテーブルを選択します。別のテーブルの場合、ULEmpCust を選択します。Mobile Link ユーザを含むカラムの場合、emp_id を選択します。ジョイン条件の場合、DBA.ULOrder.cust_id=DBA.ULEmpCust.cust_id を選択します。

一番上のウィンドウ枠でテーブルを右クリックし、イベントに移動をクリックします。テーブルの download_cursor は、次のようになります (新しい行を太字で示しています)。

```
SELECT "DBA"."ULOrder"."order_id",
    "DBA"."ULOrder"."cust_id",
    "DBA"."ULOrder"."prod_id",
    "DBA"."ULOrder"."emp_id",
    "DBA"."ULOrder"."disc",
    "DBA"."ULOrder"."quant",
    "DBA"."ULOrder"."notes",
    "DBA"."ULOrder"."status"
FROM "DBA"."ULOrder", "DBA"."ULEmpCust"
WHERE "DBA"."ULOrder"."last_modified" >= {ml s.last_table_download}
AND "DBA"."ULOrder"."cust_id" = "DBA"."ULEmpCust"."cust_id"
AND "DBA"."ULEmpCust"."cust_id" = {ml s.username}
```

カスタムの例

Mobile Link ユーザによる **Customer** というテーブルのダウンロードをサブセットし、active=1 である場合だけローをダウンロードするとします。Mobile Link ユーザ名はサブセットするテーブルに存在しないため、ユーザ名が含まれる SalesRep というテーブルへのジョインを作成する必要があります。

マッピングタブで、**Customer** テーブルのマッピングを選択します。詳細ウィンドウ枠のダウンロード方式タブを開きます。ダウンロードタイプをタイムスタンプに設定します。詳細ウィンドウ枠のサブセットのダウンロードタブを開きます。サブセットのダウンロードドロップダウンリストのカスタムを選択します。1 つ目のテキストボックス (*SQL* 式) に、次のように入力します。

```
SalesRep.ml_username = {ml s.username}
   AND Customer.active = 1
   AND Customer.cust_id = SalesRep.cust_id
```

2 つ目のテキストボックス (別のテーブル) に、次のように入力します。

```
SalesRep
```

cust_id カラムが両方のテーブルにあるので、これらのカラムへの参照は、式でテーブル名をプレフィクスとして付ける必要があります。削除の追跡をダウンロードするためにシャドウテーブルを使用する場合、シャドウテーブルが Customer では

なく Customer_del と呼ばれるので、Customer テーブルマッピングのダウンロード削除サブセットカラムにあるなしまたはカスタムを使用する必要があります。

一番上のウィンドウ枠でテーブルを右クリックし、イベントに移動をクリックします。テーブルの download_cursor は、次のようになります。

```
SELECT "DBA"."Customer"."cust_id",
 "DBA"."Customer"."cust_name"
FROM "DBA"."Customer", SalesRep
WHERE "DBA"."Customer"."last_modified" >= {ml s.last_table_download}
 AND SalesRep.ml_username = {ml s.username}
 AND Customer.active = 1
 AND Customer.cust_id = SalesRep.cust_id
```

WHERE 句の最後の行により、Customer から SalesRep へのキージョインが作成されます。

1.2.3.2.6 ダウンロード削除サブセットの変更

ダウンロードサブセットを使用して統合データベースでデータのサブセットを同期している場合は、デフォルトによりダウンロード削除サブセットがダウンロード済みに設定され、ダウンロードサブセットとまったく同じになります。この設定は、すべてまたはカスタムに変更できます。

ダウンロードサブセットがカスタムであり、削除の追跡にシャドウテーブルを使用している場合は、ローを削除するときに、ダウンロードサブセット論理によって使用されるカラム値をいずれもシャドウテーブルにコピーする必要があります。プライマリキーカラム値は、シャドウテーブルに自動的にコピーされます。

余計なカラムを指定してシャドウテーブルに追加することを除けば、カスタムダウンロードサブセットを定義することは、カスタムダウンロード削除サブセットを設定することと同じです。

関連情報

ダウンロードサブセットの変更 [46 ページ]

1.2.3.2.7 競合の検出と解決

リモートデータベースと統合データベースの両方でローが更新された場合は、次にデータベースを同期するときに競合が発生します。

競合の検出には、次のオプションがあります。

[ローベースの競合検出]

最後の同期後に、リモートデータベースと統合データベースの両方でローが更新されていた場合に競合が検出されます。

このオプションは、upload_fetch スクリプトと upload_update スクリプトを定義します。

[カラムベースの競合検出]

リモートデータベースと統合データベースの両方で、ローの同じカラムが更新されていた場合に競合が検出されます。

このオプションは、upload_fetch_column_conflict スクリプトを定義します。

テーブルに BLOB があり、カラムベースの競合検出を選択した場合は、ローベースの競合検出が使用されます。

競合の解決には、次のオプションがあります。

[統合]

先入れ勝ちです。アップロードされた更新が競合する場合は拒否されます。

[リモート]

後入れ勝ちです。アップロードされた更新が常に適用されます。

[タイムスタンプ]

最新の更新が適用されます。このオプションを使用するには、テーブルの TIMESTAMP カラムを作成し、維持する必要があります。この TIMESTAMP カラムに、ローが最後に変更された時刻が記録されます。統合データベースとリモートデータベースの両方にカラムが存在し、タイムスタンプベースのダウンロードで使用されるカラムと異なっている必要があります。これを機能させるには、リモートデータベースと統合データベースで同じタイムゾーン (UTC を推奨) を使用し、かつクロックが同期されている必要があります。

[カスタム]

独自の resolve_conflict スクリプトを作成します。スクリプトは、イベントタブで作成できます。

このセクションの内容:

競合の検出と解決の変更 [50 ページ]
SQL Central を使用して競合の検出および解決の設定をカスタマイズします。

1.2.3.2.7.1 競合の検出と解決の変更

SQL Central を使用して競合の検出および解決の設定をカスタマイズします。

前提条件

同期モデルが必要です。

手順

1. Mobile Link プロジェクト名をダブルクリックします。
2. 同期モデル名をダブルクリックします。
3. マッピングタブをクリックします。
4. テーブルマッピングウィンドウ枠で、テーブルマッピングを選択します。

5. 詳細ウィンドウ枠で、競合の処理タブを開きます。
6. 競合を検出する方式を選択しますで、ローベースまたはカラムベースを選択します。
7. 検出された競合を解決する方式を選択しますドロップダウンリストから、先入れ勝ち、後入れ勝ち、カスタムのいずれかを選択します。
8. カスタムを選択した場合は、イベントタブを開き、テーブルの resolve_conflict スクリプトを作成します。

結果

競合の検出と解決のための設定が更新されます。

1.2.3.2.8 同期モデル内のスクリプトの変更

同期モデル内のスクリプトは、イベントタブを使用して変更します。

イベントタブでは、次のタスクを実行できます。

- 現在のテーブルマッピングに基づいて生成されたスクリプトを表示および変更します。
- 新しいスクリプトを作成します。

イベントタブの上部には、選択したスクリプトが属するグループが表示されます。単一テーブルのスクリプトはすべて 1 つにまとめられています。イベントタブの上部には、選択したスクリプトの名前が表示されます。また、このスクリプトが Mobile Link プラグインによって生成されたものかどうか、ユーザによって定義されたものかどうか、または生成したスクリプトが上書きされているかどうかも表示されます。また、同期ロジックが SQL、.NET、Java のどれを使用して作成されているかも表示されます。

このスクリプトは、生成されたスクリプトを上書きするスクリプトを追加すると、完全に制御できるようになります。したがって、関連する設定を変更しても、このスクリプトが自動的に変更されることはありません。たとえば、モデルの download_delete_cursor を変更した後に、マッピングタブのテーブルマッピングウィンドウ枠で削除カラムの選択を解除しても、カスタマイズした download_delete_cursor に影響はありません。

ファイルメニューのオプションを使用すると、上書き生成スクリプトをリストアしたり、無視されるように設定したスクリプトをリストアしたり、追加した新しいスクリプトを削除したりすることができます。リストアまたは削除するスクリプトを選択してからファイルをクリックして、オプションを表示します。

このセクションの内容:

1.2.3.2.8.1 特定のテーブルのスクリプトの検索

SQL Central を使用して、特定のテーブルに対して定義されたスクリプトを検索します。

前提条件

同期モデルが必要です。

手順

1. Mobile Link プロジェクトをダブルクリックします。
2. 同期モデル名をダブルクリックします。
3. イベントタブを開きます。
4. グループドロップダウンで、確認するイベントが含まれているテーブルを選択します。
5. イベントフィールドで、検索するスクリプト名を選択します。既存のスクリプトは太字で強調表示されます。

結果

選択したスクリプトにカーソルが移動します。

次のステップ

選択したスクリプトに変更を加えます。

1.2.3.3 スキーマの更新

スキーマ更新ウィザードを使用して、同期モデル内の統合データベースとリモートデータベースのスキーマを更新します。

前提条件

同期モデルが必要です。

コンテキスト

スキーマ更新ウィザードは、モデルを展開した後と、次の場合に最適です。

- モデルに追加する必要があるリモートデータベースのスキーマを変更した場合。
- モデルに追加する必要がある統合データベースのスキーマを変更した場合。
  たとえば、1 つまたは複数のテーブルにタイムスタンプベースのダウンロードを作成したモデルを再展開する前に、[スキーマの更新] を実行したとします。この場合は、以前に展開したときに、TIMESTAMP カラムまたはシャドウテーブルを追加して、統合データベースのスキーマを変更したために、スキーマを更新する必要があります。

手順

1. Mobile Link プロジェクト名をダブルクリックします。
2. 同期モデルをダブルクリックします。
3. 同期モデル名をダブルクリックし、 ファイル スキーマ更新 をクリックします。
4. 次のオプションのうちの 1 つを選択してください。

   統合データベーススキーマ

   モデル内の統合スキーマは更新されます。モデル内のリモートスキーマは変更されません。

   リモートデータベーススキーマ

   モデル内のリモートスキーマは更新されます。モデル内の統合スキーマは変更されません。

   統合データベーススキーマとリモートデータベーススキーマ

   統合およびリモートのスキーマの両方が、既存のデータベースのスキーマに一致するようにモデル内で更新されます。
5. スキーマ更新ウィザードの指示に従います。

   完了をクリックしても、同期モデルを展開するまでモデル以外での変更は行われません。そのときまで統合データベースは変更されず、リモートデータベースは作成も変更もされません。
6. マッピングタブで、新しいリモートテーブルをマッピングします。

結果

スキーマが更新されます。

関連情報

テーブルとカラムのマッピング [38 ページ]

1.2.3.4 同期モデルの展開

同期モデルは、同期モデル展開ウィザードを使用して展開します。

次の項目を展開できます。

- 統合データベースの変更内容
- SQL Anywhere または Ultra Light リモートデータベース (データベースを作成するか、既存の空のデータベースにテーブルを追加するか、リモートテーブルがすでに格納されている既存のデータベースを使用)
- モデルを展開するバッチファイル (生成されるバッチファイルの先頭には変数宣言があり、バッチ ファイルの実行前にこの変数宣言を編集できます)
- Mobile Link サーバと Mobile Link クライアントを実行するためのバッチファイル

Mobile Link 17 プラグインでは、同期モデルを展開すると、スキーマの比較が行われます。データベースオブジェクトの中でも展開対象のオブジェクトと異なるもの (たとえば、テーブルやインデックスなど) のみが変更されます。新しいオブジェクトを展開することにより既存の同期システムが壊れる場合は警告が表示されます。操作は中止することができます。この機能は、IBM DB2 LUW 統合データベースでは利用できません。

同じスクリプトバージョンに更新を再展開すると、新しいスクリプトバージョンで必要なくなったスキーマは自動的に削除されます。たとえば、ダウンロードをサポートするために last_modified カラムがテーブルに追加されている場合、同期しかアップロードしないように変更すると、last_modified カラムは展開中に削除されます。

また、ml_model_drop システムプロシージャを使用すれば、統合データベースから同期モデルとそのスキーマを削除することができます。このシステムプロシージャは、同期スクリプトを削除するほか、同期モデルが展開されたときに作成されたスキーマ (シャドウテーブル、トラッキングカラム、トリガ、インデックスなど) も削除します。別の script_version と共有されているスキーマは削除されません。Mobile Link 17 プラグインを使用してインストールしたスキーマしか削除できません。この機能は、IBM DB2 LUW 統合データベースでは利用できません。

統合データベースへの配備

同期モデル展開ウィザードでは、次の 2 つの方法で統合データベースに展開できます。

- Mobile Link システムテーブルにデータを追加し、必要なシャドウテーブル、カラム、トリガ、ストアドプロシージャを作成することで、同期モデルを統合データベースに直接適用します。
- 同じ変更内容をすべて含む UTF-8 でエンコードされたファイルとバッチファイルを作成して、統合データベースに対して SQL ファイルを実行します。このファイルは、いつでも確認、変更、実行することができます。結果は変更を直接適用する場合と同じです。

このセクションの内容:

リモートデータベースの展開 [59 ページ]
既存のリモートデータベースを使用するか、ウィザードでリモートデータベースを作成できます。ウィザードでは、リモートデータベースを直接作成するか、または、リモートデータベースを作成または更新するために実行する、UTF-8 でエンコードされた SQL ファイルとバッチファイルを作成できます。

SQL ファイルからのリモートデータベースの展開 [59 ページ]
同期モデル展開ウィザードを使用して、リモートデータベースを展開します。

バッチファイルを展開して同期ツールを実行 [60 ページ]
同期モデル展開ウィザードでは、同期ツールの実行に使用できるバッチファイルを作成できます。

同期モデルの再展開 [60 ページ]
同期モデルを展開した後、同期モデルに変更を加えて再展開すると、同期モデルを変更できます。

展開した同期モデル [60 ページ]
同期モデルファイルは、Mobile Link プロジェクトディレクトリにあります。同期ファイルのファイル拡張子は、`.mlsm` です。同期モデルを展開すると、該当するモデル名で始まり `_deploy` で終わるディレクトリが作成されます。

バッチファイルの実行 [61 ページ]
同期モデル展開ウィザードで作成されたバッチファイルをコマンドラインから実行して、同期モデルを同期します。

1.2.3.4.1 展開前の同期モデルのテスト

テストウィンドウを使用し、展開する前に同期モデルをテストします。

前提条件

Mobile Link プロジェクトで定義された同期モデルが必要です。

コンテキスト

テスト機能を使用すると、統合データベースに変更が加えられます。

手順

1. Mobile Link プロジェクト名をダブルクリックします。
2. **同期モデル**をダブルクリックし、同期モデル名をダブルクリックします。
3. **展開**ウィンドウ枠で、**テスト**をクリックします。同期モデルをテストすると統合データベースに変更が加えられテーブル内のデータが変更されることを示す警告が表示されます。*OK* をクリックします。
4. **同期**をクリックします。

結果

同期モデルがテストされ、同期が成功したかどうかを示します。

次のステップ

テストが完了したら、次のタスクを実行できます。

- データタブに関する情報を確認し、統合データベースのデータとリモートデータベースのデータを比較します。
- テスト中の失敗に関する情報については、クライアントログタブの情報を確認してください。これは、SQL Anywhere クライアントでのみ使用できます。
- テスト中の失敗に関する情報については、*Mobile Link* ログタブの情報を確認してください。
- 同期をもう一度実行する前に、リモートデータベースと統合データベース、またはそのいずれかを変更します。データベースを変更するには、いくつかのオプションを使用できます。
  - データタブを使用して、同期されているテーブル内のデータを直接変更します。
  - アクションメニューを使用して、いずれかのデータベースで dbisql ユーティリティを開く。

1.2.3.4.2 同期モデルのテストの設定

同期モデルのテスト、同期ロジックのより詳細なテストの有効化、運用環境に近い環境の作成を行うために使用するいくつかのオプションを設定するには、ウィンドウ設定のテストウィンドウを使用します。

前提条件

Mobile Link プロジェクトで定義された同期モデルが必要です。

コンテキスト

テスト機能を使用すると、統合データベースに変更が加えられます。

手順

1. Mobile Link プロジェクト名をダブルクリックします。
2. 同期モデルをダブルクリックし、同期モデル名を選択します。
3. 展開ウィンドウ枠で、テストをクリックします。同期モデルをテストすると統合データベースに変更が加えられテーブル内のデータが変更されることを示す警告が表示されます。*OK* をクリックします。

4. アクション > 設定 をクリックします。
5. 以下のフィールドに入力します。

   Mobile Link ユーザ

   Mobile Link サーバに接続するときに Mobile Link クライアントで使用する Mobile Link ユーザ名を入力します。

   Mobile Link パスワード

   指定した Mobile Link ユーザ名のパスワードを入力します。

   認証パラメータ

   カンマで区切られたリストを使用して、認証パラメータを入力します。例: `p1,p2,p3` これらの値は Mobile Link クライアントから Mobile Link サーバに送信され、サーバ側の同期スクリプトからアクセスすることができます。

   Mobile Link サーバのコマンドライン

   Mobile Link サーバをテスト用に起動するときに使用する Mobile Link サーバのコマンドラインを選択します。Mobile Link サーバのコマンドラインに指定されたオプションを確認するには、表示をクリックします。*Mobile Link サーバのコマンドラインのプロパティ*ウィンドウが表示されます。

   クライアントネットワークプロトコル

   Mobile Link サーバに接続するときに Mobile Link クライアントで使用するネットワーク通信プロトコルを選択します。Mobile Link サーバのコマンドラインで定義されたプロトコルのみを選択できます。

   クライアントネットワークオプション

   Mobile Link サーバに接続するときに Mobile Link クライアントで使用するネットワークプロトコルオプションを選択します。オプションを直接編集するか、編集をクリックしてクライアントネットワークオプションページからオプションを編集します。

   クライアントネットワークプロトコルを選択すると、Mobile Link サーバのコマンドラインに基づいて、このフィールドにデフォルトオプションが生成されます。常にデフォルトオプションを確認し、それらのオプションが同期に適していることを確認してください。

   同期タイプ

   ドロップダウンリストから、次の同期タイプのいずれかを選択します。

   [双方向]

   データベース操作は、リモートデータベースから統合データベースの方向へ、および統合データベースからリモートデータベースの方向へ同期されます。

   [ダウンロード専用]

   変更は統合データベースからリモートデータベースの方向へのみ同期されます。

   [アップロード専用]

   変更はリモートデータベースから統合データベースの方向へのみ同期されます。

   この統合データベースでこの同期モデルをテストするときに、常にこれらの設定を使用する

   この同期モデルをテストするときに常にこの設定を使用する場合は、このオプションを選択します。指定されたオプションは、設定が変更されるまで使用されます。
6. *OK* をクリックします。同期モデルが再配備され、リモートデータベースのデータが失われることを示す警告が表示されます。*OK* をクリックします。

結果

新しい設定を使用して、同期モデルがテストウィンドウで再配備されます。

次のステップ

同期モデルが再配備されたら、次のタスクを実行できます。

- アクション > 設定 を使用して、テストウィンドウ設定を再度変更します。
- リモートデータベースと統合データベース、またはそのいずれかを変更します。データベースを変更するには、いくつかのオプションを使用できます。
  - データタブを使用して、同期されているテーブル内のデータを直接変更します。
  - アクションメニューを使用して、いずれかのデータベースで dbisql ユーティリティを開く。
- 同期をクリックして、同期のテストを実行します。

関連情報

展開前の同期モデルのテスト [55 ページ]
Mobile Link サーバのコマンドラインを作成 [30 ページ]

1.2.3.4.3 SQL ファイルからの統合データベースの展開

同期モデル展開ウィザードを使用して、統合データベースを展開します。

前提条件

展開時にシャドウテーブルを作成した場合は、シャドウテーブルが作成されるベーステーブルの所有者または管理者として、統合データベースに接続する必要があります。

手順

同期モデル展開ウィザードの実行時に、(データベースの同期を準備する方法を選択しますページで) 後で実行できるようにファイルを作成することを選択した場合は、モデルの `project-name_deploy` サブフォルダにあるバッチファイルを実行する必要があります。このファイルによって、同期スクリプト、シャドウテーブル、トリガなど、統合データベースに作成することを選択したオブジェクトがすべて作成されます。また、Mobile Link ユーザが統合データベースに登録される場合もあります。

このファイルを実行するには、project-name_deploy ディレクトリに移動し、cons_setup.bat ファイルまたは cons_setup.sh ファイルを実行します。このとき、接続情報を指定する必要があります。たとえば、次のように入力します。

```
cons_setup "DSN=my_odbc_datasource;UID=myuserid;PWD=mypassword"
```

一部のドライバでは、ODBC データソースにユーザ ID とパスワードがあるため、これらの指定は不要です。

結果

統合データベースが展開されます。

1.2.3.4.4 リモートデータベースの展開

既存のリモートデータベースを使用するか、ウィザードでリモートデータベースを作成できます。ウィザードでは、リモートデータベースを直接作成するか、または、リモートデータベースを作成または更新するために実行する、UTF-8 でエンコードされた SQL ファイルとバッチファイルを作成できます。

ウィザードでは、モデルで指定したデータベース所有者を使用して、デフォルトのデータベース作成オプションでリモートデータベース (SQL Anywhere または Ultra Light) が作成されます。また、同期モデル展開ウィザードを使用しないでカスタム設定でリモートデータベースを作成し、ウィザードを使用して必要なリモートテーブルを追加するか、すでにリモートテーブルがある既存のリモートデータベースに展開することもできます。

1.2.3.4.5 SQL ファイルからのリモートデータベースの展開

同期モデル展開ウィザードを使用して、リモートデータベースを展開します。

手順

同期モデル展開ウィザードの実行時に、(データベースの同期を準備する方法を選択しますページで) 後で実行できるようにファイルを作成することを選択した場合は、project-name_deploy ディレクトリにある SQL ファイルで作成されたバッチファイルを実行する必要があります。このファイルによって、テーブル、パブリケーション、サブスクリプション、Mobile Link ユーザなど、リモートデータベースに作成することを選択したオブジェクトがすべて作成されます。

このファイルを実行するには、project-name_deploy ディレクトリに移動し、remote_setup.bat ファイルまたは remote_setup.bat.sh ファイルを実行します。たとえば、次のように入力します。

```
remote_setup.bat
```

既存のリモートデータベースを使用している場合は、パスワードの入力を要求されます。

結果

リモートデータベースが展開されます。

1.2.3.4.6 バッチファイルを展開して同期ツールを実行

同期モデル展開ウィザードでは、同期ツールの実行に使用できるバッチファイルを作成できます。

次のバッチファイルを同期モデル展開ウィザードで作成できます。

- 指定するオプションで Mobile Link サーバを実行するバッチファイル
- SQL Anywhere のリモートデータベースの場合、指定するオプションで dbmlsync を実行するバッチファイル
- Ultra Light のリモートデータベースの場合、指定するオプションで ulsync を実行するバッチファイル ulsync は、同期のテストに使用されるので、正常に機能する Ultra Light アプリケーションがない場合に初めて同期するときに役立ちます。

1.2.3.4.7 同期モデルの再展開

同期モデルを展開した後、同期モデルに変更を加えて再展開すると、同期モデルを変更できます。

また、システムプロシージャや他のメソッドを使用して変更することもできます。ただし、展開後のモデルを SQL Central 外で変更した場合、変更内容をリバースエンジニアリングで同期モデルに戻すことはできません。SQL Central 外で加えられた変更は、モデルを再展開するときに上書きされます。

展開することでスキーマが変更されてしまう場合があるので、その他の変更を行っていないときでも、スキーマを更新する必要があります。たとえば、統合データベース内の各同期テーブルに TIMESTAMP カラムを追加するモデルを展開する場合は(モデル作成時のデフォルトの動作)、モデル内の統合スキーマを更新してから再展開してください。同様に、統合データベースにテーブルを追加してから再展開する場合は、モデル内の統合スキーマを更新してから、新しいリモートテーブルを作成する必要があります。

関連情報

スキーマの更新 [52 ページ]

1.2.3.4.8 展開した同期モデル

同期モデルファイルは、Mobile Link プロジェクトディレクトリにあります。同期ファイルのファイル拡張子は、`.mlsm` です。同期モデルを展開すると、該当するモデル名で始まり `_deploy` で終わるディレクトリが作成されます。

この場合、次のようなファイルが作成されます。作成されるファイルは選択した展開オプションによって異なります。

ファイル	説明
cons_setup.bat	SQL ファイルを実行して統合データベースを設定するバッチファイル。
cons_setup.sql	統合データベースの設定に使用する SQL ファイル。
mlsrv.bat	Mobile Link サーバを実行するためのバッチファイル。
remote_setup.bat	SQL ファイルを実行してリモートデータベースを設定するバッチファイル。
remote_setup.sql	統合データベースの設定に使用する SQL ファイル。
summary.txt	同期モデルの詳細を要約するファイル。
sync.bat	SQL Anywhere リモートデータベースを展開した場合の、SQL Anywhere データベースを dbmlsync と同期するバッチファイル。
modelname_remote.db	新しい SQL Anywhere リモートデータベースを作成する場合のデータベースファイル。
modelname_remote.udb	新しい Ultra Light リモートデータベースを作成する場合のデータベースファイル。
modelname_ulsync.bat	Ultra Light データベースを展開した場合の、ulsync ユーティリティを使用して Ultra Light リモートデータベースとの同期をテストするためのバッチファイル。

1.2.3.4.9 バッチファイルの実行

同期モデル展開ウィザードで作成されたバッチファイルをコマンドラインから実行して、同期モデルを同期します。

前提条件

同期モデルが必要です。

コンテキスト

統合データベースで Mobile Link 設定スクリプトを実行していない場合は、実行してから展開します。

バッチファイルの多くは実行時に接続情報の指定を必要とします。これらのバッチファイルを実行する前に ODBC データソースを作成する必要がある場合もあります。

手順

1. 同期モデル展開ウィザードの実行時に、(データベースの同期を準備する方法を選択しますページで) 後で実行できるようにファイルを作成することを選択した場合は、モデルの project-name_deploy サブフォルダにあるバッチファイルを実行する必要があります。このファイルによって、同期スクリプト、シャドウテーブル、トリガなど、統合データベースに作成することを選択したオブジェクトがすべて作成されます。また、Mobile Link ユーザが統合データベースに登録される場合もあります。

   このファイルを実行するには、project-name_deploy ディレクトリに移動し、cons_setup.bat ファイルまたは cons_setup.sh ファイルを実行します。コマンドラインには、接続情報を含めてください。たとえば、次のように入力します。

   ```
   cons_setup.bat "DSN=MY_ODBC_DATASOURCE"
   ```

2. 同期モデル展開ウィザードの実行時に、(データベースの同期を準備する方法を選択しますページで) 後で実行できるようにファイルを作成することを選択した場合は、project-name_deploy ディレクトリにあるバッチファイルを実行する必要があります。このファイルによって、テーブル、パブリケーション、サブスクリプション、Mobile Link ユーザなど、リモートデータベースに作成することを選択したオブジェクトがすべて作成されます。

   このファイルを実行するには、project-name_deploy ディレクトリに移動し、remote_setup.bat ファイルまたは remote_setup.sh ファイルを実行します。たとえば、次のように入力します。

   ```
   remote_setup.bat
   ```

   既存のリモートデータベースを使用している場合は、パスワードの入力を要求されます。
3. mlsrv.bat を実行して、Mobile Link サーバを起動します。コマンドラインには、統合データベースの接続情報を含めてください。たとえば、次のように入力します。

   ```
   mlsrv.bat "DSN=MY_ODBC_DATASOURCE"
   ```

4. 同期を実行します。

   SQL Anywhere リモートデータベースの場合

   - dbmlsync を実行できるようにユーザに SYS_RUN_REPLICATION_ROLE システムロールを付与します。たとえば、Interactive SQL で次のコマンドを実行します。

     ```
     GRANT ROLE SYS_RUN_REPLICATION_ROLE
     TO userid
     ```

   - ロールを付与されたユーザとしてデータベースに接続します。
   - project-name_deploy ディレクトリにあるリモートデータベースを起動します。たとえば、次のように入力します。

     ```
     dbsrv17 MyModel_remote.db
     ```

   - SQL Anywhere Mobile Link クライアント dbmlsync を起動します。project-name_deploy ディレクトリにある sync.bat ファイルを実行します。コマンドラインには、接続情報を含めてください。たとえば、次のように入力します。

     ```
     sync.bat "UID=userid;PWD=password;SERVER=MyModel_remote"
     ```

   Ultra Light リモートデータベースの場合

   - 同期をテストするには、remote ディレクトリに存在する _ulsync.bat で終わるファイルを実行します。

- Ultra Light アプリケーションを実行することもできます。

結果

同期モデルが再展開されます。

1.2.3.5 同期モデルの制限事項

同期モデルにはいくつかの制限事項があります。

モデル以外で行われた変更の再展開はできない

同期モデルを展開し、その後でモデル以外で変更を加えた場合、この変更はモデルに保存されません。モデルを開始ポイントとして使用して展開し、すべての変更をモデル以外で行う場合はこれで問題ありません。ただし、モデルを再展開する場合は、Mobile Link プロジェクトで変更を行い、変更の保存と再展開ができるようにする方が適切です。

バージョン

1 つの同期モデルに設定できるバージョンは 1 つのみです。

Mobile Link システムデータベース

同期モデルを展開する場合、統合データベースとは別の Mobile Link システムデータベースを使用することはできません。

複数のパブリケーション

複数のパブリケーションを作成することはできません。モデルの展開後、CREATE PUBLICATION 文などの非モデルメソッドを使用して、パブリケーションをさらに追加することはできます。ただし、この方法で追加したパブリケーションをリバースエンジニアリングでモデルに戻すことはできません。

ビュー

テーブルマッピング用に統合データベースのテーブルを選択する場合、ビューは選択できません。

計算カラム

統合データベースのテーブルで、計算カラムはアップロードできません。計算カラムがある同期モデルを展開すると、タイムスタンプベースのダウンロードに使用するトリガの作成に失敗する場合があります。カラムを同期対象から除外するか、テーブルをダウンロード専用に設定します (このとき、スナップショットダウンロードを使用するか、生成済み統合 SQL ファイルを編集して、計算カラムをトリガ定義から削除します)。

計算カラムをコピーすると、新しいリモートスキーマを展開して新しいリモートデータベースを作成する際に構文エラーが発生します。計算カラムを扱う際は、次のいずれかを行います。

- 同期モデルを既存のリモートデータベースに展開します。
- リモートスキーマから計算カラムを除外します。計算カラムがある統合データベースのテーブルを同期する場合、テーブルにアップロードすることはできません。

Microsoft SQL Server AdventureWorks サンプルデータベースには、計算カラムが含まれています。このデータベースを使用してモデルを作成する場合、カラムをダウンロード専用に設定するか、カラムを同期対象から除外します。

Oracle の XMLTYPE データ型

SQL Central の Mobile Link プラグインでは、Oracle XMLTYPE が次のようにサポートされています。

- 新しいリモートスキーマに XMLTYPE カラムが選択された場合、またはカラムマッピングを編集するとき、表示されるデータタイプ名は *XMLTYPE* となります。
- XMLTYPE カラムは、新しい SQL Anywhere リモートデータベースの XML タイプのカラムにマッピングされます。
- 生成された同期スクリプトは、XMLTYPE カラム値が 4 KB 以下の場合に動作します。値がそれよりも大きい場合は、同期モデル内の生成されたスクリプトを、Oracle XMLTYPE データ型に関するトピックに記載されている技法を使用したスクリプトで上書きしてください。

プロキシテーブル

リモートデータベースでは、プロキシテーブルはパブリケーションの一部にはなりません。

配備に関する考慮事項

空間カラム

空間カラムはコピーされます。ただし、空間サブタイプと SRID については、それらを取得するためのメタデータサポート (SQL/MM 標準の ST_GEOMETRY_COLUMNS ビューなど) が統合 RDBMS にない場合は、コピーされないことがあります。Ultra Light での空間サポートは、SRID=0 または SRID=4326 のポイント値とカラム SRID 制約のみをサポートするタイプ (ST_GEOMETRY) に制限されています。このため、互換性のない空間タイプを新しい Ultra Light データベースに展開するときに警告またはエラーが表示されることがあります。

長いオブジェクト名

配備時に作成されるデータベースオブジェクトには、データベースでサポートされている長さより長い名前が割り当てられる場合があります (新しいオブジェクト名がベーステーブル名にサフィックスを追加して作成されるため)。この場合、(データベースに直接ではなく) ファイルのみに配備し、生成された SQL ファイルを編集して、長すぎる名前をすべて置換します。

新しいリモートスキーマ

同期モデル作成ウィザードで新しいリモートスキーマを作成する場合、新しいリモートデータベースのカラムには統合データベース内のカラムのインデックスは格納されません。外部キーとデフォルトのカラム値は新しいリモートデータベースにコピーされます。ただし、このサポートは ODBC ドライバによって返されるデータベースのメタデータに依存しており、ドライバの問題が原因で構文エラーやその他のエラーが生じる可能性があります。たとえば、ドライバがレポートしたデフォルトのカラム値が、SQL Anywhere または Ultra Light リモートデータベースではデフォルト値の宣言に使用できないフォーマットだった場合、エラーが起こる可能性があります (展開時の構文エラーなど)。

Ultra Light では、NCHAR(n)、NVARCHAR(n)、LONG NVARCHAR の各カラムタイプはサポートされていません。同期モデルを新しい Ultra Light データベースに展開する場合、リモートスキーマ内のこのようなカラムは CHAR(4n)、VARCHAR(4n)、または LONG VARCHAR に変換されます。4n が CHAR と VARCHAR の最大長を超える場合は、最大長が使用され、警告が表示されます。

既存のリモートデータベースを使用して、同期モデルを作成するか、モデル内でリモートスキーマを更新できます。

プロキシテーブル

プロキシテーブルである統合データベーステーブルを別のデータベースと同期することはできますが、タイムスタンプベースのダウンロードのために TIMESTAMP カラムを使用する場合は、ベーステーブルとプロキシテーブルの両方に TIMESTAMP カラムを追加してください。同期モデル展開ウィザードではカラムをプロキシテーブルやそのベーステーブルに追加することはできません。そのため、ベーステーブルとプロキシテーブルの両方に存在するカラムを使用するか、シャドウテーブルまたはスナップショットダウンロードを使用する必要があります。

マテリアライズドビュー

タイムスタンプベースのダウンロードを使用していて、TIMESTAMP カラムを統合テーブルに追加するように選択した場合は、テーブルに依存するマテリアライズドビューを無効にしてから展開を開始します。こうしないと、テーブルの変更時にエラーが発生する場合があります。SQL Anywhere 統合データベースでは、sa_dependent_views システムプロシージャを使用して、テーブルに依存するマテリアライズドビューがあるかどうかを判別します。

その他の考慮事項

Oracle 統合データベースに基づくリモートデータベースの作成

Oracle 統合データベースに基づいて SQL Anywhere または Ultra Light リモートデータベースを作成する場合、統合データベースの DATE カラムを TIMESTAMP に変更することが必要になる場合があります。この処理を行わないと、秒未満の情報が更新時に失われます。

1.3 Mobile Link の CustDB サンプル

CustDB は販売管理アプリケーションです。CustDB サンプルは、Mobile Link 開発者にとって貴重なリソースです。このサンプルを使用して、Mobile Link アプリケーションの開発時に必要なさまざまな方法の実装例を紹介します。

このアプリケーションを使用して、いくつかの一般的な同期方法を説明します。同期方法について理解を深めながら、サンプルアプリケーションについて学ぶことができます。

サポートされるオペレーティングシステムとデータベースの種類ごとに、CustDB が用意されています。

CustDB 統合データベースを使用する Mobile Link プロジェクトは、*%SQLANYSAMP17%*`¥MobiLink¥CustDB ¥project.mlp` ディレクトリにあります。このプロジェクトを SQL Central で開いて、CustDB プロジェクトとともに使用し、データベーススクリプトを表示できます。

CustDB シナリオ

統合データベースは、本社に配置されています。以下のデータが統合データベースに格納されます。

- 同期されるメタデータを格納する Mobile Link システムテーブル。同期ロジックを実装する同期スクリプトが含まれています。
- すべての顧客、製品、注文の情報を含み、ベーステーブルのローに格納される CustDB のデータ

リモートデータベースは、モバイル管理者用と営業担当者用の 2 種類があります。

各モバイル営業担当者のデータベースにはすべての製品とその営業担当者に対応する注文のみが格納されていますが、モバイル管理者のデータベースには特定の顧客セットに対するすべての製品と注文が格納されています。

同期の設計

CustDB サンプルアプリケーションでは、以下の機能を使用して同期を行います。

完全なテーブルのダウンロード

ULProduct テーブルのすべてのローとカラムが、リモートデータベースでも完全に共有されます。

カラムのサブセット

ULCustomer テーブルの一部のカラムのすべてのローが、リモートデータベースでも共有されます。

ローのサブセット

ULOrder テーブルから、リモートユーザごとに異なるローセット (ローのサブセット) を取得します。

タイムスタンプベースの同期

最後のデバイス同期以降に実行された統合データベースに対する変更を識別する方法です。ULCustomer テーブルと ULOrder テーブルは、タイムスタンプベースの方法で同期されます。

スナップショットを使った同期

同期を実行するたびにすべてのローをダウンロードする単純な同期方法です。ULProduct テーブルは、この方法で同期されます。

プライマリキープール (ユニークなプライマリキー管理用)

プライマリキーの値を、Mobile Link インストール環境全体で確実にユニークにする必要があります。このアプリケーションで使われるプライマリキープールメソッドは、プライマリキーをユニークにする方法の 1 つです。

このセクションの内容:

CustDB データベースのリストア方法 [82 ページ]
サンプルをリストアするには、*%SQLANYSAMP17%*¥UltraLite¥CustDB ディレクトリから次のコマンドを実行します。

関連情報

CustDB 統合データベースの設定 [67 ページ]

1.3.1 CustDB ファイル

CustDB サンプルアプリケーションのコードとデータベースを構成する部分には、次のものがあります。

- サンプル SQL スクリプト。*%SQLANYSAMP17%*¥MobiLink¥CustDB にあります。
- アプリケーションコード。*%SQLANYSAMP17%*¥UltraLite¥CustDB にあります。
- プラットフォーム固有のユーザインタフェースのコード。*%SQLANYSAMP17%*¥UltraLite¥CustDB の各オペレーティングシステムの名前が付いたサブディレクトリにあります。

このセクションの内容:

CustDB 統合データベースの設定 [67 ページ]
Mobile Link でサポートされている任意の統合データベースを CustDB 統合データベースとして使用できます。

Ultra Lite のリモートデータベースの設定 [71 ページ]
以下の例では、CustDB のリモートデータベースを作成します。CustDB リモートデータベースは、Ultra Light データベースでなければなりません。

1.3.1.1 CustDB 統合データベースの設定

Mobile Link でサポートされている任意の統合データベースを CustDB 統合データベースとして使用できます。

SQL Anywhere 17 CustDB

SQL Anywhere 統合データベースは *%SQLANYSAMP17%*¥UltraLite¥CustDB¥custdb.db にあります。SQL Anywhere 17 CustDB という DNS はインストール環境に含まれています。

データベースは、*%SQLANYSAMP17%*¥UltraLite¥CustDB¥makedbs.cmd ファイルを使用して再構築できます。

CustDB サンプルの作りを詳しく調べるには、*%SQLANYSAMP17%*¥MobiLink¥CustDB¥custdb.sql ファイルを参照してください。

その他の RDBMS 用の CustDB

次の SQL スクリプトを使用すると、サポートされている RDBMS のいずれかで、CustDB 統合データベースを構築できます。これらのスクリプトは、*%SQLANYSAMP17%*¥MobiLink¥CustDB にあります。

RDBMS	CustDB 設定スクリプト
Adaptive Server Enterprise	custase.sql
Microsoft SQL Server	custmss.sql
Oracle	custora.sql
DB2 LUW	custdb2.sql
MySQL	custmys.sql

このセクションの内容:

1.3.1.1.1 CustDB を統合データベースとして構築 (Adaptive Server Enterprise、MySQL、Oracle、または Microsoft SQL Server の場合)

次の手順を実行すると、サポートされている任意の RDBMS 用の CustDB 統合データベースが作成されます。

前提条件

サポートされている RDBMS のいずれであっても、CustDB 統合データベースを構築する場合に使用される SQL スクリプトへのアクセス権が必要です。SQL スクリプトは、*%SQLANYSAMP17%*¥MobiLink¥CustDB にあります。

手順

1. 使用している RDBMS でデータベースを作成します。
2. 以下の SQL スクリプトのいずれかを実行して Mobile Link システムオブジェクトを追加します。これらのスクリプトは、SQL Anywhere17 インストール環境の MobiLink¥setup サブディレクトリにあります。

- Adaptive Server Enterprise 統合データベースの場合は、`syncase.sql` を実行します。
- MySQL 統合データベースの場合は、`syncmys.sql` を実行します。
- Oracle 統合データベースの場合は、`syncora.sql` を実行します。
- Microsoft SQL Server 統合データベースの場合は、`syncmss.sql` を実行します。

3. 以下の SQL スクリプトのいずれかを実行して、サンプルユーザテーブル、ストアドプロシージャ、Mobile Link 同期スクリプトを CustDB データベースに追加します。これらは、*%SQLANYSAMP17%*`¥MobiLink¥CustDB` にあります。
   - Adaptive Server Enterprise 統合データベースの場合は、`custase.sql` を実行します。
   - MySQL 統合データベースの場合は、`custmys.sql` を実行します。
   - Oracle 統合データベースの場合は、`custora.sql` を実行します。
   - Microsoft SQL Server 統合データベースの場合は、`custmss.sql` を実行します。
4. クライアントコンピュータ上で、データベースを参照する CustDB という ODBC データソースを作成します。
   a. スタート ▸ プログラム ▸ *SQL Anywhere17* ▸ 管理ツール ▸ *ODBC* データソースアドミニストレータ (32 ビットまたは 64 ビット) を選択します。
   b. 追加をクリックします。
   c. リストから適切なドライバを選択します。

      終了をクリックします。
   d. この ODBC データソースに CustDB という名前を付けます。
   e. ログインタブをクリックします。データベースのユーザ *ID* とパスワードを入力します。
5. *OK* をクリックし、もう一度 *OK* をクリックします。

結果

選択した RDBMS 用に CustDB 統合データベースが作成されます。

1.3.1.1.2 CustDB を統合データベースとして構築 (DB2 LUW の場合)

この手順を使用して、DB2 LUW 統合データベースを設定します。

手順

1. DB2 LUW サーバ上に **`CustDB`** という統合データベースを作成します。
2. デフォルトのテーブル領域 (通常は USERSPACE1) が 8 KB ページを使用することを確認します。

   デフォルトのテーブル領域が 8 KB ページを使用しない場合は、次の手順を行います。

   a.
   b. 8 KB ページの新しいテーブル領域とテンポラリテーブル領域を作成します。8 KB ページのバッファプールが少なくとも 1 つはあることを確認します。ない場合は、8 KB ページのバッファプールを作成してください。

詳細については、DB2 LUW のマニュアルを参照してください。

3. `MobiLink¥setup¥syncdb2.sql` ファイルを使用して、Mobile Link システムオブジェクトを DB2 LUW 統合データベースに追加します。
   a. `syncdb2.sql` ファイルの先頭にある接続コマンドを変更します。`DB2Database` を、お使いのデータベース名 (またはそのエイリアス) に置き換えます。この例では、このデータベースを CustDB と呼びます。以下に示すように、DB2 のユーザ名とパスワードを追加することもできます。

      ```
      connect to CustDB user userid using password ~
      ```

   b. 8 KB ページのバッファプールが少なくとも 1 つはあることを確認します。なければ、サーバまたはクライアントコンピュータで、DB2 LUW コマンドウィンドウを開きます。`syncdb2.sql` を実行します。

      ```
      db2cmd db2 -c -ec -td~ +s -v -f syncdb2.sql
      ```

4. データテーブル、ストアドプロシージャ、Mobile Link 同期スクリプトの CustDB データベースへの追加を入力します。
   a. 必要に応じて、custdb2.sql の接続コマンドを変更します。たとえば、以下に示すように、ユーザ名とパスワードを追加できます。`userid` と `password` を、使用するユーザ名とパスワードに置き換えてください。

      ```
      connect to CustDB user userid using password
      ```

   b. サーバまたはクライアントコンピュータで、DB2 コマンドウィンドウを開きます。
   c. 次のコマンドを入力して custdb2.sql を実行します。

      ```
      db2cmd db2 -c -ec -td~ +s -v -f custdb2.sql
      ```

   d. 処理が完了したら、次のコマンドを入力してコマンドウィンドウを閉じます。

      ```
      exit
      ```

5. DB2 LUW クライアント上で、DB2 LUW データベースを参照する CustDB という ODBC データソースを作成します。
   a. ODBC データソースアドミニストレータを起動します。

      スタート > **プログラム** > *SQL Anywhere17* > **管理ツール** > *ODBC* **データソースアドミニストレータ** を選択します。

      ODBC データソースアドミニストレータが表示されます。
   b. **ユーザ** *DSN* タブで、**追加**をクリックします。
   c. **データソースの新規作成**ウィンドウで、DB2 LUW データベース用の ODBC ドライバを選択します。たとえば、IBM DB2 UDB ODBC ドライバを選択します。**終了**をクリックします。

      ODBC ドライバの設定方法については、次のマニュアルを参照してください。

      - DB2 LUW のマニュアル
      - Mobile Link の推奨 ODBC ドライバ

結果

DB2 LUW 統合データベースが設定されます。

関連情報

Mobile Link の推奨 ODBC ドライバ

1.3.1.2 Ultra Lite のリモートデータベースの設定

以下の例では、CustDB のリモートデータベースを作成します。CustDB リモートデータベースは、Ultra Light データベースでなければなりません。

リモートデータベースのアプリケーション論理は *%SQLANYSAMP17%*¥UltraLite¥CustDB にあります。これには、以下のファイルが含まれています。

C++ API の論理

ファイル custdbcpp.cpp には、C++ API の論理が格納されています。

ユーザインタフェース機能

この機能は、Samples¥UltraLite¥CustDB のプラットフォーム固有のサブディレクトリに、別々に格納されています。

Ultra Light が実行されているリモートデバイスにサンプルアプリケーションをインストールするには、次の手順を実行します。

1. 統合データベースを起動します。
2. Mobile Link サーバを起動します。
3. サンプルアプリケーションをクライアントデバイスにインストールして、起動します。
4. サンプルアプリケーションを同期します。

例

次の例では、DB2 統合データベースを対象として動作している Windows デスクトップに CustDB サンプルをインストールする方法を示します。

1. 次のようにして、統合データベースが稼働していることを確認します。
   DB2 LUW データベースの DB2 コマンドウィンドウを開きます。次のコマンドを入力します。ここでは、userid と password は DB2 LUW データベースに接続するためのユーザ ID とパスワードです。

   ```
   db2 connect to CustDB user userid using password
   ```

2. Mobile Link サーバを起動します。
   DB2 LUW データベースと同期できるようにするため、コマンドプロンプトで次のコマンドを実行します。

   ```
   mlsrv17 -c "DSN=SQL Anywhere 17 CustDB;uid=ml_server;pwd=sql" -zp
   ```

3. CustDB サンプルアプリケーションを起動します。
   1. スタート > プログラム > *SQL Anywhere17* > *Mobile Link* > 同期サーバのサンプル をクリックします。
   2. Employee ID に値 50 を入力し、*OK* をクリックします。
      アプリケーションは自動的に同期を実行し、一連の顧客、製品、注文情報が CustDB 統合データベースからアプリケーションにダウンロードされます。
4. リモートアプリケーションを統合データベースと同期します。
   ファイル同期を選択します。
   この手順は、データベースを変更したときのみ行ってください。

1.3.2 CustDB データベース内のテーブル

CustDB データベースのテーブル定義は、*%SQLANYSAMP17%*¥MobiLink¥CustDB にあるプラットフォーム固有のファイル内に格納されています。

次の 5 つのテーブルは統合データベースとリモートデータベースの両方にありますが、その定義は両者で少し異なります。

ULCustomer

ULCustomer テーブルには、顧客リストがあります。

リモートデータベースの ULCustomer には、次のカラムがあります。

cust_id

顧客を識別するユニークな整数を保持するプライマリキーカラム。

cust_name

顧客の名前を保持する 30 バイトの文字列。

統合データベースの ULCustomer には、次のカラムもあります。

last_modified

ローが最後に変更されたときのタイムスタンプ。このカラムは、タイムスタンプベースの同期に使用されます。

ULProduct

ULProduct テーブルには、製品リストがあります。

リモートデータベースと統合データベースの両方の ULProduct に、次のカラムがあります。

prod_id

製品を識別するユニークな整数を保持するプライマリキーカラム。

price

単価を識別する整数。

prod_name

製品の名前を保持する 30 バイトの文字列。

ULOrder

ULOrder テーブルには受注リストがあります。注文した顧客の情報、受注した従業員、注文された製品についての詳細が含まれています。

リモートデータベースの ULOrder には、次のカラムがあります。

order_id

注文を識別するユニークな整数を保持するプライマリキーカラム。

cust_id

ULCustomer を参照する外部キーカラム。

prod_id

ULProduct を参照する外部キーカラム。

emp_id

ULEmployee を参照する外部キーカラム。

disc

注文に適用される値引きを保持する整数。

quant

注文された製品の数を保持する整数。

notes

注文に関する注記を保持する 50 バイトの文字列。

status

注文のステータスが記述された 20 バイトの文字列。

統合データベースの ULOrder には、次のカラムもあります。

last_modified

ローが最後に変更されたときのタイムスタンプ。このカラムは、タイムスタンプベースの同期に使用されます。

ULOrderIDPool

ULOrderIDPool テーブルは、ULOrder のプライマリキープールです。

リモートデータベースの ULOrderIDPool には、次のカラムがあります。

pool_order_id

注文 ID を識別するユニークな整数を保持するプライマリキーカラム。

統合データベースの ULOrderIDPool には、次のカラムもあります。

pool_emp_id

注文 ID が割り当てられたリモートデータベースの所有者の従業員 ID を保持する整数カラム。

last_modified

ローが最後に変更されたときのタイムスタンプ。

ULCustomerIDPool

ULCustomerIDPool テーブルは、ULCustomer のプライマリキープールです。

リモートデータベースの ULCustomerIDPool には、次のカラムがあります。

pool_cust_id

顧客 ID を識別するユニークな整数を保持するプライマリキーカラム。

統合データベースの ULCustomerIDPool には、次のカラムもあります。

pool_emp_id

リモートデータベースで生成された新しい従業員に使用される従業員 ID を保持する整数カラム。

last_modified

ローが最後に変更されたときのタイムスタンプ。

以下のテーブルは、統合データベースにのみ存在します。

ULIdentifyEmployee_nosync

ULIdentifyEmployee_nosync テーブルは、統合データベースとリモートデータベースの両方に存在します。これには、次の 1 つのカラムがあります。

emp_id

このプライマリキーカラムには、リモートデータベースの従業員 ID を示す整数が保持されています。

ULEmployee

ULEmployee テーブルは、統合データベースにのみ存在します。これには、営業担当者リストが格納されています。

ULEmployee には、次のカラムがあります。

emp_id

従業員を識別するユニークな整数を保持するプライマリキーカラム。

emp_name

従業員の名前を保持する 30 バイトの文字列。

ULEmpCust

ULEmpCust テーブルは、どの顧客の注文をダウンロードするかを制御します。従業員が新しい顧客の注文を必要とする場合は、従業員 ID と顧客 ID を挿入すると、その顧客の注文がダウンロードされます。

emp_id

ULEmployee.emp_id の外部キー。

cust_id

ULCustomer.cust_id の外部キー。プライマリキーは、emp_id と cust_id で構成されています。

action

従業員のレコードをリモートデータベースから削除するかどうかを決定するのに使用される文字。従業員が顧客の注文を必要としなくなった場合は、D (削除) に設定します。注文が必要な場合、action は NULL に設定してください。

この場合は、ULOrder テーブルから削除するローを統合データベースが識別できるようにするため、論理削除を使用する必要があります。削除情報がダウンロードされると、action が D に設定されたその従業員のすべてのレコードは統合データベースからも削除できます。

last_modified

ローが最後に変更されたときのタイムスタンプ。このカラムは、タイムスタンプベースの同期に使用されます。

ULOldOrder と ULNewOrder

これらのテーブルは、統合データベースにのみ存在します。また、競合を解決するために使用され、ULOrder と同じカラムが含まれています。SQL Anywhere と Microsoft SQL Server では、これらはテンポラリテーブルです。Adaptive Server Enterprise の場合、これらのテーブルは通常のテーブルであり、@@spid です。DB2 LUW と Oracle にはテンポラリテーブルがないため、同期ユーザに属するローを Mobile Link が識別できるようになっている必要があります。これらのテーブルはベーステーブルであるため、5 人のユーザが同期している場合は、これらのテーブルのローを各ユーザが同時に保持することがあります。

1.3.3 CustDB サンプル内のユーザ

CustDB サンプルのユーザには、営業担当者とモバイル管理者の 2 つのタイプがあります。

相違点は次のとおりです。

営業担当者

営業担当者に関連付けられたリモートデータベースは、ユーザ ID 50、51、52 で識別されます。営業担当者は、次のタスクを実行できます。

- 顧客と製品のリスト表示
- 新規顧客の追加
- 注文の追加や削除
- 未処理の注文リストのスクロール
- 変更内容に関する統合データベースとの同期

モバイル管理者

モバイル管理者に関連付けられたリモートデータベースは、ユーザ ID 53 で識別されます。モバイル管理者は、営業担当者と同じタスクを実行できます。このほか、モバイル管理者は次のタスクを実行できます。

- 注文の承認や拒否

1.3.4 同期ロジックのソースコード

SQL Central を使用すると、統合データベース内の同期スクリプトを調べることができます。

スクリプトタイプとイベント

`custdb.sql` ファイルは、ml_add_connection_script または ml_add_table_script を呼び出して、各同期スクリプトを統合データベースに追加します。

例

`custdb.sql` の次の行は、ULProduct テーブル用のテーブルレベルのスクリプトを追加します。このスクリプトは、download_cursor イベントで実行されます。スクリプトには、SELECT 文が 1 つあります。

```
call ml_server.ml_add_table_script(
'CustDB 17.0',
'ULProduct', 'download_cursor',
'SELECT prod_id, price, prod_name FROM ULProduct' )
go
```

1.3.5 CustDB サンプルの注文の同期

ULOrder テーブルのビジネスルールは、次のとおりです。

- 注文は、承認されていないか、ステータスが NULL である場合にかぎりダウンロードされます。
- 注文は、統合データベースでもリモートデータベースでも修正できます。
- 各リモートデータベースには、従業員に対応する注文のみが保持されます。

ダウンロード

統合データベースでは、注文を挿入、削除、更新できます。これらの操作に対応するスクリプトは、次のとおりです。

download_cursor

download_cursor スクリプトの最初のパラメータは、最終ダウンロードタイムスタンプです。これは、最後の同期以後にリモートデータベースまたは統合データベースのいずれかで修正されたローのみをダウンロードするために使用されます。2 番目のパラメータは従業員 ID です。ダウンロードするローを決定するために使用されます。

CustDB の download_cursor スクリプトを次に示します。

```
CALL ULOrderDownload( {ml s.last_table_download}, {ml s.username} )
```

CustDB の ULOrderDownload プロシージャを次に示します。

```
CREATE PROCEDURE ULOrderDownload ( IN LastDownload timestamp, IN EmployeeID
integer )
BEGIN
  SELECT o.order_id, o.cust_id, o.prod_id, o.emp_id, o.disc, o.quant, o.notes,
o.status
    FROM ULOrder o, ULEmpCust ec
    WHERE o.cust_id = ec.cust_id
    AND ec.emp_id = EmployeeID
    AND ( o.last_modified >= LastDownload
    OR ec.last_modified >= LastDownload)
    AND ( o.status IS NULL  OR  o.status != 'Approved' )
    AND ( ec.action IS NULL )
END
```

download_delete_cursor

CustDB の download_delete_cursor スクリプトを次に示します。

```
SELECT o.order_id
    FROM ULOrder o, dba.ULEmpCust ec
  WHERE o.cust_id = ec.cust_id
    AND ( ( o.status = ''Approved'' AND o.last_modified >= {ml
s.last_table_download} )
    OR ( ec.action = ''D''  )  )
    AND ec.emp_id = {ml s.username}
```

