SYBASE iAnywhere

Mobile Link
クイックスタート

バージョン 12.0.1

2012 年 1 月

バージョン 12.0.1
2012 年 1 月

目次

はじめに

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。

Mobile Link テクノロジ

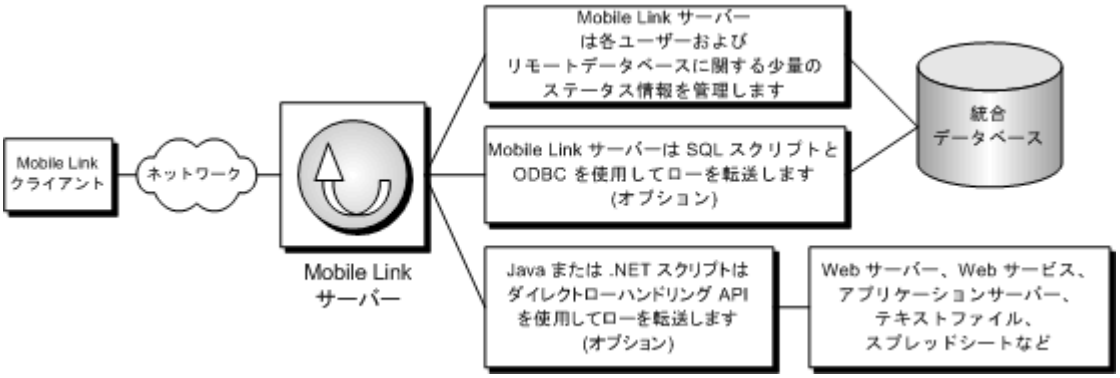
この項では、Mobile Link 同期テクノロジの概要と、Mobile Link 同期テクノロジを使用して 2 つ以上のデータベース間でデータをレプリケートする方法について説明します。

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。

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 サーバー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、モバイルデバイス、サーバー、デスクトップコンピューター、またはスマートフォンにインストールできます。Ultra Light データベースと SQL Anywhere データベースという 2 種類のクライアントがあります。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。 「Mobile Link クライアント」『Mobile Link クライアント管理』を参照してください。
- **ネットワーク**　Mobile Link サーバーと 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。 次の項を参照してください。
  - Mobile Link サーバー：「-x mlsrv12 オプション」『Mobile Link サーバー管理』 「-x mlsrv12 オプション」『Mobile Link サーバー管理』
  - Ultra Light と 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 クライアント管理』 「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 クライアント管理』

- **Mobile Link サーバー**　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ー間のインターフェイスを提供します。 「Mobile Link サーバー」『Mobile Link サーバー管理』を参照してください。

- **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ー、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。「Mobile Link 統合データベース」『Mobile Link サーバー管理』を参照してください。

- **ステータス情報**　Mobile Link サーバーは通常、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。 「Mobile Link システムデータベース」『Mobile Link サーバー管理』を参照してください。

- **SQL ローハンドリング**　Mobile Link サーバー用に SQL スクリプトを作成すると、サーバー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。 「サーバー側の同期論理の作成オプション」 12 ページを参照してください。

- **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。 「ダイレクトローハンドリング」『Mobile Link サーバー管理』を参照してください。

- **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。 次の項を参照してください。

  - 「Mobile Link イベントの概要」『Mobile Link サーバー管理』
  - 「同期スクリプトの作成」『Mobile Link サーバー管理』
  - 「同期イベント」『Mobile Link サーバー管理』
  - 「サーバー側の同期論理の作成オプション」 12 ページ

Mobile Link の機能

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。Sybase Central を使用すると、同期モデルをオフラインで表示し、簡単なインターフェイスで変更を行い、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。

- **モニターとレポート**　Mobile Link には、同期をモニターする 3 つのメカニズムが用意されています。Mobile Link モニター、Mobile Link 用 SQL Anywhere モニター、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。

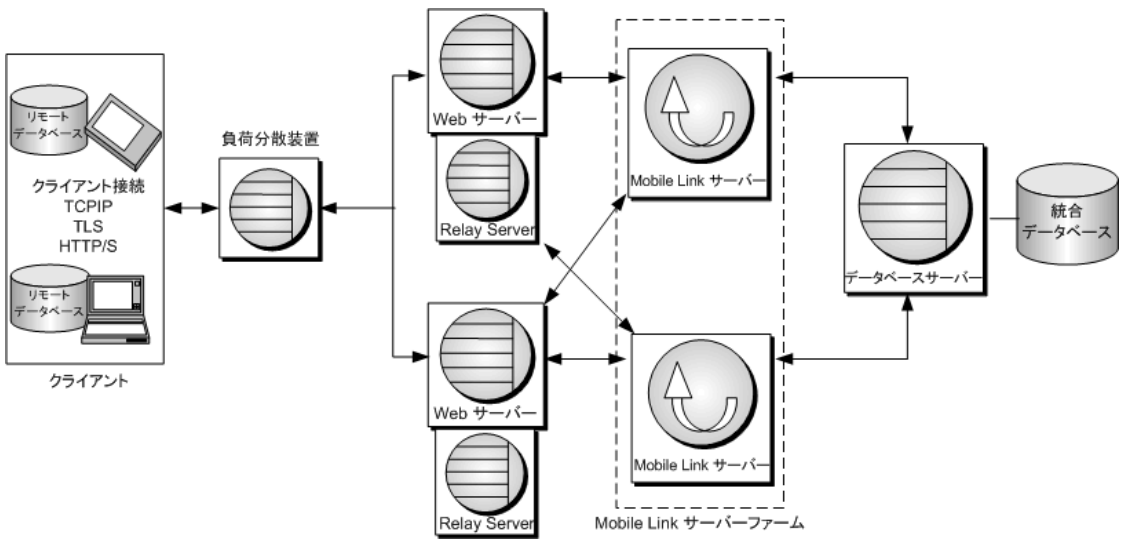
- **スケーラビリティ**　Mobile Link はスケーラビリティに優れ、堅牢な同期プラットフォームです。Mobile Link の単一のサーバーが数千の同期を同時に処理したり、負荷分散を使用して複数の Mobile Link サーバーを同時に稼働したりできます。Mobile Link サーバーはマルチスレッド化されており、統合データベースで接続プールを使用します。

- **セキュリティ**　Mobile Link には、既存の認証に統合できるユーザー認証、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ーセキュリティなど、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 と Sybase Relay Server のホスティングサービス**　Relay Server は、Web サーバー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ー間で、安全で負荷分散された通信を実現します。「Relay Server の概要」『Relay Server』を参照してください。

  Sybase Relay Server のホスティングサービスは、Sybase をホストとする Relay Server のファームです。特に公共無線ネットワークを使用してデータを送信する場合に、Mobile Link データ同期を使用するモバイルアプリケーションの開発と評価が従来より容易になります。「Sybase ホストのリレーサービス」『Relay Server』を参照してください。

  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。



アーキテクチャー

- **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論理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、ユーザー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。

- **サーバー起動同期**　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。Mobile Link サーバー起動同期を参照してください。

  サーバー起動同期への代替機能としてサーバー起動リモートタスク (SIRT) を使用できます。詳細については、「リモートデータベースの集中管理」『Mobile Link サーバー管理』と「サーバー起動リモートタスク (SIRT)」『Mobile Link サーバー管理』を参照してください。