アップロード

リモートデータベースでは、注文を挿入、削除、更新できます。これらの操作に対応するスクリプトは、次のとおりです。

upload_insert

CustDB の upload_insert スクリプトを次に示します。

```
INSERT INTO ULOrder ( order_id, cust_id, prod_id, emp_id, disc, quant, notes,
status )
   VALUES( {ml r.order_id, r.cust_id, r.prod_id, r.emp_id, r.disc, r.quant,
r.notes, r.status } )
```

upload_update

CustDB の upload_update スクリプトを次に示します。

```
UPDATE ULOrder
  SET cust_id = {ml r.cust_id},
      prod_id = {ml r.prod_id},
      emp_id = {ml r.emp_id},
      disc = {ml r.disc},
      quant = {ml r.quant},
      notes = {ml r.notes},
      status = {ml r.status}
    WHERE order_id = {ml r.order_id}
```

upload_delete

CustDB の upload_delete スクリプトを次に示します。

```
DELETE FROM ULOrder WHERE order_id = {ml r.order_id}
```

upload_fetch

CustDB の upload_fetch スクリプトを次に示します。

```
SELECT order_id, cust_id, prod_id, emp_id, disc, quant, notes, status
   FROM ULOrder WHERE order_id = {ml r.order_id}
```

upload_old_row_insert

CustDB の upload_old_row_insert スクリプトを次に示します。

```
INSERT INTO ULOldOrder ( order_id, cust_id, prod_id, emp_id, disc, quant, notes,
status )
   VALUES( {ml r.order_id, r.cust_id, r.prod_id, r.emp_id, r.disc, r.quant,
r.notes, r.status } )
```

upload_new_row_insert

CustDB の upload_new_row_insert スクリプトを次に示します。

```
INSERT INTO ULNewOrder ( order_id, cust_id, prod_id, emp_id, disc, quant, notes,
status )
   VALUES( {ml r.order_id, r.cust_id, r.prod_id, r.emp_id, r.disc, r.quant,
r.notes, r.status } )
```

競合解決

resolve_conflict

CustDB の resolve_conflict スクリプトを次に示します。

```
CALL ULResolveOrderConflict
```

CustDB の ULResolveOrderConflict プロシージャを次に示します。

```
CREATE PROCEDURE ULResolveOrderConflict()
BEGIN
  -- approval overrides denial
  IF 'Approved' = (SELECT status FROM ULNewOrder) THEN
    UPDATE ULOrder o
    SET o.status = n.status, o.notes = n.notes
    FROM ULNewOrder n
    WHERE o.order_id = n.order_id;
  END IF;
  DELETE FROM ULOldOrder;
  DELETE FROM ULNewOrder;
END
```

1.3.6 CustDB サンプルの顧客の同期

顧客を規定するビジネスルールは、次のとおりです。顧客情報は、統合データベースでもリモートデータベースでも修正できます。リモートデータベースと統合データベースの両方に、完全な顧客リストがあります。

ダウンロード

統合データベースでは、顧客情報を挿入または更新できます。これらの操作に対応するスクリプトは、次のとおりです。

download_cursor

次の download_cursor スクリプトは、ユーザが最後に情報をダウンロードした後で情報が変更された顧客をすべてダウンロードします。

```
SELECT cust_id, cust_name FROM ULCustomer WHERE last_modified >= {ml
s.last_table_download}
```

アップロード

リモートデータベース側で顧客情報を挿入、更新、または削除できます。これらの操作に対応するスクリプトは、次のとおりです。

upload_insert

CustDB の upload_insert スクリプトを次に示します。

```
INSERT INTO ULCustomer( cust_id, cust_name )
   VALUES( {ml r.cust_id, r.cust_name } )
```

upload_update

CustDB の upload_update スクリプトを次に示します。

```
UPDATE ULCustomer SET cust_name = {ml r.cust_name}
   WHERE cust_id = {ml r.cust_id}
```

このテーブルに対する競合検出は実行されません。

upload_delete

CustDB の upload_delete スクリプトを次に示します。

```
DELETE FROM ULCustomer WHERE cust_id = {ml r.cust_id}
```

1.3.7 CustDB サンプルの製品の同期

ULProduct に関するすべてのローがダウンロードされます。これはスナップショット同期と呼ばれます。

ULProduct テーブルのビジネスルールは、次のとおりです。

- 統合データベースでは、製品の修正のみが可能です。
- 各リモートデータベースには、すべての製品が含まれています。

ダウンロード

統合データベースでは、製品情報を挿入、削除、更新できます。これらの操作に対応するスクリプトは、次のとおりです。

download_cursor

次の download_cursor スクリプトは、同期が行われるたびに ULProduct テーブルのすべてのローとカラムをダウンロードします。

```
SELECT prod_id, price, prod_name FROM ULProduct
```

1.3.8 顧客と注文のプライマリキープールの管理

CustDB サンプルデータベースでは、ULCustomer テーブルと ULOrder テーブルのユニークなプライマリキーを管理するために、プライマリキープールが使用されます。プライマリキープールは、ULCustomerIDPool テーブルと ULOrderIDPool テーブルです。

このセクションの内容:

1.3.8.1 ULCustomerIDPool

以下のスクリプトは、ULCustomerIDPool テーブルで定義されています。

ダウンロード

begin_download

CustDB の begin_download スクリプトを次に示します。

```
CALL ULCustomerIDPool_maintain( {ml s.username} )
```

CustDB の ULCustomerIDPool_maintain プロシージャを次に示します。

```
CREATE PROCEDURE ULCustomerIDPool_maintain ( IN syncuser_id INTEGER )
BEGIN
  DECLARE pool_count INTEGER;
  -- Determine how many ids to add to the pool
  SELECT COUNT(*) INTO pool_count
    FROM ULCustomerIDPool
    WHERE pool_emp_id = syncuser_id;
  -- Top up the pool with new ids
  WHILE pool_count < 20 LOOP
    INSERT INTO ULCustomerIDPool ( pool_emp_id )
      VALUES ( syncuser_id );
    SET pool_count = pool_count + 1;
  END LOOP;
END
```

download_cursor

```
SELECT pool_cust_id FROM ULCustomerIDPool
   WHERE last_modified >= {ml s.last_table_download}
     AND pool_emp_id = {ml s.username}
```

アップロード

upload_delete

CustDB の upload_delete スクリプトを次に示します。

```
DELETE FROM ULCustomerIDPool
   WHERE pool_cust_id = {ml r.pool_cust_id}
```

1.3.8.2 ULOrderIDPool

以下のスクリプトは、ULOrderIDPool テーブルで定義されています。

ダウンロード

begin_download

CustDB の begin_download スクリプトを次に示します。

```
CALL ULOrderIDPool_maintain( {ml s.username} )
```

CustDB の ULOrderIDPool_maintain プロシージャを次に示します。

```
ALTER PROCEDURE ULOrderIDPool_maintain ( IN syncuser_id INTEGER )
BEGIN
  DECLARE pool_count INTEGER;
  -- Determine how many ids to add to the pool
  SELECT COUNT(*) INTO pool_count
    FROM ULOrderIDPool
    WHERE pool_emp_id = syncuser_id;
  -- Top up the pool with new ids
  WHILE pool_count < 20 LOOP
    INSERT INTO ULOrderIDPool ( pool_emp_id )
      VALUES ( syncuser_id );
    SET pool_count = pool_count + 1;
  END LOOP;
END
```

download_cursor

CustDB の download_cursor スクリプトを次に示します。

```
SELECT pool_order_id FROM ULOrderIDPool
  WHERE last_modified >= {ml s.last_table_download}
    AND pool_emp_id = {ml s.username}
```

アップロード

upload_delete

CustDB の upload_delete スクリプトを次に示します。

```
DELETE FROM ULOrderIDPool
   WHERE pool_order_id = {ml r.pool_order_id}
```

1.3.9 CustDB データベースのリストア方法

サンプルをリストアするには、*%SQLANYSAMP17%*¥UltraLite¥CustDB ディレクトリから次のコマンドを実行します。

```
makedbs
```

1.4 Mobile Link Contact サンプル

Contact サンプルは、Mobile Link 開発者にとって貴重なリソースです。このサンプルを使用して、Mobile Link アプリケーションの開発時に必要なさまざまな方法の実装例を紹介します。

Contact サンプルアプリケーションは、1 つの SQL Anywhere 統合データベースと、2 つの SQL Anywhere リモートデータベースから構成されています。このサンプルでは、同期の一般的な方法をいくつか説明します。利点を最大限に活用するために、サンプルアプリケーションのソースコードを読んで理解してください。

統合データベースは SQL Anywhere データベースですが、同期スクリプトは他のデータベース管理システムの最小限の変更を処理する SQL 文で構成されています。

Contact サンプルは *%SQLANYSAMP17%*¥MobiLink¥Contact にあります。

同期の設計

Contact サンプルアプリケーションの同期の設計では、以下の機能を使用します。

カラムのサブセット

統合データベース上の Customer、Product、SalesRep、Contact の各テーブルのカラムのサブセットが、リモートデータベースで共有されます。

ローのサブセット

統合データベース上の SalesRep テーブルのローのうち 1 つのみについて、すべてのカラムが各リモートデータベースで共有されます。

タイムスタンプベースの同期

最後のデバイス同期以降に実行された統合データベースに対する変更を識別する方法。Customer、Contact、Product の各テーブルは、タイムスタンプベースの方法を使用して同期されます。

権限

次のロールおよび権限が必要です。

- SYS_REPLICATION_ADMIN_ROLE システムロール
- SYS_RUN_REPLICATION_ROLE システムロール
- CREATE ANY TRIGGER システム権限

このセクションの内容:

Customer テーブルの同期スクリプトは、タイムスタンプベースの同期とローの分割の例を示しています。これらの方法では、同期中に転送されるデータの量が最小限になり、テーブルデータの整合性が保持されます。

Contact サンプルの顧客窓口の同期 [92 ページ]
Contact テーブルには、顧客の会社の社員名、顧客を参照するための外部キー、窓口を識別するユニークな整数が含まれています。また、last_modified タイムスタンプと、窓口がアクティブであるかどうかを示すマーカもあります。

Contact サンプルの製品の同期 [94 ページ]
Product テーブル用のスクリプトは、競合の検出と解決の例を示しています。

Contact サンプルの統計とエラーのモニタリング [96 ページ]
Contact サンプルには、単純なエラーレポートスクリプトとモニタリングスクリプトがいくつか用意されています。これらのスクリプトを作成するための SQL 文は、`MobiLink¥Contact¥mlmaint.sql` ファイルにあります。

1.4.1 Contact サンプルの設定

Contact サンプルデータベースを構築できるように、Windows バッチファイル `build.bat` が用意されています。UNIX システムでは `build.sh` です。

コンテキスト

このバッチファイルの内容を調べることができます。このバッチファイルによって次のアクションが実行されます。

- 統合データベースと 2 つのリモートデータベース用に、ODBC データソース定義が作成されます。
- `consol.db` という統合データベースが作成され、Mobile Link システムテーブル、データベーススキーマ、データ、同期スクリプト、Mobile Link ユーザ名がデータベースにロードされます。
- `remote.db` という名前の 2 つのリモートデータベースが、サブディレクトリ `remote_1` と `remote_2` に作成されます。両方のデータベースに共通の情報がロードされ、カスタマイズ内容が適用されます。カスタマイズ内容には、グローバルデータベース識別子、Mobile Link ユーザ名、2 つのパブリケーションに対するサブスクリプションが含まれます。

手順

1. コマンドプロンプトで、*%SQLANYSAMP17%*`¥MobiLink¥Contact` に移動します。
2. `build.bat` (Windows) または `build.sh` (UNIX) を実行します。

結果

Contact サンプルデータベースが構築されます。

このセクションの内容:

Contact サンプルの実行 [85 ページ]
Contact サンプルには最初の同期を実行するバッチファイルが含まれており、Mobile Link サーバと dbmlsync のコマンドラインが例示されています。

1.4.1.1 Contact サンプルの実行

Contact サンプルには最初の同期を実行するバッチファイルが含まれており、Mobile Link サーバと dbmlsync のコマンドラインが例示されています。

前提条件

Contact サンプルデータベースを構築するための `build.bat` が実行されていることを確認します。

コンテキスト

%SQLANYSAMP17%`¥MobiLink¥Contact` に次のバッチファイルがあり、その内容はテキストエディタで確認できます。

- `step1.bat`
- `step2.bat`
- `step3.bat`

手順

1. Mobile Link サーバを起動します。
   a. コマンドプロンプトで、*%SQLANYSAMP17%*`¥MobiLink¥Contact` に移動します。
   b. 次のコマンドを実行します。

      ```
      step1 -pwd=sql
      ```

      このコマンドは、Mobile Link サーバを冗長モードで起動するバッチファイルを実行します。このモードは、開発中やトラブルシューティング中には便利ですが、パフォーマンスが低下するため、通常の運用環境では使用しません。
2. 両方のリモートデータベースを同期します。
   a. コマンドプロンプトで、*%SQLANYSAMP17%*`¥MobiLink¥Contact` に移動します。
   b. 次のコマンドを実行します。

      ```
      step2
      ```

      これは、両方のリモートデータベースを同期するバッチファイルです。
3. Mobile Link サーバを停止します。

a. コマンドプロンプトで、*%SQLANYSAMP17%*¥MobiLink¥Contact に移動します。
b. 次のコマンドを実行します。

```
step3
```

これは、Mobile Link サーバを停止させるバッチファイルです。

結果

Contact サンプルが実行され、リモートデータベースと統合データベースが同期されます。

次のステップ

Contact サンプルで同期の動作方法を調べるには、Interactive SQL を使用してリモートデータベースと統合データベースでデータを修正し、バッチファイルを使用して同期を行います。

1.4.2 Contact データベース内のテーブル

Contact データベースのテーブル定義は、MobiLink¥Contact¥build_consol.sql ファイルおよび MobiLink¥Contact¥build_remote.sql ファイルにあります。これらのファイルはすべてサンプルのフォルダにあります。

次の 3 つのテーブルは統合データベースとリモートデータベースの両方にありますが、その定義は両者で少し異なります。

SalesRep

SalesRep テーブルには、営業担当者ごとに 1 つのローがあります。リモートデータベースは、それぞれ 1 人の営業担当者が所持します。

各リモートデータベースの SalesRep には、次のカラムがあります。

rep_id

営業担当者の識別番号が格納されるプライマリキーカラム。

name

担当者の名前。

統合データベース側には、この他に営業担当者の Mobile Link ユーザ名を保持する ml_username カラムもあります。

Customer

このテーブルには、顧客ごとに 1 つのローがあります。顧客は、それぞれ 1 人の営業担当者が担当する会社です。SalesRep テーブルと Customer テーブルは 1 対多の関係になっています。

各リモートデータベースの Customer には、次のカラムがあります。

cust_id

顧客の識別番号を保持するプライマリキーカラム。

name

顧客名。これは会社名です。

rep_id

SalesRep テーブルを参照する外部キーカラム。顧客に割り当てられた営業担当者を識別します。

統合データベースには、この他に last_modified カラムと active カラムがあります。

last_modified

ローを最後に変更した時刻。このカラムは、タイムスタンプベースの同期に使用されます。

active

顧客が現在アクティブであるか (1)、またはこの顧客との取引がなくなったか (0) を示すビットカラム。このカラムに非アクティブ (0) のマークが付いている場合は、この顧客に対応するすべてのローがリモートデータベースから削除されます。

Contact

このテーブルには、窓口ごとに 1 つのローがあります。窓口担当者は、顧客の会社の従業員です。Customer テーブルと Contact テーブルは 1 対多の関係になっています。

各リモートデータベースの Contact には、次のカラムがあります。

contact_id

窓口担当者の識別番号を保持するプライマリキーカラム。

name

各窓口担当者の氏名。

cust_id

窓口担当者が所属する顧客の識別子。

統合データベースでは、このテーブルに次のカラムもあります。

last_modified

ローを最後に変更した時刻。このカラムは、タイムスタンプベースの同期に使用されます。

active

窓口が現在アクティブであるか (1)、またはこの窓口との取引がなくなったか (0) を示すビットカラム。このカラムに非アクティブ (0) のマークが付いている場合は、この窓口に対応するローがリモートデータベースから削除されます。

Product

Product テーブルには、会社で販売される製品ごとに 1 つのローがあります。Product テーブルは別のパブリケーションに保持されるため、リモートデータベースはこのテーブルを別途同期できます。

各リモートデータベースの Product には、次のカラムがあります。

id

製品を識別するユニークな数値を保持するプライマリキーカラム。

name

品目の名前。

size

品目のサイズ。

quantity

品目の在庫数量。営業担当者が注文を受け取った時点で、このカラムは更新されます。

unit_price

製品の単価。

統合データベースの Product テーブルには、次のカラムもあります。

supplier

製品を製造している会社。

last_modified

ローを最後に変更した時刻。このカラムは、タイムスタンプベースの同期に使用されます。

active

製品が現在アクティブ (1) であるかどうかを示すビットカラム。このカラムに非アクティブ (0) のマークが付いている場合は、この製品に対応するローがリモートデータベースから削除されます。

統合データベースには、これらのテーブルに加えてテーブルセットが作成されます。これには、競合解決中に使用されるテンポラリテーブル product_conflict と、ユーザ mlmaint が所有する Mobile Link アクティビティをモニタリングするためのテーブルセットが含まれます。Mobile Link モニタリングテーブルを作成するためのスクリプトは、*%SQLANYSAMP17%*`¥MobiLink¥Contact¥mlmaint.sql` ファイルにあります。

1.4.3 Contact サンプル内のユーザ

Contact サンプルには、複数のデータベースユーザ ID と Mobile Link ユーザ名が含まれています。

データベースユーザ ID

2 つのリモートデータベースは、営業担当者 Samuel Singer (rep_id 856) と Pamela Savarino (rep_id 949) に割り当てられます。

この 2 人のユーザはどちらも、それぞれのリモートデータベースへの接続時に、デフォルトの SQL Anywhere ユーザ ID **dba** とパスワード **SQL** を使用します。

また、各リモートデータベースには、ユーザ ID **sync_user** (パスワードは **sync_user**) もあります。このユーザ ID は、dbmlsync コマンドラインでのみ使用します。**sync_user** 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。すなわち、dbmlsync からの接続時にはあらゆる操作を実行できますが、他のアプリケーションからの接続時には何の権限もありません。そのため、**sync_user** の ID およびパスワードを使用しても問題にはなりません。

統合データベースには、**mlmaint** というユーザが存在します。このユーザは Mobile Link 同期統計とエラーのモニタリングに使用されるテーブルの所有者です。**mlmaint** ユーザは接続権限を持っていません。テーブルを個々のユーザ ID に割り当てるには、スキーマ内でオブジェクトを他のオブジェクトから分離するだけであり、SQL Central や他のユーティリティで管理しやすくなっています。

Mobile Link ユーザ名

Mobile Link ユーザ名は、データベースユーザ ID とは異なります。各リモートデバイスには、データベースへの接続時に使用するユーザ ID の他に、Mobile Link ユーザ名があります。Samuel Singer の Mobile Link ユーザ名は SSinger です。Pamela Savarino の Mobile Link ユーザ名は PSavarino です。Mobile Link ユーザ名は、次のロケーションで格納または使用されます。

- リモートデータベース。Mobile Link ユーザ名が、CREATE SYNCHRONIZATION USER 文を使用して追加されます。
- 統合データベース。Mobile Link ユーザ名とパスワードが、mluser ユーティリティを使用して追加されます。
- `MobiLink¥Contact¥step2.bat` 内の dbmlsync コマンドライン。同期時に、接続ユーザの Mobile Link パスワードが指定されます。
- Mobile Link サーバ。同期時、Mobile Link ユーザ名がパラメータとして多数のスクリプトに指定されます。
- 統合データベース側の SalesRep テーブル。ml_username カラムがあります。同期スクリプトは、このカラムの値と Mobile Link ユーザ名パラメータを比較します。

1.4.4 Contact サンプルの営業担当者の同期

SalesRep テーブルの同期スクリプトは、スナップショットを使った同期の例を示しています。営業担当者の情報は、変更されたかどうかに関係なくダウンロードされます。

ビジネスルール

SalesRep テーブルのビジネスルールは、次のとおりです。

- リモートデータベース側ではテーブルを修正しません。
- 営業担当者の Mobile Link ユーザ名と rep_id 値を変更しません。
- 各リモートデータベースの SalesRep テーブルには、リモートデータベース所有者の Mobile Link ユーザ名に対応するローが 1 つだけ存在します。

ダウンロード

download_cursor

各リモートデータベースの SalesRep テーブルには、ローが 1 つだけ存在します。単一のローのダウンロードに伴うオーバヘッドはほとんどないため、単純なスナップショットの download_cursor スクリプトを使用します。

```
SELECT rep_id, name
FROM SalesRep
WHERE ? IS NOT NULL
AND ml_username = ?
```

このスクリプトの最初のパラメータは、最終ダウンロードタイムスタンプですが、これは使用されません。IS NOT NULL は、パラメータを使用するために指定されたダミー式です。2 番目のパラメータは Mobile Link ユーザ名です。

アップロード

このテーブルはリモートテーブル側では更新しないため、アップロードスクリプトはありません。

1.4.5 Contact サンプルの顧客の同期

Customer テーブルの同期スクリプトは、タイムスタンプベースの同期とローの分割の例を示しています。これらの方法では、同期中に転送されるデータの量が最小限になり、テーブルデータの整合性が保持されます。

ビジネスルール

顧客を規定するビジネスルールは、次のとおりです。

- 顧客情報は、統合データベースでもリモートデータベースでも修正できます。
- 営業担当者間で、顧客の再割り当てを定期的に変更できます。このプロセスは、一般に領域の再編成と呼ばれます。
- 各リモートデータベースには、割り当てられている顧客のみが保持されます。

ダウンロード

download_cursor

次の download_cursor スクリプトは、最後の正常なダウンロード以後に情報が変更されたアクティブな顧客のみをダウンロードします。また、営業担当者別に顧客をフィルタリングします。

```
SELECT cust_id, Customer.name, Customer.rep_id
FROM Customer key join SalesRep
WHERE Customer.last_modified >= ?
```

```
AND SalesRep.ml_username = ?
AND Customer.active = 1
```

download_delete_cursor

次の download_delete_cursor スクリプトは、最後の正常なダウンロード以後に情報が変更された顧客のみをダウンロードします。また、非アクティブのマークが付いているか、指定された営業担当者に割り当てられていない顧客を、すべて削除します。

```
SELECT cust_id
FROM Customer key join SalesRep
WHERE Customer.last_modified >= ?
AND ( SalesRep.ml_username != ? OR Customer.active = 0 )
```

統合データベースにある Customer テーブルからローが削除されると、この結果セットには表示されないため、リモートデータベースからは削除されません。代わりに、顧客には非アクティブのマークが付きます。

領域が再編成されると、このスクリプトは営業担当者への割り当てから外れた顧客を削除します。また、他の営業担当者に移された顧客も削除します。このような追加の削除にはフラグとして SQLCODE 100 が設定されますが、同期の妨げにはなりません。より複雑なスクリプトを作成すれば、現在の営業担当者から外された顧客のみを識別できます。

Mobile Link クライアントはリモートデータベースでカスケード削除を実行するため、このスクリプトによって、他の営業担当者に割り当てられた顧客のすべての窓口が削除されます。

アップロード

リモートデータベース側で顧客情報を挿入、更新、または削除できます。これらの操作に対応するスクリプトは、次のとおりです。

upload_insert

次の upload_insert スクリプトは、Customer テーブルにローを 1 つ追加して、顧客にアクティブのマークを付けます。

```
INSERT INTO Customer(
 cust_id, name, rep_id, active )
VALUES ( ?, ?, ?, 1 )
```

upload_update

次の upload_update スクリプトは、統合データベースにある顧客情報を修正します。このテーブルでは競合検出は実行されません。

```
UPDATE Customer
SET name = ?, rep_id = ?
WHERE cust_id = ?
```

upload_delete

次の upload_delete スクリプトは、統合データベースで顧客に非アクティブのマークを付けます。ローは削除されません。

```
UPDATE Customer
SET active = 0
WHERE cust_id = ?
```

1.4.6 Contact サンプルの顧客窓口の同期

Contact テーブルには、顧客の会社の社員名、顧客を参照するための外部キー、窓口を識別するユニークな整数が含まれています。また、last_modified タイムスタンプと、窓口がアクティブであるかどうかを示すマーカもあります。

ビジネスルール

このテーブルのビジネスルールは、次のとおりです。

- 窓口情報は、統合データベースでもリモートデータベースでも修正できます。
- 各リモートデータベースには、営業担当者が割り当てられている顧客の窓口のみが含まれます。
- 営業担当者間で顧客を再割り当てした場合は、窓口の再割り当ても行います。

トリガ

Customer テーブルのトリガは、顧客情報に変更があったときに窓口が確実に選択されるようにするために使用されます。トリガは、各窓口の last_modified カラムを、その窓口に対応する顧客の情報が変更されるたびに明示的に変更します。

```
CREATE TRIGGER UpdateCustomerForContact
AFTER UPDATE OF rep_id ORDER 1
ON DBA.Customer
REFERENCING OLD AS old_cust NEW as new_cust
FOR EACH ROW
BEGIN
  UPDATE Contact
  SET Contact.last_modified = new_cust.last_modified
  FROM Contact
  WHERE Contact.cust_id = new_cust.cust_id
END
```

顧客が修正されるたびにすべての窓口レコードを更新することで、トリガは顧客とその関連窓口を結合します。そのため、顧客が修正されるとすべての関連窓口も修正され、次回の同期時に顧客とその関連窓口が一括してダウンロードされます。

ダウンロード

download_cursor

Contact の download_cursor スクリプトを次に示します。

```
SELECT contact_id, contact.name, contact.cust_id
FROM ( contact JOIN customer ) JOIN salesrep
ON contact.cust_id = customer.cust_id
 AND customer.rep_id = salesrep.rep_id
WHERE Contact.last_modified >= ?
 AND salesrep.ml_username = ?
 AND Contact.active = 1
```

このスクリプトは、アクティブな窓口、営業担当者が最後にダウンロードした後に (明示的に、または対応する顧客の修正によって) 変更された窓口、営業担当者に割り当てられている窓口をすべて取り出します。この営業担当者に関連付けられている窓口を識別するには、Customer テーブルと SalesRep テーブルのジョインが必要です。

download_delete_cursor

Contact の download_delete_cursor スクリプトを次に示します。

```
SELECT contact_id
FROM ( Contact JOIN Customer ) JOIN SalesRep
ON Contact.cust_id = Customer.cust_id
 AND Customer.rep_id = SalesRep.rep_id
WHERE Contact.last_modified >= ?
 AND Contact.active = 0
```

リモートデータベースから顧客が削除されると、Mobile Link クライアントでは、カスケード参照整合性が自動的に使用され、対応する窓口が削除されます。このため、download_delete_cursor スクリプトは、非アクティブのマークが付いている窓口のみを削除します。

アップロード

リモートデータベース側で窓口情報を挿入、更新、または削除できます。これらの操作に対応するスクリプトは、次のとおりです。

upload_insert

次の upload_insert スクリプトは、Contact テーブルにローを 1 つ追加して、窓口にアクティブのマークを付けます。

```
INSERT INTO Contact (
 contact_id, name, cust_id, active )
VALUES ( ?, ?, ?, 1 )
```

upload_update

次の upload_update スクリプトは、統合データベースにある窓口情報を修正します。

```
UPDATE Contact
SET name = ?, cust_id = ?
WHERE contact_id = ?
```

このテーブルでは競合検出は実行されません。

upload_delete

次の upload_delete スクリプトは、統合データベースで窓口に非アクティブのマークを付けます。ローは削除されません。

```
UPDATE Contact
SET active = 0
WHERE contact_id = ?
```

1.4.7 Contact サンプルの製品の同期

Product テーブル用のスクリプトは、競合の検出と解決の例を示しています。

Product テーブルは他のテーブルとは別のパブリケーションに保管されているため、別個にダウンロードできます。たとえば、価格変更と営業担当者が低速リンク経由で同期している場合は、それぞれ顧客と窓口の変更をアップロードしなくても、製品変更をダウンロードできます。

ビジネスルール

リモートデータベース側で可能な変更は、注文を受けた時点で quantity カラムの値を変更することだけです。

ダウンロード

download_cursor

次の download_cursor スクリプトは、最後のリモートデータベースの同期以後に変更されたすべてのローをダウンロードします。

```
SELECT id, name, size, quantity, unit_price
FROM product
WHERE last_modified >= ?
AND active = 1
```

download_delete_cursor

次の download_delete_cursor スクリプトは、会社が販売を中止した製品をすべて削除します。これらの製品には、統合データベース内で非アクティブのマークが付きます。

```
SELECT id, name, size, quantity, unit_price
FROM product
WHERE last_modified >= ?
AND active = 0
```

アップロード

リモートデータベースからは UPDATE 操作のみがアップロードされます。これらのアップロードスクリプトの主な機能は、競合の検出と解決のためのプロシージャです。

2 人の営業担当者が注文を受けて同期を行うと、それぞれの注文数が Product テーブルの quantity カラムから減算されます。たとえば、Samuel Singer が野球帽 (製品 ID 400) 20 個の注文を受けると、数量は 90 から 70 に変更されます。Pamela Savarino が Samuel Singer の変更を受け取る前に野球帽 10 個の注文を受けると、自分のデータベース内のカラムの値が 90 から 80 に変更されます。

Samuel Singer が自分の変更を同期すると、統合データベース内の quantity カラムは 90 から 70 に変更されます。Pamela Savarino が自分の変更を同期した場合の正しい値は 60 です。この設定は、競合を検出することで行われます。

競合検出スキーマには、次のスクリプトが含まれています。

upload_update

次の upload_update スクリプトは、統合データベース側で単純な UPDATE を実行します。

```
UPDATE product
SET name = ?, size = ?, quantity = ?, unit_price = ?
WHERE product.id = ?
```

upload_fetch

次の upload_fetch スクリプトは、Product テーブルから単一のローをフェッチして、アップロードされるローの古い値と比較します。2 つのローが異なる場合は、競合が検出されます。

```
SELECT id, name, size, quantity, unit_price
FROM Product
WHERE id = ?
```

upload_old_row_insert

競合が検出されると、古い値が product_conflict テーブルに挿入されます。これは resolve_conflict スクリプトによって使用されます。row_type カラムに、Old を表す値 O を持つローが追加されます。

```
INSERT INTO DBA.product_conflict(
 id, name, size, quantity, unit_price, row_type )
VALUES( ?, ?, ?, ?, ?, 'O' )' )
```

upload_new_row_insert

次のスクリプトは、アップロードされるローの新しい値を product_conflict テーブルに追加します。これは、resolve_conflict スクリプトによって使用されます。

```
INSERT INTO DBA.product_conflict(
 id, name, size, quantity, unit_price, row_type )
VALUES( ?, ?, ?, ?, ?, 'N' )
```

競合解決

resolve_conflict

次のスクリプトは、統合データベース内の数量値に新しい値と古い値の差を加算して、競合を解決します。

```
UPDATE Product
SET p.quantity = p.quantity
              - old_row.quantity
              + new_row.quantity
FROM Product p,
     DBA.product_conflict old_row,
     DBA.product_conflict new_row
WHERE p.id = old_row.id
    AND p.id = new_row.id
    AND old_row.row_type = 'O'
    AND new_row.row_type = 'N'
```

1.4.8 Contact サンプルの統計とエラーのモニタリング

Contact サンプルには、単純なエラーレポートスクリプトとモニタリングスクリプトがいくつか用意されています。これらのスクリプトを作成するための SQL 文は、`MobiLink¥Contact¥mlmaint.sql` ファイルにあります。

各スクリプトは、値を保持するように作成されたテーブルにローを挿入します。便宜上、各テーブルの所有者は個別ユーザ mlmaint となっています。

1.5 Mobile Link チュートリアル

以下のチュートリアルは、新しいユーザのための入門用チュートリアルから、高度な機能の使用方法の説明にまで及びます。

追加の Mobile Link チュートリアルは、オンラインで提供されています。

> **注記**
>
> オンラインチュートリアルは、バージョン 12.0.0 の SQL Anywhere に基づいています。図やプロシージャのいくつかは SQL Anywhere 17.0.4 とは異なります。

1. チュートリアル: Mobile Link の概要 [97 ページ]
   このチュートリアルでは、同期スクリプトの作成、Mobile Link ログの解釈、Mobile Link プロファイラを使用した統合データベースと 2 つのリモートデータベース間での同期のモニタリングに関する基本手順について説明します。SQL Central を使用した、データベースと同期の設定手順を説明します。
2. チュートリアル: SQL Anywhere 統合データベースでの Mobile Link の使用 [117 ページ]
   このチュートリアルでは、Mobile Link を使用して SQL Anywhere データベースを活用する方法について説明します。ここでは、SQL Anywhere 統合データベースと Ultra Light リモートデータベースの間の同期を設定します。SQL Anywhere リモートデータベースを使用することもできます。
3. チュートリアル: Oracle Database 11g での Mobile Link の使用 [131 ページ]
   このチュートリアルでは、Mobile Link を使用して Oracle Database 10g を活用する方法について説明します。ここでは、Oracle Database 11g と SQL Anywhere リモートデータベースの間の同期を設定します。また、Ultra Light リモートデータベースを設定することもできます。
4. チュートリアル: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]
   このチュートリアルでは、Mobile Link を使用して Adaptive Server Enterprise データベースを活用する方法について説明します。ここでは、Adaptive Server Enterprise 統合データベースと SQL Anywhere リモートデータベースの間の同期を設定します。また、Ultra Light クライアントも使用できます。
5. チュートリアル: カスタムユーザ認証用の Java と .NET の使用 [174 ページ]
   Mobile Link 同期スクリプトは、SQL、Java、または .NET で作成できます。Java または .NET を使用すると、同期処理の任意の時点にカスタムアクションを追加できます。
6. チュートリアル: ダイレクトローハンドリングの使用 [184 ページ]
   ダイレクトローハンドリングを使用すると、サポートされている統合データベース以外にも、リモートデータと中央のデータソース、アプリケーション、または Web サービスとの通信ができます。
7. チュートリアル: Microsoft Excel との同期 [210 ページ]
   ダイレクトローハンドリングを使用して、リモートデータと中央のデータソース、アプリケーション、または Web サービスとの通信ができます。

8. チュートリアル: XML との同期 [231 ページ]
   このチュートリアルでは、XML ファイルとリモートクライアントの間でデータを同期する方法を示します。
9. チュートリアル: リモートデータベースの集中管理の使用 [253 ページ]
   このチュートリアルでは、リモートデータベースの集中管理の設定プロセスについての説明と、いくつかの一般操作の実行方法が、順を追って示されます。
10. チュートリアル: スクリプトバージョン句を使用したスキーマの変更 [283 ページ]
    このチュートリアルでは、dbmlsync ScriptVersion 拡張オプションが使用されていない同期に関係するリモートデータベースでスキーマの変更を実行する方法について説明します。
11. チュートリアル: ScriptVersion 拡張オプションを使用したスキーマの変更 [294 ページ]
    このチュートリアルでは、ScriptVersion 拡張オプションを使用している場合にスキーマを変更する方法について説明します。
12. チュートリアル: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート [302 ページ]
    このチュートリアルでは、mlreplay ユーティリティを使って複数の Mobile Link クライアントを単一のコンピュータ上でシミュレートする方法について説明します。

1.5.1 チュートリアル: Mobile Link の概要

このチュートリアルでは、同期スクリプトの作成、Mobile Link ログの解釈、Mobile Link プロファイラを使用した統合データベースと 2 つのリモートデータベース間での同期のモニタリングに関する基本手順について説明します。SQL Central を使用した、データベースと同期の設定手順を説明します。

前提条件

Mobile Link イベントスクリプトの基本的な知識が必要です。

次のソフトウェアが必要です。

- SQL Anywhere17

統合データベースで次のロールおよび権限を持つ必要があります。

- SYS_AUTH_DBA_ROLE 互換ロール

リモートデータベースで次のロールおよび権限を持つ必要があります。

- SYS_AUTH_DBA_ROLE 互換ロール
- SYS_RUN_REPLICATION_ROLE システムロール

コンテキスト

このチュートリアルでは、次の作業の方法について説明します。

- 統合データベースのスキーマのリモートデータベースへの移行
- SQL Central を使用した、同期に必要な基本スクリプトの作成と統合データベースへの保存

- Mobile Link サーバの起動
- ログファイルと Mobile Link プロファイラを使用した同期のモニタ

このセクションの内容:

すべてのチュートリアルをコンピュータから削除します。

タスクの概要: Mobile Link チュートリアル [96 ページ]

次のタスク: チュートリアル: SQL Anywhere 統合データベースでの Mobile Link の使用 [117 ページ]

関連情報

Mobile Link 同期 [4 ページ]
SAP SQL Anywhere フォーラム

1.5.1.1 レッスン 1: Mobile Link 統合データベースの設定

このレッスンでは、SQL Anywhere 統合データベースを作成し、ODBC データソースを定義して、SQL Anywhere 統合データベースを設定します。

手順

1. スタート > プログラム > *SQL Anywhere17* > 管理ツール > *SQL Central* をクリックします。
2. ツール > *SQL Anywhere17* > データベースの作成 をクリックします。
3. 次へをクリックします。
4. このコンピュータにデータベースを作成をデフォルトのままにし、次へをクリックします。
5. メインデータベースファイルを保存フィールドに、`c:¥MLintro¥MLconsolidated.db` と入力します。次へをクリックします。このディレクトリが存在しない場合は、作成するよう求められます。はいをクリックします。
6. データベース作成ウィザードの残りの指示に従い、デフォルト値をそのまま使用します。DBA ユーザのユーザ ID とパスワードを指定するように求められたら、それぞれ **DBA** と **passwd** を入力します。

   データベースへの接続ページで、最終切断後にデータベースを停止オプションをオンにします。
7. 完了をクリックします。

   MLconsolidated データベースが作成されます。
8. ウィンドウが自動的に閉じない場合は、データベースの作成ウィンドウの閉じるをクリックします。
9. 接続 > 切断 をクリックして、パーソナルデータベースサーバを停止します。
10. ツール > *SQL Anywhere17* > *ODBC* アドミニストレータを開く をクリックします。
11. ユーザ *DSN* タブをクリックし、追加をクリックします。
12. データソースの新規作成ウィンドウで、*SQL Anywhere17* をクリックし、完了をクリックします。
13. *SQL Anywhere* の *ODBC* 設定ウィンドウで、次の操作を行います。

a. *ODBC* タブをクリックします。
b. **データソース名**フィールドに **mlintro_consdb** と入力します。
c. **ログイン**タブをクリックします。
d. **認証**ドロップダウンリストで、**データベース**をデフォルトのままにして、ユーザ ID とパスワードを使用して接続します。
e. **ユーザ *ID*** フィールドに、**DBA** と入力します。
f. **パスワード**フィールドに、**passwd** と入力します。
g. **アクション**ドロップダウンリストで、**このコンピュータのデータベースを起動して接続**をデフォルトのままにします。
h. **データベースファイル**フィールドに、`c:¥MLintro¥MLconsolidated.db` と入力します。
i. **サーバ名**フィールドに、**MLconsolidated** と入力します。
j. **開始行**フィールドに、dbsrv17 と入力します。
k. *ODBC* タブをクリックし、**接続テスト**ボタンをクリックして、接続が成功したことを確認します。*OK* をクリックします。
l. *OK* をクリックし、*OK* をもう一度クリックして、*ODBC* **データソースアドミニストレータ**ウィンドウを閉じます。

14. **接続** > *SQL Anywhere 17* **に接続** をクリックします。
15. **アクション**ドロップダウンリストで、*ODBC* **データソースを使用した接続**を選択します。
16. *ODBC* **データソース名**フィールドに **mlintro_consdb** と入力します。
17. **接続**をクリックします。

結果

SQL Anywhere 統合データベースと ODBC データソ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。

1.5.1.2 レッスン 2: Mobile Link 統合データベースでのテーブルの作成と移植

このレッスンでは、Mobile Link 統合データベースで **Product** テーブルを作成し、サンプルデータを挿入します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. Interactive SQL で統合データベースに接続します。コマンドプロンプトで次のコマンドを実行します。

   Interactive SQL は、SQL Central またはコマンドプロンプトから起動できます。

   - SQL Central から、*MLconsolidated - DBA* データベースを右クリックし、*Interactive SQL* を開くをクリックします。
   - ```
     dbisql -c "DSN=mlintro_consdb"
     ```

2. Interactive SQL で次の SQL 文を実行し、**Product** テーブルを作成します。

   ```
   CREATE TABLE Product (
   name VARCHAR(128) NOT NULL PRIMARY KEY,
   quantity INTEGER
   );
   ```

   Product テーブルには次のカラムがあります。

カラム	説明
name	The name of the product.
quantity	The number of items sold.

   テーブルを作成したら、**Product** テーブルにサンプルデータを移植します。

3. Interactive SQL で次の SQL 文を実行して、**Product** テーブルにサンプルデータを移植します。

   ```
   INSERT INTO Product(name, quantity)
       VALUES ( 'Screwmaster Drill', 10);
   INSERT INTO Product(name, quantity)
       VALUES ( 'Drywall Screws 10lb', 30);
   INSERT INTO Product(name, quantity)
       VALUES ( 'Putty Knife x25', 12);
   COMMIT;
   ```

4. **Product** テーブルが、前の手順で挿入したデータを含んでいることを確認します。

   次の SQL 文を実行して、内容を確認します。

   ```
   SELECT * FROM Product
   ```

   Product テーブルの内容は、Interactive SQL に表示されます。

5. Interactive SQL を閉じます。SQL 文を保存する必要はありません。

結果

統合データベースに Products テーブルが作成されます。
   ```

次のステップ

次のレッスンに進みます。

1.5.1.3 レッスン 3: Mobile Link プロジェクトと同期モデルの作成

このレッスンでは、プロジェクト作成ウィザードを使用して新しい Mobile Link プロジェクトを作成します。また、プロジェクト作成ウィザードでは、後で編集可能なデフォルトを使用して同期モデルも作成されます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central で、ツール > *Mobile Link17* > 新しいプロジェクト をクリックします。

   プロジェクト作成ウィザードが表示されます。
2. 名前フィールドに **mlintro_project** と入力します。
3. ロケーション フィールドに `C:¥MLintro` と入力します。次へをクリックします。
4. データベースの表示名フィールドに **mlintro_consdb** と入力します。
5. 接続文字列フィールドに **dsn=mlintro_consdb** と入力し、次へをクリックします。
6. リモートデータベースに含める統合データベースのテーブルとカラムを指定してください。リストから *Product* テーブルが選択されていることを確認し、次へをクリックします。
7. リモートスキーマ名をプロジェクトに追加オプションを選択します。
8. 新しいリモートスキーマ名を指定してください。フィールドに **sync_mlintro** と入力し、次へをクリックします。
9. 使用するリモートデータベースのタイプを指定してください。オプションの *SQL Anywhere* を選択し、完了をクリックします。
10. はいをクリックして Mobile Link システムテーブルをインストールしてから、*OK* をクリックします。

    リモートスキーマと同じ名前を持つ同期モデルが作成されます。
11. マッピングタブから *Product* テーブルローを選択します。
12. 詳細ウィンドウ枠のダウンロード方式タブで、方式としてタイムスタンプを選択します。
13. タイムスタンプカラムをシャドウテーブルに保存するを選択します。

    シャドウテーブルを使用すると既存のテーブルを変更する必要がないため、推奨されることが多くあります。反対に、デフォルト設定の場合は、テーブルにタイムスタンプカラムが追加され、通常、パフォーマンスが向上します。

14. 同期モデルに加えた変更を保存するには、ファイル > 保存 を選択します。

結果

Mobile Link プロジェクト、リモートスキーマ名、同期モデルがすべて作成されます。

次のステップ

次のレッスンに進みます。

関連情報

同期モデル [33 ページ]
同期モデルタスク [37 ページ]
テーブルとカラムのマッピング [38 ページ]
ダウンロード方式の変更 [43 ページ]
競合の検出と解決の変更 [50 ページ]

1.5.1.4 レッスン 4: 同期モデルのテスト

このレッスンでは、同期モデルを迅速にテストする方法を説明します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central の左ウィンドウ枠の *Mobile Link 17* で、`mlintro_project` を展開します。同期モデルをクリックしてから、`sync_mlintro` を選択します。
2. ファイル > テスト をクリックします。

3. *OK* をクリックして、統合データベースが変更されることを示す警告を解除します。

   同期モデルが統合データベースに展開され、テストのためにリモートデータベースが作成されます。Mobile Link サーバが起動します。
4. データタブをクリックします。上部ウィンドウ枠に、統合データベースの Product テーブルからのローが表示されます。下部ウィンドウ枠からは、リモートデータベースに Product テーブルのローが現在何も含まれていないことがわかります。

   複数のテーブルが定義されている場合は、表示ドロップダウンリストで、表示するテーブルを選択します。
5. 同期をクリックします。これで、リモートデータベースに 3 つのローが表示されます。
6. クライアントログタブを選択します。エラーメッセージまたは警告のログをスキャンします。*Mobile Link* ログタブでも同じ操作を実行します。
7. データタブをクリックします。下ウィンドウ枠で、**`Drywall Screws 10lb`** という名前を持つローを右クリックし、ローの編集を選択し、数量を 99 に変更します。
8. 同期をクリックします。リモートデータベースで行った変更と、統合データベースとの同期が完了します。
9. アクションボタンをクリックし、統合データベースで *Interactive SQL* を開くを選択します。統合データベースに接続する Interactive SQL ウィンドウが開きます。
10. 次の SQL 文を実行して、Interactive SQL を閉じます。

```
update product set quantity = quantity + 1;
commit;
```

11. 再び SQL Central のデータタブで、アクション > データの再表示 タブをクリックします。統合データベース内の更新されたローが表示されます。
12. 同期をクリックします。リモートデータベースが新しい数量値に更新されます。
13. テストウィンドウを閉じます。

結果

同期モデルのテストが正常に完了しました。

次のステップ

次のレッスンに進みます。

関連情報

同期モデルの展開 [54 ページ]

1.5.1.5 レッスン 5: 同期モデルの調整

このレッスンでは、同期モデルに変更を加える方法を説明するとともに、モデルを展開するときに使用できる選択肢の一部を解説します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central の左ウィンドウ枠の *Mobile Link 17* で、**mlintro_project** を展開します。同期モデルをクリックしてから、**sync_mlintro** を選択します。
2. マッピングタブで、Product テーブルのローを選択します。
3. 下ウィンドウ枠で、競合の処理タブを選択します。
4. 検出された競合を解決する方式を選択しますで、先入れ勝ちを選択します。このオプションを選択すると、あるローが統合データベースとリモートデータベースの両方で変更された場合に統合データベースの値が正しい値であるとみなされます。
5. 展開タブで、テストをクリックします。同期モデルを保存するように求めるプロンプトが表示された場合は、はいをクリックします。
6. データタブを選択し、同期をクリックしてリモートデータベースを更新します。
7. *Screwmaster Drill* という名前のローを編集し、統合データベースでの数量を 20 に設定し、リモートデータベースでの数量を 10 に設定します。
8. 同期をクリックします。リモートデータベースと統合データベースの両方で、数量は 20 と表示されるはずです。
9. テストウィンドウを閉じます。
10. **sync_mlintro** モデルのイベントタブを選択します。このページは、この同期モデルのために Mobile Link サーバによって実行される SQL 文を表示します。左側のマージンにある緑色のバーは、マッピングタブで選択した内容に基づいて SQL 文が自動的に生成されることを示しています。

    マッピングタブで使用できるオプションが、シナリオの実現に十分でない場合もあります。このような場合は、イベントタブを使用して同期スクリプトをさらにカスタマイズすることができます。
11. イベントタブでは、Product テーブルの upload_insert イベントを探します。これは、Mobile Link サーバがリモートデータベースから新しいローを受信するときに実行する SQL 文を示します。このイベントに変更を加えて、新しい注文の最大数量を 50 に制限します。テキスト **{ml r."quantity"}** は、アップロードされたリモート quantity カラムを示します。これを次のように変更します。

    ```
    If {ml r."quantity"} < 50 then {ml r."quantity"} else 50 end if
    ```

    完全な upload_insert イベントは、次のようになります。

    ```
    Product (DBA): upload_insert
    ```

```
/* Insert the row into the consolidated database. */
INSERT INTO "DBA"."Product" ( "name", "quantity" )
VALUES ( {ml r."name"}, If {ml r."quantity"} < 50 then {ml r."quantity"} else 50
end if )
```

upload_insert イベントに対するマージンのバーが黄色になります。これは、マッピングタブからの設定が上書きされたことを示します。

12. 展開タブで、テストをクリックします。データタブを選択し、同期をクリックします。次に、リモートデータベースの下ウィンドウ枠で右クリックします。ローの追加を選択し、**Hammer** という名前と数量 **200** が含まれているローを追加します。同期をクリックします。統合データベースとリモートデータベースは両方とも **Hammer** ローの値が 50 になります。
13. テストウィンドウを閉じて、イベントタブに戻ります。upload_insert イベントスクリプトを右クリックし、*'Product (DBA): upload_insert'* スクリプトのリストアを選択します。マージン内のバーの色が再び緑になり、前に行ったカスタマイズが取り消されました。

結果

同期モデルに変更が加えられます。

次のステップ

次のレッスンに進みます。

関連情報

同期モデルの展開 [54 ページ]

1.5.1.6 レッスン 6: Mobile Link サーバオプションの選択

このレッスンでは、同期モデルを展開するときに Mobile Link サーバを実行するために後で使用されるオプションの選択方法を説明します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central の左ウィンドウ枠の *Mobile Link 17* で、**mlintro_project** を展開してから *Mobile Link* サーバのコマンドライン をクリックします。同期モデルを保存するように求めるプロンプトが表示された場合は、はいをクリックします。
2. 右ウィンドウ枠でデフォルトを右クリックし、プロパティを選択します。
3. デフォルト *Mobile Link* サーバのコマンドラインのプロパティウィンドウで、次のタスクを実行します。
   - 一般タブを選択します。冗長性ドロップダウンリストから、*High (-v+)* を選択します。
   - 詳細タブを選択します。
   - -dl オプションを *true* に設定します。
   - -o オプションを *mlsrv.mls* に設定します。
   - -zf オプションを *true* に設定します。
   - -zu オプションを *true* に設定します。

   注記

   -zf オプションは、デバッグと開発の目的でのみ使用してください。後のレッスンで新しいスクリプトを統合データベースに追加するときに Mobile Link サーバを停止する必要をなくするため、このチュートリアルでは -zf オプションが必要です。-zu+ オプションは、新しい Mobile Link ユーザを同期環境に自動的に追加します。
4. *OK* をクリックします。

結果

同期モデルの展開時に Mobile Link サーバを実行するためのオプションが設定されます。

次のステップ

次のレッスンに進みます。

関連情報

同期モデルの展開 [54 ページ]

1.5.1.7 レッスン 7: 同期モデルの展開

このレッスンでは、同期モデル展開ウィザードを使用して同期モデルを展開して同期のために統合データベースを設定し、リモートデータベースを作成して展開します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central の左ウィンドウ枠の *Mobile Link 17* で、**mlintro_project** を展開します。同期モデルをクリックしてから、**sync_mlintro** を選択します。
2. ファイル > 展開 をクリックします。
3. ウィザードによって生成されたファイルを保管するフォルダを選択します。フィールドのデフォルト設定を使用し、次へをクリックします。
4. クライアントネットワークオプションページのデフォルト設定をそのまま使用し、次へをクリックします。
5. どのような *Mobile Link* のユーザとパスワードを使用しますかの下の *Mobile Link* ユーザとパスワードでこれらの使用を選択し、次のタスクを実行します。
   a. *Mobile Link* ユーザフィールドに、**mlintro_user** と入力します。
   b. *Mobile Link* パスワードフィールドに、**passwd** と入力します。
   c. "このユーザを統合データベースに登録します。登録されたユーザは同期することができます" をオンにして、次へをクリックします。
6. 同期プロファイルページで、同期プロファイル名フィールドに **mlintro_remote_syncprofile** と入力します。参照をクリックし、冗長性を高に設定します。*OK* をクリックしてから、次へをクリックします。
7. データベースの同期を準備する方法を選択しますページで次のタスクを実行します。
   a. 統合データベースの同期を準備するために作成された *SQL* スクリプトの処理方法を指定してくださいで、統合データベースに対して実行しますを選択します。
   b. リモートデータベースの同期を準備するために作成された *SQL* スクリプトの処理方法を指定してくださいで、新しいリモートデータベースに対して実行しますを選択します。
   c. ユーザ *ID* フィールドに、**DBA** と入力します。
   d. パスワードフィールドとパスワードの確認フィールドに、**passwd** と入力します。
   e. 次へをクリックします。
8. 選択内容の確認ページでは、ウィザードで選択した内容を確認できます。また、ビューボタンを使用すれば、データベースに対して実行される SQL 文を確認できます。完了をクリックします。

   `sync_mlintro_remote.db` と呼ばれるリモートデータベースが作成されます。

結果

リモートデータベースが正常に作成され、展開されました。

次のステップ

次のレッスンに進みます。

関連情報

同期モデルの展開 [54 ページ]

1.5.1.8 レッスン 8: Mobile Link サーバの起動

このレッスンでは、mlsrv17 -c オプションを使って Mobile Link サーバを起動し、統合データベースに接続します。追加のオプションを使用して、Mobile Link サーバの動作を設定します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. コマンドプロンプトで、`c:¥MLintro¥mlintro_project¥sync_mlintro_deploy` ディレクトリに移動します。
2. 次のコマンドを実行します。

```
mlsrv.bat
```

> **注記**
>
> -zu オプションを設定すると、任意のユーザが接続できるようになるので、サーバのセキュリティが低下します。運用環境で -zu オプションを使用しないでください。

結果

Mobile Link サーバが起動します。

次のステップ

次のレッスンに進みます。

1.5.1.9 レッスン 9: Mobile Link クライアントの起動

このレッスンでは、同期の準備をするためにリモートデータベースを起動します。このレッスンでは、リモートデータベース、統合データベース、Mobile Link サーバが同一のコンピュータ上に存在することを前提としています。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. コマンドプロンプトで、`c:¥MLintro¥mlintro_project¥sync_mlintro_deploy` ディレクトリに移動します。
2. 次のコマンドを実行して、**sync_mlintro_remote** データベースを起動します。

```
dbsrv17 sync_mlintro_remote
```

結果

リモートデータベースが起動します。

次のステップ

次のレッスンに進みます。

1.5.1.10 レッスン 10: Mobile Link プロファイラの起動

このレッスンでは、同期の発生を監視するために、Mobile Link プロファイラを開始して設定します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

Mobile Link プロファイラは、同期の統計情報を収集するために使用できます。グラフィックチャートでは、水平軸に時間の経過が示され、垂直軸にはタスクが示されます。Mobile Link プロファイラを使用することによって、エラーを引き起こしたり、特定の条件を満たしたりする同期を迅速に突き止めることができます。Mobile Link プロファイラによってパフォーマンスが大幅に低下することはないので、開発環境でも運用環境でもこのプロファイラを使用してください。

手順

1. スタート > プログラム > *SQL Anywhere17* > 管理ツール > *Mobile Link* プロファイラ をクリックします。
2. Mobile Link プロファイラを Mobile Link サーバに接続します。ファイル > プロファイリングセッションの開始 をクリックします。

   Mobile Link サーバへの接続ウィンドウが表示されます。
3. ユーザ フィールドに `monitor_user` と入力します。このユーザにはパスワードは不要です。
4. ホストフィールドに `localhost` と入力します。*OK* をクリックします。

結果

前のレッスンで Mobile Link サーバを -zu+ オプション付きで起動したので、このユーザは自動的に追加されています。プロファイラによって同期データの収集が開始されます。

次のステップ

次のレッスンに進みます。

1.5.1.11 レッスン 11: 同期

このレッスンでは、dbmlsync ユーティリティを使用して、リモートデータベースを統合データベースと同期します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. コマンドプロンプトで、`c:¥MLintro¥mlintro_project¥sync_mlintro_deploy` ディレクトリに移動します。
2. 次のコマンドを実行して、**sync_mlintro_remote** データベースを同期します。

   ```
   sync.bat "server=sync_mlintro_remote;UID=DBA;PWD=passwd"
   ```

   sync_mlintro_remote クライアントの統合データベースとの同期に関連するすべての情報を表示するウィンドウが表示されます。この情報は `remote.dbs` ファイルに保存されます。このファイルはクライアント同期ウィンドウを閉じた後でアクセス可能になります。
3. クライアント同期ウィンドウを閉じます。シャットダウンをクリックします。
4. Mobile Link プロファイラで、一時停止ボタンをクリックして水平方向のスクロールを止め、下部ウィンドウ枠を使用して最新の同期にスクロールし戻します。
5. 最新の同期のプロパティを確認します。同期プロパティを表示するには、色つきの縦線をダブルクリックします。