- **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。Windows Mobile デバイスは、Microsoft ActiveSync を使用して同期できます。

- **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。)

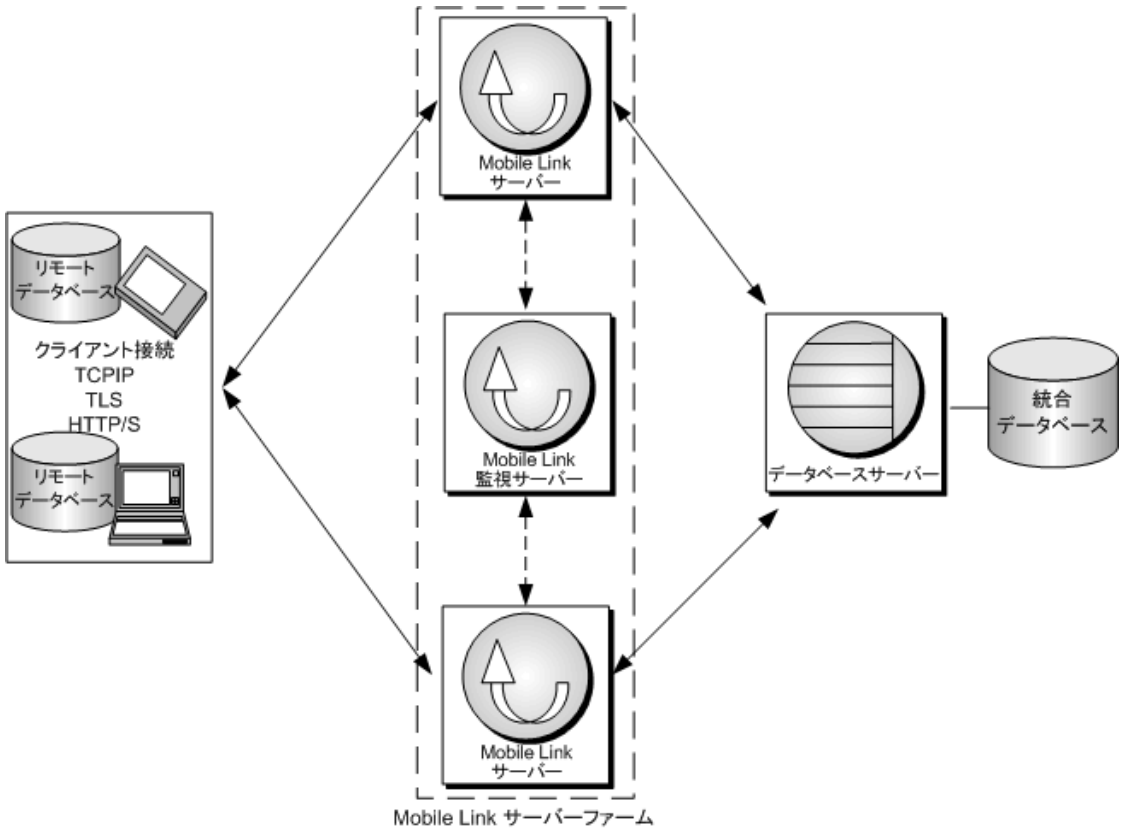
  トランザクション全体が同期されるか、トランザクション全体がまったく同期されないかのどちらかになります。これにより、各データベースでトランザクション単位の整合性が確保されます。

- **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

バー 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 サーバーは、Windows、UNIX、Linux、Mac OS X 上で動作します。SQL Anywhere は、Windows、Windows Mobile、UNIX、Linux、Mac OS X 上で動作します。Ultra Light は、Windows Mobile または BlackBerry 上で動作します。「サポートされるプラットフォーム」『SQL Anywhere 12 紹介』を参照してください。

- **Mobile Link 監視サーバー**　Mobile Link 監視サーバーは、サーバーファーム内の 1 つの Mobile Link サーバーだけがプライマリサーバーとして動作するようにします。そうすることで、サーバー起動同期環境での冗長な通知を防止し、QAnywhere メッセージング環境のメッセージを保持できます。次の図は、サーバーファーム環境の Mobile Link 監視サーバーを示しています。



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 サーバー API を使用してこのアーキテクチャーを拡張すると、サーバー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 サーバー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 サーバーへの接続を開くと、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 サーバーは、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。 「同期モデル」27 ページを参照してください。

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

♦ Mobile Link アプリケーションの作成

1. 統合データベースを設定します。

   ●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。

   「Mobile Link 統合データベース」『Mobile Link サーバー管理』を参照してください。

2. リモートデータベースを設定します。

   ●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 ユーザーを作成します。 「Mobile Link ユーザー」『Mobile Link クライアント管理』を参照してください。

   ●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。 「パブリケーション」『Mobile Link クライアント管理』を参照してください。

   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。 「Ultra Light でのデータのパブリッシュ」『Ultra Light データベース管理とリファレンス』を参照してください。

3. アップロードの適用形式を特定するには、サーバー同期論理を作成します。 「同期スクリプトの作成」『Mobile Link サーバー管理』を参照してください。

4. 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するには、タイムスタンプベースの同期を設定します。 「タイムスタンプベースのダウンロードの実装」『Mobile Link サーバー管理』を参照してください。

5. Mobile Link サーバーを起動します。「Mobile Link サーバーの実行」『Mobile Link サーバー管理』を参照してください。

6. クライアントの同期を開始します。

   ●SQL Anywhere リモートデータベースの詳細については、「同期の開始」『Mobile Link クライアント管理』を参照してください。

   ●Ultra Light リモートデータベースの詳細については、「Ultra Light クライアントの同期設計」『Ultra Light データベース管理とリファレンス』を参照してください。

入門情報

- 「Mobile Link 同期」 1 ページ
- 「同期の方法」『Mobile Link サーバー管理』

チュートリアル

- 「チュートリアル：Mobile Link の概要」 79 ページ
- 「Mobile Link の CustDB サンプル」 51 ページ
- 「チュートリアル: Ultra Light CustDB サンプルアプリケーションの構築」『Ultra Light データベース管理とリファレンス』
- 「Mobile Link Contact サンプル」 66 ページ
- 「チュートリアル：Oracle Database 10g での Mobile Link の使用」 114 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」 131 ページ
- 「チュートリアル：Java 同期論理の使用」 149 ページ
- 「チュートリアル：.NET 同期論理の使用」 157 ページ
- 「チュートリアル：カスタムユーザー認証用の Java と .NET の使用」 167 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」 174 ページ
- 「チュートリアル：Microsoft Excel との同期」 197 ページ
- 「チュートリアル：XML との同期」 214 ページ

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 *%SQLANYSAMP12%¥MobiLink* にインストールされます。

●Mobile Link のコード交換サンプルは、http://www.ianywhere.jp/dl/dl_evl.html にあります。ダウンロードにはアイエニウェアホームページのユーザー登録が必要です。

Mobile Link アプリケーションの設計

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ーがあります。

- **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ー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、ユーザー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**　各ユーザーがローカルデータベースを保有しています。各ユーザーは接続状態に関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。このセクションでは、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について説明します。また、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も説明します。