結果

リモートデータベースが統合データベースと同期されます。

次のステップ

次のレッスンに進みます。

1.5.1.12 レッスン 12: Mobile Link サーバログファイルビューアを使用したエラーと警告の確認

テーブルの同期が完了したら、各コマンドラインを使用してメッセージログファイル、つまり `mlsrv.mls` と `remote.dbs` をそれぞれ使用して同期の経過を表示できます。これらのファイルのデフォルトロケーションは、コマンドが実行されたディレクトリです。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central で、**ツール** > *Mobile Link 17* > *Mobile Link* **サーバログファイルビューア** をクリックします。
2. テキストエディタでメッセージログファイルを開いて `c:¥MLintro¥mlintro_project¥sync_mlintro_deploy¥mlsrv.mls` を選択し、**開く**をクリックします。
   Mobile Link **サーバログファイルビューア**ウィンドウが表示されます。
3. **同期**タブをクリックして、同期中に発生したエラーや警告を調べます。
4. **メッセージ**タブをクリックすると、Mobile Link サーバから報告されたエラーと警告を検索できます。

   情報を表示オプションをクリアし、**適用**をクリックします。

   エラーと警告を含む同期だけが、**メッセージ**ウィンドウ枠に表示されます。たとえば、次の内容を示す警告が表示されることがあります。

   ```
   [10093] Mobile Link サーバは現在、パフォーマンスを低下させる -zf で実行中です。
   ```
5. **概要**タブをクリックすると、メッセージログファイルに示される全体の統計情報を検索できます。
6. テキストエディタで、`remote.dbs` などのクライアントログファイルを開きます。
7. ファイルの左側を下へスキャンします。**E.** で始まる行が表示された場合、エラーが発生したことを示します。ログファイルにエラーがない場合、同期は正常に完了しています。

結果

メッセージログファイルを使用して同期の経過を確認しました。

次のステップ

次のレッスンに進みます。

1.5.1.13 レッスン 13: SQL Anywhere モニタによる Mobile Link リソースのモニタリング

このレッスンでは、Mobile Link サーバと Mobile Link サーバファームのモニタリングを設定します。このレッスンでは、モニタ Developer Edition を使用しています。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. モニタを起動します。次の手順では、モニタが現在バックグラウンドで実行されていないことを前提としています。

   モニタ Developer Edition を起動するには、次の手順に従います (Windows の場合)。

   スタート > **プログラム** > *SQL Anywhere17* > **管理ツール** > *SQL Anywhere* **モニタ** をクリックします。

   モニタ Developer Edition を起動するには、次の手順に従います (Linux の場合)。

   モニタのインストールディレクトリの `bin32` または `bin64` ディレクトリから `samonitor.sh` スクリプトを実行します。

   ```
   samonitor.sh launch
   ```

   モニタによってメトリックの収集が開始され、モニタにログインするためのデフォルト URL がブラウザに表示されます。
   `http://localhost:4950`

   > **注記**
   >
   > ネットワークを経由してモニタにアクセスしている場合は、**`http://`**`computer-name`**`:4950`** をブラウズします。`computer-name` は、モニタが実行されているコンピュータの名前です。

2. デフォルトの管理者ユーザとしてモニタにログインします。

   ユーザ名フィールドに **`admin`** と入力し、**パスワード**フィールドに **`admin`** と入力します。

> **注記**
>
> 次の手順を実行するには、モニタに管理者としてログインしてください。読み込み専用ユーザとオペレータユーザの持つ権限では、一部のタスクしか実行できません。
>
> 1. モニタにログインします。
> 2. ツール > ユーザ設定 をクリックし、ユーザのタイプ設定を確認します。

3. Mobile Link サーバリソースをモニタに追加するには、次の手順に従います。
   a. 左のナビゲーションメニューで ツール > 管理 をクリックします。
   b. リソースをクリックして、追加をクリックします。
   c. *Mobile Link* サーバをクリックして次へをクリックします。
   d. 名前フィールドに **MobiLinkServerSample** と入力し、次へをクリックします。
   e. ホストフィールドに **localhost** と入力し、次へをクリックします。
   f. 必要な認証情報が要求されたら、ユーザ *ID* フィールドに **monitor_user** などのユーザ名を入力し、パスワードフィールドに **passwd** などのパスワードを入力します。

      これらのクレデンシャルは、Mobile Link サーバにユーザを作成するために使用されます。モニタでは、このユーザ ID とパスワードを保存し、Mobile Link サーバへの接続とモニタリングに使用します。
   g. 作成をクリックします。
   h. 新しいリソース **MobiLinkServerSample** が作成され、モニタリングが開始されます。
   i. 閉じるをクリックします。
   j. 閉じるをクリックします。
   k. 概要 > リソースリスト をクリックします。**MobiLinkServerSample** をクリックしてリソースのダッシュボードを作成し、開きます。
4. 2 つの Mobile Link サーバをモニタリングするための Mobile Link サーバファームリソースを追加するには、次の手順に従います。
   a. 2 つの Mobile Link サーバをモニタリング対象のリソースとして追加します。最初のリソースとして、前の手順で追加した *MobiLinkServerSample* リソースを使用します。

      2 番目の Mobile Link サーバリソースを追加します。

      1. コマンドプロンプトで次のコマンドを実行し、ポート 8039 で受信する Mobile Link サーバを起動します。

         ```
         mlsrv17 -vcrs -zu+ -c "DSN=mlintro_consdb" -ot ml_tcpip.txt -zs ml_tcpip -
         x tcpip{port=8039}
         ```

      2. ツール > 管理 をクリックします。
      3. リソースをクリックして、追加をクリックします。
      4. *Mobile Link* サーバをクリックして次へをクリックします。
      5. 名前フィールドに **ml_tcpip** と入力し、次へをクリックします。
      6. ホストフィールドに、**localhost** と入力します。
         ポート フィールドに **8039** と入力し、次へをクリックします。
      7. 必要な認証情報が要求されたら、ユーザ *ID* フィールドに **monitor_user** などのユーザ名を入力し、パスワードフィールドに **passwd** などのパスワードを入力します。
         これらのクレデンシャルは、Mobile Link サーバにユーザを作成するために使用されます。モニタでは、このユーザ ID とパスワードを保存し、Mobile Link サーバへの接続とモニタリングに使用します。
      8. 作成をクリックします。

概要ダッシュボードのリソースリストに **ml_tcpip** リソースが追加されます。

9. 閉じるをクリックします。
10. 閉じるをクリックします。

b. Mobile Link サーバファームリソースを追加します。

1. 管理ウィンドウを開きます。
   ツール > 管理 をクリックします。
2. リソースをクリックして、追加をクリックします。
3. *Mobile Link* サーバのファームをクリックして次へをクリックします。
4. 名前フィールドに **MobiLink_Test_Farm** と入力し、次へをクリックします。
5. **MobiLinkServerSample** と **ml_tcpip** をクリックして、作成をクリックします。
6. 閉じるをクリックします。
7. 閉じるをクリックします。

c. ダッシュボード > 概要 をクリックします。

リソースリストに **MobiLink_Test_Farm** リソースが表示されます。

Mobile Link サーバリソースは、リソースリストに残ります。

d. **MobiLink_Test_Farm** の左にある矢印をクリックすると、ファームに含まれている Mobile Link サーバリソースのリストが表示されます。

e. **MobiLink_Test_Farm** をクリックすると、**MobiLink_Test_Farm** ダッシュボードが開き、収集されたメトリックが表示されます。

警告リスト、リソースウィジェット、サーバ情報ウィジェットが表示されます。

5. 警告のテストまたはその他のモニタ機能については、次のチュートリアルを参照してください。モニタによるリソースのモニタリング

結果

SQL Anywhere モニタを使用して、Mobile Link サーバと Mobile Link サーバファームをモニタリングしました。

次のステップ

次のレッスンに進みます。

1.5.1.14 レッスン 14: クリーンアップ

すべてのチュートリアルをコンピュータから削除します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 以下のアプリケーションのインスタンスをすべて閉じます。
   - Mobile Link プロファイラ
   - Mobile Link 用 SQL Anywhere モニタ
   - SQL Central
   - Interactive SQL
2. SQL Anywhere、Mobile Link、同期クライアントの各ウィンドウを閉じます。
3. 次の手順で、チュートリアルに関連するすべてのデータソースを削除します。
   a. ODBC データソースアドミニストレータを起動します。
   b. スタート > プログラム > *SQL Anywhere17* > 管理ツール > *ODBC* データソースアドミニストレータ をクリックします。
   c. ユーザデータ ソースリストから **mlintro_consdb** を選択し、削除をクリックします。
4. 統合データベースとリモートデータベースが保存されているディレクトリを削除します。

結果

チュートリアルがコンピュータから削除されます。

1.5.2 チュートリアル: SQL Anywhere 統合データベースでの Mobile Link の使用

このチュートリアルでは、Mobile Link を使用して SQL Anywhere データベースを活用する方法について説明します。ここでは、SQL Anywhere 統合データベースと Ultra Light リモートデータベースの間の同期を設定します。SQL Anywhere リモートデータベースを使用することもできます。

前提条件

次のソフトウェアが必要です。

- SQL Anywhere17

統合データベースで次のロールおよび権限を持つ必要があります。

- SYS_AUTH_DBA_ROLE 互換ロール

コンテキスト

このチュートリアルの目的は、多くの地域で動作する携帯電話会社のデータを活用することです。このシナリオで、各地域には次のような特徴があります。

- リモート同期環境です。
- Mobile Link を使って、中央の SQL Anywhere 統合データベースと同期するローカルの Ultra Light データベースがあります。
- 新しい顧客がアカウントをアクティブにしたり、既存の顧客が新しいモバイルデバイスをアクティブにしたときに、そのロケーションから製品情報にアクセスしたり、リモートデータベースからデータを操作することができます。

このチュートリアルでは、次の作業の方法について説明します。

- リモートスキーマの設計時に、リモートテーブルの同期方向などの重要な考慮事項を決定します。
- 統合データベースとリモートデータベースにユニークなプライマリキーを追加します。
- 同期モデル作成ウィザードを使用して、統合データベースとリモートデータベースの間の同期を設定します。
- SQL Central を使用して同期設定をカスタマイズします。
- 同期モデル展開ウィザードを使用して統合データベースとリモートデータベースを展開します。
- リモートクライアントを統合データベースと同期します。

1. スキーマの設計 [119 ページ]
   このチュートリアルでは、SQL Anywhere が動作しているコンピュータにサンプルデータベースがインストールされていることを前提としています。
2. レッスン 1: 統合データベースの準備 [120 ページ]
   このレッスンでは、統合データベースに接続し、CustomerProducts テーブルを作成し、地域情報を含めるように Customers テーブルを変更します。
3. レッスン 2: 同期モデルの作成 [122 ページ]
   このレッスンでは、プロジェクト作成ウィザードを使用して新しい Mobile Link プロジェクトを作成します。また、プロジェクト作成ウィザードでは、後で編集可能なデフォルトを使用して同期モデルも作成されます。
4. レッスン 3: 同期モデルの展開 [125 ページ]
   同期モデル展開ウィザードを使用すると、統合データベースとリモートデータベースを展開できます。各データベースの展開は 1 つずつ行うこともできますが、両方一度に行うこともできます。同期モデル展開ウィザードでは、展開のオプションを順を追って設定できます。
5. レッスン 4: Mobile Link サーバの起動 [127 ページ]
   このレッスンでは、mlsrv17 -c オプションを使って Mobile Link サーバを起動し、統合データベースに接続します。追加のオプションを使用すると、Mobile Link サーバの動作を設定できます。

6. レッスン 5: 同期 [128 ページ]
   このレッスンでは、ulsync ユーティリティで同期を開始することによって、Mobile Link クライアントを Mobile Link サーバと同期します。
7. レッスン 6: クリーンアップ [130 ページ]
   チュートリアルをコンピュータから削除します。

タスクの概要: Mobile Link チュートリアル [96 ページ]

前のタスク: チュートリアル: Mobile Link の概要 [97 ページ]

次のタスク: チュートリアル: Oracle Database 11g での Mobile Link の使用 [131 ページ]

関連情報

Mobile Link 同期 [4 ページ]

1.5.2.1 スキーマの設計

このチュートリアルでは、SQL Anywhere が動作しているコンピュータにサンプルデータベースがインストールされていることを前提としています。

サンプルデータベースは統合データベースとして使用されます。次の表は、SQL Anywhere 統合データベースの各テーブルの説明です。

テーブル	説明
Customers	レコードに情報が記録されている顧客。
SalesOrders	アカウントアクティブ化レコード。
Products	購入可能なすべての製品のレコード。
CustomerProducts	それぞれの顧客が所有している製品のリスト。

リモートスキーマの設計

地域ごとに統合データベース全体をコピーしておくことは不要であり、非効率的です。リモートスキーマでは同じテーブル名を使用しますが、各地域に関する情報だけが格納されます。この設定を実現するため、リモートスキーマは統合データベースのサブセットとして次のように設計されています。

統合テーブル	リモートテーブル
Customers	地域によるフィルタ。

統合テーブル	リモートテーブル
SalesOrders	適切な地域にいる顧客の顧客 *ID* によるフィルタ。
Products	すべてのローを含みます。
CustomerProducts	適切な地域にいる顧客の顧客 *ID* によるフィルタ。

それぞれの営業担当者は、すべての顧客に提供された製品の情報と同様、担当の地域の顧客に関する情報を維持する必要があります。しかし、営業担当者は別の地域の顧客に関する情報は必要としないので、この情報は各地域のオフィスとは同期されません。そのため、ローは地域の識別子に基づいてフィルタされます。

注記

リモートデータベースで不要になるカラムがある場合は、テーブルからカラムのサブセットを取得することもできます。

次に、各テーブルの同期の方向を選択します。リモートデータベースで読み込む情報と、リモートデータベースで作成、変更、または削除する情報について考慮します。この例では、地域は、顧客に提供される製品のリストにアクセスする必要がありますが、新製品をシステムに入力することはありません。これは、製品情報の入力は必ず中央の統合データベースから行うという制限を生み出します。一方で、営業担当者が、新しいアカウントのアクティブ化を常時記録する必要があります。これらの要因から、各テーブルの同期の方向は次のようになります。

テーブル	同期
Customers	統合データベースへのアップロードのみ。
SalesOrders	統合データベースへのアップロードのみ。
Products	リモートデータベースへのダウンロードのみ。
CustomerProducts	統合データベースへのアップロードのみ。

親トピック チュートリアル: SQL Anywhere 統合データベースでの Mobile Link の使用 [117 ページ]

次のタスク: レッスン 1: 統合データベースの準備 [120 ページ]

1.5.2.2 レッスン 1: 統合データベースの準備

このレッスンでは、統合データベースに接続し、CustomerProducts テーブルを作成し、地域情報を含めるように Customers テーブルを変更します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

同期システムでは、テーブルのプライマリキーは、異なるデータベース内の同じローを識別する唯一の方法であり、競合を検出する唯一の方法です。使用する各テーブルには、プライマリキーが必要です。プライマリキーが更新されることはありません。また、1 つのデータベースに挿入されたプライマリキーの値が別のデータベースに挿入されないようにする必要があります。

あとのレッスンでは、統合スキーマからリモートスキーマを作成するため、リモートスキーマは統合スキーマと同じプライマリキーを持つことになります。

プライマリキーがすべてのデータベースに対してユニークになるようにカラムが選択されています。Customers テーブルでは、プライマリキーは ID カラムで構成されています。リモートの Customers テーブルに挿入されるすべての値には、ユニークな顧客 ID 番号が必要です (地域の値は常に同じです)。これにより、リモートの各 Customers テーブルで一意性が確保されます。統合データベースの Customers テーブルのプライマリキーは、複数の販売担当者がデータがアップロードした場合の競合を防止する役割があります。ある地域からの各アップロードは、地域値が異なるので、他の地域とは異なっています。

手順

1. **スタート** > **プログラム** > *SQL Anywhere17* > **管理ツール** > *SQL Central* をクリックします。
2. **接続** > *SQL Anywhere17* **に接続** をクリックします。
3. **接続**ウィンドウで次のタスクを実行します。
   a. **アクション**ドロップダウンリストで、*ODBC* **データソースを使用した接続**を選択します。
   b. **ユーザ** *ID* フィールドに、**DBA** と入力します。
   c. **パスワード**フィールドに、**sql** と入力します。
   d. *ODBC* **データソース名**フィールドに *SQL Anywhere 17 Demo* と入力します。
   e. **接続**をクリックします。
4. Interactive SQL の統合データベースに接続します。

   コマンドプロンプトで次のコマンドを実行します。

   ```
   dbisql -c "DSN=SQL Anywhere 17 Demo;UID=DBA;PWD=sql"
   ```

5. Interactive SQL で、次の文を実行し、CustomerProducts テーブルを作成してデータを挿入します。

   ```
   CREATE TABLE CustomerProducts
       (ID int default AUTOINCREMENT PRIMARY KEY,
       SalesOrderID int NOT NULL,
       CustomerID int NOT NULL,
       ProductID int);
   INSERT INTO CustomerProducts (SalesOrderID,CustomerID,ProductID)
       SELECT SalesOrders.ID, SalesOrders.CustomerID, SalesOrderItems.ProductID
       FROM SalesOrders, SalesOrderItems
       WHERE SalesOrders.ID = SalesOrderItems.ID;
   ```

6. Interactive SQL で、次の文を実行し、Customers テーブルのそれぞれの顧客に地域情報を追加します。

   ```
   ALTER TABLE Customers
       ADD Region VARCHAR(255);
   UPDATE Customers
       SET Region = (SELECT TOP 1 SalesOrders.Region
       FROM SalesOrders
   ```

```
    WHERE Customers.ID = SalesOrders.CustomerID
    ORDER BY Region);
COMMIT;
```

結果

サンプルデータベースとの接続が確立され、CustomerProducts という名前のテーブルが作成され、地域情報を含むように Customers テーブルが変更されます。

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 Anywhere 統合データベースでの Mobile Link の使用 [117 ページ]

前のトピック: スキーマの設計 [119 ページ]

次のタスク: レッスン 2: 同期モデルの作成 [122 ページ]

1.5.2.3 レッスン 2: 同期モデルの作成

このレッスンでは、プロジェクト作成ウィザードを使用して新しい Mobile Link プロジェクトを作成します。また、プロジェクト作成ウィザードでは、後で編集可能なデフォルトを使用して同期モデルも作成されます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. *SQL Central* で、ツール > *Mobile Link17* > 新しいプロジェクト をクリックします。
2. プロジェクト作成ウィザードが表示されます。
3. 新しいプロジェクトの名前を指定してください。フィールドに **`mlsqla_project`** と入力します。

4. 新しいプロジェクトの保存場所を指定してください。フィールドに `C:¥mlsqla` と入力し、次へをクリックします。
5. データベースの表示名フィールドに *demo* と入力します。
6. 編集をクリックします。
7. 汎用 *ODBC* データベースに接続ページで次のタスクを実行します。
   a. ユーザ *ID* フィールドに、**DBA** と入力します。
   b. パスワードフィールドに、**sql** と入力します。
   c. *ODBC* データソース名フィールドで、参照をクリックして *SQL Anywhere 17 Demo* を選択します。
   d. *OK* をクリックし、保存をクリックします。
8. パスワードを記憶オプションを選択し、次へをクリックします。
9. 新しいリモートデータベーススキーマページのリモートデータベースに含める統合データベースのテーブルとカラムを指定してください。リストで、次のテーブルを選択します。
   - CustomerProducts
   - Customers
   - Products
   - SalesOrders

   次へをクリックします。
10. リモートスキーマ名をプロジェクトに追加オプションを選択します。
11. 新しいリモートスキーマ名を指定してください。フィールドに **mlsqla_remote_schema** と入力し、次へをクリックします。
12. 使用するリモートデータベースのタイプを指定してください。オプションの *Ultra Light* を選択し、完了をクリックします。
13. Mobile Link 設定スクリプトをインストールするように要求されたらはいをクリックします。
14. リモートスキーマをインポートするように要求された場合は、はいをクリックします。
15. *OK* をクリックします。
16. 新しい同期モデルを右クリックして、プロパティを選択します。
    a. 最初のフィールドに **sync_mlsqla** と入力します。
    b. パブリケーション名フィールドに **sync_mlsqla_publication** と入力します。
    c. スクリプトバージョンに **sync_mlsqla_scriptversion** と入力します。

       パブリケーションは、同期するデータを指定するリモートデータベース上のオブジェクトです。Mobile Link サーバのスクリプトにより、リモートデータベースからアップロードされたデータを統合データベースに適用する方法と、スクリプトバージョンによりスクリプトをグループ化する方法が定義されます。アプリケーションごとに異なるスクリプトバージョンを使用できるため、1 つの Mobile Link サーバを管理するだけで複数のアプリケーションを同期できます。
    d. 適用をクリックしてから *OK* をクリックします。
17. SQL Central の右ウィンドウ枠で次のタスクを実行します。
    a. イベントタブをクリックします。
    b. CustomerProducts download_cursor を更新して、東部地域の顧客向けの顧客製品だけをダウンロードするようにします。

       CustomerProducts テーブル用の download_cursor イベントの既存の SQL スクリプトを次のクエリに置き換えます。

```
SELECT "DBA"."CustomerProducts"."ID",
    "DBA"."CustomerProducts"."SalesOrderID",
    "DBA"."CustomerProducts"."CustomerID",
    "DBA"."CustomerProducts"."ProductID"
FROM "DBA"."CustomerProducts"
```

```
INNER JOIN "GROUPO"."Customers" ON "GROUPO"."Customers"."ID" =
    "DBA"."CustomerProducts"."CustomerID"
WHERE "GROUPO"."Customers"."Region" = 'Eastern';
```

c. Customers テーブルのダウンロードカーソルを更新して、東部地域の顧客情報だけをダウンロードするようにします。

Customers テーブル用の download_cursor イベントの既存の SQL スクリプトを次のクエリに置き換えます。

```
SELECT "GROUPO"."Customers"."ID",
    "GROUPO"."Customers"."Surname",
    "GROUPO"."Customers"."GivenName",
    "GROUPO"."Customers"."Street",
    "GROUPO"."Customers"."City",
    "GROUPO"."Customers"."State",
    "GROUPO"."Customers"."Country",
    "GROUPO"."Customers"."PostalCode",
    "GROUPO"."Customers"."Phone",
    "GROUPO"."Customers"."CompanyName",
    "GROUPO"."Customers"."Region"
FROM "GROUPO"."Customers"
WHERE Region = 'Eastern';
```

d. SalesOrders ダウンロードカーソルを更新して、東部地域の顧客向けの注文情報だけをダウンロードするようにします。

SalesOrders テーブル用の download_cursor イベントの既存の SQL スクリプトを次のクエリに置き換えます。

```
SELECT "GROUPO"."SalesOrders"."ID",
     "GROUPO"."SalesOrders"."CustomerID",
     "GROUPO"."SalesOrders"."OrderDate",
     "GROUPO"."SalesOrders"."FinancialCode",
     "GROUPO"."SalesOrders"."Region",
     "GROUPO"."SalesOrders"."SalesRepresentative"
FROM "GROUPO"."SalesOrders"
WHERE "GROUPO"."SalesOrders"."Region" = 'Eastern'
AND "GROUPO"."SalesOrders"."ID" IN
(SELECT "DBA"."CustomerProducts"."SalesOrderID"
FROM "DBA"."CustomerProducts");
```

18. 同期モデルを保存します。ファイル > 保存 をクリックします。

同期モデルが完成して、展開の準備が完了します。

結果

Mobile Link プロジェクトおよび同期モデルが作成されます。

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 Anywhere 統合データベースでの Mobile Link の使用 [117 ページ]

前のタスク: レッスン 1: 統合データベースの準備 [120 ページ]

次のタスク: レッスン 3: 同期モデルの展開 [125 ページ]

関連情報

同期モデル [33 ページ]
同期モデルタスク [37 ページ]
テーブルとカラムのマッピング [38 ページ]
ダウンロード方式の変更 [43 ページ]

1.5.2.4 レッスン 3: 同期モデルの展開

同期モデル展開ウィザードを使用すると、統合データベースとリモートデータベースを展開できます。各データベースの展開は 1 つずつ行うこともできますが、両方一度に行うこともできます。同期モデル展開ウィザードでは、展開のオプションを順を追って設定できます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central の左ウィンドウ枠の *Mobile Link 17* で、**mlsqla_project**、同期モデル、**sync_mlsqla** の順に展開します。
2. ファイル > 展開 をクリックします。

   同期モデル展開ウィザードが表示されます。生成されたファイルのデフォルトロケーションを選択し、次へをクリックします。
3. クライアントネットワークオプションページのデフォルトをそのまま使用し、次へをクリックします。
4. *Mobile Link* のユーザとパスワードページで、次のタスクを実行します。
   a. どのような *Mobile Link* のユーザとパスワードを使用しますかで、これらの使用を選択します。
   b. *Mobile Link* ユーザフィールドに、**mlsqla_remote** と入力します。
   c. *Mobile Link* パスワードフィールドに、**mlsqla_pass** と入力します。
   d. "このユーザを統合データベースに登録します。登録されたユーザは同期することができます" をオンにします。

e. 次へをクリックします。

5. 同期プロファイル名を **mlsqla_remote_syncprofile** に変更し、次へをクリックします。
6. データベースの同期を準備する方法を選択しますページにある、統合データベースの同期を準備するために作成された *SQL* スクリプトの処理方法を指定してくださいオプションで、統合データベースに対して実行しますを選択します。
7. リモートデータベースの同期を準備するために作成された *SQL* スクリプトの処理方法を指定してくださいオプションで、新しいリモートデータベースに対して実行しますを選択します。
8. ユーザ *ID* フィールドに、**DBA** と入力します。
9. パスワードおよびパスワードの確認フィールドに、**passwd** と入力します。次へをクリックします。
10. ウィザードによって作成された SQL スクリプトを確認するには、表示をクリックします。
11. 完了をクリックします。
12. 閉じるをクリックします。

結果

統合データベースを複数のリモートクライアントと同期するために設定し、1 つのリモートクライアントが正常に展開されました。他のリモートクライアントを展開する場合は、もう一度このウィザードを実行し、新しい Mobile Link ユーザを作成して統合データベースと Mobile Link サーバの展開を終了します。統合データベースとリモートデータベースはすでに展開されているため、他のリモート同期クライアントの展開のみを実行します。

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: SQL Anywhere 統合データベースでの Mobile Link の使用 [117 ページ]

前のタスク: レッスン 2: 同期モデルの作成 [122 ページ]

次のタスク: レッスン 4: Mobile Link サーバの起動 [127 ページ]

関連情報

同期モデルの展開 [54 ページ]

1.5.2.5 レッスン 4: Mobile Link サーバの起動

このレッスンでは、mlsrv17 -c オプションを使って Mobile Link サーバを起動し、統合データベースに接続します。追加のオプションを使用すると、Mobile Link サーバの動作を設定できます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

手順

1. コマンドプロンプトで、`c:¥mlsqla` ディレクトリに移動します。
2. 次のコマンドを実行して、統合データベースに接続します。

```
mlsrv17 -c "DSN=SQL Anywhere 17 Demo;UID=DBA;PWD=sql" -o mlsrv.mls -v+ -dl -zf -
zu+ -x tcpip
```

このチュートリアルで使用している Mobile Link サーバの各オプションの説明を次に示します。オプション -o、-v、-dl は、デバッグとトラブルシューティングの情報を提供します。これらのロギングオプションは、開発環境での使用に適しています。パフォーマンス上の理由から、一般的に -v+ と -dl は運用環境では使用しません。

オプション	説明
-c	続いて接続文字列を指定します。
-o	メッセージログファイル `mlsrv.mls` を指定します。
-v+	ログを取る対象となる情報を指定します。-v+ を使用して、最大冗長ロギングをオンに設定します。
-dl	画面にすべてのログメッセージを表示します。
-zf	各同期の始めに Mobile Link サーバがスクリプトの変更をチェックします。
-zu+	自動的に新しいユーザを追加します。
-x	Mobile Link クライアントの通信プロトコルとパラメータを設定します。

> **注記**
>
> -zf および -zu+ オプションは、デバッグと開発の目的でのみ使用してください。後のレッスンで新しいスクリプトを統合データベースに追加するときにサーバを停止する必要をなくするため、このチュートリアルでは -zf オプションが必要です。-zu+ オプションは、新しい Mobile Link ユーザを同期環境に自動的に追加します。

Mobile Link サーバメッセージウィンドウが表示されます。

結果

Mobile Link サーバが起動し、統合データベースに接続されます。

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 Anywhere 統合データベースでの Mobile Link の使用 [117 ページ]

前のタスク: レッスン 3: 同期モデルの展開 [125 ページ]

次のタスク: レッスン 5: 同期 [128 ページ]

1.5.2.6 レッスン 5: 同期

このレッスンでは、ulsync ユーティリティで同期を開始することによって、Mobile Link クライアントを Mobile Link サーバと同期します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

手順

1. `c:¥mlsqla¥mlsqla_project¥sync_mlsqla_deploy` ディレクトリに移動し、次のコマンドを実行して **`sync_mlsqla_remote`** データベースを同期します。

   ```
   ulsync -c "DBF=sync_mlsqla_remote.udb"
   "Publications=sync_mlsqla_publication;MobiLinkUid=mlsqla_remote;MobiLinkPwd=mlsql
   a_pass;ScriptVersion=sync_mlsqla_scriptversion;Stream=tcpip{port=2439}"
   ```

   DBF

   実行していないデータベースを起動したときにロードして接続するデータベースファイルを指定します。

   Publications

   同期の実行に使用するリモートデバイスのパブリケーション (このパブリケーションは、同期モデル作成ウィザードで作成されています)。

MobiLinkUid

Mobile Link サーバによる認証に使用するユーザ名。

MobiLinkPwd

Mobile Link サーバによる認証に使用するパスワード。

ScriptVersion

同期の実行時に使用するリモートデバイスのスクリプトバージョン (このパブリケーションは、同期モデル作成ウィザードで作成されています)。

Stream

ネットワークプロトコルを設定するオプションを設定します。

Mobile Link サーバのメッセージウィンドウに同期の進行状況が表示されます。このコマンドが正常に実行されると、ulsync アプリケーションによりリモートデータベースに統合データベースの情報のサブセットが格納されます。

同期が失敗した場合は、ulsync アプリケーションに渡した接続情報、および Mobile Link ユーザ名とパスワードを確認します。それでも失敗する場合は、使用したパブリケーション名を確認し、統合データベースと Mobile Link サーバが実行中であることを確認します。また、同期ログ (サーバ、クライアントとも) の内容を確認することもできます。

> **注記**
>
> 別のコンピュータにある ulsync アプリケーションを Mobile Link サーバから実行している場合、Mobile Link サーバのロケーションを指定する引数を渡す必要があります。

Mobile Link サーバを使用してリモートクライアントを正常に統合データベースに同期すると、リモートデータベースには 1 件の地域に関する情報が格納されます。SQL Anywhere17 プラグインを使用すると、SQL Central でデータベースにデータが移植されているかどうかを確認できます。

2. SQL Central を開きます。
3. 次の手順でリモートデータベースに接続します。
   a. 左ウィンドウ枠で *Ultra Light 17* を右クリックし、接続をクリックします。
   b. ユーザ *ID* に **DBA** と入力し、パスワードに **passwd** と入力します。
   c. データベースファイルフィールドに **C:¥mlsqla¥mlsqla_project¥sync_mlsqla_deploy¥sync_mlsqla_remote.udb** と入力します。
   d. 接続をクリックします。
4. 左ウィンドウ枠で、*Ultra Light 17*、**sync_mlsqla_remote**、テーブル、*Customers* の順に展開します。
5. 右ウィンドウ枠で、データタブをクリックします。

   Customers テーブルで、すべてのレコードは東部地域に関係する顧客のレコードです。この地域は、他の地域の顧客情報とは関係がありません。このため、地域を基準にしてローをフィルタ処理で除外するよう同期スクリプトを設定し、このデータベースのリモート ID を特定の地域識別子の値に設定します。この地域のデータベースの容量は小さくなり、同期に必要な時間も短くなります。リモートデータベースのサイズは最小限に抑えられているため、新しい顧客の入力やモバイルデバイスの変更の処理などの頻繁に行われる処理が迅速かつ効率的に実行されます。

結果

リモートデータベースと統合データベースの間で、データが同期されます。

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 Anywhere 統合データベースでの Mobile Link の使用 [117 ページ]

前のタスク: レッスン 4: Mobile Link サーバの起動 [127 ページ]

次のタスク: レッスン 6: クリーンアップ [130 ページ]

1.5.2.7 レッスン 6: クリーンアップ

チュートリアルをコンピュータから削除します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 以下のアプリケーションのインスタンスをすべて閉じます。
   - SQL Central
   - Interactive SQL
2. 統合データベースとリモートデータベースが保存されている `C:¥mlsqla` ディレクトリを削除します。
3. 次のコマンドを実行して、サンプルデータベースを消去し、新しいサンプルデータベースのコピーを元のオブジェクトおよびデータとともに作成します。

   ```
   newdemo "%SQLANYSAMP17%¥demo.db"
   ```

   プロンプトが表示されたら、既存のファイルをすべて消去することを選択します。

結果

チュートリアルがコンピュータから削除されます。

タスクの概要: チュートリアル: SQL Anywhere 統合データベースでの Mobile Link の使用 [117 ページ]

前のタスク: レッスン 5: 同期 [128 ページ]

1.5.3 チュートリアル: Oracle Database 11g での Mobile Link の使用

このチュートリアルでは、Mobile Link を使用して Oracle Database 10g を活用する方法について説明します。ここでは、Oracle Database 11g と SQL Anywhere リモートデータベースの間の同期を設定します。また、Ultra Light リモートデータベースを設定することもできます。

前提条件

次のソフトウェアが必要です。

- SQL Anywhere17
- Oracle Database 11g Release 2 以降

Oracle データベースで次のパーミッションを持つ必要があります。

- SYS.GV_$TRANSACTION に対する SELECT
- SYS.GV_$SESSION に対する SELECT
- SYS.V_$SESSION に対する SELECT
- SYS.GV_$LOCK に対する SELECT
- SYS.DBMS_UTILITY に対する EXECUTE
- DBA_OBJECTS に対する SELECT

リモートデータベースで次のロールおよび権限を持つ必要があります。

- SYS_REPLICATION_ADMIN_ROLE システムロール
- SYS_RUN_REPLICATION_ROLE システムロール

コンテキスト

このチュートリアルは、販売チームに関するデータを活用することを目的としています。このシナリオでは、各販売担当者はリモート同期クライアントです。各販売担当者にはローカルの SQL Anywhere データベースが用意されています。このデータベースは Mobile Link を使用して本社にある社内 Oracle データベースと同期されます。各販売担当者はラップトップまたはモバイルデバイスを使用して社内データにアクセスし、リモートデータベースからデータを操作します。

このチュートリアルでは、Oracle Database 11g の基本インストールが行われていることを前提としています。このインストールでは、orcl という名前のスターターデータベースが作成されます。orcl データベースには OE (注文エントリ) と HR (人材) というサンプル スキーマがあります。または、Oracle Database Configuration Assistant を使用して新しいデータベースを作成してサンプルスキーマをインストールしたり、SQL*Plus を介して空のデータベースにサンプルスキーマを手動でインストールすることもできます。サンプルスキーマ SQL ファイルは、個別の **Example** ダウンロードパッケージを介して Oracle から直接入手できます。

このチュートリアルでは、SYSDBA 権限がある SYS ユーザとして Oracle に接続できることを前提としています。これは Oracle システムビュー GV_$TRANSACTION に対するパーミッションを付与するための必要条件です。SYS ユーザのパスワードは Oracle データベースのインストール時に設定されます。

このチュートリアルでは、次の作業の方法について説明します。

- リモートスキーマの設計時に、リモートテーブルの同期方向などの重要な考慮事項を決定します。
- 統合データベースとリモートデータベースにユニークなプライマリキーを追加します。
- Mobile Link を Oracle Database 11g に接続する ODBC データソースを作成します。
- 同期モデル作成ウィザードを使用して、統合データベースとリモートデータベースの間の同期を設定します。
- SQL Central を使用して同期モデルをカスタマイズします。
- 同期モデル展開ウィザードを使用して統合データベースとリモートデータベースを展開します。
- リモートクライアントを統合データベースと同期します。

1. スキーマの設計 [133 ページ]
   このチュートリアルでは、OE (注文エントリ) と HR (人材) というサンプルスキーマがインストールされていることを前提としています。OE スキーマは統合データベースとして使用します。このスキーマには従業員、注文、顧客、製品に関する情報がまとめられています。このチュートリアルでは、主に OE スキーマを使用します。
2. レッスン 1: 統合データベースの準備 [134 ページ]
   OE データベースは Mobile Link で使用できるよう変更する必要があります。ユーザ定義型として作成されたカラムは削除されます。また、Mobile Link では OE のクレデンシャルを使用してトリガを作成する必要があるため、トリガを作成するための権限を OE ユーザに付与する必要があります。
3. レッスン 2: ユニークなキーの統合データベースへの追加 [136 ページ]
   同期システムでは、テーブルのプライマリキーは、異なるデータベース内の同じローを識別する唯一の方法であり、競合を検出する唯一の方法です。
4. レッスン 3: Mobile Link の接続 [138 ページ]
   このレッスンでは、Mobile Link を統合データベースに接続する ODBC データソースを作成します。
5. レッスン 4: Mobile Link プロジェクトと同期モデルの作成 [139 ページ]
   このレッスンでは、新しい Mobile Link プロジェクトを作成して、統合データベースに接続します。プロジェクトを作成すると、同期モデルが自動的に作成されます。
6. レッスン 5: 同期モデルの変更 [141 ページ]
   このレッスンでは、新規 Mobile Link プロジェクトの作成時に構築された統合データベースの同期モデルを変更します。
7. レッスン 6: 同期モデルの展開 [143 ページ]
   同期モデル展開ウィザードを使用すると、統合データベースとリモートデータベースを展開できます。各データベースの展開は 1 つずつ行うこともできますが、両方一度に行うこともできます。同期モデル展開ウィザードでは、展開のオプションを順を追って設定できます。
8. レッスン 7: サーバとクライアントの起動 [144 ページ]
   ここまでのレッスンで、ダウンロードカーソルスクリプトを変更して、1 人の販売担当者に関する情報をダウンロードしています。このレッスンでは、リモート ID を販売担当者識別子に設定して販売担当者を指定し、Mobile Link 統合データベースとリモートデータベースを起動します。
9. レッスン 8: リモート ID の設定 [147 ページ]
   このレッスンでは、データベースのリモート ID を有効な販売担当者識別子の値に設定します。
10. レッスン 9: リモートクライアントの同期 [148 ページ]

このレッスンでは、dbmlsync ユーティリティを使用してリモートクライアントを同期します。dbmlsync はリモートデータベースに接続して Mobile Link サーバにより認証されると、リモートデータベースのパブリケーションに基づいてリモートデータベースと統合データベースの同期に必要なすべてのアップロードとダウンロードを実行します。

11. レッスン 10: リモートデータベースのデータの表示 [150 ページ]
    Mobile Link サーバを使用してリモートクライアントを正常に統合データベースに同期すると、リモートデータベースには 1 人の販売担当者に関する情報が格納されます。SQL Anywhere17 プラグインを使用すると、SQL Central でデータベースにデータが正しく移植されているかどうかを確認できます。
12. レッスン 11: クリーンアップ [151 ページ]
    注文エントリのデータベースを再生成して、チュートリアルのすべての教材をコンピュータから削除します。

タスクの概要: Mobile Link チュートリアル [96 ページ]

前のタスク: チュートリアル: SQL Anywhere 統合データベースでの Mobile Link の使用 [117 ページ]

次のタスク: チュートリアル: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

関連情報

Mobile Link 同期 [4 ページ]
SQL Central の Mobile Link プラグイン [24 ページ]
Oracle のサンプルスキーマ

1.5.3.1 スキーマの設計

このチュートリアルでは、OE (注文エントリ) と HR (人材) というサンプルスキーマがインストールされていることを前提としています。OE スキーマは統合データベースとして使用します。このスキーマには従業員、注文、顧客、製品に関する情報がまとめられています。このチュートリアルでは、主に OE スキーマを使用します。

OE スキーマのテーブルのうち、このチュートリアルに関連するテーブルの概要を次に示します。

テーブル	説明
CUSTOMERS	レコードに情報が記録されている顧客。
INVENTORIES	各倉庫に保管されている各製品の数量。
ORDER_ITEMS	各注文の対象となっている製品のリスト。
ORDERS	特定の日付に販売担当者と顧客の間で成立した販売の記録。
PRODUCT_DESCRIPTIONS	各製品に関する、各種言語での説明。
PRODUCT_INFORMATION	システム内の各製品の記録。

リモートスキーマの設計

販売担当者ごとに統合データベース全体をコピーしておくことは不要であり、非効率的です。リモートスキーマは、1 人の特定の販売担当者に関連する情報のみを格納するように設計します。そのため、リモートスキーマは次のように設計します。

統合テーブル	リモートテーブル
CUSTOMERS	すべての行を抽出します。
INVENTORIES	リモートでは使用しません。
ORDER_ITEMS	sales_rep_id を基準にフィルタ処理。
ORDERS	すべての行を抽出します。
PRODUCT_DESCRIPTIONS	リモートでは使用しません。
PRODUCT_INFORMATION	すべての行を抽出します。

各販売担当者はすべての顧客と製品の記録を保持し、すべての製品をどの顧客にも販売できるようにする必要があります。このチュートリアルでは、販売担当者は常に顧客と同じ言語を使用していることを前提としているため、PRODUCT_DESCRIPTIONS テーブルは必要ありません。各販売担当者には注文に関する情報が必要ですが、他の販売担当者に関する注文の情報は不要です。そのため、ローは販売担当者の識別子に基づいてフィルタされます。

次に、各テーブルの同期の方向を選択します。リモートデータベースで読み込む情報と、リモートデータベースで作成、変更、または削除する情報について考慮する必要があります。この例では、各販売担当者は製品と顧客のリストを必要としていますが、新しい製品の情報をシステムに入力することはありません。製品と顧客の情報の入力は必ず本社の統合データベースから行うという制限が設けられています。一方で、販売担当者は新しい販売情報を常時記録する必要があります。このような理由から、各テーブルの同期については次のように決められています。

テーブル	同期
CUSTOMERS	リモートデータベースへのダウンロードのみ。
ORDER_ITEMS	ダウンロードとアップロード。
ORDER	ダウンロードとアップロード。
PRODUCT_INFORMATION	リモートデータベースへのダウンロードのみ。

親トピック チュートリアル: Oracle Database 11g での Mobile Link の使用 [131 ページ]

次のタスク: レッスン 1: 統合データベースの準備 [134 ページ]

1.5.3.2 レッスン 1: 統合データベースの準備

OE データベースは Mobile Link で使用できるよう変更する必要があります。ユーザ定義型として作成されたカラムは削除されます。また、Mobile Link では OE のクレデンシャルを使用してトリガを作成する必要があるため、トリガを作成するための権限を OE ユーザに付与する必要があります。

前提条件

このチュートリアルでは、OE (注文エントリ) サンプルデータベースがインストールされていることを前提としています。

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

Oracle 11g 用のサンプルスキーマのインストールについては、Oracle のマニュアルを参照してください。

SQL Anywhere が認識できる型に変換されたユーザ定義型としてカラムを作成することもできますが、このチュートリアルではこの操作は行いません。

手順

1. SYSDBA 権限を持つ SYS ユーザとして、Oracle SQL Plus アプリケーションを使用して接続します。コマンドプロンプトで次のコマンドを実行します。

   ```
   sqlplus SYS/your password for sys as SYSDBA
   ```

2. ユーザ定義型として作成されたカラムを削除するには、次の文を実行します。

   ```
   ALTER TABLE OE.CUSTOMERS DROP COLUMN CUST_ADDRESS;
   ALTER TABLE OE.CUSTOMERS DROP COLUMN PHONE_NUMBERS;
   ALTER TABLE OE.CUSTOMERS DROP COLUMN CUST_GEO_LOCATION;
   ALTER TABLE OE.PRODUCT_INFORMATION DROP COLUMN WARRANTY_PERIOD;
   ```

3. OE ユーザのロックを解除し、パスワードを sql に設定するには、次の文を実行します。

   ```
   ALTER USER OE IDENTIFIED BY sql ACCOUNT UNLOCK;
   ```

4. OE ユーザがトリガを作成できるようにするには、次の文を実行します。

   ```
   GRANT CREATE ANY TRIGGER TO OE;
   ```

5. orders_customer 外部キーを削除して、CUSTOMERS テーブルの customer_id を参照する新しい外部キーを作成するには、次のコマンドを実行します。

   ```
   ALTER TABLE OE.ORDERS DROP CONSTRAINT ORDERS_CUSTOMER_ID_FK;
   ALTER TABLE OE.ORDERS ADD CONSTRAINT ORDERS_CUSTOMER_ID_FK
     FOREIGN KEY (CUSTOMER_ID) REFERENCES OE.CUSTOMERS (CUSTOMER_ID);
   ```

結果

ユーザ定義型として作成されたカラムが削除され、OE ユーザはトリガを作成できるようになります。

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: Oracle Database 11g での Mobile Link の使用 [131 ページ]

前のトピック: スキーマの設計 [133 ページ]

次のタスク: レッスン 2: ユニークなキーの統合データベースへの追加 [136 ページ]

関連情報

Oracle Database 10g および 11g のマニュアル

1.5.3.3 レッスン 2: ユニークなキーの統合データベースへの追加

同期システムでは、テーブルのプライマリキーは、異なるデータベース内の同じローを識別する唯一の方法であり、競合を検出する唯一の方法です。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

使用する各テーブルには、プライマリキーが必要です。プライマリキーが更新されることはありません。また、1 つのデータベースに挿入されたプライマリキーの値が別のデータベースに挿入されないようにする必要があります。

ユニークなプライマリキーを生成する方法は複数あります。このチュートリアルでは、簡単に操作を行うため、複合プライマリキー方式を使用します。この方式では、統合データベースとリモートデータベースにまたがってユニークな複数のカラムを使用してプライマリキーを作成します。

手順

1. コマンドプロンプトで次のコマンドを実行します。

```
sqlplus SYS/your password for sys as SYSDBA
```

2. SALES_REP_ID に追加する値は HR.EMPLOYEES テーブルに存在する必要があります。ORDERS_SALES_REP_FK 外部キーによりこのルールが強制的に適用されます。次の文を実行して外部キーを削除します。

```
ALTER TABLE OE.ORDERS
DROP CONSTRAINT ORDERS_SALES_REP_FK;
```

3. SALES_REP_ID カラムには NULL 値が含まれているため、このカラムをプライマリキーとして追加することはできません。このチュートリアルでは、NULL 値は 1 に置き換えます。次の文を実行します。

```
UPDATE OE.ORDERS
SET SALES_REP_ID = 1
WHERE SALES_REP_ID IS NULL;
```

4. ORDER_ID カラムは ORDERS テーブルの現在のプライマリキーです。現在のプライマリキーを削除するには、次の文を実行します。

```
ALTER TABLE OE.ORDERS
DROP PRIMARY KEY CASCADE;
```

5. 複合プライマリキーは SALES_REP_ID カラムと ORDER_ID カラムにより構成されます。複合プライマリキーを追加するには、次の文を実行します。

```
ALTER TABLE OE.ORDERS
ADD CONSTRAINT salesrep_order_pk PRIMARY KEY (sales_rep_id, order_id);
```

結果

これらの文を実行した後、Mobile Link サーバでは統合データベースに接続し、任意の数のリモートデータベースとの同期を設定できるようになります。

あとのレッスンでは、統合スキーマからリモートスキーマを作成するため、リモートスキーマは統合スキーマと同じプライマリキーを持つことになります。

プライマリキーがすべてのデータベースに対してユニークになるようにカラムが選択されています。ORDERS テーブルでは、プライマリキーは SALES_REP_ID カラムと ORDER_ID カラムで構成されています。リモートデータベースの sales テーブルに挿入されるすべての値には、ユニークな注文番号が必要です (SALES_REP_ID の値は常に同じです)。これにより、リモートの各 ORDERS テーブルで一意性が確保されます。統合データベースの ORDERS テーブルのプライマリキーは、複数の販売担当者がデータがアップロードした場合の競合を防止する役割があります。販売担当者ごとに SALES_REP_ID の値が異なるため、1 人の販売担当者が行うアップロードはいずれも他の販売担当者に対してユニークです。

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: Oracle Database 11g での Mobile Link の使用 [131 ページ]

前のタスク: レッスン 1: 統合データベースの準備 [134 ページ]

次のタスク: レッスン 3: Mobile Link の接続 [138 ページ]

1.5.3.4 レッスン 3: Mobile Link の接続

このレッスンでは、Mobile Link を統合データベースに接続する ODBC データソースを作成します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. ODBC データソースを作成します。

   SQL Anywhere17 に付属の SQL Anywhere17 - Oracle ODBC ドライバを使用してください。次の設定を使用してください。

ODBC タブのフィールド	値
Data Source Name	oracle_cons
User ID	OE
Password	passwd
TNS Service Name	orcl
Procedure Returns Results Or Uses VARRAY Parameters	selected
Array Size	60000

   このチュートリアルでは、Oracle Database 11g の基本インストールが行われていることを前提としています。このインストールでは、orcl という名前のスターターデータベースが作成されます。OE (注文エントリ) スキーマは自動的に orcl にインストールされます。OE スキーマを別のデータベースにインストールした場合、データベース名を TNS サービス名の値として使用します。

2. テスト接続をクリックして ODBC 接続をテストします。

結果

ODBC データソースが作成され、テストされます。

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: Oracle Database 11g での Mobile Link の使用 [131 ページ]

前のタスク: レッスン 2: ユニークなキーの統合データベースへの追加 [136 ページ]

次のタスク: レッスン 4: Mobile Link プロジェクトと同期モデルの作成 [139 ページ]

関連情報

Mobile Link の推奨 ODBC ドライバ

1.5.3.5 レッスン 4: Mobile Link プロジェクトと同期モデルの作成

このレッスンでは、新しい Mobile Link プロジェクトを作成して、統合データベースに接続します。プロジェクトを作成すると、同期モデルが自動的に作成されます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. スタート ▸ プログラム ▸ *SQL Anywhere17* ▸ 管理ツール ▸ *SQL Central* をクリックします。
2. ツール ▸ *Mobile Link 17* ▸ 新しいプロジェクト をクリックします。

   プロジェクト作成ウィザードが表示されます。

3. 新しいプロジェクトの名前を指定してください。フィールドに **oracle_project** と入力します。
4. 新しいプロジェクトの保存場所を指定してください。フィールドに `C:¥mlora` と入力し、次へをクリックします。
5. データベースの表示名フィールドに **oracle_cons** と入力します。
6. 編集をクリックします。
7. 汎用 *ODBC* データベースに接続ページで次のタスクを実行します。
   a. ユーザ *ID* フィールドに **OE** と入力します。
   b. パスワードフィールドに **passwd** アカウントのパスワードを入力します。
   c. *ODBC* データソース名フィールドで、参照をクリックして **oracle_cons** を選択します。
   d. *OK* をクリックし、保存をクリックします。
8. パスワードを記憶オプションを選択し、次へをクリックします。
9. 統合スキーマの所有者ウィンドウで、選択した所有者のみを対象にデータベーススキーマをロードするを選択し、*OE* ユーザをオンにします。
10. 新しいリモートデータベーススキーマページのリモートデータベースに含める統合データベースのテーブルとカラムを指定してください。リストで、次のテーブルを選択し、次へをクリックします。
    - CUSTOMERS
    - ORDERS
    - ORDER_ITEMS
    - PRODUCT_INFORMATION
11. リモートスキーマ名をプロジェクトに追加オプションを選択します。
12. リモートスキーマ名に **oracle_remote_schema** と入力し、次へをクリックします。
13. *SQL Anywhere* オプションを選択して、完了をクリックします。

    Mobile Link が今回初めて統合データベースを使用する場合、Mobile Link システム設定をインストールするかどうかを確認するメッセージが表示されます。Mobile Link システム設定をインストールすると、Mobile Link システムテーブルと Mobile Link システムプロシージャが追加されます。はいをクリックし、*OK* をクリックします。

結果

Mobile Link プロジェクトおよび同期モデルが作成されます。

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: Oracle Database 11g での Mobile Link の使用 [131 ページ]

前のタスク: レッスン 3: Mobile Link の接続 [138 ページ]

次のタスク: レッスン 5: 同期モデルの変更 [141 ページ]

1.5.3.6 レッスン 5: 同期モデルの変更

このレッスンでは、新規 Mobile Link プロジェクトの作成時に構築された統合データベースの同期モデルを変更します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. *oracle_remote_schema* 同期モデルを右クリックして、プロパティを選択します。
2. 最初のフィールドに **sync_oracle** と入力します。
3. 次のタスクを実行します。
   a. パブリケーション名フィールドに、**sync_oracle_publication** と入力します。
   b. スクリプトバージョン フィールドに **sync_oracle_scriptversion** と入力します。

      パブリケーションは、同期するデータを指定するリモートデータベース上のオブジェクトです。Mobile Link サーバのスクリプトにより、リモートデータベースからアップロードされたデータを統合データベースに適用する方法と、スクリプトバージョンによりスクリプトをグループ化する方法が定義されます。アプリケーションごとに異なるスクリプトバージョンを使用できるため、1 つの Mobile Link サーバを管理するだけで複数のアプリケーションを同期できます。
   c. 適用をクリックしてから *OK* をクリックします。
4. 同期モデルのテーブルごとに、データの同期方向を設定します。

   右ウィンドウ枠で マッピングタブをクリックし、マッピング方向カラムのローを次のように設定します。

   - **ORDERS** と **ORDER_ITEMS** の各テーブルは、双方向 (アップロードとダウンロードの両方) に設定します。
   - 残りのテーブルは、リモートにのみダウンロードに設定します。
5. すべての所有者の統合スキーマのロードのために時間がかかるというウィンドウが表示された場合、**HR** および **OE** ユーザのデータベーススキーマをロードすることを選択します。
6. リモートデータベースにダウンロードされたローを、リモート ID を基準として次のようにフィルタします。
   a. **ORDERS** テーブルを含むローを選択し、右ウィンドウ枠の下部にあるサブセットのダウンロードタブをクリックします。
   b. サブセットのダウンロードカラムをカスタムに変更します。
   c. download_cursor スクリプトの WHERE 句に制限を追加することで、リモート ID を基準としてローをフィルタします。これで、リモートデータベースがユニークに識別されます。

      ダウンロードカーソルの *WHERE* 句で使用する *SQL* 式フィールドに探索条件を入力します。たとえば、次の SQL スクリプトは、**ORDERS** テーブルに使用できます。

      ```
      OE.ORDERS.SALES_REP_ID = {ml s.remote_id}
      ```

ダウンロードカーソルスクリプトは、各テーブルからどのカラムとローをリモートデータベースにダウンロードするかを指定します。探索条件の指定により、1 人の営業担当者 (データベースのリモート ID が一致する営業担当者) に関する情報のみをダウンロードするようにできます。

d. ダウンロード削除サブセットタブをクリックし、ダウンロード削除サブセットをダウンロード済みからすべてに変更します。

7. 同期モデルを保存します。

   ファイル > 保存 をクリックします。

結果

同期モデルが完成して、展開の準備が完了します。

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: Oracle Database 11g での Mobile Link の使用 [131 ページ]

前のタスク: レッスン 4: Mobile Link プロジェクトと同期モデルの作成 [139 ページ]

次のタスク: レッスン 6: 同期モデルの展開 [143 ページ]

関連情報

同期モデルタスク [37 ページ]
テーブルとカラムのマッピング [38 ページ]
ダウンロード方式の変更 [43 ページ]
競合の検出と解決の変更 [50 ページ]

1.5.3.7 レッスン 6: 同期モデルの展開

同期モデル展開ウィザードを使用すると、統合データベースとリモートデータベースを展開できます。各データベースの展開は 1 つずつ行うこともできますが、両方一度に行うこともできます。同期モデル展開ウィザードでは、展開のオプションを順を追って設定できます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central の左ウィンドウ枠の *Mobile Link 17* で、**oracle_project**、同期モデル、**sync_oracle** の順に展開します。
2. ファイル > 展開をクリックします。