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。各ユーザー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、各ユーザー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、ユーザー別、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、これをサーバー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 (順序数など) 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、トリガー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ーを協定世界時 (UTC) にします。

参照

- 「同期の方法」『Mobile Link サーバー管理』

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。

- **Sybase Central**　Sybase Central の Mobile Link 12 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ー**　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。ここにはサーバー側の 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 スクリプトへの Mobile Link システムテーブルの挿入やリモートユーザーの登録などの作業を行うためのシステムプロシージャー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**　上級ユーザーの場合は、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。

参照

- 「Sybase Central の Mobile Link プラグイン」 22 ページ
- 「同期モデルタスク」 30 ページ
- 「Mobile Link サーバーシステムプロシージャー」『Mobile Link サーバー管理』
- 「Mobile Link ユーティリティ」『Mobile Link サーバー管理』
- 「Mobile Link サーバーのシステムテーブル」『Mobile Link サーバー管理』

サーバー側の同期論理の作成オプション

Mobile Link 同期スクリプトは、SQL で記述することも、Java (Java 用 Mobile Link サーバー API を使用) または .NET (.NET 用 Mobile Link サーバー 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 同期論理が最適です。

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 の同期論理は、SQL 論理と同様に機能します。Mobile Link サーバーは、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、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 と .NET の同期論理では、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 と .NET の同期論理では、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ー、Web サーバー、ファイルなど) にアクセスできます。
- **認証**　ユーザー認証プロシージャーを 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ー**　RDBMS でユーザー定義のストアドプロシージャーを使用できない場合は、Java や .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ーへの接続を必要とする場合は、Java または .NET 同期論理を使用して、同期イベントによりトリガーされるアクションを実行できます。Java または .NET 同期論理は、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 や .NET で作成できます。また、SQL スクリプトでユーザー定義の名前付きパラメー

ター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。 「ユーザー定義の名前付きパラメーター」『Mobile Link サーバー管理』を参照してください。

Mobile Link サーバー API

Java と .NET 同期論理は、Mobile Link サーバー API を介して使用できます。Mobile Link サーバー API は、Mobile Link 同期用のクラスとインターフェイスのセットです。

Java 用 Mobile Link サーバー 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 などのインターフェイスを使用して、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 サーバーで実行できます。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。

「Mobile Link サーバー Java API リファレンス」『Mobile Link サーバー管理』を参照してください。

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

- .NET から ODBC を呼び出す iAnywhere 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 などのインターフェイスを使用して、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 サーバーで実行できます。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。

「Mobile Link サーバー .NET API リファレンス」『Mobile Link サーバー管理』を参照してください。

参照

- 「同期スクリプトの作成」『Mobile Link サーバー管理』
- 「同期の方法」『Mobile Link サーバー管理』
- 「Java による同期スクリプトの作成」『Mobile Link サーバー管理』
- 「.NET での同期スクリプトの作成」『Mobile Link サーバー管理』
- 「ダイレクトローハンドリング」『Mobile Link サーバー管理』

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 サーバー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 サーバー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 サーバー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 サーバー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 サーバーは、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 サーバー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 サーバーは、ユーザーが作成した同期論理を使用して、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 サーバー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。

Mobile Link 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 サーバー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。

注意
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。 「Mobile Link 独立性レベル」『Mobile Link サーバー管理』を参照してください。

参照

- 「Mobile Link イベントの概要」『Mobile Link サーバー管理』
- 「アップロード中のイベント」『Mobile Link サーバー管理』
- 「ダウンロード中のイベント」『Mobile Link サーバー管理』

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。同期イベントが発生すると、Mobile Link はその同期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 サーバー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。

注意
[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ーします。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}
```

イベントとスクリプトの詳細については、次の項を参照してください。

- 「同期スクリプトの作成」『Mobile Link サーバー管理』
- 「同期イベント」『Mobile Link サーバー管理』

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 の同期論理を使用して、スクリプトを記述できます。Java と .NET の同期論理を使用すると、Mobile Link サーバー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 Mobile Link サーバー API (Java 用と .NET 用) があります。

「サーバー側の同期論理の作成オプション」 12 ページを参照してください。

RDBMS 依存のスクリプト記述については、「Mobile Link 統合データベース」『Mobile Link サーバー管理』を参照してください。

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 サーバー 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ーを使用して、スクリプトを手動でシステムテーブルに追加できます。

●Sybase 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。

「スクリプトの追加と削除」『Mobile Link サーバー管理』を参照してください。

同期処理のトランザクション

Mobile Link サーバーは、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 サーバーは、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ーやトリガーで、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ベースの最新の状態にします。

Mobile Link サーバーは、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。

参照

- 「スクリプトでの最終ダウンロード時刻」『Mobile Link サーバー管理』
- 「Mobile Link イベントの概要」『Mobile Link サーバー管理』
- 「アップロード中のイベント」『Mobile Link サーバー管理』
- 「ダウンロード中のイベント」『Mobile Link サーバー管理』

同期の障害処理の方法

Mobile Link 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。サーバー側では、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、サーバー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 サーバーは、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、サーバー側には影響はありません。

障害が発生しても、データは失われません。Mobile Link サーバーと Mobile Link クライアントが障害時のデータ管理を行います。開発者やユーザーは、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。

アップロードの処理方法

Mobile Link サーバーが 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ー**　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 サーバー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 サーバー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 サーバー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。

> **注意**
>
> **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。複数のユーザー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。

参照整合性と同期

BlackBerry または J2SE 用の Ultra Light 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。 次の項を参照してください。
  - 「sp_hook_dbmlsync_download_ri_violation」『Mobile Link クライアント管理』
  - 「sp_hook_dbmlsync_download_log_ri_violation」『Mobile Link クライアント管理』
- 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ーが含まれます。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します。さらに、削除されたローの数を取得することもできます。

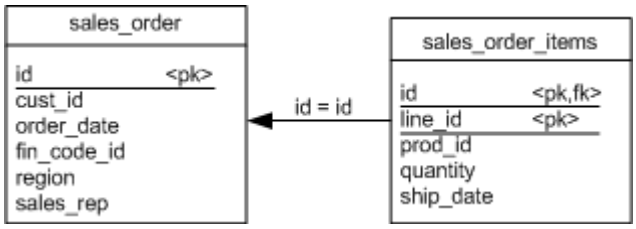
  警告が発生したときに同期を失敗させるには、同期 observer を実装し、observer に (グローバル変数などを使用して) エラーコールバック関数から信号を送信する必要があります。この場合、同期は observer への次回の呼び出しで失敗します。

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。

次の項を参照してください。

- 「sp_hook_dbmlsync_download_log_ri_violation」『Mobile Link クライアント管理』
- 「sp_hook_dbmlsync_download_ri_violation」『Mobile Link クライアント管理』

セキュリティ

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ー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。SQL Anywhere 統合データベースを使用している場合は、「データのセキュリティ」『SQL Anywhere サーバー データベース管理』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／サーバー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、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する確実な手段とはなりません。

  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。

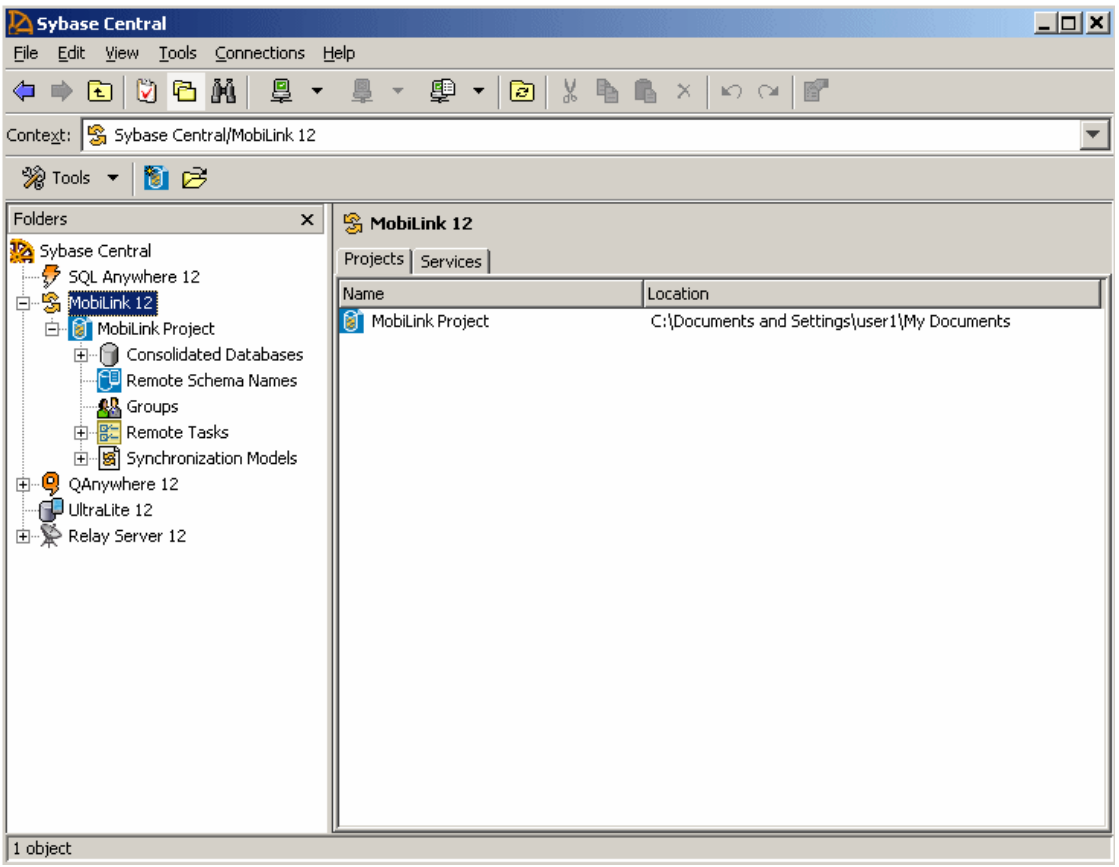
  SQL Anywhere リモートデータベースを使用している場合は、「データのセキュリティ」『SQL Anywhere サーバー データベース管理』を参照してください。

  Ultra Light データベースを使用している場合は、「Ultra Light データベースのセキュリティ」『Ultra Light データベース管理とリファレンス』を参照してください。

- **同期中のデータ保護**　Mobile Link クライアントから Mobile Link サーバーへの通信は、Mobile Link トランスポートレイヤーセキュリティ機能で保護できます。「トランスポートレイヤーセキュリティ」『SQL Anywhere サーバー データベース管理』を参照してください。
- **権限のないユーザー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ー認証システムで保護できます。このメカニズムにより、権限のないユーザーはデータを同期できなくなります。「Mobile Link ユーザー」『Mobile Link クライアント管理』を参照してください。

Sybase Central の Mobile Link プラグイン

Sybase Central の Mobile Link プラグインは、バージョン 12 で再設計されました。以前のバージョンでは、このプラグインにはモデルモードと管理モードの 2 種類のモードがありました。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。次に示すように、バージョン 12 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 **[フォルダー]** 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。 「Mobile Link プロジェクトの作成」23 ページを参照してください。

●統合データベースの操作。 「統合データベースの追加」25 ページを参照してください。

●リモートスキーマ名の操作。 「Sybase Central でのエージェントの操作」『Mobile Link サーバー管理』を参照してください。

●グループの操作。 「Sybase Central でのエージェントの操作」『Mobile Link サーバー管理』を参照してください。

●リモートタスクの操作。 「リモートタスク」『Mobile Link サーバー管理』を参照してください。

●同期モデルの操作。 「同期モデル」27 ページを参照してください。

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。 「リモートデータベースの集中管理」『Mobile Link サーバー管理』を参照してください。

Sybase 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 ユーザーのリスト

●ユーザー定義グループ、リモートデータベース、またはデバイスのリスト

●リモートデータベースのリスト

●Notifier のリスト

Mobile Link プロジェクトの作成

Sybase Central の Mobile Link を操作する前に、Mobile Link プロジェクトを作成する必要があります。

前提条件

この作業を実行するための前提条件は、ありません。

内容と備考

サンプルの Mobile Link プロジェクトは *%sqlanysamp12%¥MobiLink¥CustDB¥project.mlp* にあります。

♦ Mobile Link プロジェクトの作成

1. Sybase Central の **[フォルダ]** ビューで、**[Mobile Link 12]** を右クリックし、 » **[新規]** » **[プロジェクト]** を選択します。
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ー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ーを選択します。
4. **[完了]** をクリックして名前とロケーションのみを設定してプロジェクトを保存するか、**[次へ]** をクリックして次の手順に従って追加情報を指定します。
5. **[次へ]** をクリックした場合も、プロジェクトに関連付ける統合データベースがわかっている場合は、**[統合データベースをプロジェクトに追加]** をクリックします。
   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ーを入力するか、**[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. データベース接続に使用したパスワードを保存する場合は、**[パスワードを記憶]** を選択します。
   d. **[次へ]** をクリックします。
6. 同期モデルを追加するか、後で作成するかを決めます。次のオプションのうちの 1 つを選択してください。
   - **[モデルを追加しない]**　同期モデルの追加または作成をすぐに行なわず、後から追加する場合は、このオプションを選択します。
   - **[新しいモデルを作成する]**　新しい同期モデルを作成する場合は、このオプションを選択します。プロジェクトの作成が完了すると、**[同期モデル作成ウィザード]** が起動します。「同期モデル」27 ページを参照してください。
   - **[次のファイルからモデルをインポートする]**　既存の同期モデルをインポートする場合は、このオプションを選択します。空のフィールドに、インポートする同期モデルのパスとファイル名を入力するか、**[参照]** をクリックして同期モデルファイルを選択します。これにより、プロジェクトフォルダーに同期モデルファイルのコピーが作成されます。