   同期モデル展開ウィザードが表示されます。
3. ウィザードによって生成されたファイルを保管するフォルダを選択します。フィールドのデフォルト設定を使用し、次へをクリックします。
4. クライアントネットワークオプションページのデフォルト設定をそのまま使用し、次へをクリックします。
5. どのような *Mobile Link* のユーザとパスワードを使用しますかの下の *Mobile Link* ユーザとパスワードでこれらの使用を選択し、次のタスクを実行します。
   a. *Mobile Link* ユーザフィールドに、**oracle_remote** と入力します。
   b. *Mobile Link* パスワードフィールドに、**oracle_pass** と入力します。
   c. "このユーザを統合データベースに登録します。登録されたユーザは同期することができます" をオンにして、次へをクリックし、もう一度次へをクリックします。
6. 同期プロファイルページで、デフォルト同期プロファイル名を **sync_oracle_publication_oracle_remote** のまま使用します。次へをクリックします。
7. データベースの同期を準備する方法を選択しますページで次のオプションを選択します。
   a. 統合データベースの同期を準備するために作成された *SQL* スクリプトの処理方法を指定してくださいで、統合データベースに対して実行しますを選択します。
   b. リモートデータベースの同期を準備するために作成された *SQL* スクリプトの処理方法を指定してくださいで、新しいリモートデータベースに対して実行しますを選択します。
   c. ユーザ *ID* フィールドに **remoteuser** と入力します。
   d. パスワードおよびパスワードの再入力フィールドに、**remotepass** と入力し、次へをクリックします。
8. 展開ファイルの生成ウィンドウが表示されたら、閉じるをクリックします。
9. 完了をクリックします。
10. 閉じるをクリックします。

結果

統合データベースを複数のリモートクライアントと同期するために設定し、1 つのリモートクライアントが正常に展開されました。他のリモートクライアントを展開する場合は、もう一度このウィザードを実行し、新しい Mobile Link ユーザを作成して統合データベースと Mobile Link サーバの展開を終了します。統合データベースと Mobile Link サーバはすでに展開されているので、実行する必要があるのは、他のリモート同期クライアントを展開することだけです。

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: Oracle Database 11g での Mobile Link の使用 [131 ページ]

前のタスク: レッスン 5: 同期モデルの変更 [141 ページ]

次のタスク: レッスン 7: サーバとクライアントの起動 [144 ページ]

関連情報

同期モデルの展開 [54 ページ]

1.5.3.8 レッスン 7: サーバとクライアントの起動

ここまでのレッスンで、ダウンロードカーソルスクリプトを変更して、1 人の販売担当者に関する情報をダウンロードしています。このレッスンでは、リモート ID を販売担当者識別子に設定して販売担当者を指定し、Mobile Link 統合データベースとリモートデータベースを起動します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

デフォルトでは、Mobile Link は、アップロードとダウンロードに対してスナップショットアイソレーション/READ COMMITTED 独立性レベルを使用します。Mobile Link サーバがスナップショットアイソレーションを最大限有効に利用できるようにするに

は、Mobile Link サーバが使用する Oracle アカウントは、Oracle システムビュー GV_$TRANSACTION にアクセスできる必要があります。アクセスできない場合には、警告が表示され、ローはダウンロードで失われることがあります。

手順

1. SYSDBA 権限を持つ SYS ユーザとして、Oracle SQL Plus アプリケーションを使用して接続します。コマンドプロンプトで次のコマンドを実行します。

   ```
   sqlplus SYS/your password for sys as SYSDBA
   ```

2. Oracle システムビュー GV_$TRANSACTION へのアクセスを許可するには、次の文を実行します。

   ```
   GRANT SELECT ON SYS.GV_$TRANSACTION TO OE;
   ```

3. Oracle システムビュー V$SESSION および GV_$SESSION へのアクセスを許可するには、次の文を実行します。

   ```
   GRANT SELECT ON SYS.V_$SESSION TO OE;
         GRANT SELECT ON SYS.GV_$SESSION TO OE;
   ```

4. 他のシステムオブジェクトへのアクセスを許可するには、次の文を実行します。

   ```
   GRANT SELECT ON SYS.GV_$LOCK TO OE;
   GRANT EXECUTE ON SYS.DBMS_UTILITY TO OE;
   GRANT SELECT ON DBA_OBJECTS TO OE;
   ```

5. コマンドプロンプトで、同期モデルを作成したディレクトリに移動します (このフォルダは、同期モデル作成ウィザードの最初の手順で選択したルートディレクトリです)。

   所定のディレクトリ名を使用している場合は、次のディレクトリに移動します。`mlorc¥oracle_project¥sync_oracle_deploy¥`

6. Mobile Link サーバを起動するには、次のコマンドを実行します。

   ```
   mlsrv.bat "DSN=oracle_cons;UID=OE;PWD=passwd"
   ```

   mlsrv.bat

   Mobile Link サーバを起動するために作成されたコマンドファイル。
   DSN

   ODBC データソース名。
   UID

   統合データベースへの接続に使用するユーザ名。
   PWD

   統合データベースへの接続に使用するパスワード。

   Mobile Link サーバが起動しなかった場合は、統合データベースの接続情報を確認します。

7. コマンドプロンプトで、同期モデル展開ウィザードによりリモートデータベースを作成したディレクトリに移動します。

   所定のディレクトリ名を使用している場合は、次のディレクトリに移動します。`mlora¥oracle_project¥sync_oracle_deploy¥`

8. 次のコマンドを実行して、SQL Anywhere リモートデータベースを起動します。

```
dbsrv17 -n remote_eng sync_oracle_remote.db -n remote_db
```

dbsrv17

SQL Anywhere データベースの起動に使用するデータベースサーバ。
remote_eng

データベースサーバ名。
sync_oracle_remote.db

remote_eng で起動するデータベースファイル。
remote_db

remote_eng にあるデータベースの名前。

結果

このコマンドが正常に実行されると、remote_eng という名前の SQL Anywhere データベースサーバが起動し、remote_db という名前のデータベースがロードされます。

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: Oracle Database 11g での Mobile Link の使用 [131 ページ]

前のタスク: レッスン 6: 同期モデルの展開 [143 ページ]

次のタスク: レッスン 8: リモート ID の設定 [147 ページ]

関連情報

展開した同期モデル [60 ページ]

1.5.3.9 レッスン 8: リモート ID の設定

このレッスンでは、データベースのリモート ID を有効な販売担当者識別子の値に設定します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

リモートスキーマでは、各リモートデータベースは 1 人の販売担当者を表しています。作成した同期スクリプトに含まれている論理により、Mobile Link サーバはリモートデータベースのリモート ID に基づいてデータのサブセットをダウンロードします。リモートデバイスが最初に同期するとき、選択した販売担当者に関するすべての情報がダウンロードされるため、最初の同期の前に、データベースのリモート ID を有効な販売担当者識別子の値に設定する必要があります。

手順

1. 有効な販売担当者識別子を選択します。
   a. SYSDBA 権限を持つ SYS ユーザとして、Oracle SQL Plus アプリケーションを使用して接続します。コマンドプロンプトで次のコマンドを実行します。

      ```
      sqlplus SYS/your-password-for-sys as SYSDBA
      ```

   b. ORDERS テーブルで有効な販売担当者識別子のリストを表示するには、次の文を実行します。

      ```
      SELECT COUNT( SALES_REP_ID ), SALES_REP_ID
      FROM OE.ORDERS GROUP BY SALES_REP_ID;
      ```

      この例では、リモートデータベースは SALES_REP_ID が 154 である販売担当者を表しています。

   c. Oracle SQL Plus を終了するには、次のコマンドを実行します。

      ```
      exit
      ```

2. データベースのリモート ID の値を 154 に設定するには、次のコマンドを実行します。

   ```
   dbisql
   -c "SERVER=remote_eng;DBN=remote_db;UID=DBA;PWD=passwd"
   "SET OPTION PUBLIC.ml_remote_id='154';"
   ```

   dbisql

   SQL Anywhere データベースに対して SQL コマンドを実行するためのアプリケーション。

   ENG

データベースサーバ名として remote_eng を指定します。

DBN

データベース名として remote_db を指定します。

UID

リモートデータベースへの接続に使用するユーザ名。

PWD

リモートデータベースへの接続に使用するパスワード。

SET OPTION PUBLIC.ml_remote_id='154'

リモート ID を 154 に設定するための SQL 文。

結果

データベースのリモート ID が有効な販売担当者識別子の値に設定されます。

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: Oracle Database 11g での Mobile Link の使用 [131 ページ]

前のタスク: レッスン 7: サーバとクライアントの起動 [144 ページ]

次のタスク: レッスン 9: リモートクライアントの同期 [148 ページ]

1.5.3.10 レッスン 9: リモートクライアントの同期

このレッスンでは、dbmlsync ユーティリティを使用してリモートクライアントを同期します。dbmlsync はリモートデータベースに接続して Mobile Link サーバにより認証されると、リモートデータベースのパブリケーションに基づいてリモートデータベースと統合データベースの同期に必要なすべてのアップロードとダウンロードを実行します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

コマンドプロンプトで次のコマンドを実行します。

```
dbmlsync
-c "SERVER=remote_eng;DBN=remote_db;UID=DBA;PWD=sql"
-n sync_oracle_publication
-u oracle_remote -mp oracle_pass
```

dbmlsync

同期アプリケーション。
SERVER

リモートデータベースサーバ名を指定します。
DBN

リモートデータベース名を指定します。
UID

リモートデータベースへの接続に使用するユーザ名を指定します。
PWD

リモートデータベースへの接続に使用するパスワードを指定します。
sync_oracle_publication

同期の実行時に使用するリモートデバイスのパブリケーション (このパブリケーションは、同期モデル作成ウィザードで作成されています)。
oracle_remote

Mobile Link サーバによる認証に使用するユーザ名。
oracle_pass

Mobile Link サーバによる認証に使用するパスワード。

> **注記**
>
> 別のコンピュータにある dbmlsync アプリケーションを Mobile Link サーバから実行している場合、Mobile Link サーバのロケーションを指定する引数を渡す必要があります。

結果

SQL Anywhere Mobile Link クライアントのメッセージウィンドウに同期の進行状況が表示されます。このコマンドが正常に実行されると、dbmlsync アプリケーションによりリモートデータベースに統合データベースの情報のサブセットが格納されます。

同期が失敗した場合は、dbmlsync アプリケーションに渡す接続情報、および Mobile Link ユーザ名とパスワードを確認します。それでも失敗する場合は、使用したパブリケーション名を確認し、統合データベースと Mobile Link サーバが実行中であることを確認します。また、同期ログ (サーバ、クライアントとも) の内容を確認することもできます。

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: Oracle Database 11g での Mobile Link の使用 [131 ページ]

前のタスク: レッスン 8: リモート ID の設定 [147 ページ]

次のタスク: レッスン 10: リモートデータベースのデータの表示 [150 ページ]

関連情報

同期処理 [17 ページ]

1.5.3.11 レッスン 10: リモートデータベースのデータの表示

Mobile Link サーバを使用してリモートクライアントを正常に統合データベースに同期すると、リモートデータベースには 1 人の販売担当者に関する情報が格納されます。SQL Anywhere17 プラグインを使用すると、SQL Central でデータベースにデータが正しく移植されているかどうかを確認できます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central を起動します。
2. 次の手順でリモートデータベースに接続します。
   a. 左ウィンドウ枠で *SQL Anywhere17* を右クリックし、接続をクリックします。
   b. 認証ドロップダウンリストでデータベースをクリックし、ユーザ *ID* に `DBA` と入力し、パスワードに `passwd` と入力します。
   c. アクションドロップダウンリストで、このコンピュータで稼働しているデータベースに接続をクリックします。サーバ名に `remote_eng` と入力し、データベース名に `remote_db` と入力します。
   d. 接続をクリックします。

3. SQL Central の左ウィンドウ枠の *remote_db - DBA* で、テーブルを展開し、ORDERS テーブルをクリックし、右ウィンドウ枠のデータタブをクリックします。

   ORDERS テーブルのすべてのレコードが、識別子が 154 である販売担当者を対象としたものです。この販売担当者は、他の販売担当者の販売情報とは関係ありません。このため、リモート ID を基準にしてローをフィルタ処理で除外するよう同期スクリプトを設定し、このデータベースのリモート ID を特定の販売担当者識別子の値に設定します。この販売担当者のデータベースの容量は小さくなり、同期に必要な時間も短くなります。リモートデータベースのサイズは最小限に抑えられているため、新しい販売記録の入力や過去の販売に対する払い戻し処理などの頻繁に行われる処理が迅速かつ効率的に実行されます。

結果

リモートデータベースに、識別子 154 の販売担当員に関する情報のみが取り込まれます。

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: Oracle Database 11g での Mobile Link の使用 [131 ページ]

前のタスク: レッスン 9: リモートクライアントの同期 [148 ページ]

次のタスク: レッスン 11: クリーンアップ [151 ページ]

1.5.3.12 レッスン 11: クリーンアップ

注文エントリのデータベースを再生成して、チュートリアルのすべての教材をコンピュータから削除します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. Oracle サンプルデータベースを再作成するには、Oracle サンプルデータベース で説明する手順に従ってください。
2. Mobile Link プロジェクトを削除します。
   a. SQL Central を起動します。
   b. 右ウィンドウ枠で、*Mobile Link 17* をダブルクリックします。
   c. **oracle_project** が右ウィンドウ枠に表示されます。
   d. **oracle_project** を右クリックして、削除をクリックします。
   e. プロジェクトの削除ウィンドウで、リストとコンピュータから削除を選択し、はいをクリックします。

結果

注文エントリのデータベースが再生成され、チュートリアルのすべての教材がコンピュータから削除されます。

タスクの概要: チュートリアル: Oracle Database 11g での Mobile Link の使用 [131 ページ]

前のタスク: レッスン 10: リモートデータベースのデータの表示 [150 ページ]

1.5.4 チュートリアル: Adaptive Server Enterprise 統合データベースと Mobile Link の使用

このチュートリアルでは、Mobile Link を使用して Adaptive Server Enterprise データベースを活用する方法について説明します。ここでは、Adaptive Server Enterprise 統合データベースと SQL Anywhere リモートデータベースの間の同期を設定します。また、Ultra Light クライアントも使用できます。

前提条件

次のソフトウェアが必要です。

- SQL Anywhere17
- Adaptive Server Enterprise 15.7

統合データベースで次のロールおよび権限を持つ必要があります。

- MASTER..SYSTRANSACTIONS と MASTER..SYSPROCESSES への SELECT パーミッション

SQL Anywhere リモートで次のロールおよび権限を持つ必要があります。

- SYS_REPLICATION_ADMIN_ROLE システムロール
- SYS_RUN_REPLICATION_ROLE システムロール

コンテキスト

このチュートリアルでは、次の作業の方法について説明します。

- リモートスキーマの設計時に、リモートテーブルの同期方向などの重要な考慮事項を決定します。
- 統合データベースとリモートデータベースにユニークなプライマリキーを追加します。
- Mobile Link を Adaptive Server Enterprise データベースに接続する ODBC データソースを作成します。
- 同期モデル作成ウィザードを使用して、統合データベースとリモートデータベースの間の同期を設定します。
- SQL Central を使用して同期設定をカスタマイズします。
- 同期モデル展開ウィザードを使用して統合データベースとリモートデータベースを展開します。
- リモートクライアントを統合データベースと同期します。

このチュートリアルでは、書店チェーンのデータを活用することを目的とします。このシナリオに登場する各書店は、リモート同期環境です。各書店にはローカル SQL Anywhere データベースがあり、本社の Adaptive Server Enterprise データベースと同期しています。各書店には、リモートデータベースのデータにアクセスして操作できるコンピュータを複数設置することもできます。

このチュートリアルでは、pubs2 サンプルスキーマが Adaptive Server Enterprise 15.7 サーバにインストールされていることを前提としています。pubs2 サンプルスキーマは Adaptive Server Enterprise 15.0 に付属しており、オプションとしてインストールされます。このチュートリアルでは、統合データベースとして使用します。このサンプルに関する情報は、Adaptive Server Enterprise のマニュアルに記載されています。

このチュートリアルでは、デフォルトの **sa** アカウントを使用します。Adaptive Server Enterprise のインストール時点では、**sa** アカウントのパスワードは NULL です。このチュートリアルでは、NULL のパスワードが有効なパスワードに変更されていることを前提としています。

1. スキーマの設計 [154 ページ]
   pubs2 サンプルスキーマは統合データベーススキーマとして使用します。このスキーマには書店、タイトル、著者、出版社、販売に関する情報が格納されています。次の表は、Adaptive Server Enterprise データベースの各テーブルの説明です。
2. レッスン 1: 統合データベースの準備 [156 ページ]
   このレッスンでは、Mobile Link 同期のために統合データベースのサイズを増やします。
3. レッスン 2: ユニークなキーの統合データベースへの追加 [157 ページ]
   このレッスンでは、ユニークなプライマリキーを統合データベースに追加します。
4. レッスン 3: Mobile Link との接続 [160 ページ]
   このレッスンでは、Mobile Link を統合データベースに接続する ODBC データソースを作成します。
5. レッスン 4: Mobile Link プロジェクトと同期モデルの作成 [161 ページ]
   このレッスンでは、新しい Mobile Link プロジェクトを作成して、統合データベースに接続します。新しいプロジェクトを作成すると、同期モデルが自動的に作成されます。
6. レッスン 5: 同期モデルの変更 [163 ページ]
   このレッスンでは、同期モデルを変更します。
7. レッスン 6: 同期モデルの展開 [165 ページ]
   同期モデル展開ウィザードを使用すると、統合データベースとリモートデータベースを展開できます。これらのデータベースの展開は 1 つずつ行うこともできますが、両方一度に行うこともできます。同期モデル展開ウィザードでは、展開のオプションを順を追って設定できます。
8. レッスン 7: サーバとクライアントの起動 [166 ページ]
   このレッスンでは、Mobile Link サーバとリモートデータベースを起動します。

9. レッスン 8: リモート ID の設定 [168 ページ]
   リモートスキーマでは、各リモートデータベースは 1 軒の書店を表しています。同期スクリプトに含まれている論理により、Mobile Link サーバはリモートデータベースのリモート ID に基づいてデータのサブセットをダウンロードします。データベースのリモート ID は有効な書店識別子の値に設定する必要があります。
10. レッスン 9: 同期 [170 ページ]
    このレッスンでは、dbmlsync ユーティリティを使用して、リモートクライアントを初めて同期します。
11. レッスン 10: リモートデータベースのデータの表示 [172 ページ]
    Mobile Link サーバを使用してリモートクライアントを正常に統合データベースに同期すると、リモートデータベースには 1 軒の書店に関する情報が格納されます。SQL Anywhere 17 プラグインを使用すると、SQL Central のリモートデータベースの内容を確認できます。
12. レッスン 11: クリーンアップ [173 ページ]
    pubs2 データベースを再生成して、チュートリアルのすべての教材をコンピュータから削除します。

タスクの概要: Mobile Link チュートリアル [96 ページ]

前のタスク: チュートリアル: Oracle Database 11g での Mobile Link の使用 [131 ページ]

次のタスク: チュートリアル: カスタムユーザ認証用の Java と .NET の使用 [174 ページ]

関連情報

Mobile Link 同期 [4 ページ]

1.5.4.1 スキーマの設計

pubs2 サンプルスキーマは統合データベーススキーマとして使用します。このスキーマには書店、タイトル、著者、出版社、販売に関する情報が格納されています。次の表は、Adaptive Server Enterprise データベースの各テーブルの説明です。

テーブル	説明
au_pix	著者の写真。
authors	システムに登録されている各タイトルの著者。
discounts	特定の書店で行われている各種の割引の記録。
sales	各販売レコードは、特定の書店での 1 件の販売を表します。
salesdetail	1 件の販売で対象となった各タイトルに関する情報。
stores	各店舗レコードは、システムに登録されている 1 軒の書店または支店を表しています。
titleauthor	どのタイトルがどの著者によって執筆されたかに関する情報。
titles	システムに登録されているすべての本の記録。

テーブル	説明
blurbs, publishers, and roysched	このチュートリアルでは必要ない情報。

リモートスキーマの設計

書店ごとに統合データベース全体をコピーしておくことは不要であり、非効率的です。リモートスキーマでは同じテーブル名を使用しますが、各書店に関する情報だけが格納されます。この設定を実現するため、リモートスキーマは統合データベースのサブセットとして次のように設計されています。

統合テーブル	リモートテーブル
au_pix	すべての行を抽出します。
authors	すべての行を抽出します。
discounts	stor_id を基準にフィルタ処理。
sales	stor_id を基準にフィルタ処理。
salesdetail	stor_id を基準にフィルタ処理。
stores	stor_id を基準にフィルタ処理。
titleauthor	すべての行を抽出します。
titles	すべての行を抽出します。
blurbs	リモートでは使用しません。
publishers	リモートでは使用しません。
roysched	リモートでは使用しません。

各書店では、すべてのタイトルと著者の記録を保持し、顧客が書店の在庫を検索できるようにする必要があります。一方で、出版社や印税に関する情報は書店では必要ないため、これらの情報は各書店に対しては同期されません。各書店では販売と割引に関する情報が必要ですが、他の書店の販売や割引に関する情報は必要ありません。そのため、ローは書店の識別子に基づいてフィルタされます。

> **注記**
>
> リモートデータベースで不要になるカラムがある場合は、テーブルからカラムのサブセットを取得することもできます。

次に、各テーブルの同期の方向を選択します。リモートデータベースで読み込む情報と、リモートデータベースで作成、変更、または削除する情報について考慮します。この例では、書店は著者とタイトルのリストにアクセスする必要がありますが、新しい著者名をシステムに入力することはありません。このため、著者とタイトルは必ず本社の統合データベースから入力するという制限が適用されています。一方で、書店では新しい販売情報を常時記録する必要があります。これらの要因から、各テーブルの同期の方向は次のようになります。

テーブル	同期
titleauthor	リモートデータベースへのダウンロードのみ。
authors	リモートデータベースへのダウンロードのみ。
au_pix	リモートデータベースへのダウンロードのみ。

テーブル	同期
titles	リモートデータベースへのダウンロードのみ。
stores	リモートデータベースへのダウンロードのみ。
discounts	リモートデータベースへのダウンロードのみ。
sales	ダウンロードとアップロード。
salesdetail	ダウンロードとアップロード。

親トピック チュートリアル: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

次のタスク: レッスン 1: 統合データベースの準備 [156 ページ]

1.5.4.2 レッスン 1: 統合データベースの準備

このレッスンでは、Mobile Link 同期のために統合データベースのサイズを増やします。

前提条件

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

Mobile Link では、同期のためにシステムテーブルなどのオブジェクトを pubs2 データベースに追加する必要があります。これらのオブジェクトを追加する場合は、pubs2 データベースのサイズを増やす必要があります。

手順

1. Adaptive Server Enterprise の isql を使用して、pubs2 データベースに **sa** として接続します。コマンドプロンプトで、次のコマンドをすべて 1 行に入力して実行します。

   ```
   isql
   -U sa
   -P your-password-for-sa-account
   -D pubs2
   ```

   Adaptive Server Enterprise にリモートでアクセスしている場合は、-S オプションでサーバ名を指定します。

2. データベースのサイズを増やすための所定のパーミッションを得るには、master データベースにアクセスする必要があります。isql で次のコマンドを実行します。

```
use master
go
sp_dboption pubs2, "select into/bulkcopy/pllsort", true
go
```

3. Adaptive Server Enterprise では、データベースはディスクまたはディスクの一部に保存されます。pubs2 データベースのサイズを増やすには、次の文を実行します (pubs2 が格納されているディスクを指定する必要があります)。

```
ALTER DATABASE pubs2 ON disk-name = 33
```

結果

Mobile Link 同期のための統合データベースのサイズが増えます。

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: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

前のトピック: スキーマの設計 [154 ページ]

次のタスク: レッスン 2: ユニークなキーの統合データベースへの追加 [157 ページ]

1.5.4.3 レッスン 2: ユニークなキーの統合データベースへの追加

このレッスンでは、ユニークなプライマリキーを統合データベースに追加します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

同期システムでは、テーブルのプライマリキーは、異なるデータベース内の同じローを識別する唯一の方法であり、競合を検出する唯一の方法です。使用する各テーブルには、プライマリキーが必要です。プライマリキーが更新されることはありません。また、1 つのデータベースに挿入されたプライマリキーの値が別のデータベースに挿入されないようにする必要があります。

ユニークなプライマリキーを生成する方法は複数あります。このチュートリアルでは、簡単に操作を行うため、複合プライマリキー方式を使用します。この方式では、統合データベースとリモートデータベースにまたがってユニークな複数のカラムを使用してプライマリキーを作成します。

手順

1. Adaptive Server Enterprise の isql を使用して、pubs2 データベースに **sa** として接続します。コマンドプロンプトで、次のコマンドをすべて 1 行に入力して実行します。

   ```
   isql
   -U sa
   -P your-password-for-sa-account
   -D pubs2
   ```

   Adaptive Server Enterprise にリモートでアクセスしている場合は、-S オプションでサーバ名を指定します。
2. 次のローは、salesdetail テーブルに対して作成した複合プライマリキーを基準とするとユニークではありません。操作を簡単にするために、次の文を実行してこのローを削除します。

   ```
   DELETE FROM salesdetail
   WHERE stor_id = '5023'
   AND ord_num = 'NF-123-ADS-642-9G3'
   AND title_id = 'PC8888'
   DELETE FROM salesdetail
   WHERE stor_id = '5023'
   AND ord_num = 'ZS-645-CAT-415-1B2'
   AND title_id = 'BU2075'
   ```
3. 次のインデックスは、前の手順でのプライマリキーの作成に干渉しています。このインデックスを削除するには、次の文を実行します。

   ```
   DROP INDEX authors.auidind
   DROP INDEX titleauthor.taind
   DROP INDEX titles.titleidind
   DROP INDEX sales.salesind
   ```
4. 次の文を実行して、ユニークなプライマリキーを追加します。

   ```
   ALTER TABLE au_pix ADD PRIMARY KEY (au_id)
   ALTER TABLE authors ADD PRIMARY KEY (au_id)
   ALTER TABLE titleauthor ADD PRIMARY KEY (au_id, title_id)
   ALTER TABLE titles ADD PRIMARY KEY (title_id)
   ALTER TABLE discounts ADD PRIMARY KEY (discounttype)
   ALTER TABLE stores ADD PRIMARY KEY (stor_id)
   ALTER TABLE sales ADD PRIMARY KEY (stor_id, ord_num)
   ALTER TABLE salesdetail ADD PRIMARY KEY (stor_id, ord_num, title_id)
   ```

   これらの文を実行した後、Mobile Link サーバでは統合データベースに接続し、任意の数のリモートデータベースとの同期を設定できるようになります。

> **注記**
>
> プライマリキーがない統合データベースとデータを同期することも可能です。ただし、他のテーブルでローを一意に識別するよう設計されたシャドウテーブルで機能する同期イベントを自分で作成する必要があります。

あとのレッスンでは、統合スキーマからリモートスキーマを作成するため、リモートスキーマは統合スキーマと同じプライマリキーを持つことになります。

プライマリキーがすべてのデータベースに対してユニークになるようにカラムが選択されています。sales テーブルでは、プライマリキーは stor_id と ord_num カラムで構成されています。リモートデータベースの sales テーブルに挿入されるすべての値には、ユニークな注文番号が必要です (stor_id の値は常に同じです)。これにより、リモートの各 sales テーブルで一意性が確保されます。統合データベースの sales テーブルのプライマリキーは、複数の書店でデータがアップロードされた場合の競合を防止する役割があります。書店ごとに stor_id の値が異なるため、1 軒の書店からのアップロードはいずれも他の書店に対してユニークです。

salesdetail テーブルでは、プライマリキーは stor_id、ord_num、title_id の各カラムで構成されています。1 件の注文に複数の本のタイトルが存在する場合もあります。リモートデータベースの sales テーブルでは、stor_id と ord_num については複数のローで同じ値になってもかまいませんが、title_id の値はローごとに異なる必要があります。この設定により、各リモートデータベースの salesdetail テーブルで一意性が確保されます。sales テーブルと同様に、書店ごとに stor_id の値が異なるため、1 軒の書店から統合データベースへのアップロードはいずれも他の書店に対してユニークです。

結果

ユニークでないローは削除され、ユニークなプライマリキーが統合データベースに追加されます。

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: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

前のタスク: レッスン 1: 統合データベースの準備 [156 ページ]

次のタスク: レッスン 3: Mobile Link との接続 [160 ページ]

1.5.4.4 レッスン 3: Mobile Link との接続

このレッスンでは、Mobile Link を統合データベースに接続する ODBC データソースを作成します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. ODBC データソースを作成します。

   Adaptive Server Enterprise に付属の ODBC ドライバを使用する必要があります。このチュートリアルでは、次の設定を使用します。

[一般] タブのフィールド	値
Data Source Name	ase_cons
Description	
Server Name (ASE Host Name)	localhost
Server Port	5000
Database Name	pubs2
Logon ID	sa
Use Cursors	not selected

[トランザクション] タブのフィールド	値
Server Initiated Transactions	not selected

2. ODBC 接続をテストします。
   a. 一般タブでテスト接続をクリックします。

      Adaptive Server Enterprise のログオン画面が表示されます。
   b. **sa** アカウントのパスワードを入力し、*OK* をクリックします。

      ログインに成功しましたというメッセージが表示されます。

結果

ODBC データソースが作成され、テストされます。

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: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

前のタスク: レッスン 2: ユニークなキーの統合データベースへの追加 [157 ページ]

次のタスク: レッスン 4: Mobile Link プロジェクトと同期モデルの作成 [161 ページ]

関連情報

Mobile Link の推奨 ODBC ドライバ

1.5.4.5 レッスン 4: Mobile Link プロジェクトと同期モデルの作成

このレッスンでは、新しい Mobile Link プロジェクトを作成して、統合データベースに接続します。新しいプロジェクトを作成すると、同期モデルが自動的に作成されます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. スタート > プログラム > *SQL Anywhere17* > 管理ツール > *SQL Central* をクリックします。
2. ツール > *Mobile Link 17* > 新しいプロジェクト をクリックします。

   プロジェクト作成ウィザードが表示されます。
3. 名前フィールドに **ase_project** と入力します。
4. ロケーションフィールドに `C:¥mlase` と入力し、次へをクリックします。
5. データベースの表示名フィールドに **ase_cons** と入力します。
6. 編集をクリックします。
7. 汎用 *ODBC* データベースに接続ページで次のタスクを実行します。

a. ユーザ *ID* フィールドに **sa** と入力します。
b. パスワードフィールドに sa アカウントのパスワードを入力します。
c. *ODBC* データソース名フィールドで、参照をクリックして **ase_cons** を選択します。
d. *OK* をクリックし、保存をクリックします。

8. パスワードを記憶オプションを選択し、次へをクリックします。固定長の文字カラムについての警告が表示されたら、*OK* をクリックして次の手順に進みます。
9. 新しいリモートデータベーススキーマページのリモートデータベースに含める統合データベースのテーブルとカラムを指定してください。リストで、次のテーブルを選択します。
   - au_pix
   - authors
   - discounts
   - sales
   - salesdetail
   - stores
   - titleauthor
   - titles

   次へをクリックします。
10. リモートスキーマ名をプロジェクトに追加オプションを選択します。
11. リモートスキーマ名に **ase_remote_schema** と入力し、次へをクリックします。
12. *SQL Anywhere* をオンにして、完了をクリックします。

    Mobile Link が今回初めて統合データベースを使用する場合、Mobile Link システム設定をインストールするかどうかを確認するメッセージが表示されます。Mobile Link システム設定をインストールすると、Mobile Link システムテーブルとプロシージャが追加されます。はいをクリックし、*OK* をクリックします。

結果

Mobile Link プロジェクトおよび同期モデルが作成されます。

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: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

前のタスク: レッスン 3: Mobile Link との接続 [160 ページ]

次のタスク: レッスン 5: 同期モデルの変更 [163 ページ]

1.5.4.6 レッスン 5: 同期モデルの変更

このレッスンでは、同期モデルを変更します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. ase_remote_schema 同期モデルを右クリックして、プロパティを選択します。
2. 次のタスクを実行します。
   a. 最初のフィールドに、**sync_ase** と入力します。
   b. パブリケーション名フィールドに、**sync_ase_publication** と入力します。
   c. スクリプトバージョン フィールドに **sync_ase_scriptversion** と入力します。

      パブリケーションは、同期するデータを指定するリモートデータベース上のオブジェクトです。Mobile Link サーバのスクリプトにより、リモートデータベースからアップロードされたデータを統合データベースに適用する方法と、スクリプトバージョンによりスクリプトをグループ化する方法が定義されます。アプリケーションごとに異なるスクリプトバージョンを使用できるため、1 つの Mobile Link サーバを管理するだけで複数のアプリケーションを同期できます。
   d. 適用をクリックしてから *OK* をクリックします。
3. 同期モデルのテーブルごとに、データの同期方向を設定します。

   右ウィンドウ枠で マッピングタブをクリックし、マッピング方向カラムのローを次のように設定します。
   - **sales** と **salesdetail** の各テーブルは、双方向 (アップロードとダウンロードの両方) に設定します。
   - 残りのテーブルは、リモートにのみダウンロードに設定します。
4. リモートデータベースにダウンロードされたローを、リモート ID を基準として次のようにフィルタします。
   a. **stores** テーブルが含まれるローを選択し、サブセットのダウンロードタブをクリックします。
   b. サブセット方式をカスタムに変更します。
   c. download_cursor スクリプトの WHERE 句に制限を追加することで、リモート ID を基準としてローをフィルタします。これで、リモートデータベースがユニークに識別されます。

      ダウンロードカーソルの *WHERE* 句で使用する *SQL* 式フィールドに探索条件を入力します。たとえば、次の SQL スクリプトは、**stores** テーブルに使用できます。

      ```
      "dbo"."stores"."stor_id" = {ml s.remote_id}
      ```

      ダウンロードカーソルスクリプトは、各テーブルからどのカラムとローをリモートデータベースにダウンロードするかを指定します。探索条件の指定により、1 つの書店 (データベースのリモート ID が一致する書店) に関する情報のみをダウンロードするようにできます。
   d. ダウンロード削除サブセットタブをクリックし、ダウンロード削除サブセットをダウンロード済みからすべてに変更します。

5. **sales**、**salesdetail**、**discounts** の各テーブルを含むローについて、前の手順を繰り返します。

   注記

   SQL スクリプトで指定されたテーブルの名前は、編集するロー内のテーブル名に変更してください。

   sales テーブルには、次の WHERE 句スクリプトを使用します。

   ```
   "dbo"."sales"."stor_id" = {ml s.remote_id}
   ```

   salesdetail テーブルには、次の WHERE 句スクリプトを使用します。

   ```
   "dbo"."salesdetail"."stor_id" = {ml s.remote_id}
   ```

   discounts テーブルには、次の WHERE 句スクリプトを使用します。

   ```
   "dbo"."discounts"."stor_id" = {ml s.remote_id}
   ```

6. 同期モデルを保存します。

   ファイル > 保存 をクリックします。

結果

同期モデルが完成して、展開の準備が完了します。

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: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

前のタスク: レッスン 4: Mobile Link プロジェクトと同期モデルの作成 [161 ページ]

次のタスク: レッスン 6: 同期モデルの展開 [165 ページ]

関連情報

同期モデルタスク [37 ページ]
テーブルとカラムのマッピング [38 ページ]
ダウンロード方式の変更 [43 ページ]
競合の検出と解決の変更 [50 ページ]

1.5.4.7 レッスン 6: 同期モデルの展開

同期モデル展開ウィザードを使用すると、統合データベースとリモートデータベースを展開できます。これらのデータベースの展開は 1 つずつ行うこともできますが、両方一度に行うこともできます。同期モデル展開ウィザードでは、展開のオプションを順を追って設定できます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central の左ウィンドウ枠の *Mobile Link 17* で、**ase_project**、同期モデル、**sync_ase** の順に展開します。
2. ファイル > 展開をクリックします。
3. ウィザードによって生成されたファイルを保管するフォルダを選択します。フィールドのデフォルト設定を使用し、次へをクリックします。
4. クライアントネットワークオプションページのデフォルト設定をそのまま使用し、次へをクリックします。
5. どのような *Mobile Link* のユーザとパスワードを使用しますかの下の *Mobile Link* ユーザとパスワードでこれらの使用を選択し、次のタスクを実行します。
   - *Mobile Link* ユーザフィールドに、**ase_remote** と入力します。
   - *Mobile Link* パスワードフィールドに、**ase_pass** と入力します。
   - "このユーザを統合データベースに登録します。登録されたユーザは同期することができます" をオンにして、次へをクリックします。
6. 同期プロファイルページで、同期プロファイル名フィールドに **sync_ase_profile** と入力し、次へをクリックします。
7. データベースの同期を準備する方法を選択しますページで次のタスクを実行します。
   a. 統合データベースの同期を準備するために作成された *SQL* スクリプトの処理方法を指定してくださいで、統合データベースに対して実行しますを選択します。
   b. リモートデータベースの同期を準備するために作成された *SQL* スクリプトの処理方法を指定してくださいで、新しいリモートデータベースに対して実行しますを選択します。
   c. ユーザ *ID* フィールドに、**sa** と入力します。
   d. パスワードおよびパスワードの再入力フィールドに、**passwd** と入力し、次へをクリックします。
8. 展開ファイルの生成ウィンドウが表示されたら、閉じるをクリックします。
9. 完了をクリックします。
10. 閉じるをクリックします。

結果

統合データベースを複数のリモートクライアントと同期するための設定がすべて完了し、1 つのリモートクライアントが正常に展開されました。他のリモートクライアントを展開する場合は、もう一度このウィザードを実行し、新しい Mobile Link ユーザを作成して統合データベースと Mobile Link サーバの展開を終了します。これらのものについては展開がすでに終了しているため、あとは他のリモート同期クライアントを展開するだけです。

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: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

前のタスク: レッスン 5: 同期モデルの変更 [163 ページ]

次のタスク: レッスン 7: サーバとクライアントの起動 [166 ページ]

関連情報

同期モデルの展開 [54 ページ]

1.5.4.8 レッスン 7: サーバとクライアントの起動

このレッスンでは、Mobile Link サーバとリモートデータベースを起動します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

ここまでのレッスンで、ダウンロードカーソルスクリプトを変更して、1 軒の書店に関する情報をダウンロードしています。このレッスンでは、リモート ID を書店識別子に設定して、書店を指定します。

手順

1. コマンドプロンプトで、同期モデルを作成したフォルダに移動します (このフォルダは、同期モデル作成ウィザードの最初の手順で選択したルートディレクトリです)。

   所定のディレクトリ名を使用している場合は、次のディレクトリに移動します。`mlase¥ase_project¥sync_ase_deploy¥`
2. Mobile Link サーバを起動するには、次のコマンドを実行します。

   ```
   mlsrv.bat "DSN=ase_cons;UID=sa;PWD=passwd;"
   ```

   mlsrv.bat

   Mobile Link サーバを起動するためのコマンドファイル。
   dsn

   ODBC データソース名。
   uid

   統合データベースへの接続に使用するユーザ名。
   pwd

   指定したユーザとして接続するためのパスワード。

   Mobile Link サーバが起動しなかった場合は、統合データベースの接続情報を確認します。
3. コマンドプロンプトで、同期モデル展開ウィザードによりリモートデータベースを作成したディレクトリに移動します。

   所定のディレクトリ名を使用している場合は、次のディレクトリに移動します。`mlase¥ase_project¥sync_ase_deploy¥`
4. SQL Anywhere リモートデータベースを起動するには、次のコマンドを入力します。

   ```
   dbsrv17 -n remote_eng sync_ase_remote.db -n remote_db
   ```

   dbsrv17

   SQL Anywhere データベースの起動に使用するデータベースサーバ。
   remote_eng

   データベースサーバ名。
   sync_ase_remote.db

   remote_eng で起動するデータベースファイル。
   remote_db

   remote_eng にあるデータベースの名前。

結果

remote_eng という名前の SQL Anywhere データベースサーバが起動し、remote_db と呼ばれるデータベースをロードします。

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: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

前のタスク: レッスン 6: 同期モデルの展開 [165 ページ]

次のタスク: レッスン 8: リモート ID の設定 [168 ページ]

関連情報

展開した同期モデル [60 ページ]

1.5.4.9 レッスン 8: リモート ID の設定

リモートスキーマでは、各リモートデータベースは 1 軒の書店を表しています。同期スクリプトに含まれている論理により、Mobile Link サーバはリモートデータベースのリモート ID に基づいてデータのサブセットをダウンロードします。データベースのリモート ID は有効な書店識別子の値に設定する必要があります。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

リモートデバイスが最初に同期する際に、書店 (ここでは Thoreau Reading Discount Chain) に関するすべての情報がダウンロードされるため、この手順は最初の同期の前に完了している必要があります。

手順

1. 有効な書店識別子を選択します。

a. Adaptive Server Enterprise の isql を使用して、pubs2 データベースに **sa** として接続します。コマンドプロンプトで、次のコマンドをすべて 1 行に入力して実行します。

```
isql
-U sa
-P your-password-for-sa-account
-D pubs2
```

Adaptive Server Enterprise にリモートでアクセスしている場合は、-S オプションでサーバ名を指定します。

b. stores テーブルで有効な書店識別子のリストを表示するには、次の文を実行します。

```
SELECT * FROM stores
```

このチュートリアルでは、リモートデータベースは Thoreau Reading Discount Chain という名前の書店を表しています。書店識別子は 5023 です。

c. isql を終了するには、次のコマンドを実行します。

```
exit
```

2. データベースのリモート ID を 5023 に設定するには、次のコマンドをすべて 1 行で入力して実行します。

```
dbisql
    -c "SERVER=remote_eng;DBN=remote_db;UID=DBA;PWD=sql"
    "SET OPTION PUBLIC.ml_remote_id='5023'"
```

dbisql

SQL Anywhere データベースに対して SQL コマンドを実行するためのアプリケーション。

server

データベースサーバ名として remote_eng を指定します。

dbn

データベース名として remote_db を指定します。

uid

リモートデータベースへの接続に使用するユーザ名を指定します。

pwd

リモートデータベースへの接続に使用するパスワードを指定します。

SET OPTION PUBLIC.ml_remote_id='5023'

リモート ID を 5023 に設定するための SQL コマンド。

結果

データベースのリモート ID が 5023 に設定されます。これは書店識別子の値です。

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: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

前のタスク: レッスン 7: サーバとクライアントの起動 [166 ページ]

次のタスク: レッスン 9: 同期 [170 ページ]

1.5.4.10 レッスン 9: 同期

このレッスンでは、dbmlsync ユーティリティを使用して、リモートクライアントを初めて同期します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

dbmlsync は、リモートデータベースに接続し、リモートデータベースから同期情報をロードし、トランザクションログをスキャンして、アップロードデータを生成します。次に、dbmlsync は、Mobile Link サーバに接続し、Mobile Link サーバにより認証されると、リモートデータベースのパブリケーションに基づいてリモートデータベースと統合データベースを同期するのに必要なすべてのアップロードとダウンロードを実行します。

手順

コマンドプロンプトで、次のコマンドをすべて 1 行に入力して実行します。

```
dbmlsync -c "SERVER=remote_eng;DBN=remote_db;UID=DBA;PWD=sql;"
    -n sync_ase_publication
    -u ase_remote -mp ase_pass
```

dbmlsync

同期アプリケーション。

SERVER

リモートデータベースサーバ名を指定します。
DBN

リモートデータベース名を指定します。
UID

リモートデータベースへの接続に使用するユーザ名を指定します。
PWD

リモートデータベースへの接続に使用するパスワードを指定します。
sync_ase_publication

同期の実行に使用するリモートデバイスのパブリケーション名 (このパブリケーションは、同期モデル作成ウィザードで作成されています)。
ase_remote

Mobile Link サーバによる認証に使用するユーザ名。
ase_pass

Mobile Link サーバによる認証に使用するパスワード。

> **注記**
>
> 別のコンピュータにある dbmlsync アプリケーションを Mobile Link サーバから実行している場合、Mobile Link サーバのロケーションを指定する引数を渡す必要があります。

結果

SQL Anywhere Mobile Link クライアントのメッセージウィンドウに同期の進行状況が表示されます。このコマンドが正常に実行されると、dbmlsync アプリケーションによりリモートデータベースに統合データベースの情報のサブセットが格納されます。

同期が失敗した場合は、dbmlsync アプリケーションに渡す接続情報、および Mobile Link ユーザ名とパスワードを確認します。それでも失敗する場合は、使用したパブリケーション名を確認し、統合データベースと Mobile Link サーバが実行中であることを確認します。また、同期ログ (サーバ、クライアントとも) の内容を確認することもできます。

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: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

前のタスク: レッスン 8: リモート ID の設定 [168 ページ]

次のタスク: レッスン 10: リモートデータベースのデータの表示 [172 ページ]

関連情報

同期処理 [17 ページ]

1.5.4.11 レッスン 10: リモートデータベースのデータの表示

Mobile Link サーバを使用してリモートクライアントを正常に統合データベースに同期すると、リモートデータベースには 1 軒の書店に関する情報が格納されます。SQL Anywhere 17 プラグインを使用すると、SQL Central のリモートデータベースの内容を確認できます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central を起動します。
2. 次の手順でリモートデータベースに接続します。
   a. 左ウィンドウ枠で *SQL Anywhere17* を右クリックし、接続をクリックします。
   b. 認証ドロップダウンリストからデータベースを選択し、次の手順を実行します。
      1. ユーザ *ID* フィールドに、**DBA** と入力します。
      2. パスワードフィールドに、**passwd** と入力します。
   c. アクションドロップダウンリストから、このコンピュータで稼働しているデータベースに接続を選択します。
   d. サーバ名フィールドに **remote_eng** と入力し、データベース名フィールドに **remote_db** と入力します。
   e. 接続をクリックします。
3. 統合データベースから作成されたテーブルが表示されていない場合は、次の手順を実行します。
   a. `remote_db` を右クリックして所有者フィルタの設定をクリックします。
   b. *dbo* を選択して *OK* をクリックします。

      統合データベースから作成されたテーブルが左ウィンドウ枠に表示されます。dbo がこれらのテーブルに対して持っている所有権はリモートデータベースで保持されます。
4. 任意のリモートテーブルを選択して、右ウィンドウ枠のデータタブをクリックします。

   sales、salesdetail、stores の各テーブルでは、すべてのレコードが識別子 5023 の書店に関するものです。この書店は、他の書店の販売情報とは関係ありません。このため、リモート ID を基準にしてローをフィルタ処理で除外するよう同期スクリプトを設定し、このデータベースのリモート ID を特定の書店識別子の値に設定します。この書店のデータベースは容量が少なく、同期に必要な時間も短くなります。リモートデータベースのサイズは最小限に抑えられているため、新し

い販売記録の入力や過去の販売に対する払い戻し処理などの頻繁に行われる処理が迅速かつ効率的に実行されます。

結果

リモートデータベース内のデータが表示されます。

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: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

前のタスク: レッスン 9: 同期 [170 ページ]

次のタスク: レッスン 11: クリーンアップ [173 ページ]

1.5.4.12 レッスン 11: クリーンアップ

pubs2 データベースを再生成して、チュートリアルのすべての教材をコンピュータから削除します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. pubs2 データベースを再生成します。

   pubs2 データベースをインストールするスクリプトを実行するには、次のコマンドを実行します。

   ```
   isql
   -U sa
   -P your-password-for-sa-account
   -i %SYBASE%¥%SYBASE_ASE%¥scripts¥instpbs2
   ```

Adaptive Server Enterprise にリモートでアクセスしている場合は、-S オプションでサーバ名を指定します。また、instpbs2 ファイルをローカルのコンピュータにコピーする必要があります。-i オプションを更新して、instpbs2 ファイルの新しいロケーションを指定する必要があります。

2. 同期モデルを削除します。
   a. SQL Central を起動します。
   b. 右ウィンドウ枠で *Mobile Link 17* をダブルクリックします。

      sync_ase モデルが表示されます。
   c. **sync_ase** を右クリックして、削除を選択します。
3. dberase ユーティリティを使用して、リモートデータベースを消去します。

   次のコマンドを実行します。

```
dberase sync_ase¥remote¥sync_ase_remote.db
```

結果

pubs2 データベースが再生成され、チュートリアルのすべての教材がコンピュータから削除されます。

タスクの概要: チュートリアル: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

前のタスク: レッスン 10: リモートデータベースのデータの表示 [172 ページ]

1.5.5 チュートリアル: カスタムユーザ認証用の Java と .NET の使用

Mobile Link 同期スクリプトは、SQL、Java、または .NET で作成できます。Java または .NET を使用すると、同期処理の任意の時点にカスタムアクションを追加できます。

前提条件

次の知識と経験が必要です。

- Java または .NET の知識
- Mobile Link イベントスクリプトの基本的な知識

次のソフトウェアが必要です。

- SQL Anywhere17
- Java ソフトウェア開発キットまたは Microsoft .NET Framework

CustDB データベースで次の権限を持つ必要があります。

- SYS_REPLICATION_ADMIN_ROLE システムロール

- CREATE ANY TRIGGER システム権限
- CREATE ANY VIEW システム権限
- EXECUTE ANY PROCEDURE システム権限

コンテキスト

このチュートリアルでは、authenticate_user 接続イベントに対する Java または .NET メソッドを追加します。authenticate_user イベントを使用すると、カスタム認証スキームを指定して、Mobile Link の組み込みクライアント認証を無効にできます。

このチュートリアルの目的は、Mobile Link カスタム認証について、知識と経験を得ることです。このチュートリアルでは、次の作業の方法について説明します。

- Mobile Link サーバ API リファレンスを使用した、ソースファイルのコンパイル
- 特定のテーブルレベルイベント用のクラスメソッドの指定
- -sl オプションを使用した、Mobile Link サーバ (mlsrv17) の実行
- サンプル Windows クライアントアプリケーションを使用した、同期のテスト

1. レッスン 1: カスタム認証用の Java クラスの作成 (サーバ側) [176 ページ]
   このレッスンでは、カスタム認証用の Java 論理を記述するクラスをコンパイルします。
2. レッスン 2: カスタム認証用の .NET クラスの作成 (サーバ側) [177 ページ]
   このレッスンでは、カスタム認証用の .NET 論理を記述するクラスをコンパイルします。
3. レッスン 3: authenticate_user イベント用の Java または .NET スクリプトの登録 [179 ページ]
   SQL Anywhere には、同期用に設定されるサンプルデータベース (CustDB) が付属しています。たとえば、CustDB の ULCustomer テーブルは、さまざまなテーブルレベルスクリプトをサポートする同期テーブルです。このレッスンでは、authenticate_user 同期イベント用の MobiLinkAuth authenticateUser メソッドを登録します。このスクリプトを CustDB に追加します。
4. レッスン 4: Mobile Link サーバの起動 [181 ページ]
   このレッスンでは、-sl オプションを指定して Mobile Link サーバを実行し、コンパイル済みファイルを検索するための一連のディレクトリを指定します。
5. レッスン 5: 認証のテスト [182 ページ]
   Ultra Light には、同期を開始できるサンプル Windows クライアントが用意されています。このレッスンでは、前のレッスンで起動した CustDB 統合データベースに対してアプリケーションを実行します。
6. レッスン 6: クリーンアップ [183 ページ]
   すべてのチュートリアルをコンピュータから削除し、Windows サンプルアプリケーションのデータベースをリセットします。

タスクの概要: Mobile Link チュートリアル [96 ページ]

前のタスク: チュートリアル: Adaptive Server Enterprise 統合データベースと Mobile Link の使用 [152 ページ]

次のタスク: チュートリアル: ダイレクトローハンドリングの使用 [184 ページ]

1.5.5.1 レッスン 1: カスタム認証用の Java クラスの作成 (サーバ側)

このレッスンでは、カスタム認証用の Java 論理を記述するクラスをコンパイルします。

前提条件

Java 同期ロジックを実行するには、Mobile Link サーバが `mlscript.jar` のクラスにアクセスできることが必要です。`mlscript.jar` には、Java メソッドで利用する Mobile Link サーバ API クラスのレポジトリが含まれています。Java クラスをコンパイルするときは、`mlscript.jar` を参照します。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

顧客認証用の .NET クラスを作成するには、カスタム認証用の .NET クラスの作成 (サーバ側) を参照してください。

手順

1. MobiLinkAuth というクラスを作成し、authenticateUser メソッドを作成します。

   MobiLinkAuth クラスには、authenticate_user 同期イベントで使用する authenticateUser メソッドが含まれています。authenticate_user イベントは、ユーザパラメータとパスワードパラメータを提供します。認証結果は、authentication_status inout パラメータを使用して返します。

   > **注記**
   >
   > authenticate_user 同期イベントの authenticateUser メソッドは、レッスン 2 で登録します。authenticate_user イベント用の Java または .NET スクリプトを登録します。

   サーバアプリケーションに次のコードを使用します。

```
import com.sap.ml.script.*;
public class MobiLinkAuth {
    public void authenticateUser (
        com.sap.ml.script.InOutInteger authentication_status,
        String user,
        String pwd,
        String newPwd ) {
        if (user.startsWith("128")) {
            // success: an auth status code of 1000
            authentication_status.setValue(1000);
        } else {
            // fail: an authentication_status code of 4000
            authentication_status.setValue(4000);
        }
    }
```

```
}
```

このコードは、簡単なカスタムユーザ認証の例を示します。`128` で始まるユーザ名を使用してクライアントが統合データベースにアクセスすると、認証に成功します。

2. コードを保存します。このチュートリアルでは、`c:¥MLauth` をサーバ側コンポーネントの作業ディレクトリとします。ファイルを `MobiLinkAuth.java` という名前でこのディレクトリに保存します。
3. クラスファイルをコンパイルします
   a. Java ファイルが含まれるディレクトリに移動します。
   b. MobiLinkAuth クラスをコンパイルし、Mobile Link サーバ Java API ライブラリを参照します。

      次のコマンドを実行して、*C:¥Program Files¥SQL Anywhere 17¥* を SQL Anywhere17 ディレクトリに置き換えます。

      ```
      javac MobiLinkAuth.java -classpath "C:¥Program Files¥SQL Anywhere 17¥java
      ¥mlscript.jar"
      ```

結果

`MobiLinkAuth.class` ファイルが生成されます。

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: カスタムユーザ認証用の Java と .NET の使用 [174 ページ]

次のタスク: レッスン 2: カスタム認証用の .NET クラスの作成 (サーバ側) [177 ページ]

1.5.5.2 レッスン 2: カスタム認証用の .NET クラスの作成 (サーバ側)

このレッスンでは、カスタム認証用の .NET 論理を記述するクラスをコンパイルします。

前提条件

.NET 同期ロジックを実行するには、Mobile Link サーバが `Sap.MobiLink.Script.dll` のクラスにアクセスできることが必要です。`Sap.MobiLink.Script.dll` には、.NET メソッドで利用する Mobile Link .NET サーバ API クラスのレポジトリが含まれています。.NET クラスをコンパイルするときは、`Sap.MobiLink.Script.dll` を参照します。

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

コンテキスト

顧客認証用の .NET クラスを作成するには、カスタム認証用の Java クラスの作成 (サーバ側) を参照してください。

手順

1. MobiLinkAuth というクラスを作成し、authenticateUser メソッドを作成します。

   MobiLinkAuth クラスには、authenticate_user 同期イベントで使用する authenticateUser メソッドが含まれています。authenticate_user イベントは、ユーザパラメータとパスワードパラメータを提供します。認証結果は、authentication_status inout パラメータを使用して返します。

   > **注記**
   >
   > authenticate_user 同期イベントの authenticateUser メソッドは、レッスン 2 で登録します。authenticate_user イベント用の Java または .NET スクリプトを登録します。

   サーバアプリケーションに次のコードを使用します。

   ```
   using Sap.MobiLink.Script;
   public class MobiLinkAuth {
       public void authenticateUser(
           ref int authentication_status,
           string user,
           string pwd,
           string newPwd ) {
           if(user.StartsWith("128")) {
               // success: an auth status code of 1000
               authentication_status = 1000;
           } else {
               // fail: and authentication_status code of 4000
               authentication_status = 4000;
           }
       }
   }
   ```

   このコードは、簡単なカスタムユーザ認証の例を示します。**128** で始まるユーザ名を使用してクライアントが統合データベースにアクセスすると、認証に成功します。
2. コードを保存します。このチュートリアルでは、`c:¥MLauth` をサーバ側コンポーネントの作業ディレクトリとします。ファイルを `MobiLinkAuth.cs` という名前でこのディレクトリに保存します。
3. クラスファイルをコンパイルします
   a. C# ファイルが含まれるディレクトリに移動します。
   b. MobiLinkAuth クラスをコンパイルし、Mobile Link サーバ .NET API ライブラリを参照します。