7. 同じスキーマ名を持つリモートデータベースのグループを特定するか後から追加する場合は、**[リモートスキーマ名をプロジェクトに追加]** を選択します。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すすめします。
   - **[リモートスキーマ名を適用するデータベースのタイプを指定してください。]** リモートスキーマ名を適用するデータベースのタイプを選択します。**[SQL Anywhere]** または **[Ultra Light]** のいずれかを指定できます。
8. **[完了]** をクリックして新しいプロジェクトを保存します。

結果

Mobile Link プロジェクトが作成されます。

次の手順

なし。

参照

- 「集中管理の概念」『Mobile Link サーバー管理』

統合データベースの追加

Sybase 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できます。

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。

内容と備考

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。

♦ 統合データベースの追加

1. 左ウィンドウ枠の **[フォルダ]** ビューで、使用している Mobile Link プロジェクトが選択されていることを確認し、プロジェクト名を右クリックして **[新規]** » **[統合データベース]** をクリックします。
2. 必要なデータベース接続パラメーターを入力し、**[次へ]** をクリックしてデータベースに接続します。

3. **[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場合は、**[説明]** フィールドに入力します。

4. データベース接続に使用したパスワードを保存する場合は、**[パスワードを記憶]** を選択します。

5. **[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。

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。

次の手順

なし。

参照

- 「集中管理の概念」『Mobile Link サーバー管理』

Mobile Link システム設定

テーブル、カラム、トリガー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 *%SQLANY12%¥MobiLink¥setup* フォルダーにあります。スクリプトによる処理の内容を詳細に確認するには、テキストエディター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。このインストールは後で実行することもできます。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。

参照

- 「Mobile Link システム設定のチェック」 26 ページ
- 「統合データベースの設定」『Mobile Link サーバー管理』
- 「同期モデルの展開」 43 ページ

Mobile Link システム設定のチェック

システム設定は、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 プロジェクトは、定義されている必要があります。

内容と備考

次のとおりです。

♦ Mobile Link システム設定のチェック

1. 左ウィンドウ枠の **[フォルダ]** ビューで、使用している Mobile Link プロジェクトが選択されていることを確認し、チェックする統合データベースを右クリックして **[Mobile Link システム設定のチェック]** をクリックします。

2. 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。

次の手順

なし。

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、Sybase Central の **[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。

[同期モデル作成ウィザード] を終了した後で、モデルをカスタマイズすることができます。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同期モデル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 サーバー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実行するファイル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 Sybase Central 外で変更することもできます。

[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

Sybase 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論理を設定します。

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。

内容と備考

次のとおりです。

♦ **[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設定します。**

1. 統合データベース用に Mobile Link プロジェクトをまだ作成していない場合は、Mobile Link プラグインを使用して作成します。

   「Mobile Link プロジェクトの作成」23 ページを参照してください。

2. Sybase Central の **[フォルダー]** ビューで、**[Mobile Link 12]** と Mobile Link プロジェクト名を展開し、**[同期モデル]** をクリックします。

3. **[ファイル]** » **[新規]** » **[同期モデル]** をクリックして、**[同期モデル作成ウィザード]** を開始します。

4. **[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。

5. **[プライマリキー要件]** ページで、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。 プライマリキーの詳細については、「ユニークなプライマリキー」『Mobile Link サーバー管理』を参照してください。

6. **[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リストから選択し、**[次へ]** をクリックします。統合データベースからスキーマがロードされます。

   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。

7. **[リモートデータベーススキーマ]** ページが表示されます。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。 決定方法の詳細については、「リモートスキーマ」29 ページを参照してください。

8. **[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。

9. **[完了]** をクリックします。

結果

[完了] をクリックすると、同期モデルが Sybase Central の左ウィンドウ枠で開きます。これでオフラインで作業している状態になり、モデルに変更を加えることができます。モデルを展開するまで、モデル以外で変更は行われません。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そのときまで作成されないか、変更されません。 「同期モデルタスク」30 ページを参照してください。

次の手順

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。 「同期モデルの展開」43 ページを参照してください。

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てください。

> **注意**
> 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。 「スキーマの更新」42 ページを参照してください。

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する。** 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。
- **ユーザーテーブルがない既存のリモートデータベースを更新する。** 空のリモートデータベースに展開すると、モデルのリモートスキーマがデータベース内で作成されます。このオプションは、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。 備考、初期化ユーティリティ (dbinit)『SQL Anywhere サーバー データベース管理』を参照してください。

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。 「Ultra Light 作成パラメーターの指定」『Ultra Light データベース管理とリファレンス』を参照してください。

- **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する。** 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した場合、既存のリモートデータベースは変更されません。既存のリモートデータベースのスキーマがモデルのリモートスキーマと異なる場合に、既存のリモートデータベースに直接展開しようとすると、モデル内のリモートスキーマを更新するよう要求されます。

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。

参照

- 「同期モデルの展開」 43 ページ

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。

Sybase 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ーを使用して Mobile Link スクリプトの追加や変更を行うことができます。 「Mobile Link サーバーシステムプロシージャー」『Mobile Link サーバー管理』を参照してください。

テーブルマッピングとカラムマッピングの変更

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。

♦ テーブルマッピングの方向の変更

1. Sybase Central の **[フォルダー]** ビューで、**[Mobile Link 12]**、Mobile Link プロジェクト名、**[同期モデル]** を展開してから、同期モデル名を選択します。
2. 右ウィンドウ枠で、**[マッピング]** タブを開きます。
3. **[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。
4. **[マッピング方向]** 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]**
   - **[リモートにのみダウンロード]**
   - **[統合にのみアップロード]**

♦ テーブルマッピングの削除

1. Sybase Central の **[フォルダー]** ビューで、**[Mobile Link 12]**、Mobile Link プロジェクト名、**[同期モデル]** を展開してから、同期モデル名を選択します。
2. 右ウィンドウ枠で、**[マッピング]** タブを開きます。
3. **[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。
4. **[マッピング方向]** 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。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。

テーブルマッピングとカラムマッピングの変更

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。必ず確認し、必要に応じてカスタマイズする必要があります。

♦ テーブルマッピングの変更

1. Sybase Central の **[フォルダー]** ビューで、**[Mobile Link 12]**、Mobile Link プロジェクト名、**[同期モデル]** を展開してから、同期モデル名を選択します。
2. 右ウィンドウ枠で、**[マッピング]** タブを開きます。
3. **[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。
4. マッピングされたリモートテーブルを変更するには、**[リモートテーブル]** コンテキストメニューで、同期されていないリモートテーブルのリストから異なる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。