      次のコマンドを実行して、*C:¥Program Files¥SQL Anywhere 17¥* を SQL Anywhere17 ディレクトリに置き換えます。

      ```
      csc /out:MobiLinkAuth.dll /target:library /reference:"C:¥Program Files¥SQL
      Anywhere 17¥Assembly¥v3.5¥Sap.MobiLink.Script.dll" MobiLinkAuth.cs
      ```

結果

`MobiLinkAuth.dll` アセンブリが生成されます。

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: カスタムユーザ認証用の Java と .NET の使用 [174 ページ]

前のタスク: レッスン 1: カスタム認証用の Java クラスの作成 (サーバ側) [176 ページ]

次のタスク: レッスン 3: authenticate_user イベント用の Java または .NET スクリプトの登録 [179 ページ]

1.5.5.3 レッスン 3: authenticate_user イベント用の Java または .NET スクリプトの登録

SQL Anywhere には、同期用に設定されるサンプルデータベース (CustDB) が付属しています。たとえば、CustDB の ULCustomer テーブルは、さまざまなテーブルレベルスクリプトをサポートする同期テーブルです。このレッスンでは、authenticate_user 同期イベント用の MobiLinkAuth authenticateUser メソッドを登録します。このスクリプトを CustDB に追加します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

CustDB は、Ultra Light クライアントと SQL Anywhere クライアントの両方の統合データベースサーバとなるように設計されています。CustDB 統合データベースは、ODBC データソース SQL Anywhere 17 CustDB を使用します。

手順

1. Interactive SQL からサンプルデータベースに接続します。

   次のコマンドを実行します。

   ```
   dbisql -c "DSN=SQL Anywhere 17 CustDB;uid=ml_server;pwd=sql"
   ```

2. ml_add_java_connection_script または ml_add_dnet_connection_script ストアドプロシージャを使用して、authenticate_user イベント用の authenticateUser メソッドを登録します。

   Java の場合は、次の SQL 文を実行します。

   ```
   CALL ml_server.ml_add_java_connection_script(
       'custdb 17.0',
       'authenticate_user',
       'MobiLinkAuth.authenticateUser');
   COMMIT;
   ```

   .NET の場合は、次の SQL 文を実行します。

   ```
   CALL ml_add_dnet_connection_script(
       'custdb 17.0',
       'authenticate_user',
       'MobiLinkAuth.authenticateUser');
   COMMIT;
   ```

結果

authenticate_user イベント用の authenticateUser メソッドが登録されます。

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: カスタムユーザ認証用の Java と .NET の使用 [174 ページ]

前のタスク: レッスン 2: カスタム認証用の .NET クラスの作成 (サーバ側) [177 ページ]

次のタスク: レッスン 4: Mobile Link サーバの起動 [181 ページ]

1.5.5.4 レッスン 4: Mobile Link サーバの起動

このレッスンでは、-sl オプションを指定して Mobile Link サーバを実行し、コンパイル済みファイルを検索するための一連のディレクトリを指定します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

手順

CustDB サンプルデータベースに接続して、mlsrv17 コマンドラインで Java クラスまたは .NET アセンブリをロードします。

`c:¥MLauth` は、ソースファイルがある実際のディレクトリに置き換えてください。

Java の場合は、次のコマンドを実行します。

```
mlsrv17 -c "DSN=SQL Anywhere 17 CustDB;uid=ml_server;pwd=sql" -o serverOut.txt -v+ -
sl java(-cp c:¥MLauth)
```

.NET の場合は、次のコマンドを実行します。

```
mlsrv17 -c "DSN=SQL Anywhere 17 CustDB;uid=ml_server;pwd=sql" -o serverOut.txt -v+ -
sl dnet(-MLAutoLoadPath=c:¥MLauth)
```

結果

authenticate_user synchronization イベントが発生すると、MobiLinkAuth メソッドが実行されます。

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: カスタムユーザ認証用の Java と .NET の使用 [174 ページ]

前のタスク: レッスン 3: authenticate_user イベント用の Java または .NET スクリプトの登録 [179 ページ]

次のタスク: レッスン 5: 認証のテスト [182 ページ]

1.5.5.5 レッスン 5: 認証のテスト

Ultra Light には、同期を開始できるサンプル Windows クライアントが用意されています。このレッスンでは、前のレッスンで起動した CustDB 統合データベースに対してアプリケーションを実行します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. サンプルアプリケーションを起動します。

   スタート ▶ プログラム ▶ *SQL Anywhere17* ▶ *Ultra Light* ▶ *Windows* サンプルアプリケーション をクリックします。
2. 無効な従業員 ID を入力して同期します。

   このアプリケーションでは、従業員 ID は Mobile Link ユーザ名でもあります。`128` で始まらないユーザ名の場合、論理により同期が失敗します。Employee ID に値 `50` を入力し、*OK* をクリックします。

結果

authenticate_user スクリプトからエラー 4000 が返されたことが、Mobile Link サーバのメッセージウィンドウに表示されます。

ユーザ ID またはパスワードが無効であることを示す SQLCODE -1497 の同期エラーが *Ultra Light CustDB* デモウィンドウに表示されます。

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: カスタムユーザ認証用の Java と .NET の使用 [174 ページ]

前のタスク: レッスン 4: Mobile Link サーバの起動 [181 ページ]

次のタスク: レッスン 6: クリーンアップ [183 ページ]

関連情報

Mobile Link の CustDB サンプル [65 ページ]

1.5.5.6 レッスン 6: クリーンアップ

すべてのチュートリアルをコンピュータから削除し、Windows サンプルアプリケーションのデータベースをリセットします。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. Java または .NET のソースファイルを削除します。

   たとえば、`c:¥mlauth` ディレクトリを削除します。

   > **警告**
   >
   > このディレクトリには、チュートリアル関連のファイルのみが含まれていることを確認してください。

2. Interactive SQL と Ultra Light Windows クライアントアプリケーションを終了します。

   各アプリケーションで、ファイル > 終了 をクリックします。

3. SQL Anywhere、Mobile Link、同期クライアントの各ウィンドウを閉じます。

   それぞれのタスクバーを右クリックして、閉じるをクリックします。

4. Windows サンプルアプリケーションのデータベースをリセットします。

   %SQLANYSAMP17%`¥UltraLite¥CustDB` ディレクトリから次のコマンドを実行します。

   ```
   makedbs
   ```

結果

すべてのチュートリアルがコンピュータから削除され、Windows サンプルアプリケーションのデータベースがリセットされます。

タスクの概要: チュートリアル: カスタムユーザ認証用の Java と .NET の使用 [174 ページ]

前のタスク: レッスン 5: 認証のテスト [182 ページ]

1.5.6 チュートリアル: ダイレクトローハンドリングの使用

ダイレクトローハンドリングを使用すると、サポートされている統合データベース以外にも、リモートデータと中央のデータソース、アプリケーション、または Web サービスとの通信ができます。

前提条件

次の知識と経験が必要です。

- Java または .NET の知識
- Mobile Link イベントスクリプトの基本的な知識

次のソフトウェアが必要です。

- SQL Anywhere17
- Java ソフトウェア開発キットまたは Microsoft .NET Framework

統合データベースで次のロールおよび権限を持つ必要があります。

- SYS_AUTH_RESOURCE_ROLE 互換ロール
- MONITOR システム権限

リモートデータベースで次のロールおよび権限を持つ必要があります。

- SYS_REPLICATION_ADMIN_ROLE システムロール
- SYS_RUN_REPLICATION_ROLE システムロール

コンテキスト

このチュートリアルでは、Java 用または .NET 用の Mobile Link サーバ API を使用して簡単なダイレクトローハンドリングを行う方法を習得します。また、クライアントの RemoteOrders テーブルを統合データベースと同期し、OrderComments テーブル用に特別なダイレクトローハンドリング処理を追加する方法も習得します。

このチュートリアルでは、次の作業の方法について説明します。

- Java 用または .NET 用の Mobile Link サーバ API の使用
- Mobile Link ダイレクトローハンドリング用のメソッドの作成

1. レッスン 1: テキストファイルデータソースの設定 [186 ページ]
   このレッスンでは、注文情報を保存する新しいテキストファイルを作成します。

2. レッスン 2: Mobile Link 統合データベースの設定 [187 ページ]
   このレッスンでは、データベースを作成し、ODBC データソースを定義します。
3. レッスン 3: Mobile Link 統合データベースでのテーブルの作成 [188 ページ]
   このレッスンでは、Mobile Link 統合データベースに RemoteOrders テーブルを作成します。
4. レッスン 4: 同期スクリプトの追加 [190 ページ]
   ダイレクトローハンドリングを使用し、ストアドプロシージャを使用して Mobile Link 統合データベースに同期スクリプト情報を追加します。このレッスンでは、handle_UploadData、handle_DownloadData、end_download、download_cursor、download_delete_cursor の各イベントに対応するメソッド名を登録します。独自の Java または .NET クラスをレッスンの後半で作成します。
5. レッスン 5: Mobile Link のダイレクトローハンドリングのための Java または .NET クラスの作成 [193 ページ]
   このレッスンでは、ダイレクトローハンドリングを使用して、クライアントデータベース内の OrderComments テーブルのローを処理します。
6. レッスン 6: Mobile Link サーバの起動 [204 ページ]
   このレッスンでは、Mobile Link サーバを起動します。-c オプションを使って Mobile Link サーバ (mlsrv17) を起動し、統合データベースに接続します。-sl java オプションまたは -sl dnet オプションを使用して、Java クラスまたは .NET クラスをそれぞれロードします。
7. レッスン 7: Mobile Link クライアントデータベースの設定 [205 ページ]
   このレッスンでは、SQL Anywhere データベースを統合データベースと Mobile Link クライアントに使用します。また、このチュートリアルの目的上、Mobile Link クライアント、統合データベース、および Mobile Link サーバはすべて同じコンピュータに置きます。
8. レッスン 8: 同期 [207 ページ]
   dbmlsync ユーティリティを使用して、SQL Anywhere リモートデータベースの Mobile Link 同期を開始します。dbmlsync を起動する前に、注文データとコメントをリモートデータベースに追加します。
9. レッスン 9: クリーンアップ [209 ページ]
   チュートリアルをコンピュータから削除します。

タスクの概要: Mobile Link チュートリアル [96 ページ]

前のタスク: チュートリアル: カスタムユーザ認証用の Java と .NET の使用 [174 ページ]

次のタスク: チュートリアル: Microsoft Excel との同期 [210 ページ]

関連情報

Mobile Link 同期 [4 ページ]
SAP SQL Anywhere フォーラム

1.5.6.1 レッスン 1: テキストファイルデータソースの設定

このレッスンでは、注文情報を保存する新しいテキストファイルを作成します。

前提条件

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 新しい空のテキストファイルを作成します。
2. comment_id、order_id、order_comment を表す、次のタブで区切った値をファイルに追加します。

```
786    34    OK, ship promotional material.
787    35    Yes, the product is going out of production.
788    36    No, your commission cannot be increased...
```

3. そのファイルを作業ディレクトリに保存します。このチュートリアルでは、`c:¥MLdirect` をサーバ側コンポーネントの作業ディレクトリとします。ファイルを `orderResponses.txt` という名前でこのディレクトリに保存します。

結果

テキストファイルが作成されます。

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: ダイレクトローハンドリングの使用 [184 ページ]

次のタスク: レッスン 2: Mobile Link 統合データベースの設定 [187 ページ]

1.5.6.2 レッスン 2: Mobile Link 統合データベースの設定

このレッスンでは、データベースを作成し、ODBC データソースを定義します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

Mobile Link 統合データベースはデータの中央レポジトリであり、同期処理の管理に使用する Mobile Link のシステムテーブルとストアドプロシージャが含まれます。ダイレクトローハンドリングでは、統合データベース以外のデータソースと同期しますが、Mobile Link サーバが使用する情報を保持するために統合データベースも必要です。

> **注記**
>
> Mobile Link 統合データベースを Mobile Link システムオブジェクトと ODBC データソースを使用して設定済みの場合は、このレッスンは省略できます。

手順

1. スタート > プログラム > *SQL Anywhere17* > 管理ツール > *SQL Central* をクリックします。
2. ツール > *SQL Anywhere17* > データベースの作成 をクリックします。
3. 次へをクリックします。
4. デフォルト値である、このコンピュータにデータベースを作成を受け入れて、次へをクリックします。
5. メインデータベースファイルを保存フィールドに、データベースのファイル名およびパスを入力します。たとえば、`c:¥MLdirect¥MLconsolidated.db` のように入力します。
6. データベース作成ウィザードの残りの指示に従い、デフォルト値をそのまま使用します。DBA ユーザのユーザ ID とパスワードを指定するように求められたら、それぞれ **DBA** と **passwd** を入力します。

   データベースへの接続ページで、最終切断後にデータベースを停止オプションをオフにします。
7. 完了をクリックします。

   MLconsolidated データベースが SQL Central に表示されます。
8. ウィンドウが自動的に閉じない場合は、データベースの作成ウィンドウの閉じるをクリックします。
9. SQL Anywhere17 ドライバを使用して、MLconsolidated データベース用の ODBC データソースを定義します。

   SQL Central で、ツール > *SQL Anywhere17* > *ODBC アドミニストレータを開く* をクリックします。

10. **ユーザ** *DSN* タブをクリックし、**追加**をクリックします。
11. **データソースの新規作成**ウィンドウで、*SQL Anywhere17* をクリックし、**完了**をクリックします。
12. *SQL Anywhere* **の** *ODBC* **設定**ウィンドウで、次の操作を行います。
    a. *ODBC* タブをクリックします。
    b. **データソース名**フィールドに **mldirect_db** と入力します。
    c. **ログイン**タブをクリックします。
    d. **ユーザ** *ID* フィールドに、**DBA** と入力します。
    e. **パスワード**フィールドに、**passwd** と入力します。
    f. **サーバ名**フィールドに **MLconsolidated** と入力します。
    g. *OK* をクリックします。
13. ODBC データソースアドミニストレータを閉じます。

    ODBC **データソースアドミニストレータ**ウィンドウで *OK* をクリックします。

結果

統合データベースと ODBC データソ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: ダイレクトローハンドリングの使用 [184 ページ]

前のタスク: レッスン 1: テキストファイルデータソースの設定 [186 ページ]

次のタスク: レッスン 3: Mobile Link 統合データベースでのテーブルの作成 [188 ページ]

1.5.6.3 レッスン 3: Mobile Link 統合データベースでのテーブルの作成

このレッスンでは、Mobile Link 統合データベースに RemoteOrders テーブルを作成します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

作成する RemoteOrders テーブルには次のカラムが含まれます。

order_id

注文のユニークな識別子です。
product_id

製品のユニークな識別子です。
quantity

品目の販売数です。
order_status

注文のステータスです。
last_modified

ローが最後に変更された日です。このカラムはタイムスタンプベースのダウンロードに使用します。このダウンロード方法は、効率的な同期のためにローをフィルタする一般的な方法です。

手順

1. Interactive SQL からデータベースに接続します。

   Interactive SQL は、SQL Central またはコマンドプロンプトから起動できます。

   - SQL Central から Interactive SQL を起動するには、**MLconsolidated - DBA** データベースを右クリックし、*Interactive SQL* を開くをクリックします。
   - コマンドプロンプトで Interactive SQL を起動するには、次のコマンドを実行します。

     ```
     dbisql -c "DSN=mldirect_db"
     ```

2. Interactive SQL で次の SQL 文を実行し、RemoteOrders テーブルを作成します。

   ```
   CREATE TABLE RemoteOrders (
       order_id            INTEGER NOT NULL,
       product_id          INTEGER NOT NULL,
       quantity            INTEGER,
       order_status        VARCHAR(10) DEFAULT 'new',
       last_modified       TIMESTAMP DEFAULT CURRENT TIMESTAMP,
       PRIMARY KEY(order_id)
   );
   ```

   Interactive SQL によって、統合データベースに RemoteOrders テーブルが作成されます。

3. Interactive SQL で次の文を実行して Mobile Link のシステムテーブルとストアドプロシージャを作成します。

   C:¥Program Files¥SQL Anywhere 17¥ は、SQL Anywhere17 インストール環境のロケーションに置き換えてください。

   ```
   READ "C:¥Program Files¥SQL Anywhere 17¥MobiLink¥setup¥syncsa.sql";
   ```

Interactive SQL によって `syncsa.sql` が統合データベースに適用されます。`syncsa.sql` を実行すると、前に **ml_** が付いた一連のシステムテーブルとストアドプロシージャが作成されます。これらのテーブルとストアドプロシージャは、同期処理中に Mobile Link サーバによって使用されます。

結果

RemoteOrders テーブルが作成され、Mobile Link のシステムテーブルとストアドプロシージャが統合データベースに追加されます。

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: ダイレクトローハンドリングの使用 [184 ページ]

前のタスク: レッスン 2: Mobile Link 統合データベースの設定 [187 ページ]

次のタスク: レッスン 4: 同期スクリプトの追加 [190 ページ]

1.5.6.4 レッスン 4: 同期スクリプトの追加

ダイレクトローハンドリングを使用し、ストアドプロシージャを使用して Mobile Link 統合データベースに同期スクリプト情報を追加します。このレッスンでは、handle_UploadData、handle_DownloadData、end_download、download_cursor、download_delete_cursor の各イベントに対応するメソッド名を登録します。独自の Java または .NET クラスをレッスンの後半で作成します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

SQL ローハンドリングを使用すると、リモートデータを、Mobile Link 統合データベース内のテーブルと同期できます。SQL ベースのスクリプトでは、次の情報を定義します。

- Mobile Link クライアントからアップロードするデータを統合データベースに適用する方法。
- 統合データベースからダウンロードするデータ。

次の SQL ベースのアップロードイベントとダウンロードイベントが作成されます。

upload_insert

このイベントは、クライアントデータベースに挿入された新しい注文を統合データベースに適用する方法を定義します。
download_cursor

このイベントは、リモートクライアントにダウンロードする注文を定義します。
download_delete_cursor

このイベントは、アップロード専用の同期スクリプトを使用する場合に必要です。このチュートリアルでは、このイベントを無視するように Mobile Link サーバを設定します。

手順

1. まだ接続していない場合は、次のコマンドを実行して、Interactive SQL から統合データベースに接続します。

```
dbisql -c "DSN=mldirect_db"
```

2. ml_add_table_script ストアドプロシージャを使用して、upload_insert、download_cursor、download_delete_cursor の各イベント用の SQL ベースのテーブルスクリプトを追加します。

   Interactive SQL で次の SQL 文を実行します。upload_insert のスクリプトでは、アップロードされた order_id、product_id、quantity、order_status を Mobile Link 統合データベースに挿入します。download_cursor のスクリプトでは、タイムスタンプベースのフィルタを使用して、更新されたローをリモートクライアントにダウンロードします。

```
CALL ml_add_table_script( 'default', 'RemoteOrders',
    'upload_insert',
    'INSERT INTO RemoteOrders( order_id, product_id, quantity, order_status)
     VALUES( {ml r.order_id}, {ml r.product_id}, {ml r.quantity}, {ml
r.order_status} )' );

CALL ml_add_table_script( 'default', 'RemoteOrders',
    'download_cursor',
    'SELECT order_id, product_id, quantity, order_status
     FROM RemoteOrders WHERE last_modified >= {ml s.last_table_download}');
CALL ml_add_table_script( 'default', 'RemoteOrders',
    'download_delete_cursor', '--{ml_ignore}');
COMMIT;
```

3. end_download イベント用の Java または .NET メソッドを登録します。

   Mobile Link サーバが end_download 接続イベントを実行するときに、このメソッドを使用してメモリリソースを解放します。

   Java の場合は、Interactive SQL で次の文を実行します。

```
CALL ml_add_java_connection_script( 'default',
    'end_synchronization',
    'MobiLinkOrders.EndSync' );
```

   .NET の場合は、Interactive SQL で次の文を実行します。

```
CALL ml_add_dnet_connection_script( 'default',
```

```
    'end_synchronization',
    'MobiLinkOrders.EndSync' );
```

Interactive SQL によって、ユーザ定義の EndDownload メソッドが end_download イベント用に登録されます。

4. handle_UploadData と handle_DownloadData の各イベント用の Java または .NET メソッドを登録します。

   Java の場合は、Interactive SQL で次の文を実行します。

```
CALL ml_add_java_connection_script( 'default',
    'handle_UploadData',
    'MobiLinkOrders.GetUpload' );

CALL ml_add_java_connection_script( 'default',
    'handle_DownloadData',
    'MobiLinkOrders.SetDownload' );
```

   .NET の場合は、Interactive SQL で次の文を実行します。

```
CALL ml_add_dnet_connection_script( 'default',
    'handle_UploadData',
    'MobiLinkOrders.GetUpload' );

CALL ml_add_dnet_connection_script( 'default',
    'handle_DownloadData',
    'MobiLinkOrders.SetDownload' );
```

   Interactive SQL によって、ユーザ定義の GetUpload と SetDownload の各メソッドが、それぞれ handle_UploadData と handle_DownloadData の各イベント用に登録されます。これらのメソッドは次のレッスンで作成します。

5. download_cursor と download_delete_cursor の各イベントを登録します。

   Interactive SQL で次の文を実行します。

```
CALL ml_add_table_script( 'default', 'OrderComments',
    'download_cursor', '--{ml_ignore}');
CALL ml_add_table_script( 'default', 'OrderComments',
    'download_delete_cursor', '--{ml_ignore}');
```

   同期は双方向であり、アップロード専用ではないため、スクリプトを使用するときには、OrderComments テーブル用に download_cursor と download_delete_cursor の各イベントを登録してください。

6. これまでの変更内容をコミットします。

   Interactive SQL で次の文を実行します。

```
COMMIT;
```

7. Interactive SQL を閉じます。

結果

handle_UploadData、handle_DownloadData、end_download、download_cursor、および download_delete_cursor イベントに対応するメソッド名が登録されます。

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: ダイレクトローハンドリングの使用 [184 ページ]

前のタスク: レッスン 3: Mobile Link 統合データベースでのテーブルの作成 [188 ページ]

次のタスク: レッスン 5: Mobile Link のダイレクトローハンドリングのための Java または .NET クラスの作成 [193 ページ]

1.5.6.5 レッスン 5: Mobile Link のダイレクトローハンドリングのための Java または .NET クラスの作成

このレッスンでは、ダイレクトローハンドリングを使用して、クライアントデータベース内の OrderComments テーブルのローを処理します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

コンテキスト

ダイレクトローハンドリング用に次のメソッドを追加します。

GetUpload

このメソッドは handle_UploadData イベントに使用します。GetUpload では、アップロードされたコメントを `orderComments.txt` というファイルに書き込みます。

SetDownload

このメソッドは handle_DownloadData イベントに使用します。SetDownload では、`orderResponses.txt` ファイルを使用してリモートクライアントに応答をダウンロードします。

EndDownload

このメソッドを end_download イベントに使用します。EndDownload では、メモリリソースが解放されます。

次の手順では、処理用メソッドを含む Java または .NET のクラスを作成する方法を示します。

手順

1. Java または .NET で、MobiLinkOrders というクラスを作成します。

   Java の場合は、次のコードを入力します。

   ```
   import com.sap.ml.script.*;
   import java.io.*;
   import java.sql.*;

   public class MobiLinkOrders {
   ```

   .NET の場合は、次のコードを入力します。

   ```
   using Sap.MobiLink.Script;
   using System.IO;
   using System.Data;
   using System.Text;
   public class MobiLinkOrders {
   ```

2. クラスレベルの DBConnectionContext インスタンスを宣言します。

   Java の場合は、次のコードを入力します。

   ```
       // Class level DBConnectionContext
       DBConnectionContext _cc;
   ```

   .NET の場合は、次のコードを入力します。

   ```
       // Class level DBConnectionContext
       private DBConnectionContext _cc = null;
   ```

   Mobile Link サーバによって DBConnectionContext のインスタンスがクラスコンストラクタに渡されます。
   DBConnectionContext には、Mobile Link 統合データベースとの現在の接続に関する情報がカプセル化されます。

3. ファイル入出力に使用するオブジェクトを宣言します。

   Java の場合は、java.io.FileWriter と java.io.BufferedReader を次のように宣言します。

   ```
       // Java objects for file i/o
       FileWriter my_writer;
       BufferedReader my_reader;
   ```

   .NET の場合は、StreamWriter と StreamReader を次のように宣言します。

   ```
       // Instances for file I/O
       private static StreamWriter my_writer = null;
       private static StreamReader my_reader = null;
   ```

4. クラスコンストラクタを作成します。

   クラスコンストラクタが、クラスレベルの DBConnectionContext インスタンスを設定します。

   Java の場合は、次のコードを入力します。

   ```
   public MobiLinkOrders( DBConnectionContext cc )
           throws IOException, FileNotFoundException
       {
           // Declare a class-level DBConnectionContext
           _cc = cc;
   ```

```
    }
```

.NET の場合は、次のコードを入力します。

```
    public MobiLinkOrders(DBConnectionContext cc) {
        _cc = cc;
    }
```

5. GetUpload メソッドを作成します。

   GetUpload メソッドでは、OrderComments テーブルを表す UploadedTableData クラスインスタンスを取得します。OrderComments テーブルには、遠隔地の営業部員による特別なコメントが含まれます。このテーブルはレッスンの後半で作成します。

   UploadedTableData の getInserts メソッドでは、注文に対する新しいコメントの結果セットを返します。writeOrderComment メソッドでは、結果セット内の各ローをテキストファイルに書き出します。

   Java の場合は、次のコードを入力します。

```
public void writeOrderComment( int _commentID, int _orderID, String _comments )
        throws IOException
    {
        if (my_writer == null)
            // A FileWriter for writing order comments
            my_writer = new FileWriter( "C:¥¥MLdirect¥¥orderComments.txt",true);

        // Write out the order comments to remoteOrderComments.txt
        my_writer.write(_commentID + "¥t" + _orderID + "¥t" + _comments);
        my_writer.write( "¥n" );
        my_writer.flush();
    }

    //  Method for the handle_UploadData synchronization event
    public void GetUpload( UploadData ut )
        throws SQLException, IOException
    {
        // Get an UploadedTableData for OrderComments
        UploadedTableData orderCommentsTbl =
ut.getUploadedTableByName("OrderComments");

        // Get inserts uploaded by the MobiLink client
        ResultSet insertResultSet = orderCommentsTbl.getInserts();

        while ( insertResultSet.next() )
        {
            // Get order comments
            int _commentID = insertResultSet.getInt("comment_id");
            int _orderID = insertResultSet.getInt("order_id");
            String _specialComments =
insertResultSet.getString("order_comment");
            if (_specialComments != null) {
                writeOrderComment(_commentID,_orderID,_specialComments);
            }
        }
        insertResultSet.close();
    }
```

.NET の場合は、次のコードを入力します。

```
    public void WriteOrderComment(int comment_id,
        int order_id,
        string comments)
    {
        if (my_writer == null) {
```

```
            my_writer = new StreamWriter("c:¥¥MLdirect¥¥orderComments.txt");
        }
        my_writer.WriteLine("{0}¥t{1}¥t{2}", comment_id, order_id, comments);
        my_writer.Flush();
    }
// Method for the handle_UploadData synchronization event.
    public void GetUpload(UploadData ut)
    {
        // Get UploadedTableData for remote table called OrderComments
        UploadedTableData order_comments_table_data =
            ut.GetUploadedTableByName("OrderComments");
        // Get inserts uploaded by the MobiLink client
        using( IDataReader new_comment_reader =
order_comments_table_data.GetInserts() ) {
                    while (new_comment_reader.Read()) {
                                // Columns are
                                // 0 - "order_comment"
                                // 1 - "comment_id"
                                // 2 - "order_id"
                                // You can look up these values using the
DataTable returned by:
                                // order_comments_table_data.GetSchemaTable().
                                // In this example, you just use the known
column order to
                                // determine the column indexes; alternatively,
you could use
                                // the column names

                                // Only process this insert if the order_comment
is not null
                                if (!new_comment_reader.IsDBNull(2)) {
                                    int comment_id =
new_comment_reader.GetInt32(0);
                                    int order_id =
new_comment_reader.GetInt32(1);
                                    string comments =
new_comment_reader.GetString(2);
                                    WriteOrderComment(comment_id, order_id,
comments);
                                }
                    }
                }
    }
```

6. SetDownload メソッドを作成します。
   a. OrderComments テーブルを表すクラスインスタンスを取得します。

      DBConnectionContext の getDownloadData メソッドを使用して DownloadData のインスタンスを取得します。DownloadData の getDownloadTableByName メソッドを使用して、OrderComments テーブルの DownloadTableData インスタンスを返します。

      Java の場合は、次のコードを入力します。

```
    public void SetDownload()
        throws SQLException, IOException
    {
        DownloadData download_d = _cc.getDownloadData();

        DownloadTableData download_td =
download_d.getDownloadTableByName( "OrderComments" );
```

      .NET の場合は、次のコードを入力します。

```
    private const string read_file_path =
        "c:¥¥MLdirect¥¥orderResponses.txt";
```

```
    // Method for the handle_DownloadData synchronization event
    public void SetDownload() {
        if ((my_reader == null) && !File.Exists(read_file_path)) {
            System.Console.Out.Write("There is no file to read.");
            return;
        }
        DownloadTableData comments_for_download =
            _cc.GetDownloadData().GetDownloadTableByName("OrderComments");
```

> **注記**
>
> レッスン 7 では、OrderComments テーブルをリモートデータベースに作成します。Mobile Link クライアントデータベースを設定します。

b. 準備文または IDbCommand を取得します。これを使用すると、ダウンロードに操作を追加、挿入、または更新できます。

Java の場合は、DownloadTableData の getUpsertPreparedStatement メソッドを使用して java.sql.PreparedStatement のインスタンスを次のように返します。

```
PreparedStatement update_ps = download_td.getUpsertPreparedStatement();
```

.NET の場合は、DownloadTableData の GetUpsertCommand メソッドを次のように使用します。

```
        // Add upserts to the set of operation that are going to be
        // applied at the remote database
        IDbCommand comments_upsert =
            comments_for_download.GetUpsertCommand();
```

c. 各ローのダウンロードデータを設定します。

このコードは、`orderResponses.txt` を参照して、Mobile Link ダウンロードにデータを追加しています。

Java の場合は、次のコードを入力します。

```
try {
        // A BufferedReader for reading in responses
        if (my_reader == null)
        my_reader = new BufferedReader(new FileReader("C:¥¥MLdirect¥
¥orderResponses.txt"));

        // Get the next line from orderResponses
        String commentLine;
        commentLine = my_reader.readLine();

        // Send comment responses down to clients
        while (commentLine != null) {
        // Get the next line from orderResponses.txt
        String[] response_details = commentLine.split("¥t");

        if (response_details.length != 3) {
            System.err.println("Error reading from orderResponses.txt");
            System.err.println("Error setting direct row handling download");
            return;
        }
        int comment_id = Integer.parseInt(response_details[0]);
        int order_id = Integer.parseInt(response_details[1]);
        String updated_comment = response_details[2];

        // Set an order comment response in the MobiLink download
        update_ps.setInt(1, comment_id);
        update_ps.setInt(2, order_id);
        update_ps.setString(3, updated_comment);
```

```
        update_ps.executeUpdate();

        // Get next line
        commentLine = my_reader.readLine();
        }
    }
```

.NET の場合は、次のコードを入力します。

```
        if (my_reader == null) {
            my_reader = new StreamReader(read_file_path);
        }
        string comment_line;
        while ((comment_line = my_reader.ReadLine()) != null) {
            // Three values are on each line separated by '¥t'
            string[] response_details = comment_line.Split('¥t');
            if (response_details.Length != 3) {
                throw (new SynchronizationException(
                    "Error reading from orderResponses.txt"));
            }
            int comment_id = System.Int32.Parse(response_details[0]);
            int order_id = System.Int32.Parse(response_details[1]);
            string comments = response_details[2];
            // Parameters of the correct number and type have
            // already been added so you just need to set the
            // values of the IDataParameter
            ((IDataParameter)(comments_upsert.Parameters[0])).Value =
                comment_id;
            ((IDataParameter)(comments_upsert.Parameters[1])).Value =
                order_id;
            ((IDataParameter)(comments_upsert.Parameters[2])).Value =
                comments;
            // Add the upsert operation
            comments_upsert.ExecuteNonQuery();
        }
    }
```

d. ダウンロードに挿入操作または更新操作を追加する準備文を終了します。

Java の場合は、次のコードを入力します。

```
finally {
    update_ps.close();
   }
}
```

.NET の場合、IDbCommand を閉じる必要はありません。オブジェクトは、ダウンロードの終わりに自動的に破棄されます。

7. EndDownload メソッドを作成します。

このメソッドでは、end_download 接続イベントを処理し、またリソースを解放できます。

Java の場合は、次のコードを入力します。

```
 public void EndSync()
     throws IOException
 {
     // Close i/o resources
     if (my_reader != null) {
         my_reader.close();
         my_reader = null;
     }
     if (my_writer != null) {
         my_writer.close();
```

```
            my_writer = null;
        }
    }
}
```

.NET の場合は、次のコードを入力します。

```
    public void EndSync()
    {
        if (my_writer != null) {
            my_writer.Close();
            my_writer = null;
        }
        if (my_reader != null) {
            my_reader.Close();
            my_reader = null;
        }
    }
```

8. コードを保存します。

   Java の場合は、コードを `MobiLinkOrders.java` という名前で作業ディレクトリに保存します。`c:¥MLdirect`

   .NET の場合は、コードを `MobiLinkOrders.cs` という名前で作業ディレクトリに保存します。`c:¥MLdirect`

9. コードを検証するには、MobiLinkOrders コードの全リスト (Java) または MobiLinkOrders コードの全リスト (.NET) を確認します。
10. クラスファイルをコンパイルします。
    a. Java または .NET のソースファイルが含まれるディレクトリに移動します。
    b. MobiLinkOrders をコンパイルし、Java または .NET 用の Mobile Link サーバ API ライブラリを参照します。

       Java の場合、*%SQLANY17%*`¥java` にある `mlscript.jar` を参照する必要があります。

       Java の場合は、次のコマンドを実行して、*C:¥Program Files¥SQL Anywhere 17*`¥` を SQL Anywhere17 ディレクトリに置き換えます。

       ```
       javac -classpath "%SQLANY17%¥java¥mlscript.jar" MobiLinkOrders.java
       ```

       .NET の場合は、次のコマンドを実行して、*C:¥Program Files¥SQL Anywhere 17*`¥` を SQL Anywhere17 ディレクトリに置き換えます。

       ```
       csc /out:MobiLinkServerCode.dll /target:library /reference:"%SQLANY17%
       ¥Assembly¥v3.5¥Sap.MobiLink.Script.dll" MobiLinkOrders.cs
       ```

> **注記**
>
> この例では、プライマリキーの値がユニークとはかぎりません。

結果

Mobile Link のダイレクトローハンドリングのための Java クラスまたは .NET クラスが作成されます。

次のステップ

次のレッスンに進みます。

このセクションの内容:

タスクの概要: チュートリアル: ダイレクトローハンドリングの使用 [184 ページ]

前のタスク: レッスン 4: 同期スクリプトの追加 [190 ページ]

次のタスク: レッスン 6: Mobile Link サーバの起動 [204 ページ]

1.5.6.5.1 MobiLinkOrders コードの全リスト (Java)

Java でダイレクトローハンドリングを行う場合の MobiLinkOrders の全リストは次のとおりです。

```
import com.sap.ml.script.*;
    import java.io.*;
    import java.sql.*;

    public class MobiLinkOrders {

    // Class level DBConnectionContext
    DBConnectionContext _cc;

    // Java objects for file i/o
    FileWriter my_writer;
    BufferedReader my_reader;
    public MobiLinkOrders( DBConnectionContext cc )
        throws IOException, FileNotFoundException
    {
        // Declare a class-level DBConnectionContext
        _cc = cc;
    }

    public void writeOrderComment( int _commentID, int _orderID, String _comments )
        throws IOException
    {
        if (my_writer == null)
            // A FileWriter for writing order comments
            my_writer = new FileWriter( "C:¥¥MLdirect¥¥orderResponses.txt",true);

        // Write out the order comments to remoteOrderComments.txt
        my_writer.write(_commentID + "¥t" + _orderID + "¥t" + _comments);
        my_writer.write( "¥n" );
        my_writer.flush();
    }

    //  Method for the handle_UploadData synchronization event
```

```
    public void GetUpload( UploadData ut )
        throws SQLException, IOException
    {
        // Get an UploadedTableData for OrderComments
        UploadedTableData orderCommentsTbl =
ut.getUploadedTableByName("OrderComments");

        // Get inserts uploaded by the MobiLink client
        ResultSet insertResultSet = orderCommentsTbl.getInserts();

        while ( insertResultSet.next() )
        {
            // Get order comments
            int _commentID = insertResultSet.getInt("comment_id");
            int _orderID = insertResultSet.getInt("order_id");
            String _specialComments =
insertResultSet.getString("order_comment");
            if (_specialComments != null) {
                writeOrderComment(_commentID,_orderID,_specialComments);
            }
        }
        insertResultSet.close();
    }

    public void SetDownload()
        throws SQLException, IOException
    {
        DownloadData download_d = _cc.getDownloadData();

        DownloadTableData download_td =
download_d.getDownloadTableByName( "OrderComments" );

        PreparedStatement update_ps = download_td.getUpsertPreparedStatement();
    try {
        // A BufferedReader for reading in responses
        if (my_reader == null)
        my_reader = new BufferedReader(new FileReader("C:¥¥MLdirect¥
¥orderResponses.txt"));

        // Get the next line from orderResponses
        String commentLine;
        commentLine = my_reader.readLine();

        // Send comment responses down to clients
        while (commentLine != null) {
        // Get the next line from orderResponses.txt
        String[] response_details = commentLine.split("¥t");

        if (response_details.length != 3) {
            System.err.println("Error reading from orderResponses.txt");
            System.err.println("Error setting direct row handling download");
            return;
        }
        int comment_id = Integer.parseInt(response_details[0]);
        int order_id = Integer.parseInt(response_details[1]);
        String updated_comment = response_details[2];

        // Set an order comment response in the MobiLink download
        update_ps.setInt(1, comment_id);
        update_ps.setInt(2, order_id);
        update_ps.setString(3, updated_comment);
        update_ps.executeUpdate();

        // Get next line
        commentLine = my_reader.readLine();
        }
    } finally {
        update_ps.close();
```

```
    }
    }

    public void EndDownload()
        throws IOException
    {
        // Close i/o resources
        if (my_reader != null) {
            my_reader.close();
            my_reader = null;
        }
        if (my_writer != null) {
            my_writer.close();
            my_writer = null;
        }
    }
}
```

1.5.6.5.2 MobiLinkOrders コードの全リスト (.NET)

.NET でダイレクトローハンドリングを行う場合の MobiLinkOrders の全リストは次のとおりです。

```
using Sap.MobiLink.Script;
    using System.IO;
    using System.Data;
    using System.Text;

    public class MobiLinkOrders {
    // Class level DBConnectionContext
    private DBConnectionContext _cc = null;
    // Instances for file I/O
    private static StreamWriter my_writer = null;
    private static StreamReader my_reader = null;
    public MobiLinkOrders(DBConnectionContext cc) {
        _cc = cc;
    }
    public void WriteOrderComment(int comment_id,
        int order_id,
        string comments)
    {
        if (my_writer == null) {
            my_writer = new StreamWriter("c:¥¥MLdirect¥¥orderComments.txt");
        }
        my_writer.WriteLine("{0}¥t{1}¥t{2}", comment_id, order_id, comments);
        my_writer.Flush();
    }
    // Method for the handle_UploadData synchronization event.
    public void GetUpload(UploadData ut)
    {
        // Get UploadedTableData for remote table called OrderComments
        UploadedTableData order_comments_table_data =
            ut.GetUploadedTableByName("OrderComments");
        // Get inserts uploaded by the MobiLink client
        IDataReader new_comment_reader =
            order_comments_table_data.GetInserts();
        while (new_comment_reader.Read()) {
            // Columns are
            // 0 - "order_comment"
            // 1 - "comment_id"
            // 2 - "order_id"
            // You can look up these values using the DataTable returned by:
            // order_comments_table_data.GetSchemaTable().
            // In this example, you just use the known column order to
```

```
            // determine the column indexes
            // Only process this insert if the order_comment is not null
            if (!new_comment_reader.IsDBNull(2)) {
                int comment_id = new_comment_reader.GetInt32(0);
                int order_id = new_comment_reader.GetInt32(1);
                string comments = new_comment_reader.GetString(2);
                WriteOrderComment(comment_id, order_id, comments);
            }
        }
        // Always close the reader when you are done with it!
        new_comment_reader.Close();
    }
    private const string read_file_path =
        "c:¥¥MLdirect¥¥orderResponses.txt";
    // Method for the handle_DownloadData synchronization event
    public void SetDownload() {
        if ((my_reader == null) && !File.Exists(read_file_path)) {
            System.Console.Out.Write("There is no file to read.");
            return;
        }
        DownloadTableData comments_for_download =
            _cc.GetDownloadData().GetDownloadTableByName("OrderComments");
        // Add upserts to the set of operation that are going to be
        // applied at the remote database
        IDbCommand comments_upsert =
            comments_for_download.GetUpsertCommand();
        if (my_reader == null) {
            my_reader = new StreamReader(read_file_path);
        }
        string comment_line;
        while ((comment_line = my_reader.ReadLine()) != null) {
            // Three values are on each line separated by '¥t'
            string[] response_details = comment_line.Split('¥t');
            if (response_details.Length != 3) {
                throw (new SynchronizationException(
                    "Error reading from orderResponses.txt"));
            }
            int comment_id = System.Int32.Parse(response_details[0]);
            int order_id = System.Int32.Parse(response_details[1]);
            string comments = response_details[2];
            // Parameters of the correct number and type have
            // already been added so you just need to set the
            // values of the IDataParameter
            ((IDataParameter)(comments_upsert.Parameters[0])).Value =
                comment_id;
            ((IDataParameter)(comments_upsert.Parameters[1])).Value =
                order_id;
            ((IDataParameter)(comments_upsert.Parameters[2])).Value =
                comments;
            // Add the upsert operation
            comments_upsert.ExecuteNonQuery();
        }
    }
    public void EndDownload()
    {
        if (my_writer != null) {
            my_writer.Close();
            my_writer = null;
        }
        if (my_reader != null) {
            my_reader.Close();
            my_reader = null;
        }
    }
}
```

1.5.6.6 レッスン 6: Mobile Link サーバの起動

このレッスンでは、Mobile Link サーバを起動します。-c オプションを使って Mobile Link サーバ (mlsrv17) を起動し、統合データベースに接続します。-sl java オプションまたは -sl dnet オプションを使用して、Java クラスまたは .NET クラスをそれぞれロードします。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

手順

統合データベースに接続し、mlsrv17 のコマンドラインでクラスをロードします。

`c:¥MLdirect` は、ソースファイルがある実際のディレクトリに置き換えてください。

Java の場合は、次のコマンドを実行します。

```
mlsrv17 -c "DSN=mldirect_db" -o serverOut.txt -v+ -dl -zu+ -x tcpip -sl java (-cp c:
¥MLdirect)
```

.NET の場合は、次のコマンドを実行します。

```
mlsrv17 -c "DSN=mldirect_db" -o serverOut.txt -v+ -dl -zu+ -x tcpip -sl dnet (-
MLAutoLoadPath=c:¥MLdirect)
```

このチュートリアルで使用している Mobile Link サーバの各オプションの説明を次に示します。オプション -o、-v、-dl は、デバッグとトラブルシューティングの情報を提供します。これらのロギングオプションは、開発環境での使用に適しています。パフォーマンス上の理由から、一般的に -v+ と -dl は運用環境では使用しません。

オプション	説明
-c	続いて接続文字列を指定します。
-o	メッセージログファイル `serverOut.txt` を指定します。
-v+	-v オプションは、ログを取る情報を指定します。-v+ を使用して、最大冗長ロギングをオンに設定します。
-dl	画面にすべてのログメッセージを表示します。
-zu+	自動的に新しいユーザを追加します。
-x	Mobile Link クライアントの通信プロトコルとパラメータを設定します。
-sl java	クラスファイルを検索する一連のディレクトリを指定し、またサーバ起動時に Java VM をロードします。

オプション	説明
-sl dnet	.NET アセンブリのロケーションを指定し、またサーバ起動時に CLR をロードします。

Mobile Link サーバメッセージウィンドウが表示されます。

結果

統合データベースに接続する Mobile Link サーバが起動し、作成したクラスがロードされます。

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: ダイレクトローハンドリングの使用 [184 ページ]

前のタスク: レッスン 5: Mobile Link のダイレクトローハンドリングのための Java または .NET クラスの作成 [193 ページ]

次のタスク: レッスン 7: Mobile Link クライアントデータベースの設定 [205 ページ]

1.5.6.7 レッスン 7: Mobile Link クライアントデータベースの設定

このレッスンでは、SQL Anywhere データベースを統合データベースと Mobile Link クライアントに使用します。また、このチュートリアルの目的上、Mobile Link クライアント、統合データベース、および Mobile Link サーバはすべて同じコンピュータに置きます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

Mobile Link クライアントデータベースを設定するには、RemoteOrders と OrderComments の各テーブルを作成します。RemoteOrders テーブルは、統合データベースの RemoteOrders テーブルに対応します。Mobile Link サーバでは、SQL

ベースのスクリプトを使用してリモート注文が同期されます。OrderComments テーブルは、クライアントデータベースだけで使用されます。Mobile Link サーバでは、特別なイベントを使用して OrderComments テーブルが処理されます。

テーブルの作成後に、クライアントデータベースで同期ユーザ、パブリケーション、サブスクリプションを作成します。パブリケーションは、リモートデータベース上の同期対象となるテーブルとカラムを識別します。これらのテーブルとカラムをアーティクルと呼びます。同期サブスクリプションは、パブリケーションに対する Mobile Link ユーザのサブスクリプションです。

手順

1. dbinit コマンドラインユーティリティを使用して、Mobile Link クライアントデータベースを作成します。

   次のコマンドを実行します。

   ```
   dbinit -i -k -dba DBA,passwd remote1
   ```

   -i オプションと -k オプションは、それぞれ jConnect のサポートと Watcom SQL の互換ビューを省略します。
2. dbsrv17 コマンドラインユーティリティを使用して、Mobile Link クライアントデータベースを起動します。

   次のコマンドを実行します。

   ```
   dbsrv17 remote1
   ```
3. 次のコマンドを実行して、Interactive SQL から Mobile Link クライアントデータベースに接続します。

   ```
   dbisql -c "SERVER=remote1;UID=DBA;PWD=passwd"
   ```
4. Interactive SQL で次の SQL 文を実行して RemoteOrders テーブルを作成します。

   ```
   CREATE TABLE RemoteOrders (
       order_id           INTEGER NOT NULL,
       product_id         INTEGER NOT NULL,
       quantity           INTEGER,
       order_status       VARCHAR(10) DEFAULT 'new',
       PRIMARY KEY(order_id)
   );
   ```
5. Interactive SQL で次の文を実行して OrderComments テーブルを作成します。

   ```
   CREATE TABLE OrderComments (
       comment_id         INTEGER NOT NULL,
       order_id           INTEGER NOT NULL,
       order_comment      VARCHAR(255),
       PRIMARY KEY(comment_id),
       FOREIGN KEY(order_id) REFERENCES RemoteOrders(order_id)
   );
   ```
6. Interactive SQL で次の文を実行して、Mobile Link 同期ユーザ、パブリケーション、サブスクリプションを作成します。

   ```
   CREATE SYNCHRONIZATION USER ml_sales1;
   CREATE PUBLICATION order_publ (TABLE RemoteOrders, TABLE OrderComments);
   CREATE SYNCHRONIZATION SUBSCRIPTION TO order_publ FOR ml_sales1
    TYPE TCPIP ADDRESS 'host=localhost';
   ```

> **注記**
>
> Mobile Link サーバに接続する方法は、CREATE SYNCHRONIZATION SUBSCRIPTION 文の TYPE 句と ADDRESS 句を使用して指定します。

パブリケーションを使用して、同期するデータを指定できます。この例では、RemoteOrders と OrderComments の各テーブル全体を指定します。

結果

SQL Anywhere 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: ダイレクトローハンドリングの使用 [184 ページ]

前のタスク: レッスン 6: Mobile Link サーバの起動 [204 ページ]

次のタスク: レッスン 8: 同期 [207 ページ]

1.5.6.8 レッスン 8: 同期

dbmlsync ユーティリティを使用して、SQL Anywhere リモートデータベースの Mobile Link 同期を開始します。dbmlsync を起動する前に、注文データとコメントをリモートデータベースに追加します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. まだ接続していない場合は、次のコマンドを実行して、Interactive SQL から Mobile Link クライアントデータベースに接続します。

```
dbisql -c "SERVER=remote1;UID=DBA;PWD=passwd"
```

2. 次の文を実行して、クライアントデータベース内の RemoteOrders テーブルに注文を追加します。

```
INSERT INTO RemoteOrders (order_id, product_id, quantity, order_status)
 VALUES (1,12312,10,'new');
```

3. 次の文を Interactive SQL で実行して、クライアントデータベース内の OrderComments テーブルにコメントを追加します。

```
INSERT INTO OrderComments (comment_id, order_id, order_comment)
 VALUES (1,1,'send promotional material with the order');
```

4. Interactive SQL で次の文を実行して、変更をコミットします。

```
COMMIT;
```

5. 次のコマンドを実行します。

```
dbmlsync -c "SERVER=remote1;UID=DBA;PWD=passwd" -o rem1.txt -v+
```

次の表には、このレッスンで使用する各 dbmlsync オプションの説明が含まれています。

オプション	説明
-c	接続文字列を指定します。
-o	メッセージログファイル `rem1.txt` を指定します。
-v+	-v オプションは、ログを取る情報を指定します。-v+ を使用して、最大冗長ロギングをオンに設定します。

Mobile Link 同期クライアントの起動が完了すると、同期が成功したことを示す出力画面が表示されます。SQL ベースの同期によって、クライアントの RemoteOrders テーブル内のローが、統合データベース内の RemoteOrders テーブルに転送されます。

Java または .NET の処理によってコメントが `orderComments.txt` に挿入されました。

6. SQL AnywhereMobile Link クライアントの各ウィンドウを閉じます。
7. 応答を `orderResponses.txt` に挿入してリモートデータベースにダウンロードします。この操作はサーバ側で行います。

次のテキストを `orderResponses.txt` に追加します。エントリはタブ文字で区切ります。行末で、Enter キーを押します。

```
1    1    Promotional material shipped
```

8. dbmlsync クライアントユーティリティを使用して同期を実行します。

この操作はクライアント側で行います。

次のコマンドを実行します。

```
dbmlsync -c "SERVER=remote1;UID=DBA;PWD=passwd" -o rem1.txt -v+
```

> **注記**
>
> ダイレクトローハンドリングを使用してダウンロードされたローは、mlsrv17 -v+ オプションによっては出力されず、dbmlsync -v+ オプションによってリモートログに出力されます。

Mobile Link クライアントユーティリティが表示されます。

9. Interactive SQL で、OrderComments テーブルを選択して、ローがダウンロードされたことを確認します。

   次の SQL 文を実行します。

   ```
   SELECT * FROM OrderComments;
   ```

結果

SQL Anywhere リモートデータベースが更新され、統合データベースと同期されます。

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: ダイレクトローハンドリングの使用 [184 ページ]

前のタスク: レッスン 7: Mobile Link クライアントデータベースの設定 [205 ページ]

次のタスク: レッスン 9: クリーンアップ [209 ページ]

1.5.6.9 レッスン 9: クリーンアップ

チュートリアルをコンピュータから削除します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. Interactive SQL のすべてのインスタンスを閉じます。
2. SQL Anywhere、Mobile Link、同期クライアントの各ウィンドウを閉じます。
3. 次の手順で、チュートリアルに関連するすべての ODBC データソースを削除します。
   a. ODBC アドミニストレータを起動します。

      次のコマンドを実行します。

      ```
      odbcad32
      ```

   b. **mldirect_db** データソースを削除します。
4. 統合データベースとリモートデータベースを削除します。
   a. 統合データベースとリモートデータベースが保存されているディレクトリに移動します。
   b. `MLconsolidated.db`、`MLconsolidated.log`、`remote1.db`、および `remote1.log` を削除します。

結果

チュートリアルがコンピュータから削除されます。

タスクの概要: チュートリアル: ダイレクトローハンドリングの使用 [184 ページ]

前のタスク: レッスン 8: 同期 [207 ページ]

1.5.7 チュートリアル: Microsoft Excel との同期

ダイレクトローハンドリングを使用して、リモートデータと中央のデータソース、アプリケーション、または Web サービスとの通信ができます。

前提条件

次の知識と経験が必要です。

- Java の知識
- Microsoft Excel の知識
- Mobile Link イベントスクリプトの基本的な知識

次のソフトウェアが必要です。

- SQL Anywhere17
- Java ソフトウェア開発キット

- Microsoft Office Excel 2007 以降

統合データベースで次のロールおよび権限を持つ必要があります。

- SYS_AUTH_RESOURCE_ROLE 互換ロール
- MONITOR システム権限

リモートデータベースで次のロールおよび権限を持つ必要があります。

- SYS_REPLICATION_ADMIN_ROLE システムロール
- SYS_RUN_REPLICATION_ROLE システムロール

コンテキスト

このチュートリアルでは、ダイレクトローハンドリングを使用して、Microsoft Excel のスプレッドシートにあるデータを Mobile Link クライアントと同期する基本的な手順について説明します。Java 実装を例として使用し、Mobile Link ダイレクトローハンドリングを実装して、サポートされている統合データソース以外のデータソースを使用できるようにする方法について説明します。

このチュートリアルでは、次の作業の方法について説明します。

- Java 用 Mobile Link サーバ API の使用
- Mobile Link ダイレクトローハンドリング用のメソッドの作成
- Java を使用した Microsoft Excel ワークシートにあるデータへのアクセス

1. レッスン 1: Microsoft Excel ワークシートの設定 [212 ページ]
   このレッスンでは、Microsoft Excel ワークシートを作成し、Microsoft Excel ドライバを使用して ODBC データソースを定義します。Microsoft Excel ワークシートには製品情報を格納します。
2. レッスン 2: Mobile Link 統合データベースの設定 [214 ページ]
   このレッスンでは、データベースを作成し、ODBC データソースを定義します。
3. レッスン 3: Mobile Link 統合データベースでのテーブルの作成 [215 ページ]
   このレッスンでは、Mobile Link 統合データベースに RemoteOrders テーブルを作成します。
4. レッスン 4: 同期スクリプトの追加 [217 ページ]
   このレッスンでは、SQL ローハンドリングとダイレクトローハンドリング用のスクリプトを統合データベースに追加します。
5. レッスン 5: Mobile Link のダイレクトローハンドリングのための Java クラスの作成 [219 ページ]
   このレッスンでは、ダイレクトローハンドリングを使用して、クライアントデータベース内の OrderComments テーブルのローを処理します。
6. レッスン 6: Mobile Link サーバの起動 [225 ページ]
   このレッスンでは、-c オプションを使用して Mobile Link サーバ (mlsrv17) を起動して統合データベースに接続し、-sl java オプションを使用して Java クラスをロードします。
7. レッスン 7: Mobile Link クライアントデータベースの設定 [226 ページ]
   このレッスンでは、SQL Anywhere データベースを統合データベースと Mobile Link クライアントに使用します。
8. レッスン 8: 同期 [228 ページ]
   dbmlsync ユーティリティを使用して、SQL Anywhere リモートデータベースの Mobile Link 同期を開始します。dbmlsync を起動する前に、注文データとコメントをリモートデータベースに追加します。

9. レッスン 9: クリーンアップ [230 ページ]
   チュートリアルをコンピュータから削除します。

タスクの概要: Mobile Link チュートリアル [96 ページ]

前のタスク: チュートリアル: ダイレクトローハンドリングの使用 [184 ページ]

次のタスク: チュートリアル: XML との同期 [231 ページ]

関連情報

Mobile Link 同期 [4 ページ]
SAP SQL Anywhere フォーラム

1.5.7.1 レッスン 1: Microsoft Excel ワークシートの設定

このレッスンでは、Microsoft Excel ワークシートを作成し、Microsoft Excel ドライバを使用して ODBC データソースを定義します。Microsoft Excel ワークシートには製品情報を格納します。