   - リモートスキーマにテーブルを追加する場合は、「スキーマの更新」42 ページを参照してください。

5. マッピングされていない統合テーブルにテーブルマッピングを追加するには、**[ファイル]** » **[新規]** » **[テーブルマッピング]** を使用して、テーブルを選択する **[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。リモートスキーマへの変更を回避するには、対応するテーブルがまだ存在していない場合、対応するテーブルをリモートスキーマにも追加するオプションを無効にします。テーブルをリモートスキーマに追加したいものの、すべてのカラムを追加したいわけではない場合は、カラムを選択できるオプションを有効にします。

   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。

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 ユーザー名、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ー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。

♦ テーブルマッピングのカラムマッピングの変更

1. **[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。

2. 右ウィンドウ枠で、**[カラムマッピング]** タブを開きます。

3. 変更するカラムマッピングを右クリックし、コンテキストメニューから次のいずれかを選択します。

   - **[なし]**
   - **[Mobile Link ユーザー名]**
   - **[リモート ID]**
   - **[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、**[Mobile Link ユーザー名]**、**[リモート ID]**、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、統合カラムはリモートデータベースでダウンロードされません。

ダウンロードタイプの変更

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。ダウンロードタイプは、**[マッピング]** タブの **[テーブルマッピング]** ウィンドウ枠で変更できます。

- **[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。 「タイムスタンプベースのダウンロードの実装」『Mobile Link サーバー管理』を参照してください。

- **[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、すべてのローがダウンロードされます。 次の項を参照してください。

  ○「スナップショットを使った同期」『Mobile Link サーバー管理』
  ○「スナップショットを使った同期をいつ行うか」『Mobile Link サーバー管理』

- **[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。 次の項を参照してください。

  ○「同期スクリプトの作成」『Mobile Link サーバー管理』
  ○「download_cursor スクリプト」『Mobile Link サーバー管理』
  ○「download_delete_cursor スクリプト」『Mobile Link サーバー管理』

♦ ダウンロードタイプの変更

1. Sybase Central の **[フォルダー]** ビューで、**[Mobile Link 12]**、Mobile Link プロジェクト名、**[同期モデル]** を展開してから、同期モデル名を選択します。
2. 右ウィンドウ枠で、**[マッピング]** タブを開きます。
3. **[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。
4. **[ダウンロードタイプ]** 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。
5. **[カスタム]** を選択した場合は、**[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。

削除の記録方法の変更

スナップショットダウンロードを使用している場合、リモートデータベース内のローは、スナップショットがダウンロードされる前にすべて削除されます。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、削除できるようにローを記録しておく必要があります。この処理は、シャドウテーブルを使用するか、論理削除によって行うことができます。

♦ 削除の記録方法の変更

1. Sybase Central の **[フォルダー]** ビューで、**[Mobile Link 12]**、Mobile Link プロジェクト名、**[同期モデル]** を展開してから、同期モデル名を選択します。
2. 右ウィンドウ枠で、**[マッピング]** タブを開きます。
3. **[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。
4. 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、**[削除]** カラムで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。
5. 削除のダウンロードを行う場合は、下ウィンドウ枠で **[削除のダウンロード]** タブを開きます。

   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。

論理削除

Mobile Link 同期モデルでの論理削除のサポートは、論理削除カラムが統合データベースのみにあり、リモートデータベースにはないことを前提としています。統合スキーマを新しいリモートスキーマにコピーする場合、同期モデル設定の論理削除カラムに対応するすべてのカラムを除外します。

デフォルトの論理削除カラム名と一致するカラムは、新しいリモートスキーマに自動的にコピーされません。

リモートデータベースで論理削除を使用しない場合は、削除をダウンロードしないことを選択し、必要に応じて、論理削除カラムが含まれるようにリモートスキーマを更新します。

注意
リモートスキーマでの論理削除カラムのカラムマッピングを、統合スキーマでの論理削除カラムに設定してください。

参照

- 「削除」『Mobile Link サーバー管理』
- 「download_delete_cursor スクリプト」『Mobile Link サーバー管理』
- 「download_cursor テーブルイベント」『Mobile Link サーバー管理』

ダウンロードサブセットの変更

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。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。

- **[ユーザー]** このオプションを選択すると、Mobile Link ユーザー名によってデータが分割され、登録済みの Mobile Link ユーザー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 ユーザー名が統合データベースに存在している必要があります。Mobile Link ユーザー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 ユーザー名用に使用するカラムは、ユーザー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 ユーザー名がある場合は、そのテーブルにジョインする必要があります。

- **[リモート ID]** 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 が統合データベースに存在し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。

  注意
  リモート ID は、リモートコンピューター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、ユーザー別、または認証パラメーター別に分割することをおすすめします。

- **[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ー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。

♦ ダウンロードサブセットの変更

1. Sybase Central の **[フォルダー]** ビューで、**[Mobile Link 12]**、Mobile Link プロジェクト名、**[同期モデル]** を展開してから、同期モデル名を選択します。
2. 右ウィンドウ枠で、**[マッピング]** タブを開きます。
3. **[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。
4. **[サブセットのダウンロード]** ドロップダウンリストで、**[なし]**、**[ユーザー]**、**[リモート]**、**[カスタム]** のいずれかのダウンロードサブセットを選択します。
5. **[ユーザー]**、**[リモート]**、**[カスタム]** を選択する場合は、下ウィンドウ枠で **[サブセットのダウンロード]** タブを開きます。

6. **[ユーザー]** または **[リモート]** を選択した場合は、**[サブセットのダウンロード]** タブで、同期テーブルまたはジョインしたテーブルのいずれかの、Mobile Link ユーザー名またはリモート ID を含むカラムを特定できます。ジョインしたテーブルの場合は、ジョイン条件のカラムを指定する必要があります。

7. **[カスタム]** を選択した場合、**[サブセットのダウンロード]** タブに 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。

   - download_cursor にその他のテーブルへのジョインが必要な場合は、最初のテキストボックス (**[ダウンロードカーソルの FROM 句に追加するテーブル]**) 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。
   - 2 つ目のボックス (**[ダウンロードカーソルの WHERE 句で使用する 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。認証パラメーターを含む Mobile Link 名前付きパラメーター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。

参照

- 「Mobile Link ユーザー」『Mobile Link クライアント管理』
- 「リモート ID」『Mobile Link クライアント管理』
- 「リモートデータベース間でのローの分割」『Mobile Link サーバー管理』
- 「スクリプトでのリモート ID と Mobile Link ユーザー名」『Mobile Link クライアント管理』
- 「スクリプトでの最終ダウンロード時刻」『Mobile Link サーバー管理』
- 「download_cursor スクリプト」『Mobile Link サーバー管理』

例 (ユーザー)

たとえば、CustDB の ULOrder テーブルは、ユーザー間で共有できます。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、マネージャーは、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、このような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。マネージャーは、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。

どのように処理されたか確認するには、まず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。**[ダウンロードタイプ]** カラムで **[タイムスタンプ]**、**[サブセットのダウンロード]** カラムで **[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。ULOrder の **[サブセットのダウンロード]** カラムを **[ユーザー]** に変更します。下ウィンドウ枠で **[サブセットのダウンロード]** タブを開きます。**[ジョインされている関係テーブル内のカラムを使用する]** を選択します。ジョインするテーブルで、ULEmpCust を選択します。一致させるカラムで、emp_id を選択します。ジョイン条件には emp_id = emp_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"."emp_id" = "DBA"."ULEmpCust"."emp_id"
AND "DBA"."ULEmpCust"."emp_id" = {ml s.username}」
```

例 (カスタム)

たとえば、Mobile Link ユーザーによる Customer というテーブルのダウンロードをサブセットし、active=1 であるローのみをダウンロードするとします。Mobile Link ユーザー名はサブセットするテーブルに存在しないため、ユーザー名が含まれる 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 テーブルマッピングの **[ダウンロードタイプ]** で **[タイムスタンプ]**、**[サブセットのダウンロード]** で **[カスタム]** をそれぞれクリックします。下ウィンドウ枠で **[サブセットのダウンロード]** タブを開きます。最初のテキストボックス (**[ダウンロードカーソルの FROM 句に追加するテーブル]**) に、次のように入力します。

```
SalesRep
```

2 つ目のテキストボックス (**[ダウンロードカーソルの WHERE 句で使用する 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
```

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。

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 **[同じ]** 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、**[なし]** 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、シャドウテーブルに必要なカラムがあることと、式が統合テーブルを明示的に参照しないことを確認する必要があります。下側のウィンドウ枠にある **[ダウンロード削除サブセット]** タブを開いて、カスタムダウンロードサブセット SQL 式で使用されている非プライマリキーカラムがあれば、シャドウテーブルに追加します。

カスタムダウンロード削除サブセットは、削除の追跡にシャドウテーブルを使用している場合にシャドウテーブル用に追加のカラムも選択できることを除くと、カスタムダウンロードサブセットの場合と同様に設定できます。

参照

- 「ダウンロードサブセットの変更」34 ページ

競合の検出と解決の変更

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。

- **[競合検出を実行しない]**　このオプションを選択すると、競合は検出されません。アップロードされた更新は、競合を確認しないで適用されます。これにより、統合データベースから現在のローの値をフェッチする必要がなくなるため、更新の同期が高速になることがあります。
- **[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。「upload_fetch または upload_fetch_column_conflict スクリプトによる競合の検出」『Mobile Link サーバー管理』を参照してください。

- **[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。 「upload_fetch または upload_fetch_column_conflict スクリプトによる競合の検出」『Mobile Link サーバー管理』を参照してください。

  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。

  このオプションは、カラムベースの競合検出だけに使用してください。それ以外の場合に使用すると、競合検出を選択しない方が、同じ結果でパフォーマンスがよくなります。

- **[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。この処理は、ウィザード終了後に **[イベント]** タブで行います。

  「resolve_conflict スクリプトによる競合の解決」『Mobile Link サーバー管理』を参照してください。

♦ 競合の検出と解決のカスタマイズ

1. Sybase Central の **[フォルダー]** ビューで、**[Mobile Link 12]**、Mobile Link プロジェクト名、**[同期モデル]** を展開してから、同期モデル名を選択します。
2. 右ウィンドウ枠で、**[マッピング]** タブを開きます。
3. **[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。
4. **[競合の検出]** ドロップダウンリストで、**[なし]**、**[ローベース]**、**[カラムベース]** のいずれかをクリックします。**[なし]** を選択した場合は、ここで完了です。
5. **[ローベース]** または **[カラムベース]** を選択した場合は、**[競合の解決]** ドロップダウンリストから **[統合]**、**[リモート]**、**[タイムスタンプ]**、**[カスタム]** のいずれかを選択します。
6. **[タイムスタンプ]** を選択した場合は、下ウィンドウ枠の **[競合の解決]** タブを開き、使用する TIMESTAMP カラムの名前を入力します。

7. **[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。

参照

- 「競合の解決」『Mobile Link サーバー管理』

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。

- **[同期モデル作成ウィザード]** で生成されたスクリプトを表示し、変更する。
- 新しいスクリプトを作成する。

[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、このスクリプトの生成方法が **[同期モデル作成ウィザード]** による生成、ユーザー定義、生成されたスクリプトの上書きのうちどれであるかも表示されます。また、同期論理が 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。

生成スクリプトを上書きするスクリプトを追加すると、このスクリプトを完全に制御できるようになるため、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。

♦ 特定のテーブルのスクリプトの検索

1. Sybase Central の **[フォルダー]** ビューで、**[Mobile Link 12]**、Mobile Link プロジェクト名、**[同期モデル]** を展開してから、同期モデル名を選択します。
2. 右ウィンドウ枠で、**[テーブルマッピング]** タブを開きます。
3. **[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。
4. **[ファイル]** » **[イベントに移動]** をクリックします。

   [イベント] タブが開き、目的のテーブルのスクリプトがすべて表示されます。
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。

   カーソルがそのスクリプトに移動し、必要な変更を加えることができます。

既存のスクリプトは太字で強調表示されます。

外部サーバーに対する認証

♦ POP3、IMAP、LDAP の各外部サーバーの認証

[認証] タブを使用すると、同期モデルの外部サーバーに対して同期認証が有効になります。

1. Sybase Central の **[フォルダー]** ビューで、**[Mobile Link 12]**、Mobile Link プロジェクト名、**[同期モデル]** を展開してから、同期モデル名を選択します。
2. 右ウィンドウ枠で、**[認証]** タブを開きます。
3. **[この同期モデルのカスタム認証を有効にする]** を選択します。
4. 認証するサーバーを選択します。
5. 適切なホスト、ポート、URL 情報をそれぞれのフィールドに入力します。

   これらのフィールドの詳細については、「外部認証識別符号プロパティ」『Mobile Link クライアント管理』を参照してください。

同期モデルでのサーバー起動同期の設定

サーバー起動同期を使用すると、統合データベースで変更が行われたときに、クライアントデータベースで同期を起動できます。サーバー起動同期を同期モデルで有効にできます。この方法では、サーバー起動同期の設定と実行を簡単に行うことができますが、バージョンが限定されています。

サーバー起動同期を同期モデルで有効にすると、Mobile Link サーバーでテーブルの download_cursor スクリプトを使用して、同期の起動時期が判断されます。このことを行うには、download_cursor スクリプトを使用して、Notifier に request_cursor が生成されます。この方法で request_cursor をカスタマイズすることはできません。

♦ 同期モデルでのサーバー起動同期の設定

1. Sybase Central の **[フォルダー]** ビューで、**[Mobile Link 12]**、Mobile Link プロジェクト名、**[同期モデル]** を展開してから、同期モデル名を選択します。
2. 右ウィンドウ枠で、**[通知]** タブを開きます。
3. **[サーバー起動同期を有効にする]** を選択します。
4. 通知に使用する統合データベースのテーブルを選択します。

   このテーブル内のデータが変更されると、リモートデータベースに通知が送信されます。通知により、同期がトリガーされます。

注意
この目的のためには、タイムスタンプベースのダウンロードカーソルを定義したテーブルだけを選択できます (デフォルト)。通知はダウンロードカーソルの内容に基づきます。

5. ポーリング間隔を選択します。

   これは、ポーリングからポーリングまでの時間です。事前に定義されたポーリング間隔を選択するか、独自の間隔を指定できます。デフォルトは 30 秒です。

   データベースが切断された場合、Notifier はデータベースが再び使用可能になった後に、この最初のポーリング間隔で自動的にリカバリを行います。

6. Notifier のデータベース接続の独立性レベルを変更します。(省略可)

   デフォルトは、コミットされた読み出しです。独立性レベルの設定結果に注意してください。レベルが高くなると競合が増加し、パフォーマンスが逆に低下することがあります。独立性レベル 0 に設定すると、コミットされていないデータ (最終的にロールバックされるデータ) を読み込むことができます。

参照

- Mobile Link サーバー起動同期
- 「サーバー起動同期のクイックスタート」『Mobile Link サーバー起動同期』

スキーマの更新

スキーマ更新ウィザードでは、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できます。

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。

♦ 同期モデルのスキーマの更新

1. Sybase Central の **[フォルダー]** ビューで、**[Mobile Link 12]**、Mobile Link プロジェクト名、**[同期モデル]** を展開してから、同期モデル名を選択します。
2. **[ファイル]** » **[スキーマの更新]** をクリックします。
3. 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。

4. **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。

5. **[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。

参照

- 「テーブルマッピングとカラムマッピングの変更」30 ページ

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 サーバーと Mobile Link クライアントを実行するためのバッチファイル
- サーバーによって開始される同期の設定

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、トリガー、ストアドプロシージャーを作成することで、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。

♦ SQL ファイルからの統合データベースの展開

注意
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。

- **同期モデル展開ウィザード**の実行時に、(**[統合データベースの展開先]** 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 *consolidated* サブフォルダー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、トリガーなど、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 ユーザー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、*consolidated* ディレクトリに移動し、*_consolidated.bat* または *_consolidated.sh* で終わる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。