前提条件

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

Microsoft Excel ドライバは 32 ビットドライバであるため、このチュートリアルでは、32 ビットバージョンの ODBC データソースアドミニストレータが必要です。

手順

1. サーバ側コンポーネント用に `c:¥MLobjexcel` という名前の作業ディレクトリを作成します。
2. Microsoft Excel を開き、新しいワークブックを作成します。
3. デフォルトのワークシートで、**A**、**B**、**C** の各カラムヘッダで次の内容を追加します。

comment_id	order_id	order_comment
2	1	Promotional material shipped

comment_id	order_id	order_comment
3	1	More information about material required

4. デフォルトのワークシート名 **Sheet1** を **order_sheet** に変更します。
   a. **Sheet1** タブをダブルクリックします。
   b. **order_sheet** と入力します。
5. Microsoft Excel ワークブックを保存します。ワークブックを `order_central.xlsx` という名前で作業ディレクトリ `c:¥MLobjexcel` に保存します。
6. Microsoft Excel ドライバを使用して ODBC データソースを作成します。
   a. **スタート** > **プログラム** > *SQL Anywhere17* > **管理ツール** > *ODBC* **データソースアドミニストレータ** をクリックします。
   b. **ユーザ** *DSN* タブをクリックします。
   c. **追加**をクリックします。
   d. *Microsoft Excel Driver (*.xls*、*.xlsx*、*.xlsm*、*.xlsb)* をクリックします。
   e. **終了**をクリックします。
   f. **データソース名**フィールドに **excel_datasource** と入力します。
   g. **ブックの選択**をクリックし、`c:¥MLobjexcel¥order_central.xlsx` (ワークシートが含まれるファイル) を選択します。
   h. **読み込み専用**オプションをクリアします。
   i. 開いているすべての [ODBC データソースアドミニストレータ] ウィンドウで、*OK* をクリックします。
7. Microsoft Excel Driver が存在しない場合、`%windir%¥SysWOW64¥odbcad32.exe` を実行します。Unable to create DSN for Microsoft Office System Driver on 64-bit versions of Windows を確認します。

結果

Microsoft Excel ワークシートと ODBC データソ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: Microsoft Excel との同期 [210 ページ]

次のタスク: レッスン 2: Mobile Link 統合データベースの設定 [214 ページ]

1.5.7.2 レッスン 2: Mobile Link 統合データベースの設定

このレッスンでは、データベースを作成し、ODBC データソースを定義します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

Mobile Link 統合データベースはデータの中央レポジトリであり、同期処理の管理に使用する Mobile Link のシステムテーブルとストアドプロシージャが含まれます。ダイレクトローハンドリングでは、統合データベース以外のデータソースと同期しますが、Mobile Link サーバが使用する情報を保持するために統合データベースも必要です。

> **注記**
>
> Mobile Link 統合データベースを Mobile Link システムオブジェクトと DSN を使用して設定済みの場合は、このレッスンは省略できます。

手順

1. スタート ▶ プログラム ▶ *SQL Anywhere17* ▶ 管理ツール ▶ *SQL Central* をクリックします。
2. ツール ▶ *SQL Anywhere17* ▶ データベースの作成 をクリックします。
3. 次へをクリックします。
4. このコンピュータにデータベースを作成をデフォルトのままにし、次へをクリックします。
5. メインデータベースファイルを保存フィールドに、データベースのファイル名およびパスを入力します。たとえば、`c:¥MLobjexcel¥MLconsolidated.db` のように入力します。
6. データベース作成ウィザードの残りの指示に従い、デフォルト値をそのまま使用します。DBA ユーザのユーザ ID とパスワードを指定するように求められたら、それぞれ **DBA** と **passwd** を入力します。

   データベースへの接続ページで、最終切断後にデータベースを停止オプションをオフにします。
7. 完了をクリックします。

   MLconsolidated データベースが SQL Central に表示されます。
8. SQL Central で、ツール ▶ *SQL Anywhere17* ▶ *ODBC アドミニストレータを開く* をクリックします。
9. ユーザ *DSN* タブをクリックしてから、追加をクリックします。
10. データソースの新規作成ウィンドウで、*SQL Anywhere17* をクリックし、完了をクリックします。

11. *SQL Anywhere の ODBC* 設定ウィンドウで、次の操作を行います。
    a. *ODBC* タブをクリックします。
    b. データソース名フィールドに **mlexcel_db** と入力します。
    c. ログインタブをクリックします。
    d. ユーザ *ID* フィールドに、**DBA** と入力します。
    e. パスワードフィールドに、**passwd** と入力します。
    f. アクションドロップダウンリストで、このコンピュータで稼働しているデータベースに接続をクリックします。
    g. サーバ名フィールドに **MLconsolidated** と入力します。
    h. *OK* をクリックします。
12. ODBC データソースアドミニストレータを閉じます。

    ODBC データソースアドミニストレータウィンドウで *OK* をクリックします。

結果

統合データベースと統合データベース用の ODBC データソ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: Microsoft Excel との同期 [210 ページ]

前のタスク: レッスン 1: Microsoft Excel ワークシートの設定 [212 ページ]

次のタスク: レッスン 3: Mobile Link 統合データベースでのテーブルの作成 [215 ページ]

1.5.7.3 レッスン 3: Mobile Link 統合データベースでのテーブルの作成

このレッスンでは、Mobile Link 統合データベースに RemoteOrders テーブルを作成します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

このレッスンで作成する RemoteOrders テーブルには、次のカラムが含まれます。

order_id

注文のユニークな識別子です。
product_id

製品のユニークな識別子です。
quantity

品目の販売数です。
order_status

注文のステータスです。
last_modified

ローが最後に変更された日です。このカラムはタイムスタンプベースのダウンロードに使用します。このダウンロード方法は、効率的な同期のためにローをフィルタする一般的な方法です。

手順

1. Interactive SQL からデータベースに接続します。

   Interactive SQL は、SQL Central またはコマンドプロンプトから起動できます。

   - SQL Central から Interactive SQL を起動するには、**MLconsolidated - DBA** データベースを右クリックし、*Interactive SQL* を開くをクリックします。
   - コマンドプロンプトで Interactive SQL を起動するには、次のコマンドを実行します。

     ```
     dbisql -c "DSN=mlexcel_db"
     ```

2. Interactive SQL で次の SQL 文を実行し、RemoteOrders テーブルを作成します。

   ```
   CREATE TABLE RemoteOrders (
       order_id            INTEGER NOT NULL,
       product_id          INTEGER NOT NULL,
       quantity            INTEGER,
       order_status        VARCHAR(10) DEFAULT 'new',
       last_modified       TIMESTAMP DEFAULT CURRENT TIMESTAMP,
       PRIMARY KEY(order_id)
   );
   ```

   Interactive SQL によって、統合データベースに RemoteOrders テーブルが作成されます。

3. Interactive SQL で次の SQL 文を実行して Mobile Link のシステムテーブルとストアドプロシージャを作成します。

   C:¥Program Files¥SQL Anywhere 17¥ は、SQL Anywhere17 インストール環境のロケーションに置き換えてください。

   ```
   READ "C:¥Program Files¥SQL Anywhere 17¥MobiLink¥setup¥syncsa.sql";
   ```

   Interactive SQL によって `syncsa.sql` が統合データベースに適用されます。`syncsa.sql` を実行すると、前に **ml_** が付いた一連のシステムテーブルとストアドプロシージャが作成されます。これらのテーブルとストアドプロシージャは、同期処理中に Mobile Link サーバによって使用されます。

結果

統合データベースに RemoteOrders テーブルが作成されます。

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: Microsoft Excel との同期 [210 ページ]

前のタスク: レッスン 2: Mobile Link 統合データベースの設定 [214 ページ]

次のタスク: レッスン 4: 同期スクリプトの追加 [217 ページ]

1.5.7.4 レッスン 4: 同期スクリプトの追加

このレッスンでは、SQL ローハンドリングとダイレクトローハンドリング用のスクリプトを統合データベースに追加します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

SQL ローハンドリングを使用すると、リモートデータを、Mobile Link 統合データベース内のテーブルと同期できます。SQL ベースのスクリプトでは、次の情報を定義します。

- Mobile Link クライアントからアップロードするデータを統合データベースに適用する方法。
- 統合データベースからダウンロードするデータ。

このレッスンでは、次の SQL ベースのアップロードイベントとダウンロードイベント用の同期スクリプトを作成します。

upload_insert

このイベントは、クライアントデータベースに挿入された新しい注文を統合データベースに適用する方法を定義します。

download_cursor

このイベントは、リモートクライアントにダウンロードする注文を定義します。

download_delete_cursor

このイベントは、アップロード専用の同期スクリプトを使用する場合に必要です。このチュートリアルでは、このイベントを無視するように Mobile Link サーバを設定します。

ダイレクトローハンドリングを使用して特別な処理を SQL ベースの同期システムに追加します。この手順では、handle_UploadData、handle_DownloadData、download_cursor、download_delete_cursor の各イベントに対応するメソッド名を登録します。独自の Java クラスをレッスンの後半で作成します。

手順

1. 統合データベースに接続していない場合は、Interactive SQL から接続します。

   次のコマンドを実行します。

   ```
   dbisql -c "DSN=mlexcel_db"
   ```

2. ml_add_table_script ストアドプロシージャを使用して、upload_insert、download_cursor、download_delete_cursor の各イベント用の SQL ベースのテーブルスクリプトを追加します。

   Interactive SQL で次の SQL 文を実行します。upload_insert のスクリプトでは、アップロードされた order_id、product_id、quantity、order_status を Mobile Link 統合データベースに挿入します。download_cursor のスクリプトでは、タイムスタンプベースのフィルタを使用して、更新されたローをリモートクライアントにダウンロードします。

   ```
   CALL ml_add_table_script( 'default', 'RemoteOrders',
       'upload_insert',
       'INSERT INTO RemoteOrders( order_id, product_id, quantity, order_status)
       VALUES( {ml r.order_id}, {ml r.product_id}, {ml r.quantity}, {ml
   r.order_status} )' );

   CALL ml_add_table_script( 'default', 'RemoteOrders',
       'download_cursor',
       'SELECT order_id, product_id, quantity, order_status
       FROM RemoteOrders WHERE last_modified >= {ml s.last_table_download}');
   CALL ml_add_table_script( 'default', 'RemoteOrders',
       'download_delete_cursor', '--{ml_ignore}');
   COMMIT
   ```

3. handle_UploadData と handle_DownloadData の各イベント用の Java メソッドを登録します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   CALL ml_add_java_connection_script( 'default',
       'handle_UploadData',
       'MobiLinkOrders.GetUpload' );

   CALL ml_add_java_connection_script( 'default',
       'handle_DownloadData',
       'MobiLinkOrders.SetDownload' );
   ```

   Interactive SQL によって、GetUpload と SetDownload の各メソッドが、handle_UploadData と handle_DownloadData の各イベント用にそれぞれ登録されます。これらのメソッドは次のレッスンで作成します。

4. download_cursor と download_delete_cursor の各イベントを登録します。

   Interactive SQL で次の SQL スクリプトを実行します。

   ```
   CALL ml_add_table_script( 'default', 'OrderComments',
   ```

```
    'download_cursor', '--{ml_ignore}');
CALL ml_add_table_script( 'default', 'OrderComments',
    'download_delete_cursor', '--{ml_ignore}');
```

同期は双方向であり、アップロード専用ではないため、スクリプトを使用するときには、OrderComments テーブル用に download_cursor と download_delete_cursor の各イベントを登録してください。

5. これまでの変更内容をコミットします。

   Interactive SQL で次の SQL 文を実行します。

```
COMMIT;
```

結果

upload_insert、download_cursor、download_delete_cursor の各イベントがデータベースに追加されます。handle_UploadData、handle_DownloadData、download_cursor、および download_delete_cursor イベントに対応するメソッド名が登録されます。

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: Microsoft Excel との同期 [210 ページ]

前のタスク: レッスン 3: Mobile Link 統合データベースでのテーブルの作成 [215 ページ]

次のタスク: レッスン 5: Mobile Link のダイレクトローハンドリングのための Java クラスの作成 [219 ページ]

1.5.7.5 レッスン 5: Mobile Link のダイレクトローハンドリングのための Java クラスの作成

このレッスンでは、ダイレクトローハンドリングを使用して、クライアントデータベース内の OrderComments テーブルのローを処理します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

コンテキスト

このレッスンでは、ダイレクトローハンドリング用に次のメソッドを追加します。

GetUpload

このメソッドは handle_UploadData イベントに使用します。GetUpload では、アップロードされたコメントを `order_central.xlsx` という Microsoft Excel ワークシートに書き込みます。

SetDownload

このメソッドは handle_DownloadData イベントに使用します。SetDownload は、Microsoft Excel ワークシート `order_central.xlsx` に格納されたデータを取り出し、リモートクライアントに送信します。

次の手順では、処理用メソッドを含む Java クラスを作成する方法を示します。全リストについては、Java の MobiLinkOrders コードリストを確認してください。

手順

1. MobiLinkOrders という新しいクラスの作成を開始します。

   次のコードを作成します。

   ```
   import com.sap.ml.script.*;
   import java.io.*;
   import java.sql.*;
   public class MobiLinkOrders {
   ```

2. クラスレベルの DBConnectionContext インスタンスを宣言します。

   次のコードを追加します。

   ```
       // Class level DBConnectionContext
       DBConnectionContext _cc;
   ```

   Mobile Link サーバによって DBConnectionContext のインスタンスがクラスコンストラクタに渡されます。DBConnectionContext には、Mobile Link 統合データベースとの現在の接続に関する情報がカプセル化されます。

3. クラスコンストラクタを作成します。クラスコンストラクタが、クラスレベルの DBConnectionContext インスタンスを設定します。

   次のコードを追加します。

   ```
       public MobiLinkOrders( DBConnectionContext cc )
           throws IOException, FileNotFoundException {
           // Declare a class-level DBConnectionContext
           _cc = cc;
       }
   ```

4. GetUpload メソッドを作成します。

   GetUpload メソッドでは、OrderComments テーブルを表す UploadedTableData クラスインスタンスを取得します。OrderComments テーブルには、遠隔地の営業部員による特別なコメントが含まれます。このテーブルはレッスンの後半で作成します。

   UploadedTableData の getInserts メソッドでは、注文に対する新しいコメントの結果セットを返します。

次のコードを追加します。

```
    //  Method for the handle_UploadData synchronization event
    public void GetUpload( UploadData ut )
        throws SQLException, IOException {
        // Get an UploadedTableData for OrderComments
        UploadedTableData orderCommentsTbl =
ut.getUploadedTableByName("OrderComments");
        // Get inserts uploaded by the MobiLink client
        ResultSet insertResultSet = orderCommentsTbl.getInserts();

        try {
            // Connect to the excel worksheet through ODBC
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con =
DriverManager.getConnection( "jdbc:odbc:excel_datasource" );
            while( insertResultSet.next() ) {
                // Get order comments
                int _commentID = insertResultSet.getInt("comment_id");
                int _orderID = insertResultSet.getInt("order_id");
                String _specialComments =
insertResultSet.getString("order_comment");
                // Execute an insert statement to add the order comment to the
worksheet
                PreparedStatement st = con.prepareStatement("INSERT INTO
[order_sheet$]"
                    + "(order_id, comment_id, order_comment) VALUES (?,?,?)" );
                st.setString( 1, Integer.toString(_orderID) );
                st.setString( 2, Integer.toString(_commentID) );
                st.setString( 3, _specialComments );
                st.executeUpdate();
                st.close();
            }
            con.close();
        } catch(Exception ex) {
            System.err.print("Exception: ");
            System.err.println(ex.getMessage());
        } finally {
        insertResultSet.close();
    }
    }
```

5. SetDownload メソッドを作成します。
   a. OrderComments テーブルを表すクラスインスタンスを取得します。

      DBConnectionContext の getDownloadData メソッドを使用して DownloadData のインスタンスを取得します。DownloadData の getDownloadTableByName メソッドを使用して、OrderComments テーブルの DownloadTableData インスタンスを返します。

      次のコードを追加します。

      ```
          public void SetDownload() throws SQLException, IOException {
              DownloadData download_d = _cc.getDownloadData();
              DownloadTableData download_td =
      download_d.getDownloadTableByName( "OrderComments" );
      ```

      > **注記**
      >
      > Mobile Link クライアントデータベースを設定する場合、リモートデータベースにこのテーブルを作成します。

   b. 準備文または IDbCommand を取得します。これを使用すると、ダウンロードに挿入操作や更新操作を追加できます。

DownloadTableData の getUpsertPreparedStatement メソッドを使用して java.sql.PreparedStatement のインスタンスを返します。

次のコードを追加します。

```
        // Prepared statement to compile upserts (inserts or updates).
        PreparedStatement download_upserts =
download_td.getUpsertPreparedStatement();
```

c. 各ローのダウンロードデータを設定します。

次のコードでは、order_central.xlsx ワークシートを参照して、Mobile Link ダウンロードにデータを追加しています。

次のコードを追加します。

```
        try {
            // Connect to the excel worksheet through ODBC
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con =
DriverManager.getConnection( "jdbc:odbc:excel_datasource" );
            // Retrieve all the rows in the worksheet
            Statement st = con.createStatement();
            ResultSet Excel_rs = st.executeQuery( "select * from [order_sheet
$]" );
            while (Excel_rs.next()) {
                // Retrieve the row data
                int Excel_comment_id = Excel_rs.getInt(1);
                int Excel_order_id = Excel_rs.getInt(2);
                String Excel_comment = Excel_rs.getString(3);
                // Add the Excel data to the MobiLink download.
                download_upserts.setInt( 1, Excel_comment_id );
                download_upserts.setInt( 2, Excel_order_id );
                download_upserts.setString( 3, Excel_comment );
                download_upserts.executeUpdate();
            }
            // Close the excel result set, statement, and connection.
            Excel_rs.close();
            st.close();
            con.close();
        } catch (Exception ex) {
            System.err.print("Exception: ");
            System.err.println(ex.getMessage());
        }
```

d. ダウンロードに挿入操作または更新操作を追加する準備文を終了し、メソッドとクラスを終了します。

次のコードを追加します。

```
        finally {
        download_upserts.close();
    }
    }
}
```

6. Java コードを MobiLinkOrders.java の名前で作業ディレクトリ c:¥MLobjexcel に保存します。

MobiLinkOrders.java のコードを検証するには、Java の MobiLinkOrders コードリストを確認します。

7. クラスファイルをコンパイルします
   a. Java のソースファイルが含まれるディレクトリに移動します。
   b. Java 用の Mobile Link サーバ API ライブラリを参照する MobiLinkOrders をコンパイルします。

%SQLANY17%¥Java にある mlscript.jar を参照する必要があります。

次のコマンドを実行して、*C:¥Program Files¥SQL Anywhere 17*¥ を SQL Anywhere17 ディレクトリに置き換えます。

```
javac -classpath "C:¥Program Files¥SQL Anywhere 17¥java¥mlscript.jar"
MobiLinkOrders.java
```

結果

クライアントデータベース内の OrderComments テーブルのローが更新されます。

次のステップ

次のレッスンに進みます。

このセクションの内容:

タスクの概要: チュートリアル: Microsoft Excel との同期 [210 ページ]

前のタスク: レッスン 4: 同期スクリプトの追加 [217 ページ]

次のタスク: レッスン 6: Mobile Link サーバの起動 [225 ページ]

1.5.7.5.1 MobiLinkOrders コードの全リスト (Java)

このチュートリアルで使用している Java の MobiLinkOrders クラスの全コードを次に示します。

```
import com.sap.ml.script.*;
    import java.io.*;
    import java.sql.*;

    public class MobiLinkOrders {

    // Class level DBConnectionContext
    DBConnectionContext _cc;

    public MobiLinkOrders( DBConnectionContext cc )
        throws IOException, FileNotFoundException {
        // Declare a class-level DBConnectionContext
        _cc = cc;
    }
    //  Method for the handle_UploadData synchronization event
```

```
    public void GetUpload( UploadData ut )
        throws SQLException, IOException {
        // Get an UploadedTableData for OrderComments
        UploadedTableData orderCommentsTbl =
ut.getUploadedTableByName("OrderComments");
        // Get inserts uploaded by the MobiLink client
        ResultSet insertResultSet = orderCommentsTbl.getInserts();

        try {
            // Connect to the excel worksheet through ODBC
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con =
DriverManager.getConnection( "jdbc:odbc:excel_datasource" );
            while( insertResultSet.next() ) {
                // Get order comments
                int _commentID = insertResultSet.getInt("comment_id");
                int _orderID = insertResultSet.getInt("order_id");
                String _specialComments =
insertResultSet.getString("order_comment");
                // Execute an insert statement to add the order comment to the
worksheet
                PreparedStatement st = con.prepareStatement("INSERT INTO
[order_sheet$]"
                    + "(order_id, comment_id, order_comment) VALUES (?,?,?)" );
                st.setString( 1, Integer.toString(_orderID) );
                st.setString( 2, Integer.toString(_commentID) );
                st.setString( 3, _specialComments );
                st.executeUpdate();
                st.close();
            }
            con.close();
        } catch(Exception ex) {
            System.err.print("Exception: ");
            System.err.println(ex.getMessage());
        } finally {
        insertResultSet.close();
    }
    }

    public void SetDownload() throws SQLException, IOException {
        DownloadData download_d = _cc.getDownloadData();
        DownloadTableData download_td =
download_d.getDownloadTableByName( "OrderComments" );

        // Prepared statement to compile upserts (inserts or updates).
        PreparedStatement download_upserts =
download_td.getUpsertPreparedStatement();

        try {
            // Connect to the excel worksheet through ODBC
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con =
DriverManager.getConnection( "jdbc:odbc:excel_datasource" );
            // Retrieve all the rows in the worksheet
            Statement st = con.createStatement();
            ResultSet Excel_rs = st.executeQuery( "select * from [order_sheet$]" );
            while (Excel_rs.next()) {
                // Retrieve the row data
                int Excel_comment_id = Excel_rs.getInt(1);
                int Excel_order_id = Excel_rs.getInt(2);
                String Excel_comment = Excel_rs.getString(3);
                // Add the Excel data to the MobiLink download.
                download_upserts.setInt( 1, Excel_comment_id );
                download_upserts.setInt( 2, Excel_order_id );
                download_upserts.setString( 3, Excel_comment );
                download_upserts.executeUpdate();
            }
            // Close the excel result set, statement, and connection.
```

```
            Excel_rs.close();
            st.close();
            con.close();
        } catch (Exception ex) {
            System.err.print("Exception: ");
            System.err.println(ex.getMessage());
        } finally {
        download_upserts.close();
    }
    }
}
```

1.5.7.6 レッスン 6: Mobile Link サーバの起動

このレッスンでは、-c オプションを使用して Mobile Link サーバ (mlsrv17) を起動して統合データベースに接続し、-sl java オプションを使用して Java クラスをロードします。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

手順

統合データベースに接続し、mlsrv17 のコマンドラインでクラスをロードします。

統合データベースが起動していることを確認してから、次のコマンドを実行します。`c:¥MLobjexcel` は、Java ソースファイルがある実際のディレクトリに置き換えてください。

```
mlsrv17 -c "DSN=mlexcel_db" -o serverOut.txt -v+ -dl -zu+ -x tcpip  -sl java (-cp c:
¥MLobjexcel)
```

Mobile Link サーバメッセージウィンドウが表示されます。

このチュートリアルで使用している Mobile Link サーバの各オプションの説明を次に示します。オプション -o、-v、-dl は、デバッグとトラブルシューティングの情報を提供します。これらのロギングオプションは、開発環境での使用に適しています。パフォーマンス上の理由から、一般的に -v+ と -dl は運用環境では使用しません。

オプション	説明
-c	続いて接続文字列を指定します。
-o	メッセージログファイル `serverOut.txt` を指定します。
-v+	-v オプションは、ログを取る情報を指定します。-v+ を使用して、最大冗長ロギングをオンに設定します。

オプション	説明
-dl	画面にすべてのログメッセージを表示します。
-zu+	自動的に新しいユーザを追加します。
-x	Mobile Link クライアントの通信プロトコルとパラメータを設定します。
-sl java	クラスファイルを検索する一連のディレクトリを指定し、またサーバ起動時に Java VM をロードします。

Mobile Link サーバメッセージウィンドウが表示されます。

結果

Mobile Link サーバが起動し、ダイレクトローハンドリングの準備が整います。

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: Microsoft Excel との同期 [210 ページ]

前のタスク: レッスン 5: Mobile Link のダイレクトローハンドリングのための Java クラスの作成 [219 ページ]

次のタスク: レッスン 7: Mobile Link クライアントデータベースの設定 [226 ページ]

1.5.7.7 レッスン 7: Mobile Link クライアントデータベースの設定

このレッスンでは、SQL Anywhere データベースを統合データベースと Mobile Link クライアントに使用します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

また、このチュートリアルの目的上、Mobile Link クライアント、統合データベース、および Mobile Link サーバはすべて同じコンピュータに置きます。

Mobile Link クライアントデータベースを設定するには、RemoteOrders と OrderComments の各テーブルを作成します。RemoteOrders テーブルは、統合データベースの RemoteOrders テーブルに対応します。Mobile Link サーバでは、SQL ベースのスクリプトを使用してリモート注文が同期されます。OrderComments テーブルは、クライアントデータベースだけで使用されます。Mobile Link サーバでは、特別なイベントを使用して OrderComments テーブルが処理されます。

手順

1. dbinit コマンドラインユーティリティを使用して、Mobile Link クライアントデータベースを作成します。

   次のコマンドを実行します。

   ```
   dbinit -i -k -dba DBA,passwd remote1
   ```

   -i オプションと -k オプションは、それぞれ jConnect のサポートと Watcom SQL の互換ビューを省略します。
2. dbsrv17 コマンドラインユーティリティを使用して、Mobile Link クライアントデータベースを起動します。

   次のコマンドを実行します。

   ```
   dbsrv17 remote1
   ```
3. Interactive SQL を使用して Mobile Link クライアントデータベースに接続します。

   次のコマンドを実行します。

   ```
   dbisql -c "SERVER=remote1;UID=DBA;PWD=passwd"
   ```
4. RemoteOrders テーブルを作成します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   CREATE TABLE RemoteOrders (
       order_id           INTEGER NOT NULL,
       product_id         INTEGER NOT NULL,
       quantity           INTEGER,
       order_status       VARCHAR(10) DEFAULT 'new',
       PRIMARY KEY(order_id)
   );
   ```
5. OrderComments テーブルを作成します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   CREATE TABLE OrderComments (
       comment_id         INTEGER NOT NULL,
       order_id           INTEGER NOT NULL,
       order_comment      VARCHAR(255),
       PRIMARY KEY(comment_id),
       FOREIGN KEY(order_id) REFERENCES RemoteOrders(order_id)
   );
   ```

6. Mobile Link 同期ユーザ、パブリケーション、サブスクリプションを作成します。

   Interactive SQL で次の SQL 文を実行します。

```
CREATE SYNCHRONIZATION USER ml_sales1;
CREATE PUBLICATION order_publ (TABLE RemoteOrders, TABLE OrderComments);
CREATE SYNCHRONIZATION SUBSCRIPTION TO order_publ FOR ml_sales1
 TYPE TCPIP ADDRESS 'host=localhost';
```

   > **注記**
   >
   > Mobile Link サーバに接続する方法は、CREATE SYNCHRONIZATION SUBSCRIPTION 文の TYPE 句と ADDRESS 句を使用して指定します。

   パブリケーションを使用して、同期するデータを指定できます。この例では、RemoteOrders と OrderComments のテーブルをすべて指定します。

結果

SQL Anywhere クライアントデータベースが作成され、同期の準備が行われます。

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: Microsoft Excel との同期 [210 ページ]

前のタスク: レッスン 6: Mobile Link サーバの起動 [225 ページ]

次のタスク: レッスン 8: 同期 [228 ページ]

1.5.7.8 レッスン 8: 同期

dbmlsync ユーティリティを使用して、SQL Anywhere リモートデータベースの Mobile Link 同期を開始します。dbmlsync を起動する前に、注文データとコメントをリモートデータベースに追加します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. Mobile Link クライアントデータベースに接続していない場合は、Interactive SQL から接続します。

   次のコマンドを実行します。

   ```
   dbisql -c "SERVER=remote1;UID=DBA;PWD=passwd"
   ```

2. クライアントデータベース内の RemoteOrders テーブルに注文を追加します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   INSERT INTO RemoteOrders (order_id, product_id, quantity, order_status)
    VALUES (1,12312,10,'new');
   ```

3. クライアントデータベース内の OrderComments テーブルにコメントを追加します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   INSERT INTO OrderComments (comment_id, order_id, order_comment)
    VALUES (1,1,'send promotional material with the order');
   ```

4. これまでの変更内容をコミットします。

   Interactive SQL で次の SQL 文を実行します。

   ```
   COMMIT;
   ```

5. コマンドプロンプトで次のコマンドを実行します。

   ```
   dbmlsync -c "SERVER=remote1;UID=DBA;PWD=passwd" -o rem1.txt -v+
   ```

   次の表は、使用されている各 dbmlsync オプションを説明しています。

オプション	説明
-c	接続文字列を指定します。
-o	メッセージログファイル `rem1.txt` を指定します。
-v+	-v オプションは、ログを取る情報を指定します。-v+ を使用して、最大冗長ロギングをオンに設定します。

   Mobile Link 同期クライアントの起動が完了すると、同期が成功したことを示す出力画面が表示されます。SQL ベースの同期によって、クライアントの RemoteOrders テーブル内のローが、統合データベース内の RemoteOrders テーブルに転送されました。

   Java の処理によって、コメントが `order_central.xlsx` ワークシートに挿入されました。`order_central.xlsx` ワークシートに格納された情報がクライアントにダウンロードされます。

6. Interactive SQL で、OrderComments テーブルを選択して、ローがダウンロードされたことを確認します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   SELECT * FROM OrderComments;
   ```

> **注記**
>
> ダイレクトローハンドリングを使用してダウンロードされたローは、mlsrv17 -v+ オプションによっては出力されず、dbmlsync -v+ オプションによってリモートログに出力されます。

結果

リモートデータベースの注文データとコメントが更新され、リモートデータベースと統合データベースが同期されます。

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: Microsoft Excel との同期 [210 ページ]

前のタスク: レッスン 7: Mobile Link クライアントデータベースの設定 [226 ページ]

次のタスク: レッスン 9: クリーンアップ [230 ページ]

1.5.7.9 レッスン 9: クリーンアップ

チュートリアルをコンピュータから削除します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

手順

1. 以下のアプリケーションのインスタンスをすべて閉じます。
   - Interactive SQL
   - Microsoft Excel
2. Microsoft Excel ワークブック `order_central.xlsx` を削除します。

3. SQL Anywhere、Mobile Link、同期クライアントの各ウィンドウを閉じます。
4. 次の手順で、チュートリアルに関連するすべての ODBC データソースを削除します。
   a. ODBC データソースアドミニストレータを起動します。

      次のコマンドを実行します。

      ```
      odbcad32
      ```

   b. **excel_datasource** と **mlexcel_db** の各データソースを削除します。
5. 統合データベースとリモートデータベースを削除します。
   a. 統合データベースとリモートデータベースが保存されているディレクトリに移動します。
   b. `MLconsolidated.db`、`MLconsolidated.log`、`remote1.db`、および `remote1.log` を削除します。

結果

チュートリアルがコンピュータから削除されます。

タスクの概要: チュートリアル: Microsoft Excel との同期 [210 ページ]

前のタスク: レッスン 8: 同期 [228 ページ]

1.5.8 チュートリアル: XML との同期

このチュートリアルでは、XML ファイルとリモートクライアントの間でデータを同期する方法を示します。

前提条件

次の知識と経験が必要です。

- Java の知識
- XML の知識
- XML DOM の知識
- Mobile Link イベントスクリプトの基本的な知識

次のソフトウェアが必要です。

- SQL Anywhere17
- Java ソフトウェア開発キット
- XML DOM ライブラリ

統合データベースで次のロールおよび権限を持つ必要があります。

- SYS_AUTH_RESOURCE_ROLE 互換ロール

- MONITOR システム権限

リモートデータベースで次のロールおよび権限を持つ必要があります。

- SYS_REPLICATION_ADMIN_ROLE システムロール
- SYS_RUN_REPLICATION_ROLE システムロール

コンテキスト

ダイレクトローハンドリングを使用して、リモートデータと中央のデータソース、アプリケーション、または Web サービスとの通信ができます。

このチュートリアルでは、Mobile Link ダイレクトローハンドリングを実装して、サポートされている統合データソース以外のデータソースを使用できるようにします。このチュートリアルでは、例として Java 実装を使用します。

このチュートリアルでは、次の作業の方法について説明します。

- Java 用 Mobile Link サーバ Java API の使用
- Mobile Link ダイレクトローハンドリング用のメソッドの作成

1. レッスン 1: XML データソースの設定 [233 ページ]
   このレッスンでは、注文情報を保存する XML ファイルを作成します。
2. レッスン 2: Mobile Link 統合データベースの設定 [234 ページ]
   このレッスンでは、データベースを作成し、ODBC データソースを定義します。
3. レッスン 3: Mobile Link 統合データベースでのテーブルの作成 [236 ページ]
   このレッスンでは、Mobile Link 統合データベースに RemoteOrders テーブルを作成します。
4. レッスン 4: 同期スクリプトの追加 [238 ページ]
   このレッスンでは、SQL ローハンドリングとダイレクトローハンドリング用のスクリプトを統合データベースに追加します。
5. レッスン 5: Mobile Link のダイレクトローハンドリングのための Java クラスの作成 [240 ページ]
   このレッスンでは、ダイレクトローハンドリングを使用して、クライアントデータベース内の OrderComments テーブルのローを処理します。handle_UploadData イベント用のダイレクトローハンドリングの GetUpload メソッドを追加します。GetUpload では、アップロードされたコメントを XML ファイルに書き込みます。
6. レッスン 6: Mobile Link サーバの起動 [247 ページ]
   このレッスンでは、Mobile Link サーバを起動します。-c オプションを使用して Mobile Link サーバ (mlsrv17) を起動して統合データベースに接続し、-sl java オプションを使用して Java クラスをロードします。
7. レッスン 7: Mobile Link クライアントデータベースの設定 [248 ページ]
   このレッスンでは、SQL Anywhere データベースを統合データベースと Mobile Link クライアントに使用します。また、このチュートリアルの目的上、Mobile Link クライアント、統合データベース、および Mobile Link サーバはすべて同じコンピュータに置きます。
8. レッスン 8: 同期 [250 ページ]
   このレッスンでは、dbmlsync ユーティリティを使用して Mobile Link 同期を開始します。dbmlsync を起動する前に、注文データとコメントをリモートデータベースに追加します。
9. レッスン 9: クリーンアップ [252 ページ]
   チュートリアルをコンピュータから削除します。

タスクの概要: Mobile Link チュートリアル [96 ページ]

前のタスク: チュートリアル: Microsoft Excel との同期 [210 ページ]

次のタスク: チュートリアル: リモートデータベースの集中管理の使用 [253 ページ]

関連情報

Mobile Link 同期 [4 ページ]
SAP SQL Anywhere フォーラム

1.5.8.1 レッスン 1: XML データソースの設定

このレッスンでは、注文情報を保存する XML ファイルを作成します。

前提条件

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 次の内容の XML ファイルを作成します。

```
<?xml version="1.0" encoding="UTF-8"?>
<orders></orders>
```

2. XML ファイルを保存します。このチュートリアルでは、`c:¥MLobjxml` をサーバ側コンポーネントの作業ディレクトリとします。XML ファイルを `order_comments.xml` という名前でこのディレクトリに保存します。

結果

XML ファイルが作成されます。

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: XML との同期 [231 ページ]

次のタスク: レッスン 2: Mobile Link 統合データベースの設定 [234 ページ]

1.5.8.2 レッスン 2: Mobile Link 統合データベースの設定

このレッスンでは、データベースを作成し、ODBC データソースを定義します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

Mobile Link 統合データベースはデータの中央レポジトリであり、同期処理の管理に使用する Mobile Link のシステムテーブルとストアドプロシージャが含まれます。ダイレクトローハンドリングでは、統合データベース以外のデータソースと同期しますが、Mobile Link サーバが使用する情報を保持するために統合データベースも必要です。

> **注記**
>
> Mobile Link 統合データベースを Mobile Link システムオブジェクトと DSN を使用して設定済みの場合は、このレッスンは省略できます。

手順

1. SQL Central を起動します。

   スタート > プログラム > *SQL Anywhere17* > 管理ツール > *SQL Central* をクリックします。
2. ツール > *SQL Anywhere17* > データベースの作成 をクリックします。
3. 次へをクリックします。
4. このコンピュータにデータベースを作成をデフォルトのままにし、次へをクリックします。

5. **メインデータベースファイルを保存**フィールドに、データベースのファイル名およびパスを入力します。たとえば、`c:¥MLobjxml¥MLconsolidated.db` のように記述します。**次へ**をクリックします。
6. **データベース作成ウィザード**の残りの指示に従い、デフォルト値をそのまま使用します。DBA ユーザのユーザ ID とパスワードを指定するように求められたら、それぞれ **DBA** と **passwd** を入力します。

   データベースへの接続ページで、**最終切断後にデータベースを停止**オプションをオフにします。
7. **完了**をクリックします。

   MLconsolidated データベースが SQL Central に表示されます。
8. ウィンドウが自動的に閉じない場合は、**データベースの作成**ウィンドウの**閉じる**をクリックします。
9. **ツール** > *SQL Anywhere17* > *ODBC アドミニストレータを開く* をクリックします。
10. **ユーザ** *DSN* タブをクリックし、**追加**をクリックします。
11. **データソースの新規作成**ウィンドウで、*SQL Anywhere17* をクリックし、**完了**をクリックします。
12. *SQL Anywhere* の *ODBC* **設定**ウィンドウで、次の操作を行います。
    a. *ODBC* タブをクリックします。
    b. **データソース名**フィールドに **mlxml_db** と入力します。
    c. **ログイン**タブをクリックします。
    d. **ユーザ** *ID* フィールドに **DBA** と入力します。
    e. **パスワード**フィールドに、**passwd** と入力します。
    f. **サーバ名**フィールドに **MLconsolidated** と入力します。
    g. *OK* をクリックします。
13. ODBC データソースアドミニストレータを閉じます。

    ODBC **データソースアドミニストレータ**ウィンドウで *OK* をクリックします。

結果

データベースが作成され、ODBC データソースが定義されます。

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: XML との同期 [231 ページ]

前のタスク: レッスン 1: XML データソースの設定 [233 ページ]

次のタスク: レッスン 3: Mobile Link 統合データベースでのテーブルの作成 [236 ページ]

1.5.8.3 レッスン 3: Mobile Link 統合データベースでのテーブルの作成

このレッスンでは、Mobile Link 統合データベースに RemoteOrders テーブルを作成します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

RemoteOrders テーブルには次のカラムが含まれます。

order_id

注文のユニークな識別子です。
product_id

製品のユニークな識別子です。
quantity

品目の販売数です。
order_status

注文のステータスです。
last_modified

ローが最後に変更された日です。このカラムはタイムスタンプベースのダウンロードに使用します。このダウンロード方法は、効率的な同期のためにローをフィルタする一般的な方法です。

手順

1. Interactive SQL を使用してデータベースに接続します。

   Interactive SQL は、SQL Central またはコマンドプロンプトから起動できます。

   - SQL Central から Interactive SQL を起動するには、**MLconsolidated - DBA** データベースを右クリックし、*Interactive SQL* を開くをクリックします。
   - コマンドプロンプトで Interactive SQL を起動するには、次のコマンドを実行します。

     ```
     dbisql -c "DSN=mlxml_db"
     ```

2. Interactive SQL で次の SQL 文を実行し、RemoteOrders テーブルを作成します。

```
CREATE TABLE RemoteOrders (
    order_id            INTEGER NOT NULL,
    product_id          INTEGER NOT NULL,
    quantity            INTEGER,
    order_status        VARCHAR(10) DEFAULT 'new',
    last_modified       TIMESTAMP DEFAULT CURRENT TIMESTAMP,
    PRIMARY KEY(order_id)
);
```

   Interactive SQL によって、統合データベースに RemoteOrders テーブルが作成されます。

3. Interactive SQL で次の文を実行して Mobile Link のシステムテーブルとストアドプロシージャを作成します。

   C:¥Program Files¥SQL Anywhere 17¥ は、SQL Anywhere17 インストール環境のロケーションに置き換えてください。

```
READ "C:¥Program Files¥SQL Anywhere 17¥MobiLink¥setup¥syncsa.sql";
```

   Interactive SQL によって `syncsa.sql` が統合データベースに適用されます。`syncsa.sql` を実行すると、前に **ml_** が付いた一連のシステムテーブルとストアドプロシージャが作成されます。これらのテーブルとストアドプロシージャは、同期処理中に Mobile Link サーバによって使用されます。

結果

RemoteOrders テーブルが作成され、Mobile Link のシステムテーブルとストアドプロシージャがインストールされます

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: XML との同期 [231 ページ]

前のタスク: レッスン 2: Mobile Link 統合データベースの設定 [234 ページ]

次のタスク: レッスン 4: 同期スクリプトの追加 [238 ページ]

1.5.8.4 レッスン 4: 同期スクリプトの追加

このレッスンでは、SQL ローハンドリングとダイレクトローハンドリング用のスクリプトを統合データベースに追加します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

SQL ローハンドリングを使用すると、リモートデータを、Mobile Link 統合データベース内のテーブルと同期できます。SQL ベースのスクリプトでは、次の情報を定義します。

- Mobile Link クライアントからアップロードするデータを統合データベースに適用する方法。
- 統合データベースからダウンロードするデータ。

このレッスンでは、次の SQL ベースのアップロードイベントとダウンロードイベント用の同期スクリプトを作成します。

upload_insert

このイベントは、クライアントデータベースに挿入された新しい注文を統合データベースに適用する方法を定義します。

download_cursor

このイベントは、リモートクライアントにダウンロードする注文を定義します。

download_delete_cursor

このイベントは、アップロード専用の同期スクリプトを使用する場合に必要です。このチュートリアルでは、このイベントを無視するように Mobile Link サーバを設定します。

ダイレクトローハンドリングを使用して特別な処理を SQL ベースの同期システムに追加します。このレッスンでは、handle_UploadData、download_cursor、download_delete_cursor の各イベントに対応するメソッド名を登録します。

手順

1. 統合データベースに接続していない場合は、Interactive SQL で接続します。

   次のコマンドを実行します。

   ```
   dbisql -c "DSN=mlxml_db"
   ```

2. ml_add_table_script ストアドプロシージャを使用して、upload_insert、download_cursor、download_delete_cursor の各イベント用の SQL ベースのテーブルスクリプトを追加します。

Interactive SQL で次の SQL 文を実行します。upload_insert のスクリプトでは、アップロードされた order_id、product_id、quantity、order_status を Mobile Link 統合データベースに挿入します。download_cursor のスクリプトでは、タイムスタンプベースのフィルタを使用して、更新されたローをリモートクライアントにダウンロードします。

```
CALL ml_add_table_script( 'default', 'RemoteOrders',
    'upload_insert',
    'INSERT INTO RemoteOrders( order_id, product_id, quantity, order_status)
    VALUES( {ml r.order_id}, {ml r.product_id}, {ml r.quantity}, {ml
r.order_status} )' );

CALL ml_add_table_script( 'default', 'RemoteOrders',
    'download_cursor',
    'SELECT order_id, product_id, quantity, order_status
    FROM RemoteOrders WHERE last_modified >= {ml s.last_table_download}');
CALL ml_add_table_script( 'default', 'RemoteOrders',
    'download_delete_cursor', '--{ml_ignore}');
COMMIT;
```

3. handle_UploadData イベント用に Java メソッドを登録します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   CALL ml_add_java_connection_script( 'default',
       'handle_UploadData', 'MobiLinkOrders.GetUpload' );
   ```

   Interactive SQL によって、handle_UploadData イベント用の GetUpload メソッドが登録されます。次のレッスンでは、挿入されたデータを Mobile Link クライアントデータベース内の OrderComments テーブルから取得する GetUpload メソッドを作成します。

4. download_cursor と download_delete_cursor の各イベントを登録します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   CALL ml_add_table_script( 'default', 'OrderComments',
       'download_cursor', '--{ml_ignore}');
   CALL ml_add_table_script( 'default', 'OrderComments',
       'download_delete_cursor', '--{ml_ignore}');
   ```

   同期は双方向であり、アップロード専用ではないため、スクリプトを使用するときには、OrderComments テーブル用に download_cursor と download_delete_cursor の各イベントを登録してください。

5. これまでの変更内容をコミットします。

   Interactive SQL で次の SQL 文を実行します。

   ```
   COMMIT;
   ```

6. Interactive SQL を閉じます。

結果

handle_UploadData、handle_DownloadData、end_download、download_cursor、および download_delete_cursor イベントに対応するメソッド名が登録されます。

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: XML との同期 [231 ページ]

前のタスク: レッスン 3: Mobile Link 統合データベースでのテーブルの作成 [236 ページ]

次のタスク: レッスン 5: Mobile Link のダイレクトローハンドリングのための Java クラスの作成 [240 ページ]

1.5.8.5 レッスン 5: Mobile Link のダイレクトローハンドリングのための Java クラスの作成

このレッスンでは、ダイレクトローハンドリングを使用して、クライアントデータベース内の OrderComments テーブルのローを処理します。handle_UploadData イベント用のダイレクトローハンドリングの GetUpload メソッドを追加します。GetUpload では、アップロードされたコメントを XML ファイルに書き込みます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

次の手順では、処理用メソッドを含む Java クラスを作成する方法を示します。全リストについては、MobiLinkOrders Java コードリストを確認してください。

手順

1. MobiLinkOrders というクラスを作成します。

   次のコードを作成します。

```
import com.sap.ml.script.*;
import java.io.*;
import java.sql.*;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
```

```
import org.xml.sax.SAXException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
// For write operation
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
public class MobiLinkOrders {
```

2. クラスレベルの DBConnectionContext インスタンスおよび Document インスタンスを宣言します。Document クラスは、XML 文書をオブジェクトとして表します。

   次のコードを作成します。

   ```
       // Class level DBConnectionContext
       DBConnectionContext _cc;
       Document _doc;
   ```

   Mobile Link サーバによって DBConnectionContext のインスタンスがクラスコンストラクタに渡されます。DBConnectionContext には、Mobile Link 統合データベースとの現在の接続に関する情報がカプセル化されます。

3. クラスコンストラクタを作成します。

   クラスコンストラクタが、クラスレベルの DBConnectionContext インスタンスを設定します。

   次のコードを作成します。

   ```
       public MobiLinkOrders( DBConnectionContext cc ) throws IOException,
   FileNotFoundException {
           // Declare a class-level DBConnectionContext
           _cc = cc;
       }
   ```

4. GetUpload メソッドを作成します。

   GetUpload メソッドでは、OrderComments テーブルを表す UploadedTableData クラスインスタンスを取得します。OrderComments テーブルには、遠隔地の営業部員による特別なコメントが含まれます。このテーブルはレッスンの後半で作成します。

   UploadedTableData の getInserts メソッドでは、注文に対する新しいコメントの結果セットを返します。

   a. メソッドの宣言を作成します。

      次のコードを作成します。

      ```
          //  Method for the handle_UploadData synchronization event
          public void GetUpload( UploadData ut ) throws SQLException, IOException {
      ```

   b. アップロード済み挿入を Mobile Link クライアントから取得するコードを作成します。

      次のコードを作成します。

      ```
              // Get an UploadedTableData for the remote table
              UploadedTableData remoteOrdersTable =
   ut.getUploadedTableByName("OrderComments");
              // Get inserts uploaded by the MobiLink client
              // as a java.sql.ResultSet
              ResultSet insertResultSet = remoteOrdersTable.getInserts();
      ```

c. 既存の XML ファイル `order_comments.xml` を読み込むコードを作成します。

次のコードを作成します。

```
        try {
        readDom("order_comments.xml");
```

d. すべてのアップロード済み挿入を XML ファイルに追加するコードを作成します。

次のコードを作成します。

```
    // Write out each insert in the XML file
    while( insertResultSet.next() ) {
       buildXML(insertResultSet);
    }
```

e. XML ファイルに出力するコードを作成します。

次のコードを作成します。

```
        writeXML();
     }
```

f. ResultSet を閉じるコードを作成します。

次のコードを作成します。

```
        finally {
     // Close the result set of uploaded inserts
     insertResultSet.close();
}
    }
```

5. buildXML メソッドを作成します。

次のコードを作成します。

```
   private void buildXML( ResultSet rs ) throws SQLException {
      int order_id = rs.getInt(1);
      int comment_id = rs.getInt(2);
      String order_comment = rs.getString(3);
      // Create the comment object to be added to the XML file
      Element comment = _doc.createElement("comment");
      comment.setAttribute("id", Integer.toString(comment_id));
      comment.appendChild(_doc.createTextNode(order_comment));
      // Get the root element (orders)
      Element root = _doc.getDocumentElement();

      // Get each individual order
      NodeList rootChildren = root.getChildNodes();

      for(int i = 0; i < rootChildren.getLength(); i++) {
          // If the order exists, add the comment to the order
          Node n = rootChildren.item(i);
          if(n.getNodeType() == Node.ELEMENT_NODE) {
              Element e = (Element) n;
              int idIntVal = Integer.parseInt(e.getAttribute("id"));

              if(idIntVal == order_id) {
                  e.appendChild(comment);
                  // The comment has been added to the file, so exit
                  // the function.
                  return;
              }
          }
```

```
        }
        // If the order did not exist already, create it
        Element order = _doc.createElement("order");
        order.setAttribute("id", Integer.toString(order_id));
        // Add the comment to the new order
        order.appendChild(comment);
        root.appendChild(order);
    }
```

6. writeXML メソッドを作成します。

   次のコードを作成します。

```
    private void writeXML() {
        try {
            // Use a Transformer for output
            TransformerFactory tFactory = TransformerFactory.newInstance();
            Transformer transformer = tFactory.newTransformer();

            // The XML source is _doc
            DOMSource source = new DOMSource(_doc);
            // Write the xml data to order_comments.xml
            StreamResult result = new StreamResult(new
File("order_comments.xml"));
            transformer.transform(source, result);
        } catch (TransformerConfigurationException tce) {
            // Error generated by the parser
            System.out.println ("¥n** Transformer Factory error");
            System.out.println("   " + tce.getMessage() );
            // Use the contained exception, if any
            Throwable x = tce;
            if (tce.getException() != null) x = tce.getException();
            x.printStackTrace();
        } catch (TransformerException te) {
            // Error generated by the parser
            System.out.println ("¥n** Transformation error");
            System.out.println("   " + te.getMessage() );
            // Use the contained exception, if any
            Throwable x = te;
            if (te.getException() != null) x = te.getException();
            x.printStackTrace();
        }
    }
```

7. readDom メソッドを作成します。

   次のコードを作成します。

```
    private void readDom(String filename) {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        try {
            //parse the Document data into _doc
            DocumentBuilder builder = factory.newDocumentBuilder();
            _doc = builder.parse( new File(filename) );

        } catch (SAXException sxe) {
            // Error generated during parsing)
            Exception x = sxe;
            if (sxe.getException() != null) x = sxe.getException();
            x.printStackTrace();
        } catch (ParserConfigurationException pce) {
            // Parser with specified options can't be built
            pce.printStackTrace();
        } catch (IOException ioe) {
            // I/O error
            ioe.printStackTrace();
        }
```

```
    }
}
```

8. Java コードを `MobiLinkOrders.java` の名前で作業ディレクトリ `c:¥MLobjxml` に保存します。

   `MobiLinkOrders.java` のコードを検証するには、MobiLinkOrders Java コードリストを確認します。

9. クラスファイルをコンパイルします
   a. Java のソースファイルが含まれるディレクトリに移動します。
   b. Java 用の Mobile Link サーバ API ライブラリを参照する MobiLinkOrders をコンパイルします。

      %SQLANY17%`¥Java` にある `mlscript.jar` を参照し、XML DOM ライブラリが正しくインストールされていることを確認する必要があります。

      次のコマンドを実行して、*C:¥Program Files¥SQL Anywhere 17*`¥` を SQL Anywhere17 ディレクトリに置き換えます。

      ```
      javac -classpath "C:¥Program Files¥SQL Anywhere 17¥java¥mlscript.jar"
      MobiLinkOrders.java
      ```

結果

Mobile Link のダイレクトローハンドリングのための Java クラスが作成されます。

次のステップ

次のレッスンに進みます。

このセクションの内容:

MobiLinkOrders の Java コードのリスト [244 ページ]
このチュートリアルで使用している Java の MobiLinkOrders クラスの全コードを次に示します。

タスクの概要: チュートリアル: XML との同期 [231 ページ]

前のタスク: レッスン 4: 同期スクリプトの追加 [238 ページ]

次のタスク: レッスン 6: Mobile Link サーバの起動 [247 ページ]

1.5.8.5.1 MobiLinkOrders の Java コードのリスト

このチュートリアルで使用している Java の MobiLinkOrders クラスの全コードを次に示します。

```
import com.sap.ml.script.*;
```

```
    import java.io.*;
    import java.sql.*;

    import javax.xml.parsers.DocumentBuilder;
    import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.xml.sax.SAXException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
// For write operation
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
public class MobiLinkOrders {
    // Class level DBConnectionContext
    DBConnectionContext _cc;
    Document _doc;

    public MobiLinkOrders( DBConnectionContext cc ) throws IOException,
FileNotFoundException {
        // Declare a class-level DBConnectionContext
        _cc = cc;
    }
    //  Method for the handle_UploadData synchronization event
    public void GetUpload( UploadData ut ) throws SQLException, IOException {
        // Get an UploadedTableData for the remote table
        UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("OrderComments");
        // Get inserts uploaded by the MobiLink client
        // as a java.sql.ResultSet
        ResultSet insertResultSet = remoteOrdersTable.getInserts();

    try {
        readDom("order_comments.xml");

        // Write out each insert in the XML file
        while( insertResultSet.next() ) {
        buildXML(insertResultSet);
        }
        writeXML();
    } finally {
        // Close the result set of uploaded inserts
        insertResultSet.close();
    }
    }

    private void buildXML( ResultSet rs ) throws SQLException {
        int order_id = rs.getInt(1);
        int comment_id = rs.getInt(2);
        String order_comment = rs.getString(3);
        // Create the comment object to be added to the XML file
        Element comment = _doc.createElement("comment");
        comment.setAttribute("id", Integer.toString(comment_id));
        comment.appendChild(_doc.createTextNode(order_comment));
        // Get the root element (orders)
        Element root = _doc.getDocumentElement();

        // Get each individual order
        NodeList rootChildren = root.getChildNodes();

        for(int i = 0; i < rootChildren.getLength(); i++) {
            // If the order exists, add the comment to the order
            Node n = rootChildren.item(i);
```

```
            if(n.getNodeType() == Node.ELEMENT_NODE) {
                Element e = (Element) n;
                int idIntVal = Integer.parseInt(e.getAttribute("id"));

                if(idIntVal == order_id) {
                    e.appendChild(comment);
                    // The comment has been added to the file, so exit
                    // the function
                    return;
                }
            }
        }
        // If the order did not exist already, create it
        Element order = _doc.createElement("order");
        order.setAttribute("id", Integer.toString(order_id));
        // Add the comment to the new order
        order.appendChild(comment);
        root.appendChild(order);
    }
    private void writeXML() {
        try {
            // Use a Transformer for output
            TransformerFactory tFactory = TransformerFactory.newInstance();
            Transformer transformer = tFactory.newTransformer();

            // The XML source is _doc
            DOMSource source = new DOMSource(_doc);
            // Write the xml data to order_comments.xml
            StreamResult result = new StreamResult(new File("order_comments.xml"));
            transformer.transform(source, result);
        } catch (TransformerConfigurationException tce) {
            // Error generated by the parser
            System.out.println ("¥n** Transformer Factory error");
            System.out.println("   " + tce.getMessage() );
            // Use the contained exception, if any
            Throwable x = tce;
            if (tce.getException() != null) x = tce.getException();
            x.printStackTrace();
        } catch (TransformerException te) {
            // Error generated by the parser
            System.out.println ("¥n** Transformation error");
            System.out.println("   " + te.getMessage() );
            // Use the contained exception, if any
            Throwable x = te;
            if (te.getException() != null) x = te.getException();
            x.printStackTrace();
        }
    }
    private void readDom(String filename) {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        try {
            //parse the Document data into _doc
            DocumentBuilder builder = factory.newDocumentBuilder();
            _doc = builder.parse( new File(filename) );

        } catch (SAXException sxe) {
            // Error generated during parsing)
            Exception x = sxe;
            if (sxe.getException() != null) x = sxe.getException();
            x.printStackTrace();
        } catch (ParserConfigurationException pce) {
            // Parser with specified options can't be built
            pce.printStackTrace();
        } catch (IOException ioe) {
            // I/O error
            ioe.printStackTrace();
        }
    }
```

```
}
```

1.5.8.6 レッスン 6: Mobile Link サーバの起動

このレッスンでは、Mobile Link サーバを起動します。-c オプションを使用して Mobile Link サーバ (mlsrv17) を起動して統合データベースに接続し、-sl java オプションを使用して Java クラスをロードします。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

手順

統合データベースに接続し、mlsrv17 のコマンドラインでクラスをロードします。

`c:¥MLobjxml` をソースファイルがある実際のディレクトリに置き換えて、次のコマンドを実行します。

```
mlsrv17 -c "DSN=mlxml_db" -o serverOut.txt -v+ -dl -zu+ -x tcpip  -sl java (-cp c:
¥MLobjxml)
```

このチュートリアルで使用している Mobile Link サーバの各オプションの説明を次に示します。オプション -o、-v、-dl は、デバッグとトラブルシューティングの情報を提供します。これらのロギングオプションは、開発環境での使用に適しています。パフォーマンス上の理由から、一般的に -v+ と -dl は運用環境では使用しません。

オプション	説明
-c	続いて接続文字列を指定します。
-o	メッセージログファイル `serverOut.txt` を指定します。
-v+	-v オプションは、ログを取る情報を指定します。-v+ を使用して、最大冗長ロギングをオンに設定します。
-dl	画面にすべてのログメッセージを表示します。
-zu+	自動的に新しいユーザを追加します。
-x	Mobile Link クライアントの通信プロトコルとパラメータを設定します。
-sl java	クラスファイルを検索する一連のディレクトリを指定し、またサーバ起動時に Java VM をロードします。

Mobile Link サーバメッセージウィンドウが表示されます。

結果

Mobile Link サーバが起動します。

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: XML との同期 [231 ページ]

前のタスク: レッスン 5: Mobile Link のダイレクトローハンドリングのための Java クラスの作成 [240 ページ]

次のタスク: レッスン 7: Mobile Link クライアントデータベースの設定 [248 ページ]

1.5.8.7 レッスン 7: Mobile Link クライアントデータベースの設定

このレッスンでは、SQL Anywhere データベースを統合データベースと Mobile Link クライアントに使用します。また、このチュートリアルの目的上、Mobile Link クライアント、統合データベース、および Mobile Link サーバはすべて同じコンピュータに置きます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

Mobile Link クライアントデータベースを設定するには、RemoteOrders と OrderComments の各テーブルを作成します。RemoteOrders テーブルは、統合データベースの RemoteOrders テーブルに対応します。Mobile Link サーバでは、SQL ベースのスクリプトを使用してリモート注文が同期されます。OrderComments テーブルは、クライアントデータベースだけで使用されます。Mobile Link サーバでは、特別なイベントを使用して OrderComments テーブルが処理されます。

手順

1. dbinit コマンドラインユーティリティを使用して、Mobile Link クライアントデータベースを作成します。

   `c:¥MLobjxml` に移動して、次のコマンドを実行します。

   ```
   dbinit -i -k -dba DBA,passwd remote1
   ```

   -i オプションと -k オプションは、それぞれ jConnect のサポートと Watcom SQL の互換ビューを省略します。
2. dbsrv17 コマンドラインユーティリティを使用して、Mobile Link クライアントデータベースを起動します。

   次のコマンドを実行します。

   ```
   dbsrv17 remote1
   ```
3. Interactive SQL を使用して Mobile Link クライアントデータベースに接続します。

   次のコマンドを実行します。

   ```
   dbisql -c "SERVER=remote1;UID=DBA;PWD=passwd"
   ```
4. RemoteOrders テーブルを作成します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   CREATE TABLE RemoteOrders (
       order_id            INTEGER NOT NULL,
       product_id          INTEGER NOT NULL,
       quantity            INTEGER,
       order_status        VARCHAR(10) DEFAULT 'new',
       PRIMARY KEY(order_id)
   );
   ```
5. OrderComments テーブルを作成します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   CREATE TABLE OrderComments (
       comment_id          INTEGER NOT NULL,
       order_id            INTEGER NOT NULL,
       order_comment       VARCHAR(255),
       PRIMARY KEY(comment_id),
       FOREIGN KEY(order_id) REFERENCES RemoteOrders(order_id)
   );
   ```
6. Mobile Link 同期ユーザ、パブリケーション、サブスクリプションを作成します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   CREATE SYNCHRONIZATION USER ml_sales1;
   CREATE PUBLICATION order_publ (TABLE RemoteOrders, TABLE OrderComments);
   CREATE SYNCHRONIZATION SUBSCRIPTION TO order_publ FOR ml_sales1
    TYPE TCPIP ADDRESS 'host=localhost';
   ```

   注記

   Mobile Link サーバに接続する方法は、CREATE SYNCHRONIZATION SUBSCRIPTION 文の TYPE 句と ADDRESS 句を使用して指定します。

パブリケーションを使用して、同期するデータを指定できます。この例では、RemoteOrders と OrderComments のテーブルをすべて指定します。

結果

リモートデータベースに作成され、同期用に設定されます。

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: XML との同期 [231 ページ]

前のタスク: レッスン 6: Mobile Link サーバの起動 [247 ページ]

次のタスク: レッスン 8: 同期 [250 ページ]

1.5.8.8 レッスン 8: 同期

このレッスンでは、dbmlsync ユーティリティを使用して Mobile Link 同期を開始します。dbmlsync を起動する前に、注文データとコメントをリモートデータベースに追加します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. Mobile Link クライアントデータベースに接続していない場合は、Interactive SQL から接続します。

   次のコマンドを実行します。

   ```
   dbisql -c "SERVER=remote1;UID=DBA;PWD=passwd"
   ```

2. クライアントデータベース内の RemoteOrders テーブルに注文を追加します。

Interactive SQL で次の SQL 文を実行します。

```
INSERT INTO RemoteOrders (order_id, product_id, quantity, order_status)
 VALUES (1,12312,10,'new');
```

3. クライアントデータベース内の OrderComments テーブルにコメントを追加します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   INSERT INTO OrderComments (comment_id, order_id, order_comment)
    VALUES (1,1,'send promotional material with the order');
   ```

4. これまでの変更内容をコミットします。

   Interactive SQL で次の SQL 文を実行します。

   ```
   COMMIT;
   ```

5. コマンドプロンプトで次のコマンドを実行します。

   ```
   dbmlsync -c "SERVER=remote1;UID=DBA;PWD=passwd" -o rem1.txt -v+
   ```

   次の表には、このレッスンで使用する各 dbmlsync オプションの説明が含まれています。

オプション	説明
-c	接続文字列を指定します。
-o	メッセージログファイル `rem1.txt` を指定します。
-v+	-v オプションは、ログを取る情報を指定します。-v+ を使用して、最大冗長ロギングをオンに設定します。

   Mobile Link 同期クライアントの起動が完了すると、同期が成功したことを示す出力画面が表示されます。

6. SQL ベースの同期によって、クライアントの RemoteOrders テーブル内のローが、統合データベース内の RemoteOrders テーブルに転送されました。

   次の手順を実行して、クライアントの RemoteOrders テーブルに追加された情報が、統合データベース内の RemoteOrders テーブルに転送されたことを確認します。

   a. コマンドプロンプトで Interactive SQL を起動するには、次のコマンドを実行します。

      ```
      dbisql -c "DSN=mlxml_db"
      ```

   b. Interactive SQL で次の SQL 文を実行します。

      ```
      SELECT * FROM RemoteOrders;
      ```

7. Java の処理によってコメントが XML ファイルに挿入されました。

   `c:¥MLobjxml` に移動し、テキストエディタで `order_comments.xml` を開いて、コメントが挿入されていることを確認します。

結果

リモートデータベースと統合データベースが同期されます。

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: XML との同期 [231 ページ]

前のタスク: レッスン 7: Mobile Link クライアントデータベースの設定 [248 ページ]

次のタスク: レッスン 9: クリーンアップ [252 ページ]

1.5.8.9 レッスン 9: クリーンアップ

チュートリアルをコンピュータから削除します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. Interactive SQL のすべてのインスタンスを閉じます。
2. SQL Anywhere、Mobile Link、同期クライアントの各ウィンドウを閉じます。
3. 次の手順で、チュートリアルに関連するすべての ODBC データソースを削除します。
   a. ODBC アドミニストレータを起動します。

      次のコマンドを実行します。

      ```
      odbcad32
      ```

   b. **mlxml_db** データソースを削除します。
4. 統合データベースとリモートデータベースを削除します。
   a. 統合データベースとリモートデータベースが保存されているディレクトリに移動します。
   b. `MLconsolidated.db`、`MLconsolidated.log`、`remote1.db`、および `remote1.log` を削除します。

結果

チュートリアルがコンピュータから削除されます。

タスクの概要: チュートリアル: XML との同期 [231 ページ]

前のタスク: レッスン 8: 同期 [250 ページ]

1.5.9 チュートリアル: リモートデータベースの集中管理の使用

このチュートリアルでは、リモートデータベースの集中管理の設定プロセスについての説明と、いくつかの一般操作の実行方法が、順を追って示されます。

前提条件

このチュートリアルでは、チュートリアルを実行するローカルコンピュータに SQL Anywhere (Mobile Link と SQL Central を含む) が完全にインストールされていることを前提としています。

統合データベースで次のロールおよび権限を持つ必要があります。

- SYS_AUTH_RESOURCE_ROLE 互換ロール
- MONITOR システム権限

リモートデータベースで次のロールおよび権限を持つ必要があります。

- SYS_REPLICATION_ADMIN_ROLE システムロール
- SYS_RUN_REPLICATION_ROLE システムロール

コンテキスト

このチュートリアルに従って、集中管理を最初から設定したり、既存の同期システムに集中管理を追加したりできます。チュートリアルでは、手順全体を通して、既存の同期システムに集中管理を追加する場合に異なる処理が必要となる箇所では、そのことが示されます。

このチュートリアルでは、次の作業の方法について説明します。

- 統合データベースと Mobile Link プロジェクトを作成します。
- Mobile Link サーバを起動し、Mobile Link ユーザとエージェントを定義し、リモートデバイスでエージェントを設定します。
- 同期モデルを作成し、展開します。
- リモートタスクを操作します。

リモートデータベースの集中管理に関するいくつかの紹介ビデオやチュートリアル用ビデオが、オンラインで提供されています。

> **注記**
>
> ビデオチュートリアルは、バージョン 12.0.0 の SQL Anywhere に基づいています。図やプロシージャのいくつかは SQL Anywhere 17.0.4 とは異なります。

1. レッスン 1: 統合データベースの作成 [255 ページ]
   このレッスンでは、統合データベースを設定します。既存の同期システムがある場合は、レッスン 2 に進みます。Mobile Link プロジェクトを作成します。
2. レッスン 2: Mobile Link プロジェクトの作成 [256 ページ]
   集中管理を行う場合は、Mobile Link プロジェクトを作成してください。プロジェクトは、集中管理用に定義するさまざまなオブジェクトのコンテナとしての役割を果たします。
3. レッスン 3: Mobile Link サーバの起動 [258 ページ]
   このレッスンでは、Mobile Link サーバを起動します。Mobile Link サーバは、各リモートデバイスの統合データベースとエージェントデータベースの間において、リモートデータベースのデータの同期と、タスクおよびタスク結果の同期の両方を実行するために必要となります。
4. レッスン 4: Mobile Link ユーザの定義 [259 ページ]
   このレッスンでは、エージェントが使用する Mobile Link ユーザを定義します。既存の同期システムがあり、既存のいずれかの Mobile Link ユーザを Mobile Link エージェントが使用して同期する場合は、このレッスンを省略してもかまいません。
5. レッスン 5: エージェントの定義 [260 ページ]
   このレッスンでは、エージェントを定義します。このエージェントは、リモートデバイスで実行している Mobile Link エージェントのインスタンスを表します。
6. レッスン 6: リモートデバイスでのエージェントの設定 [262 ページ]
   このレッスンでは、Mobile Link エージェントを実行します。Mobile Link エージェントは、集中管理の対象となる、それぞれのリモートデバイス上で実行されている必要があります。このチュートリアルでは、Mobile Link サーバと同じコンピュータでエージェントを実行します。
7. レッスン 7: 同期モデルの作成 [264 ページ]
   このレッスンでは、同期モデルを作成します。
8. レッスン 8: 同期モデルの展開 [266 ページ]
   このレッスンでは、前のレッスンで作成した同期モデルを展開します。
9. レッスン 9:リモートタスクの作成 [267 ページ]
   このレッスンでは、リモートタスクを作成して、「Hello World」というメッセージをリモートデバイスに表示します。
10. レッスン 10: リモートタスクの展開 [269 ページ]
    このレッスンでは、リモートタスクを展開して、エージェントに割り当てできるようにします。
11. レッスン 11: リモートタスクのステータスのチェック [270 ページ]
    このレッスンでは、SQL Central でリモートタスクのステータスをチェックします。
12. レッスン 12: リモートデバイスでのリモートデータベースの作成 [271 ページ]
    このレッスンでは、リモートタスクを使用してリモートデバイスで新しいリモートデータベースを作成します。集中管理を既存の同期システムに追加している場合は、同期スケジュールのレッスンに進みます。
13. レッスン 13: 同期のスケジュール [273 ページ]
    次の手順では、リモートデータベースを定期的に同期するように Mobile Link エージェントを設定します。このことを行うには、スケジュールに基づいて実行されるリモートタスクを作成し、リモートタスクが実行されるたびにデータベースを同期します。
14. レッスン 14: スケジュールされた同期の変更 [274 ページ]

前のレッスンでは、リモートデータベースを 1 分ごとに同期するリモートタスクを作成しました。このレッスンでは、同期間隔を 1 時間に 1 回に変更します。

15. レッスン 15: 即時同期の強制 [276 ページ]
    これまでで、1 時間ごとに同期するリモートデータベースを設定しました。このレッスンでは、サーバ起動のリモートタスク (SIRT) を使用して、1 時間が経過する前に強制的に同期する方法について説明します。この方法は、特定のタスクの実行時期を集中管理する場合に便利です。
16. レッスン 16: リモートスキーマの変更 [277 ページ]
    このレッスンでは、リモートデータベースのスキーマを変更します。このチュートリアルでは、データベースのリモートスキーマ名を変更するたびに、スキーマの変更が発生します。リモートスキーマ名の変更は、強制的に実行されるものではなく、常にユーザが任意で行います。
17. レッスン 17: リモートデータベースの問い合わせ [279 ページ]
    このレッスンでは、リモートデータベースに対してクエリを行い、結果をサーバに返します。リモートデータベースの状態を正確に把握できるため、トラブルシューティングに非常に便利です。
18. レッスン 18: SIRT を使用したファイルのアップロード [281 ページ]
    このレッスンでは、サーバ起動のリモートタスク (SIRT) を使用して、リモートデバイスからファイルをアップロードします。ファイルに問題があるかどうかを管理者が検証できるため、リモートデバイスからのファイルのアップロードはトラブルシューティングに便利です。
19. レッスン 19: クリーンアップ [282 ページ]
    すべてのチュートリアルをコンピュータから削除します。

タスクの概要: Mobile Link チュートリアル [96 ページ]

前のタスク: チュートリアル: XML との同期 [231 ページ]

次のタスク: チュートリアル: スクリプトバージョン句を使用したスキーマの変更 [283 ページ]

1.5.9.1 レッスン 1: 統合データベースの作成

このレッスンでは、統合データベースを設定します。既存の同期システムがある場合は、レッスン 2 に進みます。Mobile Link プロジェクトを作成します。

前提条件

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 次のコマンドを実行して、このチュートリアルで使用するディレクトリを作成します。通常、統合ディレクトリには、中央のサーバに存在するすべてのデータベースと他のファイルが含まれます。

```
md c:¥cadmin_demo
```

```
md c:¥cadmin_demo¥consolidated
```

2. SQL Anywhere 統合データベースと、それに接続する ODBC データソースを作成します。

```
cd c:¥cadmin_demo¥consolidated
dbinit -dba DBA,passwd consol.db
start dbsrv17 consol.db
dbdsn -w cadmin_tutorial_consol consol -y -c
"UID=DBA;PWD=passwd;DBF=consol.db;SERVER=consol"
cd ..
```

3. Interactive SQL でデータベースに接続します。次のコマンドを実行します。

```
dbisql -c "DSN=cadmin_tutorial_consol"
```

4. Interactive SQL で次の文を実行して、`syncsa.sql` 設定スクリプトを使用して Mobile Link のシステムテーブルとストアドプロシージャを作成します。`C:¥Program Files¥`*SQL Anywhere17*`¥` は、SQL Anywhere17 インストール環境のロケーションに置き換えてください。

```
READ "C:¥Program Files¥SQL Anywhere 17¥MobiLink¥setup¥syncsa.sql";
```

5. *Interactive SQL* を閉じます。SQL 文を保存する必要はありません。

結果

SQL Anywhere データベースが作成され、これに接続されます。

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: リモートデータベースの集中管理の使用 [253 ページ]

次のタスク: レッスン 2: Mobile Link プロジェクトの作成 [256 ページ]

1.5.9.2 レッスン 2: Mobile Link プロジェクトの作成

集中管理を行う場合は、Mobile Link プロジェクトを作成してください。プロジェクトは、集中管理用に定義するさまざまなオブジェクトのコンテナとしての役割を果たします。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central を起動するには、スタート > プログラム > *SQL Anywhere17* > 管理ツール > *SQL Central* をクリックします。
2. ツール > *Mobile Link 17* > 新しいプロジェクト をクリックします。

   プロジェクト作成ウィザードが表示されます。
3. ようこそページで、プロジェクト名を **Central Admin Tutorial** に変更し、プロジェクトファイルのロケーションとして `C:/cadmin` を入力します。次へをクリックします。
4. 統合データベースを指定ページで、データベースの表示名に **Tutorial** を入力します。
5. 接続文字列に次の値を入力します。

   ```
   UID=DBA;PWD=passwd;DSN=cadmin_tutorial_consol
   ```
6. パスワードを記憶を選択し、次へをクリックします。

   データベースにテーブルが存在しないために同期モデルが作成されないことを示す警告を受信する場合があります。*OK* をクリックします。
7. リモートスキーマ名をプロジェクトに追加を選択し、スキーマ名として **Tutorial Application v1.0** と入力します。次へをクリックします。
8. データベースタイプとして *SQL Anywhere* を選択し、完了をクリックします。

   Mobile Link が今回初めて統合データベースを使用する場合、Mobile Link システム設定をインストールするかどうかを確認するメッセージが表示されます。Mobile Link システム設定をインストールすると、Mobile Link システムテーブルとプロシージャが追加されます。はいをクリックし、*OK* をクリックします。

結果

Mobile Link プロジェクトが作成されます。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 1: 統合データベースの作成 [255 ページ]

次のタスク: レッスン 3: Mobile Link サーバの起動 [258 ページ]

1.5.9.3 レッスン 3: Mobile Link サーバの起動

このレッスンでは、Mobile Link サーバを起動します。Mobile Link サーバは、各リモートデバイスの統合データベースとエージェントデータベースの間において、リモートデータベースのデータの同期と、タスクおよびタスク結果の同期の両方を実行するために必要となります。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

既存の同期システムがある場合は、サーバがすでに稼働しているため、このレッスンを省略してもかまいません。ただし、サーバコマンドラインで -ftr オプションと -ftru オプションが指定されていることを確認してください。これらのオプションは、リモートデバイスにファイルをダウンロードしたり、リモートデバイスからファイルをアップロードしたりする場合に必要です。

手順

コマンドプロンプトで次のコマンドを実行します。

```
md c:¥cadmin_demo¥consolidated¥upload
md c:¥cadmin_demo¥consolidated¥download
cd c:¥cadmin_demo¥consolidated
start mlsrv17.exe -c "DSN=cadmin_tutorial_consol;UID=DBA;PWD=passwd" -ftr download -
ftru upload -x tcpip(port=2439) -v+ -ot mlsrv.txt
cd ..
```

次に、使用するオプションの概要を示します。

-c

統合データベースに接続するために Mobile Link で使用される接続パラメータを指定します。

-ftr

Mobile Link でダウンロードするファイルを検索するディレクトリを指定します。

-ftru

Mobile Link でアップロードされたファイルを保存するディレクトリを指定します。

-x

同期クライアントを Mobile Link サーバに接続する方法を定義する通信パラメータを指定します。

-v+

最大冗長を指定します。この設定はデバッグに役立ちますが、運用環境ではパフォーマンスが低下する場合があります。

-ot

Mobile Link 出力メッセージが記録されるファイルを指定します。

結果

Mobile Link サーバが起動され、リモートデバイスからアップロードまたはリモートデバイスにダウンロードされるファイルが入るアップロードディレクトリおよびダウンロードディレクトリが作成されます。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 2: Mobile Link プロジェクトの作成 [256 ページ]

次のタスク: レッスン 4: Mobile Link ユーザの定義 [259 ページ]

1.5.9.4 レッスン 4: Mobile Link ユーザの定義

このレッスンでは、エージェントが使用する Mobile Link ユーザを定義します。既存の同期システムがあり、既存のいずれかの Mobile Link ユーザを Mobile Link エージェントが使用して同期する場合は、このレッスンを省略してもかまいません。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

エージェントがエージェントデータベースを同期する場合、Mobile Link サーバに対してエージェント自体を認証する必要があります。Mobile Link ユーザとオプションのパスワードを使用することによって認証します。通常、リモートデータベースの同期に使用する Mobile Link ユーザとパスワードは、エージェントデータベースの同期にエージェントが使用するものと同じです。

手順

1. SQL Central で、ビュー > フォルダ をクリックします。
2. Mobile Link17 で、*Central Admin Tutorial*、統合データベース、**Tutorial** の順に展開します。
3. ユーザを右クリックし、新規 > ユーザ をクリックします。

   ユーザ作成ウィザードが表示されます。
4. ようこそページで、新規ユーザの名前に **JOHN** と入力し、次へをクリックします。
5. 認証ページで、このユーザは、標準 *Mobile Link* 認証を使用する場合、接続のためにパスワードが必要です。をオンにし、パスワードとパスワードの確認の両方のフィールドに **passwd** と入力します。完了をクリックします。

   同期しようとするエージェントを認証しない場合は、この手順を省略し、-zu+ オプションを Mobile Link サーバコマンドラインに追加します。-zu+ を指定すると、最初に同期を行おうとするときに、各 Mobile Link ユーザが登録されます。

結果

Mobile Link 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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 3: Mobile Link サーバの起動 [258 ページ]

次のタスク: レッスン 5: エージェントの定義 [260 ページ]

1.5.9.5 レッスン 5: エージェントの定義

このレッスンでは、エージェントを定義します。このエージェントは、リモートデバイスで実行している Mobile Link エージェントのインスタンスを表します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

管理するリモートデバイスごとに別のエージェントを作成してください。

手順

1. Mobile Link17 で、*Central Admin Tutorial*、統合データベース、**Tutorial** の順に展開します。
2. エージェントを右クリックし、新規 > エージェント をクリックします。

   Mobile Link エージェント作成ウィザードが表示されます。
3. ようこそページで、エージェントを *1* つ設定するを選択し、次へをクリックします。
4. エージェント *ID* ページで、エージェント *ID* に **AID_JOHN** と入力します。エージェント ID には任意の値を指定できますが、各エージェントにはユニークな ID が必要です。エージェント ID は、プレフィクス AID_ で始まり、その後にエージェントが使用する Mobile Link ユーザ名が続くのが一般的です。必要に応じて、説明フィールドにエージェントの説明を入力することもできます。次へをクリックします。
5. リモートデータベースページでは、このエージェントが管理するリモートデータベースを定義できます。この処理ではデータベースは作成されません。実際の作成は後で行います。リモートスキーマ名で **Tutorial Application v1.0** を選択します。これは、前のレッスンでドロップダウンリストから定義した名前です。
6. データベース接続文字列フィールドに次の接続文字列を入力します。

   ```
   start=dbsrv17;SERVER=tutorial_v1;DBF={db_location}
   ¥tutorial_v1.db;UID=DBA;PWD=passwd
   ```

   この文字列値はマクロ {db_location} を使用しています。このマクロは、アプリケーションデータベースが保存される時点で、リモートデバイスでディレクトリに置き換えられます。次へをクリックします。
7. エージェント設定ページで、**30** と入力し、同期間隔に秒を選択します。同期間隔は、エージェントがエージェントデータベースを同期する頻度を制御します。エージェントデータベースの同期は、エージェントが実行する新しいタスクを受け取り、実行済みのタスクの結果をアップロードする方法となります。
8. エージェント設定ページで、**10** と入力し、管理ポーリング間隔に秒を選択します。管理ポーリング間隔によって、サーバからの同期要求や他のアクションの実行要求をエージェントがチェックする頻度が決まります。

   > **注記**
   >
   > 同期間隔または管理ポーリング間隔に選択した値が小さいと、応答性に非常に優れたエージェントとなり、デモやトラブルシューティングに効果的です。ただし、小さい値を運用環境でグローバルに使用すると、サーバの負荷が大きくなり、パフォーマンスが低下します。
9. 完了をクリックします。

結果

エージェントが作成および設定されます。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 4: Mobile Link ユーザの定義 [259 ページ]

次のタスク: レッスン 6: リモートデバイスでのエージェントの設定 [262 ページ]

1.5.9.6 レッスン 6: リモートデバイスでのエージェントの設定

このレッスンでは、Mobile Link エージェントを実行します。Mobile Link エージェントは、集中管理の対象となる、それぞれのリモートデバイス上で実行されている必要があります。このチュートリアルでは、Mobile Link サーバと同じコンピュータでエージェントを実行します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 通常はリモートデバイス上にあるファイルを含むディレクトリを作成します。

```
md c:¥cadmin_demo¥remote
cd c:¥cadmin_demo¥remote
```

2. Mobile Link エージェントを設定モードで次のように実行します。

```
mlagent -cr -db . -x tcpip{host=localhost;port=2439} -a AID_JOHN  -u JOHN -p
passwd
```

この手順では、エージェントデータベースを作成し、一部の設定情報をそこに保存します。指定されたオプションがデータベースに保存されると、エージェントが停止します。次に、使用するオプションの概要を示します。

-cr

エージェントを設定モードで実行し、設定モードの前の実行で保存された設定内容を破棄することを指定します。

-db

エージェントがアプリケーションデータベースを作成する場所を指定します。この場所は、{db_location} マクロの値になります。

-x

エージェントデータベースを同期する (新しいタスクを受け取り、実行済みのタスクの結果をアップロードする) ために、エージェントを Mobile Link サーバに接続する方法を指定します。集中管理を既存の同期システムに追加している場合は、このオプションで指定した値を、Mobile Link サーバへの接続に適した文字列に変更する必要があります。

-a

このエージェントのエージェント ID を指定します。SQL Central を使用して統合データベースで前に作成したエージェント ID と同じ値が指定されています。

-u

エージェントデータベースの同期時にエージェントが使用する Mobile Link ユーザを指定します。この値は、主にエージェントを認証するために Mobile Link サーバで使用されます。

-p

-u オプションで指定された Mobile Link ユーザのパスワードを指定します。

3. リモートデバイスで Mobile Link エージェントを実行します。このチュートリアルでは、エージェントの実行を次のように明示的に開始します。

```
start mlagent -v9 -ot agent.txt
```

次に、エージェントを実行するためにこのレッスンで使用するオプションの概要を示します。

-v9

最大冗長を使用します。このロギングオプションは、開発環境での使用に適しています。パフォーマンス上の理由から、一般的に -v9 は運用環境では使用しません。

-ot

エージェントログの出力ファイルを指定します。

4. これで、Mobile Link エージェントが稼働し、正常に同期されます。確認するには、SQL Central に戻ります。フォルダビューの *Mobile Link 17* で、*Central Admin Tutorial* を展開します。統合データベースを展開し、`Tutorial` を選択します。エージェント、*AID_JOHN* の順に選択し、右ウィンドウ枠でイベントタブを確認します。エージェントの最初の同期を示すエントリが表示されています。

> **注記**
>
> エージェント設定に関する運用時の考慮事項
>
> 集中管理を運用環境で使用する場合は、次の考慮事項に注意してください。
>
> - -u オプションと -p オプションに指定した値を、同期システムに適した Mobile Link ユーザとパスワードの組み合わせに変更する必要があります。
> - -on オプションを使用すると、エージェントが生成するログファイルのサイズを制限できます。
> - Mobile Link エージェントがリモートデバイスで稼働している場合、リモートデバイスにはリモート管理のみが可能です。エージェントが常に稼働していることを確認する措置を取ってください。このことを行うには、たとえば、エージェントをサービスとして実行したり、エージェントをレジストリで Run 起動グループに追加したりします。

結果

Mobile Link エージェントが実行され、同期されます。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 5: エージェントの定義 [260 ページ]

次のタスク: レッスン 7: 同期モデルの作成 [264 ページ]

1.5.9.7 レッスン 7: 同期モデルの作成

このレッスンでは、同期モデルを作成します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

集中管理を既存の同期システムに追加している場合は、レッスン 9 に進みます。同期モデルを展開します。

手順

1. 統合データベースでリモートデータベース用にテーブルを定義します。SQL Central のフォルダビューの *Mobile Link 17* で、*Central Admin Tutorial* 統合データベースを展開します。**Tutorial - DBA** を右クリックして、*Interactive SQL* を開くをクリックします。
2. *SQL* 文ウィンドウ枠で、次のコマンドを入力します。

```
CREATE TABLE customer(
```

```
    cust_id         INTEGER PRIMARY KEY,
    f_name          VARCHAR(100),
    l_name          VARCHAR(100)
)
```

3. F5 キーを押して SQL を実行します。Interactive SQL を閉じます。SQL 文を保存する必要はありません。
4. SQL Central のフォルダビューで、*Central Admin Tutorial* > 新規 > 同期モデル を右クリックします。
5. ようこそページで、新しい同期モデルの名前に **tutorial1** と入力します。次へをクリックします。
6. プライマリキー要件ページで、スキーマが同期要件を満たしていることを確認するために 3 つのチェックボックスをすべてオンにします。次へをクリックします。
7. 統合データベーススキーマページで **Tutorial** データベースを選択し、次へをクリックします。
8. リモートデータベーススキーマページでいいえ、新しいリモートデータベーススキーマを作成しますを選択し、次へをクリックします。
9. 新しいリモートデータベーススキーマページで、customer テーブルが選択されていることを確認し、次へをクリックします。完了をクリックします。

結果

これで、**customer** という単一のテーブルを含む同期モデルが作成されました。このテーブルは、リモートデータベースと統合データベース間で同期できます。次の手順では、このモデルを展開して統合データベースに同期オブジェクトを作成し、リモートデータベースを作成する SQL を生成します。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 6: リモートデバイスでのエージェントの設定 [262 ページ]

次のタスク: レッスン 8: 同期モデルの展開 [266 ページ]

1.5.9.8 レッスン 8: 同期モデルの展開

このレッスンでは、前のレッスンで作成した同期モデルを展開します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central のフォルダビューの *Mobile Link 17* で、 *Central Admin Tutorial* 同期モデル を展開します。 **tutorial1** を右クリックして、展開をクリックします。

   同期モデル展開ウィザードが表示されます。
2. ようこそページで、ウィザードによって生成されたファイルを保管するデフォルトロケーションをそのまま使用し、次へをクリックします。
3. クライアントネットワークオプションページで、以下のオプションを選択し、次へをクリックします。

   プロトコル

   TCP/IP

   ホスト

   localhost

   ポート

   2439
4. *Mobile Link* ユーザとパスワードページで、リモートタスクに適したマクロの値の使用を選択します。

   {ml_username} および *{ml_password}* のマクロ値は、生成された SQL ファイルで使用され、リモートデバイスで SQL が実行されるときに Mobile Link エージェントが使用する Mobile Link ユーザとパスワードに置き換えられます。同期プロファイルは tutorial1_{ml_username} という名前で自動的に作成されます。このとき、{ml_username} マクロは Mobile Link ユーザ名に置き換えられ、ここでは JOHN に置き換えられます。
5. 同期プロファイルページで、同期プロファイル名フィールドに *tutorial1_JOHN* と入力します。
6. データベースの同期を準備する方法を選択しますページが表示されるまで次へをクリックし、以下のタスクを実行します。
   - 統合データベースの同期を準備するために作成された *SQL* スクリプトの処理方法を指定してくださいで、統合データベースに対して実行しますを選択します。
   - リモートデータベースの同期を準備するために作成された *SQL* スクリプトの処理方法を指定してくださいで、実行しませんを選択します。
   - 次へをクリックしてから、完了をクリックします。

   同期モデルから移動する場合、変更内容を保存するよう求められます。はいをクリックします。

結果

これで、同期モデルの作成と展開が完了しました。モデルを展開すると、スクリプトが統合データベースに追加され、リモートデータベースを同期できるようになります。また、SQL ファイルも `c:¥cadmin_demo¥Central Admin Tutorial ¥tutorial1_deploy¥` ディレクトリに生成され、リモートデータベースの作成に使用できます。これらのファイルをこの時点で確認してください。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 7: 同期モデルの作成 [264 ページ]

次のタスク: レッスン 9:リモートタスクの作成 [267 ページ]

1.5.9.9 レッスン 9:リモートタスクの作成

このレッスンでは、リモートタスクを作成して、「Hello World」というメッセージをリモートデバイスに表示します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

集中管理におけるほとんどのアクションには、リモートタスクが関係しています。リモートタスクは、管理者が作成するコマンドのコレクションです。リモートタスクは、1 つまたは複数のエージェントに割り当てることができます。エージェントに割り当てられたリモートタスクは、エージェントが次にエージェントデータベースを同期するときにエージェントにダウンロードされます。エージェントは、タスクを適切な時間に実行し、実行に関する情報をアップロードします。

手順

1. 新しいリモートタスクを作成します。SQL Central のフォルダビューの *Mobile Link 17* で、*Central Admin Tutorial* を展開し、リモートタスクを右クリックして 新規 > リモートタスク をクリックします。

   リモートタスク作成ウィザードが表示されます。
2. ようこそページで、名前フィールドに **Hello World** と入力します。次へをクリックします。
3. トリガのメカニズムページで、エージェントで受信されたときをオンにし、完了をクリックしてウィザードを完了します。
4. フォルダビューで、新しく作成した **Hello World** タスクをクリックします。右ウィンドウ枠に、コマンドをタスクに追加できるコマンドタブが表示されます。
5. コマンドタブで、コマンドタイプドロップダウンリストからプロンプトを選択します。メッセージフィールドに **Hello World** と入力します。
6. 2 つ目のコマンドをタスクに追加するには、新しいコマンドが表示されるまで [Tab] キーを押すか、*[コマンドを追加]* ボタンをクリックします。2 つ目のコマンドのコマンドタイプをプロンプトに設定し、メッセージフィールドに **Hello Again** と入力します。

   ここで作成した Hello World タスクは、デザイン時のタスクです。このタスクは、ローカルコンピュータでプロジェクトに保存されます。タスクをエージェントに割り当てる前に、タスクを展開して統合データベースにコピーする必要があります。

結果

リモートタスクが作成され、展開の準備が完了します。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 8: 同期モデルの展開 [266 ページ]

次のタスク: レッスン 10: リモートタスクの展開 [269 ページ]

1.5.9.10 レッスン 10: リモートタスクの展開

このレッスンでは、リモートタスクを展開して、エージェントに割り当てできるようにします。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central のフォルダビューの *Mobile Link 17* で、*Central Admin Tutorial*、リモートタスクの順に展開し、*Hello World* タスクを右クリックして展開をクリックします。

   リモートタスク展開ウィザードが表示されます。
2. タスクの名前と説明ページでデフォルトをそのまま使用し、次へをクリックします。

   このようにすると、展開済みタスクが、デザイン時のタスクと同じ名前になります。同じデザイン時のタスクを 2 回目に展開しないかぎり、通常はこのようにします。2 回目に展開する場合は、展開済みタスクの名前を変更する必要があります。
3. 受信者ページでは、展開済みタスクを既存のエージェントに割り当てることができます。この操作を別の手順で行うこともできます。受信者ドロップダウンから特定のエージェントを選択します。エージェントリストで **AID_JOHN** を選択し、次へをクリックします。
4. 配信オプションページで、エージェントの次回の同期時をオンにし、次へをクリックします。
5. 結果とステータスのレポートページで、どちらの質問に対してもすぐに結果とステータスを送信をオンにします。これにより、タスクを実行したときに通知をタイムリーに受け取ることができます。ルーチンタスクや繰り返しタスクの場合は、フィードバックの受信頻度を抑える (特に成功した場合) ことで、エージェントデータベースの同期回数と Mobile Link サーバでの負荷が削減されます。
6. 完了をクリックします。

   エージェント **AID_JOHN** が次回にエージェントデータベースを同期すると、新しいタスクを受け取り、それを実行します。*Hello World* と *Hello Again* というテキストが表示されたメッセージボックスで、*OK* をクリックします。

   これで、フォルダビューのリストに **Hello World** タスクの 2 つのコピーができます。展開済みコピーは、リモートタスク > 展開済みタスク のフォルダビューに表示されます。これは、統合データベースにおけるコピーです。展開済みタスクのコピーは変更できません。デザイン時のタスクのコピーは、リモートタスクに引き続き表示されます。このタスクは変更可能であり、新しい名前でもう一度展開できます。

   展開済みタスクを右クリックして、受信者の追加を選択すると、展開済みタスクをいつでも他のエージェントに割り当てることができます。

結果

リモートタスクが実行されます。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 9:リモートタスクの作成 [267 ページ]

次のタスク: レッスン 11: リモートタスクのステータスのチェック [270 ページ]

1.5.9.11 レッスン 11: リモートタスクのステータスのチェック

このレッスンでは、SQL Central でリモートタスクのステータスをチェックします。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. SQL Central のフォルダビューの *Mobile Link 17* で、*Central Admin Tutorial* > リモートタスク > 展開済みタスク の順に展開します。`Hello World` タスクの展開済みバージョンをクリックし、右ウィンドウ枠で結果タブを選択します。タブが自動的に再表示されるまで待つか、F5 キーを押してすぐに再表示します。結果タブには、タスクのコマンドごとに行が表示され、コマンドが成功したか失敗したかを示す結果コードが表示されます。結果コードの 0 は、成功を示します。
2. タスクの実行結果を異なる方法で表示するには、展開済みタスクの受信者タブを選択するか、タスクを実行したエージェントのイベントタブまたはタスクタブを見ます。

結果

タスクの実行結果が表示されます。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 10: リモートタスクの展開 [269 ページ]

次のタスク: レッスン 12: リモートデバイスでのリモートデータベースの作成 [271 ページ]

1.5.9.12 レッスン 12: リモートデバイスでのリモートデータベースの作成

このレッスンでは、リモートタスクを使用してリモートデバイスで新しいリモートデータベースを作成します。集中管理を既存の同期システムに追加している場合は、同期スケジュールのレッスンに進みます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 新しいリモートタスクを作成します。SQL Central のフォルダビューの *Mobile Link 17* で、*Central Admin Tutorial* を展開します。リモートタスクを右クリックし、新規 > リモートタスク をクリックします。

   リモートタスク作成ウィザードが表示されます。
2. ようこそページで、名前フィールドに **Create DB** と入力します。前に作成したタスクと異なり、このタスクはリモートデータベースでの作成または操作になるため、タスクにリモートデータベースが必要、またはリモートデータベースを作成をオンにし、リモートスキーマ名 **Tutorial Application v1.0** を選択します。これにより、このタスクで実行されるデータベースアクションが識別されます。次へをクリックします。
3. トリガのメカニズムページで、エージェントで受信されたときをオンにし、完了をクリックしてウィザードを完了します。
4. フォルダビューで、新しく作成した **Create DB** タスクをクリックします。
5. 右側のコマンドウィンドウ枠からリモートタスクにコマンドを追加します。
   a. 最初のコマンドでは、リモートデバイスで新しい空のデータベースを作成します。コマンドタイプを、データベースを作成に設定します。
   b. ファイル名を **{db_location}¥tutorial_v1.db** に設定します。このファイル名は、エージェントの設定で指定した接続文字列内のファイル名に対応しています。
   c. 新しいコマンドが表示されるまで Tab キーを押します。

d. 2 つ目のコマンドでは、新しいデータベースでスキーマを作成します。コマンドタイプを、*SQL を実行*に設定します。インポートをクリックします。
e. 開くウィンドウでファイル `c:¥cadmin¥Central Admin Tutorial¥tutorial1_deploy ¥remote_setup.sql` を選択し、開くをクリックします。同期モデルをコマンドに展開したときに生成されたリモートデータベースを初期化する SQL がインポートされます。

6. これで、リモートタスクが完了しました。タスクを展開して、次の手順でエージェント **AID_JOHN** に割り当てます。
   a. フォルダビューで **Create DB** タスクを右クリックし、展開をクリックします。

      リモートタスク展開ウィザードが表示されます。
   b. タスクの名前と説明ページでデフォルトをそのまま使用し、次へをクリックします。
   c. 受信者ドロップダウンから特定のエージェントを選択します。エージェントリストで **AID_JOHN** を選択し、次へをクリックします。
   d. 配信オプションページで、エージェントの次回の同期時をオンにし、次へをクリックします。
   e. 結果とステータスのレポートページで、どちらの質問に対してもすぐに結果とステータスを送信を選択します。
   f. 完了をクリックします。
7. タスクが成功したかどうかを確認します。
   a. SQL Central の フォルダビューの *Mobile Link 17* で、*Central Admin Tutorial* > 統合データベース > チュートリアル > エージェント の順に展開し、**AID_JOHN** をクリックします。
   b. イベントタブを選択し、**Create DB** タスクを検索します。タブが自動的に再表示されるまで待つか、F5 キーを押してすぐに再表示します。

結果

リモートデバイスに新しい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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 11: リモートタスクのステータスのチェック [270 ページ]

次のタスク: レッスン 13: 同期のスケジュール [273 ページ]

1.5.9.13 レッスン 13: 同期のスケジュール

次の手順では、リモートデータベースを定期的に同期するように Mobile Link エージェントを設定します。このことを行うには、スケジュールに基づいて実行されるリモートタスクを作成し、リモートタスクが実行されるたびにデータベースを同期します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

これまでに作成した他のタスクは、1 回のみ実行されることが、このタスクと異なります。このタスクは、リモートデバイスに残り、停止するまで定期的に実行されます。

手順

1. 新しいリモートタスクを作成します。SQL Central のフォルダビューの *Mobile Link 17* で、*Central Admin Tutorial* を展開します。リモートタスクを右クリックし、新規 > リモートタスク をクリックします。

   リモートタスク作成ウィザードが表示されます。
2. ようこそページで、名前フィールドに **Sync** と入力します。タスクにリモートデータベースが必要、またはリモートデータベースを作成をオンにし、リモートスキーマ名 **Tutorial Application v1.0** を選択します。これにより、このタスクで実行されるデータベースアクションが識別されます。次へをクリックします。
3. トリガのメカニズムページで、スケジュールに従うを選択し、次へをクリックします。
4. 開始日時ページで、デフォルト値をそのまま使用します。これで、タスクの実行をすぐに開始できます。次へをクリックします。
5. 繰り返しページで、次の間隔で繰り返しをオンにし、間隔を 1 分に設定します。完了をクリックしてウィザードを完了します。
6. フォルダビューで、新しく作成した **Sync** タスクをクリックします。
7. 同期を開始する 1 つのコマンドをタスクに追加します。
   a. コマンドタブで、最初のコマンドのコマンドタイプを同期に設定します。
   b. 同期プロファイルに **tutorial1_JOHN** と入力します。これは、同期モデルを展開したときに作成した同期プロファイルです。
8. これで、同期タスクが完了しました。**Sync** を右クリックして、展開をクリックします。次へをクリックします。
9. 受信者ドロップダウンで特定のエージェントをクリックし、タスクをエージェント **AID_JOHN** に割り当てます。次へをクリックし、もう一度次へをクリックします。
10. 結果とステータスのレポートページで、タスクが成功した場合を後でステータスのみ送信に設定し、タスクが失敗した場合をすぐに結果とステータスを送信に設定します。

このタスクは頻繁に繰り返されるため、要求するフィードバックを制限してパフォーマンスを向上させてください。

11. 完了をクリックします。

結果

この新しいタスクをエージェントが受け取ると、リモートデータベースが 1 分ごとに同期されます。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 12: リモートデバイスでのリモートデータベースの作成 [271 ページ]

次のタスク: レッスン 14: スケジュールされた同期の変更 [274 ページ]

1.5.9.14 レッスン 14: スケジュールされた同期の変更

前のレッスンでは、リモートデータベースを 1 分ごとに同期するリモートタスクを作成しました。このレッスンでは、同期間隔を 1 時間に 1 回に変更します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

リモートタスクを展開すると、展開済みバージョンは変更できなくなります。代わりに、必要な変更を加えた新しいリモートタスクを作成してから既存のタスクをキャンセルし、新しいタスクを展開して置き換えます。

手順

1. まず、既存の展開済みタスクをテンプレートとして使用して、希望の繰り返し間隔で新しいリモートタスクを作成します。
   a. SQL Central のフォルダビューの *Mobile Link 17* で、*Central Admin Tutorial* > リモートタスク > 展開済みタスク の順に展開します。**Sync** を右クリックし、コピーを選択してタスクをクリップボードにコピーします。
   b. リモートタスクを右クリックし、貼り付けを選択します。リモートタスク名を変更するかどうかを尋ねるウィンドウが表示されます。**Sync every hour** と入力し、*OK* をクリックします。
   c. 新しい **Sync every hour** タスクを右クリックし、プロパティを選択します。プロパティウィンドウの繰り返しページで、次の間隔で繰り返しの値を *1* 分から *1* 時間に変更し、*OK* をクリックします。
2. 次に、1 分ごとに同期を行う既存のリモートタスクをキャンセルします。
   a. フォルダビューで、展開済みバージョンの **Sync** タスクをクリックし、右ウィンドウ枠で受信者タブをクリックします。
   b. エージェント **AID_JOHN** のテーブル内のエントリを右クリックし、キャンセルをクリックします。
3. 最後に、新しい **Sync every hour** タスクを展開してエージェント **AID_JOHN** に割り当てます。
   a. **Sync every hour** を右クリックして、展開をクリックします。次へをクリックします。
   b. 受信者ドロップダウンリストから特定のエージェントをクリックし、タスクをエージェント **AID_JOHN** に割り当てます。次へをクリックし、もう一度次へをクリックします。
   c. 結果とステータスのレポートページで、タスクが成功した場合を後でステータスのみ送信に設定し、タスクが失敗した場合をすぐに結果とステータスを送信に設定します。
   d. 完了をクリックします。

結果

この新しいタスクをエージェントが受け取ると、リモートデータベースが 1 分ごとではなく 1 時間ごとに同期されます。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 13: 同期のスケジュール [273 ページ]

次のタスク: レッスン 15: 即時同期の強制 [276 ページ]

1.5.9.15 レッスン 15: 即時同期の強制

これまでで、1 時間ごとに同期するリモートデータベースを設定しました。このレッスンでは、サーバ起動のリモートタスク (SIRT) を使用して、1 時間が経過する前に強制的に同期する方法について説明します。この方法は、特定のタスクの実行時期を集中管理する場合に便利です。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

SQL Central のフォルダビューの *Mobile Link 17* で、*Central Admin Tutorial* > リモートタスク > 展開済みタスク の順に展開します。**Sync every hour** を右クリックし、すべての受信者について開始をクリックします。

結果

タスクのすべての受信者は、次にサーバをポーリングするときに、タスクをすぐに実行するよう指示されます。エージェントがサーバをポーリングする頻度は、エージェントの管理ポーリング間隔プロパティで制御されます。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 14: スケジュールされた同期の変更 [274 ページ]

次のタスク: レッスン 16: リモートスキーマの変更 [277 ページ]

1.5.9.16 レッスン 16: リモートスキーマの変更

このレッスンでは、リモートデータベースのスキーマを変更します。このチュートリアルでは、データベースのリモートスキーマ名を変更するたびに、スキーマの変更が発生します。リモートスキーマ名の変更は、強制的に実行されるものではなく、常にユーザが任意で行います。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

1 つのリモートデータベースに対して実行できるすべてのリモートタスクは、同じリモートスキーマ名を持つ他のリモートデータベースに対しても実行できるようにしてください。タスクが失敗するか成功するかに影響を及ぼす変更をデータベースに加えた場合は、必ずデータベースのリモートスキーマ名を変更してください。リモートデータベースの状態が影響するタスク内のコマンドは、同期コマンドと SQL を実行コマンドのみです。

同期コマンドは、リモートデータベースに同期プロファイルが存在するかどうかが影響するため、同期プロファイルを追加または削除するたびにリモートスキーマ名を変更する必要があります。

SQL を実行コマンドは、スキーマの一部として通常処理する多くのデータベースオブジェクトの状態が影響します。たとえば、データベースへのテーブルの追加やデータベースからのテーブルの削除、データベース内のテーブルの定義の変更、ストアドプロシージャの追加や削除を実行すると、SQL を実行コマンドへの影響が発生し、これにより、リモートスキーマ名を変更することが必要になります。

手順

1. SQL Central の フォルダビューに戻ります。*Mobile Link 17* で、**Central Admin Tutorial** を展開し、リモートスキーマ名を右クリックし、 新規 リモートスキーマ名 をクリックします。

   リモートスキーマ名作成ウィザードが表示されます。
2. スキーマ名に **Tutorial Application v2.0** と入力し、完了をクリックします。
3. 新しいリモートタスクを作成します。SQL Central の フォルダビューの **Central Admin Tutorial** で、リモートタスクを右クリックし、 新規 リモートタスク をクリックします。リモートタスク作成ウィザードが表示されます。
4. ようこそページで、名前フィールドに **Schema Upgrade** と入力します。
5. タスクにリモートデータベースが必要、またはリモートデータベースを作成をオンにし、リモートスキーマ名を **Tutorial Application v1.0** に設定します。
6. このタスクは、管理対象リモートデータベースのスキーマを更新をオンにし、新しいリモートスキーマ名を **Tutorial Application v2.0** に設定します。完了をクリックします。

7. コマンドタブで、コマンドタイプドロップダウンリストから *SQL を実行*を選択します。*SQL* フィールドに次のように入力します。

```
CREATE TABLE  product (
    prod_id             integer primary key,
    name                varchar( 100 )
);
```

これで、スキーマの変更タスクが完了しました。

新しいスキーマ変更タスクを展開する前に、リモートデバイスにすでに割り当てられているすべてのタスクを考慮する必要があります。**Schema Upgrade** タスクが完了すると、データベースのリモートスキーマ名は **Tutorial Application v2.0** になります。古いリモートスキーマ名 **Tutorial Application v1.0** に関連付けられているリモートデバイス上のタスクは、実行しなくなり、エージェントによって破棄されます。これらのタスクに備えられている機能を保持するため、新しいバージョンのタスクを作成し、それを新しいリモートスキーマ名に関連付けてください。

8. フォルダビューの **Central Admin Tutorial** で、統合データベース > *Tutorial* > エージェント を展開し、**AID_JOHN** をクリックします。右ウィンドウ枠で、タスクタブを選択します。アクティブタスクのみがエージェントで実行されています。これらのタスクに対してのみ、新しいバージョンを作成する必要があります。この場合、アクティブタスクは **Sync every hour** タスクのみです。

タスクタブのリモートスキーマ名カラムをチェックすることで、このタスクが古いリモートスキーマ名に関連付けられているかどうかを判断できます。このタスクは、**Sync every hour** タスクのリモートスキーマ名が **Tutorial Application v1.0** であるため、古いリモートスキーマ名に関連付けられていることを示しています。スキーマの変更後に引き続き同期を行うには、このタスクの新しいバージョンを作成してエージェントに割り当てる必要があります。

9. **Sync every hour** タスクを右クリックし、タスクに移動を選択します。
10. 展開済みタスク **Sync every hour** を右クリックし、コピーを選択します。
11. リモートタスクを右クリックし、貼り付けをクリックします。コピーしたタスクの名前を尋ねられたら **Sync every hour v2** と入力し、*OK* をクリックします。
12. 新しいスキーマに対する作業を続行するために、タスク内のコマンドを変更することが必要かどうかを検討します。この場合、答えはいいえです。これは、コマンドは 1 つのみであり、このコマンドはこのスキーマ変更で修正を加えていない **tutorial1_JOHN** 同期プロファイルのみに依存するためです。
13. 新しいリモートスキーマ名に関連付けられたタスクであるとマーク付けします。**Sync every hour v2** タスクを右クリックし、プロパティを選択します。プロパティウィンドウの一般ページで、リモートスキーマ名に **Tutorial Application v2.0** を選択し、*OK* をクリックします。
14. 新しいタスクを展開するには、**Sync every hour v2** タスクを右クリックし、展開をクリックします。次へをクリックします。
15. 受信者で特定のエージェントをクリックしてから、エージェント **AID_JOHN** を選択します。次へをクリックしてから、完了をクリックします。
16. **Schema Upgrade** タスクを右クリックし、展開をクリックします。次へをクリックします。
17. 受信者ドロップダウンリストから特定のエージェントをクリックし、タスクをエージェント **AID_JOHN** に割り当てます。次へをクリックしてから、完了をクリックします。

結果

Schema Upgrade タスクが正常に実行されたことを確認してください。その後は、**Sync every hour v2** タスクが 1 時間ごとに実行し、**Sync every hour** タスクは実行を停止します。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 15: 即時同期の強制 [276 ページ]

次のタスク: レッスン 17: リモートデータベースの問い合わせ [279 ページ]

1.5.9.17 レッスン 17: リモートデータベースの問い合わせ

このレッスンでは、リモートデータベースに対してクエリを行い、結果をサーバに返します。リモートデータベースの状態を正確に把握できるため、トラブルシューティングに非常に便利です。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限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データベースに追加したテーブルにはデータが含まれていないため、代わりにデータベースシステムテーブルに対して問い合わせます。この例ではシステムテーブルに対して問い合わせますが、ユーザテーブルに対する動作とまったく同じです。

前のレッスンで実行したスキーマの変更により、期待する操作 (正しいカラムを持つ product テーブルを作成すること) が行われたかどうかを確認するとします。このことは、systable と systabcol の各システムテーブルに対してクエリを実行することによって確認できます。

手順

1. SQL Central のフォルダビューの *Mobile Link 17* で、*Central Admin Tutorial* を展開し、リモートタスクを右クリックして 新規 > リモートタスク をクリックします。

   リモートタスク作成ウィザードが表示されます。
2. ようこそページで、名前フィールドに `Table Query` と入力します。

3. タスクにリモートデータベースが必要、またはリモートデータベースを作成をオンにし、リモートスキーマ名を **Tutorial Application v2.0** に設定して次へをクリックします。
4. トリガのメカニズムページで、エージェントで受信されたときをオンにし、完了をクリックします。
5. 次の SQL を使用して、*SQL を実行*コマンドをタスクに追加します。

```
SELECT * FROM SYS.SYSTAB WHERE table_name = 'product'
go
SELECT * FROM SYS.SYSTABCOL ORDER BY table_id
```

6. 新しい **Table Query** タスクを右クリックし、展開をクリックします。次へをクリックします。
7. 受信者に特定のエージェントを選択し、エージェント **AID_JOHN** を選択して次へをクリックしてから、もう一度次へをクリックします。
8. 結果とステータスのレポートページで、タスクが成功した場合とタスクが失敗した場合をどちらもすぐに結果とステータスを送信に設定します。完了をクリックし、タスクが実行するまで待ちます。
9. フォルダビューで **Table Query** タスクの展開済みコピーをクリックし、結果タブをクリックします。タブに結果が何も表示されない場合は、タブが自動的に再表示されるまで待つか、F5 キーを押してすぐに再表示します。
10. テーブルで *SQL を実行*文の行を右クリックし、詳細を選択します。

    コマンドの結果ウィンドウが表示されます。
11. ウィンドウで結果タブをクリックします。このタブには、実行されたクエリの結果が表示されます。ウィンドウ枠上部の結果ドロップダウンを使用すると、2 つのクエリの結果を切り替えることができます。閉じるをクリックします。

結果

リモートデータベースからの結果が表示されます。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 16: リモートスキーマの変更 [277 ページ]

次のタスク: レッスン 18: SIRT を使用したファイルのアップロード [281 ページ]

1.5.9.18 レッスン 18: SIRT を使用したファイルのアップロード

このレッスンでは、サーバ起動のリモートタスク (SIRT) を使用して、リモートデバイスからファイルをアップロードします。ファイルに問題があるかどうかを管理者が検証できるため、リモートデバイスからのファイルのアップロードはトラブルシューティングに便利です。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

Mobile Link エージェントをリモートデバイスで起動したとき、ファイル `agent.txt` にメッセージを記録するよう指示しました。ここでは、そのファイルをリモートデバイスから取得して、検証します。

手順

1. SQL Central のフォルダビューの *Mobile Link 17* で、*Central Admin Tutorial* を展開し、リモートタスクを右クリックして 新規 > リモートタスク をクリックします。

   リモートタスク作成ウィザードが表示されます。
2. ようこそページで、名前フィールドに **Upload Agent Log** と入力します。
3. タスクにリモートデータベースが必要、またはリモートデータベースを作成がオンになっている場合はクリアし、完了をクリックします。
4. フォルダビューで新しいタスクをクリックし、タスクにコマンドを追加します。コマンドタイプをファイルをアップロードに設定します。
5. サーバファイル名を `{agent_id}¥agent.txt` に設定し、リモートファイル名を `{agent_log}` に設定します。コマンドエディタで省略記号 (ピリオド 3 つ) を使用すると、マクロの値を簡単に入力できます。

   `{agent_log}` マクロは、Mobile Link エージェントがリモートデバイスで保持しているログファイルの名前に置き換えられます。

   サーバファイル名フィールドには、`{agent_id}` マクロを使用してファイルが置かれるディレクトリを指定しました。このことは非常に重要です。マクロを使用しないでサーバファイル名を指定すると、タスクを実行するすべてのエージェントによって、アップロードファイルが同じ場所に配置されます。このとき、新しいエージェントは、前のエージェントによって書き込まれたファイルを毎回上書きします。マクロを使用すると、各エージェントはログファイルをサーバ上の異なる場所にアップロードするため、すべてのエージェントログファイルを表示できるようになります。
6. 新しい **Upload Agent Log** タスクを右クリックし、展開をクリックします。次へをクリックします。
7. 受信者で特定のエージェントをクリックしてから、エージェント **AID_JOHN** を選択します。次へをクリックします。

8. 配信オプションページで、エージェントの次回の同期時をクリックし、次へをクリックします。
9. 結果とステータスのレポートページで、タスクが成功した場合とタスクが失敗した場合をどちらもすぐに結果とステータスを送信に設定します。完了をクリックします。
10. 管理者は、SQL Central でタスクを開始する必要があります。タスクを開始するには、エージェントの AID_JOHN に移動します。ウィンドウ枠で、タスクタブを選択し、*Upload Agent Log* タスクを右クリックし、開始をクリックします。タスクが実行するのを待ちます。

結果

アップロードしたファイルは、Mobile Link コマンドラインで -ftru オプションを使用して指定された Mobile Link アップロードディレクトリに配置されます。アップロードディレクトリには `c:¥cadmin_demo¥consolidated¥upload` が指定されています。コマンドプロンプトまたは Windows エクスプローラーを使用して、このディレクトリを見てみます。`AID_JOHN` サブフォルダがあります。このサブフォルダに、アップロードした `agent.txt` ファイルがあります。

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: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 17: リモートデータベースの問い合わせ [279 ページ]

次のタスク: レッスン 19: クリーンアップ [282 ページ]

1.5.9.19 レッスン 19: クリーンアップ

すべてのチュートリアルをコンピュータから削除します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. タスクバー上で、Interactive SQL、SQL Central、Mobile Link、同期クライアントの各ウィンドウを右クリックし、[閉じる]を選択して閉じます。
2. 次の手順で、チュートリアルに関連するすべてのデータソースを削除します。
   a. ODBC データソースアドミニストレータを起動します。
   b. **スタート** > **プログラム** > *SQL Anywhere17* > **管理ツール** > *ODBC* **データソースアドミニストレータ** をクリックします。
   c. **ユーザデータソース**リストから *cadmin_tutorial_consol* を選択し、**削除**をクリックします。

結果

チュートリアルリソースが削除されます。

タスクの概要: チュートリアル: リモートデータベースの集中管理の使用 [253 ページ]

前のタスク: レッスン 18: SIRT を使用したファイルのアップロード [281 ページ]

1.5.10 チュートリアル: スクリプトバージョン句を使用したスキーマの変更

このチュートリアルでは、dbmlsync ScriptVersion 拡張オプションが使用されていない同期に関係するリモートデータベースでスキーマの変更を実行する方法について説明します。

前提条件

このチュートリアルでは、チュートリアルを実行するローカルコンピュータに SQL Anywhere (Mobile Link を含む) が完全にインストールされていることを前提としています。

統合データベースで次のロールおよび権限を持つ必要があります。

- SYS_AUTH_RESOURCE_ROLE 互換ロール
- MONITOR システム権限

リモートデータベースで次のロールおよび権限を持つ必要があります。

- SYS_REPLICATION_ADMIN_ROLE システムロール
- SYS_RUN_REPLICATION_ROLE システムロール

コンテキスト

このチュートリアルでは、単一テーブルを同期する同期システムを設定し、スキーマを変更して同期対象テーブルにカラムを追加して、同期を続行します。

このチュートリアルでは、次の作業の方法について説明します。

- 統合データベースの作成と設定
- リモートデータベースの作成と設定
- リモートデータベースの同期
- リモートデータベースへのデータの挿入
- 統合データベースでのスキーマ変更の実行
- リモートデータベースでのスキーマ変更の実行
- リモートデータベースへのデータの挿入
- 同期

1. レッスン 1: 統合データベースの作成と設定 [285 ページ]
   このレッスンでは、同期用に統合データベースを設定します。
2. レッスン 2: リモートデータベースの作成と設定 [286 ページ]
   このレッスンでは、同期用にリモート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同期 [288 ページ]
   この時点では、同期システムの設定が動作しているはずです。このレッスンでは、リモートデータベースにデータを挿入して同期をテストします。
4. レッスン 4: リモートデータベースへのデータの挿入 [289 ページ]
   このレッスンでは、リモートデータベースにデータを挿入して、アップロードする必要があるリモートデータベースで操作が実行されている場合でも、スキーマの変更を処理できることを示します。
5. レッスン 5: 統合データベースでのスキーマ変更の実行 [290 ページ]
   このレッスンでは、統合データベースでスキーマを変更します。
6. レッスン 6: リモートデータベースでのスキーマ変更の実行 [291 ページ]
   このレッスンでは。リモートデータベースを変更して、新しいカラムを customer テーブルに追加し、同期に使用されるスクリプトバージョンを変更します。
7. レッスン 7: データの挿入 [292 ページ]
   このレッスンでは、新しいスキーマを使用して、リモートデータベースと統合データベースにさらにデータを挿入します。
8. レッスン 8: 同期 [293 ページ]
   このレッスンでは、スキーマの変更ともう一度同期します。

タスクの概要: Mobile Link チュートリアル [96 ページ]

前のタスク: チュートリアル: リモートデータベースの集中管理の使用 [253 ページ]

次のタスク: チュートリアル: ScriptVersion 拡張オプションを使用したスキーマの変更 [294 ページ]

1.5.10.1 レッスン 1: 統合データベースの作成と設定

このレッスンでは、同期用に統合データベースを設定します。

前提条件

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 次のコマンドを実行して、統合データベースを作成し、実行を開始します。

```
md c:¥cons
cd c:¥cons
dbinit -dba DBA,passwd consol.db
dbsrv17 consol.db
```

2. 次のコマンドを実行して、統合データベースの ODBC データソースを定義します。

```
dbdsn -w dsn_consol -y -c "UID=DBA;PWD=passwd;DBF=consol.db;SERVER=consol"
```

3. データベースを統合データベースとして使用するには、Mobile Link で使用するシステムテーブル、ビュー、ストアドプロシージャを追加する設定スクリプトを実行する必要があります。次のコマンドを実行して、統合データベースとして `consol.db` を設定します。

```
dbisql -c "DSN=dsn_consol" %SQLANY17%¥MobiLink¥setup¥syncsa.sql
```

4. Interactive SQL を開き、dsn_consol ODBC データソースを使用して `consol.db` に接続します。

```
dbisql -c "DSN=dsn_consol"
```

5. 次の SQL 文を実行します。統合データベースで customer テーブルが作成され、必要な同期スクリプトが作成されます。

```
CREATE TABLE customer (
   id      unsigned integer primary key,
   name    varchar( 256),
   phone   varchar( 12 )
);
CALL ml_add_column('my_ver1', 'customer', 'id', null );
CALL ml_add_column('my_ver1', 'customer', 'name', null );
CALL ml_add_column('my_ver1', 'customer', 'phone', null );
CALL ml_add_table_script( 'my_ver1', 'customer', 'upload_insert',
      'INSERT INTO customer ( id, name, phone ) '
      || 'VALUES ({ml r.id}, {ml r.name}, {ml r.phone} )' );
CALL ml_add_table_script( 'my_ver1', 'customer', 'download_cursor',
       'SELECT id, name, phone from customer' );
CALL ml_add_table_script( 'my_ver1', 'customer', 'download_delete_cursor', '--
{ml_ignore}' );
COMMIT;
```

チュートリアルに従って作業する間に、データベースに対してさらに SQL を実行するため、この SQL の実行完了後も、引き続き Interactive SQL を実行し、データベースに接続した状態にします。

6. 次のコマンドを実行して、Mobile Link サーバを起動します。

```
start mlsrv17 -c "DSN=dsn_consol" -v+ -ot mlsrv.txt -zu+
```

結果

統合データベースが作成され、Mobile Link で使用でき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: スクリプトバージョン句を使用したスキーマの変更 [283 ページ]

次のタスク: レッスン 2: リモートデータベースの作成と設定 [286 ページ]

1.5.10.2 レッスン 2: リモートデータベースの作成と設定

このレッスンでは、同期用にリモートデータベースを設定します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

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作成し、実行を開始します。

```
cd..
md c:¥remote
cd c:¥remote
dbinit -dba DBA,passwd remote.db
dbsrv17 remote.db
```

2. Interactive SQL の別のインスタンスを開き、`remote.db` に接続します。

```
dbisql -c "SERVER=remote;DBF=remote.db;UID=DBA;PWD=passwd"
```

3. Interactive SQL で次の SQL 文を実行し、同期させるテーブルを作成します。

```
CREATE TABLE customer (
   id       UNSIGNED INTEGER PRIMARY KEY,
   name     VARCHAR( 256),
   phone    VARCHAR( 12 )
);
```

4. リモートデータベースに接続された Interactive SQL インスタンスを引き続き使用して、パブリケーション、Mobile Link ユーザ、サブスクリプションを作成します。スクリプトバージョンは、SCRIPT VERSION 句を使用したサブスクリプションに関連付けられています。このチュートリアルで示すスキーマのアップグレード手順は、SCRIPT VERSION 句を使用して設定されたスクリプトバージョンがあるサブスクリプションでのみ動作するため、このことは非常に重要です。

```
CREATE PUBLICATION p1 (
      TABLE customer
);
CREATE SYNCHRONIZATION USER u1;
CREATE SYNCHRONIZATION SUBSCRIPTION my_sub
TO p1
FOR u1
SCRIPT VERSION 'my_ver1';
```

   チュートリアルに従って作業する間に、データベースに対してさらに SQL を実行するため、この SQL の実行完了後も、引き続き Interactive SQL を実行し、データベースに接続した状態にします。

結果

リモートデータベースが作成され、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: スクリプトバージョン句を使用したスキーマの変更 [283 ページ]

前のタスク: レッスン 1: 統合データベースの作成と設定 [285 ページ]

次のタスク: レッスン 3: リモートデータベースの同期 [288 ページ]

1.5.10.3 レッスン 3: リモートデータベースの同期

この時点では、同期システムの設定が動作しているはずです。このレッスンでは、リモートデータベースにデータを挿入して同期をテストします。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 統合データベースに接続された Interactive SQL のインスタンスを使用し、次の SQL 文を実行して customer テーブルにローを挿入します。

   ```
   INSERT INTO customer VALUES( 100, 'John Jones', '519-555-1234' );
   COMMIT;
   ```

2. リモートデータベースに接続された Interactive SQL のインスタンスを使用し、次の SQL 文を実行して customer テーブルにローを挿入します。

   ```
   INSERT INTO customer VALUES( 1, 'Willie Lowman', '705-411-6372' );
   COMMIT;
   ```

3. 次のコマンドを実行して同期します。

   ```
   dbmlsync -v+ -ot sync1.txt -c UID=DBA;PWD=passwd;SERVER=remote -s my_sub -k
   ```

結果

リモートデータベースが同期されます。

customer テーブルの内容と統合データベースを比較することによって、同期が成功したことを確認できます。dbmlsync ログの `sync1.txt` を調べて、エラーがないかを確認することもできます。

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: スクリプトバージョン句を使用したスキーマの変更 [283 ページ]

前のタスク: レッスン 2: リモートデータベースの作成と設定 [286 ページ]

次のタスク: レッスン 4: リモートデータベースへのデータの挿入 [289 ページ]

1.5.10.4 レッスン 4: リモートデータベースへのデータの挿入

このレッスンでは、リモートデータベースにデータを挿入して、アップロードする必要があるリモートデータベースで操作が実行されている場合でも、スキーマの変更を処理できることを示します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

リモートデータベースに接続された Interactive SQL のインスタンスを使用し、次の SQL 文を実行して customer テーブルにローを挿入します。

```
INSERT INTO customer VALUES( 2, 'Sue Slow', '602-411-5467' );
COMMIT;
```

結果

データがリモートデータベースに挿入されます。

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: スクリプトバージョン句を使用したスキーマの変更 [283 ページ]

前のタスク: レッスン 3: リモートデータベースの同期 [288 ページ]

次のタスク: レッスン 5: 統合データベースでのスキーマ変更の実行 [290 ページ]

1.5.10.5 レッスン 5: 統合データベースでのスキーマ変更の実行

このレッスンでは、統合データベースでスキーマを変更します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 新しいカラムを customer テーブルに追加して、顧客の携帯電話番号を保存します。まず、統合データベースに接続されている Interactive SQL のインスタンスで次の SQL 文を実行して、統合データベースに新しいカラムを追加します。

   ```
   ALTER TABLE customer ADD cell_phone VARCHAR(12) DEFAULT NULL;
   ```

2. **my_ver2** という新しいスクリプトバージョンを作成して、リモートデータベースから新しいスキーマで同期を処理します。古いスキーマを使用するリモートデータベースでは、古いスクリプトバージョン **my_ver1** が引き続き使用されます。統合データベースで次の SQL 文を実行します。

   ```
   CALL ml_add_column('my_ver2', 'customer', 'id', null );
   CALL ml_add_column('my_ver2', 'customer', 'name', null );
   CALL ml_add_column('my_ver2', 'customer', 'phone', null );
   CALL ml_add_column('my_ver2', 'customer', 'cell_phone', null );
   CALL ml_add_table_script( 'my_ver2', 'customer', 'upload_insert',
         'INSERT INTO customer ( id, name, phone, cell_phone ) '
         || 'VALUES ({ml r.id}, {ml r.name}, {ml r.phone}, {ml r.cell_phone})' );
   CALL ml_add_table_script( 'my_ver2', 'customer', 'download_cursor',
          'SELECT id, name, phone, cell_phone from customer' );
   CALL ml_add_table_script( 'my_ver2', 'customer', 'download_delete_cursor', '--
   {ml_ignore}' );
   COMMIT;
   ```

結果

統合データベースのスキーマが更新されます。

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: スクリプトバージョン句を使用したスキーマの変更 [283 ページ]

前のタスク: レッスン 4: リモートデータベースへのデータの挿入 [289 ページ]

次のタスク: レッスン 6: リモートデータベースでのスキーマ変更の実行 [291 ページ]

1.5.10.6 レッスン 6: リモートデータベースでのスキーマ変更の実行

このレッスンでは。リモートデータベースを変更して、新しいカラムを customer テーブルに追加し、同期に使用されるスクリプトバージョンを変更します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 同期スキーマ変更を開始します。これは、同期対象テーブルに影響を及ぼすほとんどのスキーマ変更で必要です。この文によって、サブスクリプションの同期に使用されるスクリプトバージョンが変更され、スキーマの変更を安全に実行できるように、影響を受けるテーブルがロックされます。

   リモートデータベースに接続されている Interactive SQL のインスタンスを使用して、リモートデータベースで次の SQL 文を実行します。

   ```
   START SYNCHRONIZATION SCHEMA CHANGE
   FOR TABLES customer
   SET SCRIPT VERSION = 'my_ver2';
   ```

2. 次の SQL 文を実行して、新しいカラムを customer テーブルに追加します。

   ```
   ALTER TABLE customer ADD cell_phone VARCHAR(12) DEFAULT NULL;
   ```

3. スキーマの変更を閉じます。これにより、テーブルのロックが解除されます。

   ```
   STOP SYNCHRONIZATION SCHEMA CHANGE;
   ```

結果

リモートデータベースで同期スキーマの変更が実行されます。

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: スクリプトバージョン句を使用したスキーマの変更 [283 ページ]

前のタスク: レッスン 5: 統合データベースでのスキーマ変更の実行 [290 ページ]

次のタスク: レッスン 7: データの挿入 [292 ページ]

1.5.10.7 レッスン 7: データの挿入

このレッスンでは、新しいスキーマを使用して、リモートデータベースと統合データベースにさらにデータを挿入します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. Interactive SQL を使用して、リモートデータベースで次の SQL 文を実行します。

```
INSERT INTO customer VALUES( 3, 'Mo Hamid', '613-411-9999', '613-502-1212' );
COMMIT;
```

2. Interactive SQL を使用して、統合データベースで次の SQL 文を実行します。

```
INSERT INTO customer VALUES( 101, 'Theo Tug', '212-911-7677', '212-311-3900' );
COMMIT;
```

結果

新しいスキーマを使用してリモートデータベースと統合データベースにデータが挿入されます。

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: スクリプトバージョン句を使用したスキーマの変更 [283 ページ]

前のタスク: レッスン 6: リモートデータベースでのスキーマ変更の実行 [291 ページ]

次のタスク: レッスン 8: 同期 [293 ページ]

1.5.10.8 レッスン 8: 同期

このレッスンでは、スキーマの変更ともう一度同期します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

次のコマンドを実行して、もう一度同期します。

```
dbmlsync -v+ -ot sync2.txt -c UID=DBA;PWD=passwd;SERVER=remote -s my_sub -k
```

スキーマの変更前に挿入されたロー **Sue Slow** が、スクリプトバージョン **my_ver1** を使用してアップロードされます。スキーマの変更後に挿入されたロー **Mo Hamid** が、スクリプトバージョン **my_ver2** を使用してアップロードされます。**my_ver2** のダウンロードカーソルを使用して、ローがダウンロードされます。

結果

これで、スキーマの変更が完了し、正常に同期を続行できます。

タスクの概要: チュートリアル: スクリプトバージョン句を使用したスキーマの変更 [283 ページ]

前のタスク: レッスン 7: データの挿入 [292 ページ]

1.5.11 チュートリアル: ScriptVersion 拡張オプションを使用したスキーマの変更

このチュートリアルでは、ScriptVersion 拡張オプションを使用している場合にスキーマを変更する方法について説明します。

前提条件

このチュートリアルでは、チュートリアルを実行するローカルコンピュータに SQL Anywhere (Mobile Link を含む) が完全にインストールされていることを前提としています。

統合データベースで次のロールおよび権限を持つ必要があります。

- SYS_AUTH_RESOURCE_ROLE 互換ロール
- MONITOR システム権限

リモートデータベースで次のロールおよび権限を持つ必要があります。

- SYS_REPLICATION_ADMIN_ROLE システムロール
- SYS_RUN_REPLICATION_ROLE システムロール

コンテキスト

このチュートリアルでは、次の作業の方法について説明します。

- 統合データベースの作成と設定
- リモートデータベースの作成と設定
- リモートデータベースの同期
- 統合データベースでのスキーマ変更の実行
- リモートデータベースでのスキーマ変更の実行

> **注記**
>
> できる限り、ScriptVersion 拡張オプションは使用しないでください。代わりに、CREATE SYNCHRONIZATION SUBSCRIPTION 文の SCRIPT VERSION 句または ALTER SYNCHRONIZATION SUBSCRIPTION 文の SET SCRIPT VERSION 句を使用して、スクリプトバージョンをサブスクリプションに関連付けます。これらを実装すると、スキーマを柔軟にアップグレードできるようになります。

1. レッスン 1: 統合データベースの作成と設定 [295 ページ]
   このレッスンでは、同期用に統合データベースを設定します。
2. レッスン 2: リモートデータベースの作成と設定 [297 ページ]
   このレッスンでは、同期用にリモート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同期 [298 ページ]
   この時点では、同期システムの設定が動作しているはずです。このレッスンでは、リモートデータベースにデータを挿入して同期をテストします。
4. レッスン 4: 統合データベースでのスキーマ変更の実行 [299 ページ]
   このレッスンでは、新しいカラムを customer テーブルに追加して、顧客の携帯電話番号を保存します。
5. レッスン 5: リモートデータベースでのスキーマ変更の実行 [300 ページ]
   このレッスンでは。リモートデータベースを変更して、新しいカラムを customer テーブルに追加し、同期に使用されるスクリプトバージョンを変更します。

タスクの概要: Mobile Link チュートリアル [96 ページ]

前のタスク: チュートリアル: スクリプトバージョン句を使用したスキーマの変更 [283 ページ]

次のタスク: チュートリアル: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート [302 ページ]

1.5.11.1 レッスン 1: 統合データベースの作成と設定

このレッスンでは、同期用に統合データベースを設定します。

前提条件

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 次のコマンドを実行して、統合データベースを作成して起動します。

```
md c:¥cons
cd c:¥cons
dbinit -dba DBA,passwd consol.db
dbsrv17 consol.db
```

2. 次のコマンドを実行して、統合データベースの ODBC データソースを定義します。

```
dbdsn -w dsn_consol -y -c "UID=DBA;PWD=passwd;DBF=consol.db;SERVER=consol"
```

3. データベースを統合データベースとして使用するには、Mobile Link で使用するシステムテーブル、ビュー、ストアドプロシージャを追加する設定スクリプトを実行する必要があります。次のコマンドを実行して、統合データベースとして consol.db を設定します。

```
dbisql -c "DSN=dsn_consol" %SQLANY17%¥MobiLink¥setup¥syncsa.sql
```

4. Interactive SQL を開き、**dsn_consol** DSN を使用して consol.db に接続します。

```
dbisql -c "DSN=dsn_consol"
```

5. Interactive SQL で次の SQL 文を実行します。統合データベースで customer テーブルが作成され、必要な同期スクリプトが作成されます。

```
CREATE TABLE customer (
   id      unsigned integer primary key,
   name    varchar( 256),
   phone   varchar( 12 )
);
CALL ml_add_column('my_ver1', 'customer', 'id', null );
CALL ml_add_column('my_ver1', 'customer', 'name', null );
CALL ml_add_column('my_ver1', 'customer', 'phone', null );
CALL ml_add_table_script( 'my_ver1', 'customer', 'upload_insert',
      'INSERT INTO customer ( id, name, phone ) '
      || 'VALUES ({ml r.id}, {ml r.name}, {ml r.phone} )' );
CALL ml_add_table_script( 'my_ver1', 'customer', 'download_cursor',
       'SELECT id, name, phone from customer' );
CALL ml_add_table_script( 'my_ver1', 'customer', 'download_delete_cursor', '--
{ml_ignore}' );
COMMIT;
```

チュートリアルに従って作業する間に、データベースに対してさらに SQL を実行するため、この SQL 文の実行完了後も、引き続き Interactive SQL を実行し、データベースに接続した状態にします。

6. 次のコマンドを実行して、Mobile Link サーバを起動します。

```
start mlsrv17 -c "DSN=dsn_consol" -v+ -ot mlsrv.txt -zu+
```

結果

統合データベースが作成され、同期用に設定されます。

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: ScriptVersion 拡張オプションを使用したスキーマの変更 [294 ページ]

次のタスク: レッスン 2: リモートデータベースの作成と設定 [297 ページ]

1.5.11.2 レッスン 2: リモートデータベースの作成と設定

このレッスンでは、同期用にリモートデータベースを設定します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

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作成して起動します。

```
cd..
md c:¥remote
cd c:¥remote
dbinit -dba DBA,passwd remote.db
dbeng17 remote.db
```

2. Interactive SQL の別のインスタンスを開き、`remote.db` に接続します。

```
dbisql -c "SERVER=remote;DBF=remote.db;UID=DBA;PWD=passwd"
```

3. Interactive SQL で次の SQL 文を実行し、同期させるテーブルを作成します。

```
CREATE TABLE customer (
   id      unsigned integer primary key,
   name    varchar( 256),
   phone   varchar( 12 )
);
```

4. パブリケーション、Mobile Link ユーザ、サブスクリプションを作成します。

```
CREATE PUBLICATION p1 (
   TABLE customer
);
CREATE SYNCHRONIZATION USER u1;
CREATE SYNCHRONIZATION SUBSCRIPTION my_sub
TO p1
FOR u1
OPTION ScriptVersion='my_ver1';
```

   チュートリアルに従って作業する間に、データベースでさらに SQL 文を実行するため、この SQL 文の実行完了後も、引き続き Interactive SQL を実行し、データベースに接続した状態にします。

結果

リモートデータベースが作成され、同期用に設定されます。

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: ScriptVersion 拡張オプションを使用したスキーマの変更 [294 ページ]

前のタスク: レッスン 1: 統合データベースの作成と設定 [295 ページ]

次のタスク: レッスン 3: リモートデータベースの同期 [298 ページ]

1.5.11.3 レッスン 3: リモートデータベースの同期

この時点では、同期システムの設定が動作しているはずです。このレッスンでは、リモートデータベースにデータを挿入して同期をテストします。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 統合データベースに接続された Interactive SQL のインスタンスを使用し、次の SQL 文を実行して customer テーブルにローを挿入します。

   ```
   INSERT INTO customer VALUES( 100, 'John Jones', '519-555-1234' );
   COMMIT;
   ```

2. リモートデータベースに接続された Interactive SQL のインスタンスを使用し、次の SQL 文を実行して customer テーブルにローを挿入します。

   ```
   INSERT INTO customer VALUES( 1, 'Willie Lowman', '705-411-6372' );
   COMMIT;
   ```

3. 次のコマンドを実行して同期します。

   ```
   dbmlsync -v+ -ot sync1.txt -c UID=DBA;PWD=passwd;SERVER=remote -s my_sub -k
   ```

   customer テーブルの内容と統合データベースを比較することによって、同期が成功したことを確認できます。dbmlsync ログの `sync1.txt` を調べて、エラーがないかを確認することもできます。

結果

統合データベースとリモートデータベースが同期されます。

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: ScriptVersion 拡張オプションを使用したスキーマの変更 [294 ページ]

前のタスク: レッスン 2: リモートデータベースの作成と設定 [297 ページ]

次のタスク: レッスン 4: 統合データベースでのスキーマ変更の実行 [299 ページ]

1.5.11.4 レッスン 4: 統合データベースでのスキーマ変更の実行

このレッスンでは、新しいカラムを customer テーブルに追加して、顧客の携帯電話番号を保存します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 統合データベースに接続された Interactive SQL のインスタンスを使用し、次の SQL 文を実行して customer テーブルにローを挿入します。

   ```
   ALTER TABLE customer ADD cell_phone VARCHAR(12) DEFAULT NULL;
   ```

2. my_ver2 という新しいスクリプトバージョンを作成して、リモートデータベースから新しいスキーマで同期を処理します。古いスキーマを使用するリモートデータベースでは、古いスクリプトバージョン my_ver1 が引き続き使用されます。

   統合データベースで次の SQL 文を実行します。

   ```
   CALL ml_add_column('my_ver2', 'customer', 'id', null );
   CALL ml_add_column('my_ver2', 'customer', 'name', null );
   CALL ml_add_column('my_ver2', 'customer', 'phone', null );
   CALL ml_add_column('my_ver2', 'customer', 'cell_phone', null );
   CALL ml_add_table_script( 'my_ver2', 'customer', 'upload_insert',
   ```

```
      'INSERT INTO customer ( id, name, phone, cell_phone ) '
      || 'VALUES ({ml r.id}, {ml r.name}, {ml r.phone}, {ml r.cell_phone})' );
CALL ml_add_table_script( 'my_ver2', 'customer', 'download_cursor',
       'SELECT id, name, phone, cell_phone from customer' );
CALL ml_add_table_script( 'my_ver2', 'customer', 'download_delete_cursor', '--
{ml_ignore}' );
COMMIT;
```

結果

統合データベースに変更が加えられ、スキーマの変更を処理する新しいスクリプトバージョンが作成されます。

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: ScriptVersion 拡張オプションを使用したスキーマの変更 [294 ページ]

前のタスク: レッスン 3: リモートデータベースの同期 [298 ページ]

次のタスク: レッスン 5: リモートデータベースでのスキーマ変更の実行 [300 ページ]

1.5.11.5 レッスン 5: リモートデータベースでのスキーマ変更の実行

このレッスンでは。リモートデータベースを変更して、新しいカラムを customer テーブルに追加し、同期に使用されるスクリプトバージョンを変更します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

リモートデータベースを変更し同期用のスクリプトバージョンを作成する前に、アップロードを必要とする customer テーブルに対する操作がないことを確認してください。このための最適な方法として、sp_hook_dbmlsync_schema_upgrade フックでスキーマの変更を実行することが挙げられます。このフックを使用すると、同期の開始時点で同期対象テーブルがロックされ、スキーマの変更が完了するまでロック状態が保持されるため、スキーマの変更が安全に実行されたことを dbmlsync で確認できます。

コンテキスト

> **警告**
>
> アップロードする操作があるときにスキーマを変更すると、スキーマ変更後にリモートデータベースで常に同期できなくなります。

手順

1. リモートデータベースで次の SQL 文を実行して、sp_hook_dbmlsync_schema_upgrade フックを作成します。このフックにより、新しいカラムが customer テーブルに追加され、サブスクリプションで格納された ScriptVersion 拡張オプションの値が変更されます。このフックは、実行後に dbmlsync によって削除されます。

   ```
   CREATE PROCEDURE sp_hook_dbmlsync_schema_upgrade()
   BEGIN
       ALTER TABLE customer
       ADD cell_phone varchar(12) default null;
       ALTER SYNCHRONIZATION SUBSCRIPTION my_sub
       ALTER OPTION ScriptVersion='my_ver2';
       UPDATE #hook_dict
       SET value = 'always'
           WHERE name = 'drop hook';
   END;
   ```

2. sp_hook_dbmlsync_schema_change フックを実行して、アップロードを必要とする操作のアップロードを同期し、スキーマを変更します。次のコマンドを実行します。

   ```
   dbmlsync -v+ -ot sync2.txt -c UID=DBA;PWD=passwd;SERVER=remote -s my_sub -k
   ```

   この同期が完了したら、dbmlsync ログ `sync2.txt` を調べて、スキーマの変更が完了しなかったことを示すエラーがないことを確認してください。

結果

これで、スキーマの変更が完了し、正常に同期を続行できます。

タスクの概要: チュートリアル: ScriptVersion 拡張オプションを使用したスキーマの変更 [294 ページ]

前のタスク: レッスン 4: 統合データベースでのスキーマ変更の実行 [299 ページ]

1.5.12 チュートリアル: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート

このチュートリアルでは、mlreplay ユーティリティを使って複数の Mobile Link クライアントを単一のコンピュータ上でシミュレートする方法について説明します。

前提条件

次の知識と経験が必要です。

- Mobile Link イベントスクリプトの基本的な知識

次のソフトウェアが必要です。

- SQL Anywhere17

統合データベースで次のロールおよび権限を持つ必要があります。

- SYS_AUTH_RESOURCE_ROLE 互換ロール
- MONITOR システム権限

リモートデータベースで次のロールおよび権限を持つ必要があります。

- SYS_REPLICATION_ADMIN_ROLE システムロール
- SYS_RUN_REPLICATION_ROLE システムロール

コンテキスト

このチュートリアルでは、次の作業の方法について説明します。

- Mobile Link 統合データベースを設定
- Mobile Link サーバの起動による同期の記録とリプレイ
- mlreplay ユーティリティを使用した Mobile Link クライアントのシミュレート

1. レッスン 1: Mobile Link 統合データベースの設定 [303 ページ]
   このレッスンでは、Mobile Link 統合データベースを設定します。
2. レッスン 2: Mobile Link プロジェクトの作成 [305 ページ]
   このレッスンでは、新しい Mobile Link プロジェクトを作成して、統合データベースに接続します。
3. レッスン 3: 同期スクリプトの追加 [306 ページ]
   スクリプトは、それぞれ指定のスクリプトバージョンに属しています。スクリプトは、統合データベースにスクリプトバージョンを追加した後に追加してください。
4. レッスン 4: 記録のための Mobile Link サーバの起動 [308 ページ]
   このレッスンでは、-c オプションを使って Mobile Link サーバ (mlsrv17) を起動し、統合データベースに接続します。
5. レッスン 5: Mobile Link クライアントデータベースの設定 [309 ページ]

Mobile Link は、統合データベースサーバと多数のモバイルデータベースとの間で同期を実行できるように設計されています。このレッスンでは、リモートデータベースを作成し、`T1` テーブル (このテーブルは統合データベースと同期する) を作成し、同期パブリケーション、ユーザ、サブスクリプションを作成します。

6. レッスン 6: 同期の記録 [311 ページ]
   このレッスンでは、dbmlsync ユーティリティを実行して SQL Anywhere リモートデータベース用の Mobile Link 同期を開始します。
7. レッスン 7: リプレイのための Mobile Link サーバの再起動 [313 ページ]
   このレッスンでは、記録を停止するために Mobile Link サーバを停止してから、-rp オプションを指定せずにサーバを再起動して、サーバでリプレイの準備をします。
8. レッスン 8: 同期のリプレイ [314 ページ]
   このレッスンでは、スキーマが Mobile Link サーバにキャッシュされるように同期を実行します。シミュレートされたクライアント情報ファイルを作成し、シミュレートされたクライアントで Mobile Link プロトコル情報をリプレイします。
9. レッスン 9: クリーンアップ [317 ページ]
   チュートリアルをコンピュータから削除します。

タスクの概要: Mobile Link チュートリアル [96 ページ]

前のタスク: チュートリアル: ScriptVersion 拡張オプションを使用したスキーマの変更 [294 ページ]

1.5.12.1 レッスン 1: Mobile Link 統合データベースの設定

このレッスンでは、Mobile Link 統合データベースを設定します。

前提条件

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 新しい作業ディレクトリを作成して、このチュートリアルで作成するすべてのサンプルファイルを格納します。このチュートリアルでは、パスを `c:¥mlreplay` とします。
2. コマンドプロンプトで、作業ディレクトリを `c:¥mlreplay` に変更します。このチュートリアルでは、すべてのコマンドがこのディレクトリから実行されることを前提とします。
3. 次のコマンドを実行して、`cons.db` という名前の SQL Anywhere 統合データベースを作成します。

   ```
   dbinit -dba DBA,passwd cons.db
   ```
4. 次のコマンドを入力して、統合データベースを起動します。

   ```
   dbsrv17 cons.db
   ```

5. スタート > プログラム > *SQL Anywhere17* > 管理ツール > *ODBC* データソースアドミニストレータをクリックします。
6. ユーザ *DSN* タブをクリックし、追加をクリックします。
7. データソースの新規作成ウィンドウで、*SQL Anywhere17* をクリックし、完了をクリックします。
8. *SQL Anywhere* の *ODBC* 設定ウィンドウで、次の操作を行います。
   a. *ODBC* タブをクリックします。
   b. データソース名フィールドに **cons** と入力します。
   c. ログインタブをクリックします。
   d. ユーザ *ID* フィールドに、**DBA** と入力します。
   e. パスワードフィールドに、**passwd** と入力します。
   f. アクションドロップダウンリストから、このコンピュータで稼働しているデータベースに接続を選択します。
   g. サーバ名フィールドに **cons** と入力します。
   h. データベース名フィールドに、**cons** と入力します。
   i. *OK* をクリックします。
9. ODBC データソースアドミニストレータを閉じます。

   ODBC データソースアドミニストレータウィンドウで *OK* をクリックします。
10. Interactive SQL の統合データベースに接続します。

    次のコマンドを実行します。

    ```
    dbisql -c "DSN=cons"
    ```
11. Interactive SQL で次の文を実行して、`syncsa.sql` 設定スクリプトを使用して Mobile Link のシステムテーブルとストアドプロシージャを作成します。*C:¥Program Files¥SQL Anywhere 17¥* は、SQL Anywhere17 インストール環境のロケーションに置き換えてください。

    ```
    READ "C:¥Program Files¥SQL Anywhere 17¥MobiLink¥setup¥syncsa.sql";
    ```

    Interactive SQL によって `syncsa.sql` が統合データベースに適用されます。`syncsa.sql` を実行すると、前に **ml_** が付いた一連のシステムテーブルとストアドプロシージャが作成されます。これらのテーブルとストアドプロシージャは、同期処理中に Mobile Link サーバによって使用されます。
12. Interactive SQL で次の SQL 文を実行し、**T1** テーブルを作成します。

    ```
    CREATE TABLE T1 (
        pk1     INTEGER,
        pk2     INTEGER,
        c1      VARCHAR(30000),
        PRIMARY KEY(pk1, pk2)
    );
    ```

    Interactive SQL によって、統合データベースに **T1** テーブルが作成されます。
13. Interactive SQL を閉じます。

結果

統合データベースが設定されます。

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: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート [302 ページ]

次のタスク: レッスン 2: Mobile Link プロジェクトの作成 [305 ページ]

1.5.12.2 レッスン 2: Mobile Link プロジェクトの作成

このレッスンでは、新しい Mobile Link プロジェクトを作成して、統合データベースに接続します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. スタート > プログラム > *SQL Anywhere17* > 管理ツール > *SQL Central* をクリックします。
2. ツール > *Mobile Link 17* > 新しいプロジェクト をクリックします。
3. 名前フィールドに **mlreplay_project** と入力します。
4. ロケーションフィールドに `C:¥mlreplay` と入力し、次へをクリックします。
5. データベースの表示名フィールドに **cons** と入力します。
6. 編集をクリックします。汎用 *ODBC* データベースに接続ウィンドウが表示されます。
7. ユーザ *ID* フィールドに、**DBA** と入力します。
8. パスワードフィールドに、**passwd** と入力します。
9. *ODBC* データソース名フィールドで、参照をクリックして **cons** を選択します。
10. *OK* をクリックし、保存をクリックします。
11. パスワードを記憶オプションをオンにし、ウィザードの最後に達するまで次へをクリックしてすべてのデフォルトをそのまま使用します。
12. 完了をクリックします。

結果

Mobile Link プロジェクトが作成されます。

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: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート [302 ページ]

前のタスク: レッスン 1: Mobile Link 統合データベースの設定 [303 ページ]

次のタスク: レッスン 3: 同期スクリプトの追加 [306 ページ]

1.5.12.3 レッスン 3: 同期スクリプトの追加

スクリプトは、それぞれ指定のスクリプトバージョンに属しています。スクリプトは、統合データベースにスクリプトバージョンを追加した後に追加してください。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

SQL Central を使用して同期スクリプトを表示、作成、修正できます。このレッスンでは、次の同期スクリプトを作成します。

upload_insert

このイベントは、新しいクライアント側データを統合データベースに適用する方法を定義します。

download_cursor

このイベントは、リモートクライアントにダウンロードするデータを定義します。

download_delete_cursor

このイベントは、アップロード専用の同期スクリプトを使用する場合に必要です。このチュートリアルでは、このイベントを無視するように Mobile Link サーバを設定します。

手順

1. ビュー > フォルダ をクリックします。
2. SQL Central の左ウィンドウ枠の *Mobile Link 17* で、**mlreplay_project**、統合データベース、**cons - DBA** の順に展開します。
3. バージョンを右クリックし、新規 > バージョン を選択します。
4. 新しいスクリプトバージョンの名前を指定してください。フィールドに **MLReplayDemo** と入力します。
5. 完了をクリックします。
6. SQL Central の左ウィンドウ枠の *Mobile Link 17* で、**mlreplay_project**、統合データベース、**cons - DBA** の順に展開します。
7. 同期テーブルを右クリックし、新規 > 同期テーブル をクリックします。
8. リモートテーブルと同じ名前のテーブルを統合データベースで選択するオプションをクリックします。
9. 同期するテーブルの所有者を指定してください。リストで *DBA* をクリックします。
10. 同期するテーブルを指定してください。リストで *T1* をクリックします。
11. 完了をクリックします。

    T1 テーブルは同期テーブルとして登録され、そのテーブルにスクリプトを追加できます。
12. SQL Central の左ウィンドウ枠の *Mobile Link 17* で、**mlreplay_project**、統合データベース、**cons - DBA**、同期テーブルの順に展開します。
13. *T1* を右クリックし、新規 > テーブルスクリプト をクリックします。
14. テーブルスクリプトを作成するバージョンを指定してください。リストで **MLReplayDemo** をクリックします。
15. テーブルスクリプトを実行するイベントを指定してください。リストで *upload_insert* をクリックし、次へをクリックします。
16. 完了をクリックします。
17. SQL Central の右ウィンドウ枠で、*upload_insert* イベント用の次の SQL スクリプトを使用します。

    ```
    INSERT INTO T1 VALUES( cast({ml s.remote_id} as INTEGER), {ml r.2}, {ml r.3} );
    ```

    upload_insert イベントは、リモートデータベースに挿入されたデータが統合データベースにどのように適用されるかを決定します。
18. ファイル > 保存 をクリックします。
19. 手順 13 ～ 16 を繰り返して、手順 15 の *upload_insert* イベントの代わりに *download_cursor* イベントを指定します。
20. SQL Central の右ウィンドウ枠で、*download_cursor* イベント用の次の SQL スクリプトを使用します。

    ```
    SELECT pk1, pk2, c1 FROM T1;
    ```

    download_cursor スクリプトは、ダウンロードしてリモートデータベースに (挿入または更新によって) 取り込む必要があるローを、統合データベースから選択するためのカーソルを定義します。
21. ファイル > 保存 をクリックします。
22. 手順 13 ～ 16 を繰り返して、手順 15 の *upload_insert* イベントの代わりに *download_delete_cursor* イベントを指定します。
23. SQL Central の右ウィンドウ枠で、*download_delete_cursor* イベント用の次の SQL スクリプトを使用します。

    ```
    --{ml_ignore}
    ```

24. ファイル > 保存 をクリックします。

結果

同期スクリプトが作成されます。

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: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート [302 ページ]

前のタスク: レッスン 2: Mobile Link プロジェクトの作成 [305 ページ]

次のタスク: レッスン 4: 記録のための Mobile Link サーバの起動 [308 ページ]

1.5.12.4 レッスン 4: 記録のための Mobile Link サーバの起動

このレッスンでは、-c オプションを使って Mobile Link サーバ (mlsrv17) を起動し、統合データベースに接続します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

次のコマンドを実行して、統合データベースに接続します。

```
mlsrv17 -c "DSN=cons" -zu+ -zs mlreplay_svr -x tcpip -ot mlsrv.mls -v+ -rp .
```

使用している Mobile Link サーバの各オプションの説明を次に示します。-ot および -v オプションは、デバッグとトラブルシューティングの情報を提供します。これらのロギングオプションは、開発環境での使用に適しています。一般に、パフォーマンス上の理由から、-v は運用環境では使用しません。

オプション	説明
-c	続いて接続文字列を指定します。
-ot	メッセージログファイル `mlsrv.mls` を指定します。
-v+	ログを取る対象となる情報を指定します。-v+ を使用して、最大冗長ロギングをオンに設定します。
-rp	同期がプレイバック用に記録されるディレクトリを指定します。
-x	同期要求を受信するために使用するプロトコルを設定します。
-zs	Mobile Link サーバ名を設定します。
-zu+	自動的に新しいユーザを追加します。

結果

Mobile Link サーバが起動し、統合データベースに接続されます。Mobile Link サーバメッセージウィンドウが表示されます。

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: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート [302 ページ]

前のタスク: レッスン 3: 同期スクリプトの追加 [306 ページ]

次のタスク: レッスン 5: Mobile Link クライアントデータベースの設定 [309 ページ]

1.5.12.5 レッスン 5: Mobile Link クライアントデータベースの設定

Mobile Link は、統合データベースサーバと多数のモバイルデータベースとの間で同期を実行できるように設計されています。このレッスンでは、リモートデータベースを作成し、`T1` テーブル (このテーブルは統合データベースと同期する) を作成し、同期パブリケーション、ユーザ、サブスクリプションを作成します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

このレッスンでは、SQL Anywhere データベースを統合データベースと Mobile Link クライアントに使用します。また、このチュートリアルの目的上、Mobile Link クライアント、統合データベース、および Mobile Link サーバはすべて同じコンピュータに置きます。

Mobile Link クライアントデータベースを設定するために、リモートデータベースに対して **T1** テーブルを作成します。**T1** テーブルは、統合データベースの **T1** テーブルに対応します。Mobile Link サーバでは、SQL ベースのスクリプトを使用して製品数が同期されます。

テーブルの作成後に、クライアントデータベースで同期ユーザ、パブリケーション、サブスクリプションを作成します。パブリケーションは、リモートデータベース上の同期対象となるテーブルとカラムを識別します。これらのテーブルとカラムをアーティクルと呼びます。同期サブスクリプションは、パブリケーションに対する Mobile Link ユーザのサブスクリプションです。

手順

1. dbinit コマンドラインユーティリティを使用して、Mobile Link クライアントデータベースを作成します。

   次のコマンドを実行して、**remote** データベースを作成します。

   ```
   dbinit -dba DBA,passwd remote.db
   ```

2. dbeng17 コマンドラインユーティリティを使用して、Mobile Link クライアントデータベースを起動します。

   次のコマンドを実行して、**remote** データベースを起動します。

   ```
   dbeng17 remote
   ```

3. Interactive SQL を使用して、リモートデータベースに接続します。

   次のコマンドを実行します。

   ```
   dbisql -c "SERVER=remote;UID=DBA;PWD=passwd"
   ```

4. **remote** データベース用に **T1** テーブルを作成します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   CREATE TABLE T1 (
       pk1      INTEGER,
       pk2      INTEGER,
       c1       VARCHAR(30000),
       PRIMARY KEY(pk1,pk2)
   );
   SET OPTION PUBLIC.ml_remote_id = '0';
   ```

5. **remote** データベース用に Mobile Link 同期ユーザ、パブリケーション、サブスクリプションを作成します。

   Interactive SQL で次の SQL 文を実行します。

   ```
   CREATE PUBLICATION P1 ( TABLE T1 );
   CREATE SYNCHRONIZATION USER U1;
   CREATE SYNCHRONIZATION SUBSCRIPTION TO P1 FOR U1 TYPE 'TCPIP' ADDRESS
   'host=localhost;port=2439';
   ```

6. 次のレッスンのために Interactive SQL を開いたままにします。

結果

リモートデータベース、`T1` テーブル、同期パブリケーション、ユーザ、サブスクリプションが作成されます。

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: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート [302 ページ]

前のタスク: レッスン 4: 記録のための Mobile Link サーバの起動 [308 ページ]

次のタスク: レッスン 6: 同期の記録 [311 ページ]

1.5.12.6 レッスン 6: 同期の記録

このレッスンでは、dbmlsync ユーティリティを実行して SQL Anywhere リモートデータベース用の Mobile Link 同期を開始します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. スキーマが Mobile Link サーバ上でキャッシュされるように、最初に記録された同期を実行します。

   次のコマンドを実行して、`remote` データベースを同期します。

   ```
   dbmlsync -c "SERVER=remote;UID=DBA;PWD=passwd" -ot remote1.mls -e
   "sv=MLReplayDemo" -v+
   ```

次の表は、使用されている各 dbmlsync オプションを説明しています。

オプション	説明
-c	接続文字列を指定します。
-ot	メッセージのログの記録先ファイルを指定します。
-e	同期先のスクリプトバージョンを指定します。
-v+	ログを取る対象となる情報を指定します。-v+ を使用して、最大冗長ロギングをオンに設定します。

同期が成功したことを示す出力画面が表示されます。SQL ベースの同期によって、クライアントの **T1** テーブル内のローが、統合データベース内の **T1** テーブルに転送されました。

2. 2 回目の同期が発生するように、データ挿入のためにリモートデータベースを準備します。

   Interactive SQL で引き続き *remote* データベースに接続されている必要があります。接続されていない場合は、次のコマンドを実行して、**remote** データベースに接続します。

   ```
   dbisql -c "SERVER=remote;UID=DBA;PWD=passwd"
   ```

3. リプレイセッション中に Mobile Link サーバにアップロードするように、データを **remote** データベースにロードします。

   Interactive SQL で次の SQL 文を実行します。

   ```
   INSERT INTO T1 (pk1,pk2,c1) values (0,1,'data1');
   INSERT INTO T1 (pk1,pk2,c1) values (0,2,'data2');
   INSERT INTO T1 (pk1,pk2,c1) values (0,3,'data3');
   INSERT INTO T1 (pk1,pk2,c1) values (0,4,'data4');
   INSERT INTO T1 (pk1,pk2,c1) values (0,5,'data5');
   INSERT INTO T1 (pk1,pk2,c1) values (0,6,'data6');
   INSERT INTO T1 (pk1,pk2,c1) values (0,7,'data7');
   INSERT INTO T1 (pk1,pk2,c1) values (0,8,'data8');
   INSERT INTO T1 (pk1,pk2,c1) values (0,9,'data9');
   INSERT INTO T1 (pk1,pk2,c1) values (0,10,'data10');
   COMMIT;
   ```

4. 2 回目に記録した同期を実行します。これは、リプレイされるプロトコルです。

   次のコマンドを実行して、**remote** データベースを同期します。

   ```
   dbmlsync -c "SERVER=remote;UID=DBA;PWD=passwd" -ot remote2.mls -e
   "sv=MLReplayDemo" -v+
   ```

結果

データベースが同期されます。

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: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート [302 ページ]

前のタスク: レッスン 5: Mobile Link クライアントデータベースの設定 [309 ページ]

次のタスク: レッスン 7: リプレイのための Mobile Link サーバの再起動 [313 ページ]

1.5.12.7 レッスン 7: リプレイのための Mobile Link サーバの再起動

このレッスンでは、記録を停止するために Mobile Link サーバを停止してから、-rp オプションを指定せずにサーバを再起動して、サーバでリプレイの準備をします。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. 次のコマンドを実行して、Mobile Link サーバである **mlreplay_svr** を停止します。

   ```
   mlstop -w -t 1m mlreplay_svr
   ```

   次の表は、使用されている各オプションを説明しています。

オプション	説明
-w	コマンドプロンプトに戻る前に、サーバがシャットダウンされるのを待機します。
-t	1 分後、または現在の同期が完了した後のどちらか早い時期にサーバをシャットダウンするように指定します。

   Mobile Link サーバは、同期記録とともに停止します。

2. 次のコマンドを実行して、統合データベースに接続します。

   ```
   mlsrv17 -c "DSN=cons" -zu+ -zs mlreplay_svr -x tcpip -ot server_replay.mls -v+
   ```

   使用している Mobile Link サーバの各オプションの説明を次に示します。-ot および -v オプションは、デバッグとトラブルシューティングの情報を提供します。これらのロギングオプションは、開発環境での使用に適しています。一般に、パフォーマンス上の理由から、-v は運用環境では使用しません。

オプション	説明
-c	接続文字列を指定します。
-ot	メッセージログファイル `server_replay.mls` を指定します。
-v+	ログを取る対象となる情報を指定します。-v+ を使用して、最大冗長ロギングをオンに設定します。
-x	同期要求を受信するために使用するプロトコルを設定します。
-zs	Mobile Link サーバ名を設定します。
-zu+	自動的に新しいユーザを追加します。

Mobile Link サーバメッセージウィンドウが表示されます。

結果

Mobile Link サーバが停止してから再度起動します。

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: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート [302 ページ]

前のタスク: レッスン 6: 同期の記録 [311 ページ]

次のタスク: レッスン 8: 同期のリプレイ [314 ページ]

1.5.12.8 レッスン 8: 同期のリプレイ

このレッスンでは、スキーマが Mobile Link サーバにキャッシュされるように同期を実行します。シミュレートされたクライアント情報ファイルを作成し、シミュレートされたクライアントで Mobile Link プロトコル情報をリプレイします。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

シミュレートされたクライアント情報ファイルは、記録されたプロトコルを、複数のシミュレートされたクライアント間で同時にリプレイするときにだけ必要です。

手順

1. 次のコマンドを実行して、**remote** データベースを同期します。

   ```
   dbmlsync -c "SERVER=remote;UID=DBA;PWD=passwd" -ot remote3.mls -e
   "sv=MLReplayDemo" -v+
   ```

   次の表は、使用されている各 dbmlsync オプションを説明しています。

オプション	説明
-c	接続文字列を指定します。
-ot	メッセージのログの記録先ファイルを指定します。
-e	同期先のスクリプトバージョンを指定します。
-v+	ログを取る対象となる情報を指定します。-v+ を使用して、最大冗長ロギングをオンに設定します。

   同期が成功したことを示す出力画面が表示されます。SQL ベースの同期によって、クライアントの **T1** テーブル内のローが、統合データベース内の **T1** テーブルに転送されました。

2. mlreplay ユーティリティで使用するために、シミュレートされたクライアント情報ファイルを作成します。

   新しいテキストファイルを作成して、表示されているように次のカンマ区切りリストに書き込みます。

   ```
   mlreplay1,,1,
   mlreplay2,,2,
   mlreplay3,,3,
   mlreplay4,,4,
   mlreplay5,,5,
   mlreplay6,,6,
   mlreplay7,,7,
   mlreplay8,,8,
   mlreplay9,,9,
   mlreplay10,,10,
   ```

3. そのファイルを `mlreplay.csv` として作業ディレクトリに保存します。

   クライアント情報ファイルは、10 個のリモートクライアントをシミュレートするために使用できます。

4. シミュレートされクライアントで、記録した同期をリプレイします。

次のコマンドを実行します。

```
mlreplay -ap -x tcpip -ot mlreplay.mls -sci mlreplay.csv
recorded_protocol_mlreplay_svr_2.mlr
```

次の表は、使用されている各オプションを説明しています。

オプション	説明
-ap	mlreplay ユーティリティによって Mobile Link サーバ上で進行オフセット不一致警告が生成されないように、リプレイセッションでリプレイされている同期の進行を調整します。
-x	同期要求を受信するために使用するプロトコルを設定します。
-ot	メッセージのログを取るファイルを指定します。
-sci	クライアント情報ファイルのロケーションを指定します。

mlreplay ユーティリティは、`recorded_protocol_mlreplay_svr_2.mlr` という記録されたプロトコルファイルに、接続の開始から接続の最後までの情報を格納します。

5. テキストエディタで `mlreplay.mls` ログファイルを開き、Mobile Link リプレイの結果を確認します。

結果

同期が実行され、スキーマが Mobile Link サーバ上でキャッシュされ、シミュレートされたクライアントに関する Mobile Link プロトコル情報をリプレイするためにシミュレートされたクライアント情報ファイルが作成されます。

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: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート [302 ページ]

前のタスク: レッスン 7: リプレイのための Mobile Link サーバの再起動 [313 ページ]

次のタスク: レッスン 9: クリーンアップ [317 ページ]

1.5.12.9 レッスン 9: クリーンアップ

チュートリアルをコンピュータから削除します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. タスクバー上で、Interactive SQL、SQL Anywhere、Mobile Link、同期クライアントの各ウィンドウを右クリックし、閉じるを選択して閉じます。
2. 次の手順で、チュートリアルに関連するすべてのデータソースを削除します。
   a. ODBC データソースアドミニストレータを起動します。
   b. スタート > プログラム > *SQL Anywhere17* > 管理ツール > *ODBC* データソースアドミニストレータをクリックします。
   c. ユーザデータ ソースリストから **cons** を選択し、削除をクリックします。
3. 統合データベースとリモートデータベースが保存されているディレクトリを削除します。

結果

チュートリアルがコンピュータから削除されます。

タスクの概要: チュートリアル: Mobile Link リプレイユーティリティを使った複数の Mobile Link クライアントのシミュレート [302 ページ]

前のタスク: レッスン 8: 同期のリプレイ [314 ページ]

1.6 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