  ```
  MyModel_consolidated.bat "DSN=my_odbc_datasource;UID=myuserid;PWD=mypassword"
  ```

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

リモートデータベースの配備

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。

♦ 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**の実行時に、(**[新しい SQL Anywhere リモートデータベース]** ページまたは **[新しい Ultra Light リモートデータベース]** 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、*remote* 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 ユーザーなど、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、*remote* ディレクトリに移動し、*_remote.bat* または *_remote.sh* で終わるファイルを実行します。たとえば、次のように入力します。

  ```
  MyModel_remote.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。

同期ツールを実行するバッチファイルの配備

ウィザードでは、次のバッチファイルを作成できます。

- 指定するオプションで Mobile Link サーバー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。
- 同期モデルにサーバー起動同期を設定する場合は、**同期モデル展開ウィザード**で Notifier と Mobile Link Listener を実行するバッチファイルも作成できます。

♦ モデルの展開

1. Sybase Central の **[フォルダー]** ビューで、**[Mobile Link 12]**、Mobile Link プロジェクト名、**[同期モデル]** を展開してから、同期モデル名を選択します。
2. **[ファイル]** » **[展開]** をクリックします。
3. **同期モデル展開ウィザード**の指示に従います。
4. 完了したら、選択した変更内容が展開されます。同じ名前のファイルがすでにあった場合は、上書きされます。
5. アプリケーションを同期するには、「展開されたモデル同期」46 ページを参照してください。

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ー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 Sybase 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。Sybase 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する必要があります。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。

「スキーマの更新」42 ページを参照してください。

注意
同期モデルを再展開すると、シャドウテーブルの再作成が削除されます。シャドウテーブルデータを削除しないようにするには、ファイルを展開し、シャドウテーブルを作成しないように SQL ファイルを編集します。

展開されたモデル同期

モデルを展開するときは、**[同期モデル作成ウィザード]** の最初のページで選択したロケーションにディレクトリやファイルがオプションで作成されます。このファイルやディレクトリは、このときに選択したモデル名に従って名前が付けられます。

モデルの名前を **MyModel** にし、*c:¥SyncModels* に保存したとします。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。

ディレクトリ (この例の名前とロケーションに基づく)	説明と内容 (この例の名前に基づく)
c:¥SyncModels	*MyModel.mlsm* という名前で保存されたモデルファイルがあります。
c:¥SyncModels¥MyModel	展開ファイルを含むフォルダーがあります。
c:¥SyncModels¥MyModel¥consolidated	統合データベース用の展開ファイルがあります。 ● *MyModel_consolidated.sql* - 統合データベースの設定に使用する SQL ファイル ● *MyModel_consolidated.bat* - SQL ファイルを実行するためのバッチファイル
c:¥SyncModels¥MyModel¥mlsrv	Mobile Link サーバー用の展開ファイルがあります。 ● *MyModel_mlsrv.bat* - Mobile Link サーバーを実行するためのバッチファイル。サーバー起動同期を設定した場合は、Notifier も起動されます。

ディレクトリ (この例の名前とロケーションに基づく)	説明と内容 (この例の名前に基づく)
c:¥SyncModels¥MyModel¥remote	リモートデータベース用の展開ファイルがあります。 ● *dblsn.txt* - サーバー起動同期を設定した場合の、Listener のオプション設定を含むテキストファイル。*MyModel_dblsn.bat* によって使用されます。 ● *MyModel_dblsn.bat* - サーバー起動同期を設定した場合の、Mobile Link Listener を実行するためのバッチファイル ● *MyModel_dbmlsync.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 ● *MyModel_remote.bat* - *MyModel_remote.sql* を実行するためのバッチファイル ● *MyModel_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 ● *MyModel_remote.sql* - 新しい SQL Anywhere リモートデータベースを設定するための SQL ファイル ● *MyModel_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 ● *MyModel_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

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るバッチファイルは、コマンドラインから実行する必要があります。多くの場合、このときに接続情報を指定する必要があります。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があります。

「ODBC データソース」『SQL Anywhere サーバー データベース管理』を参照してください。

♦ バッチファイルを使用した同期モデルの同期

1. 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。

   「統合データベースの設定」『Mobile Link サーバー管理』を参照してください。

2. **同期モデル展開ウィザード**の実行時に、(**[統合データベースの展開先]** 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 *consolidated* サブフォル

ダー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、トリガーなど、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 ユーザー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、*consolidated* ディレクトリに移動し、*_consolidated.bat* または *_consolidated.sh* で終わる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。

```
MyModel_consolidated.bat "DSN=MY_ODBC_DATASOURCE"
```

3. **同期モデル展開ウィザード**を実行したときに、後で実行するためのファイルを作成する場合 (**[新しい SQL Anywhere リモートデータベース]** ページまたは **[新しい Ultra Light リモートデータベース]** ページ) は、*remote* ディレクトリにあるバッチファイルを実行します。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 ユーザーなど、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、*remote* ディレクトリに移動し、*_remote.bat* または *_remote.sh* で終わるファイルを実行します。たとえば、次のように入力します。

   ```
   MyModel_remote.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。

4. *mlsrv¥MyModel_mlsrv.bat* を実行して Mobile Link サーバーを起動します。サーバー起動同期を設定した場合は、このファイルによって Notifier も起動されます。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。

   ```
   MyModel_mlsrv.bat "DSN=MY_ODBC_DATASOURCE"
   ```

5. 同期を実行します。

   SQL Anywhere リモートデータベースの場合

   - DBA 以外のユーザー (推奨) に REMOTE DBA 権限を付与します。たとえば、Interactive SQL で次のコマンドを実行します。

     ```
     GRANT REMOTE DBA
     TO userid, IDENTIFIED BY password
     ```

   - REMOTE DBA 権限を持つユーザーで接続します。
   - *remote* 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。

     ```
     dbeng12 MyModel_remote.db
     ```

   - SQL Anywhere Mobile Link クライアント dbmlsync を起動します。*remote* ディレクトリに存在する *_dbmlsync.bat* で終わる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。

     ```
     MyModel_dbmlsync.bat "UID=DBA;PWD=sql;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。

6. サーバー起動同期を設定した場合は、最初の同期を行ってから、Mobile Link Listener を起動します。最初の同期は、リモート ID ファイルを作成するために必要です。Mobile Link Listener を起動するには、*remote* ディレクトリに存在する *_dblsn.bat* で終わるファイルを実行します。たとえば、次のように入力します。

```
MyModel_dblsn.bat
```

参照

- 「GRANT REMOTE DBA 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「dbmlsync のパーミッション」『Mobile Link クライアント管理』

同期モデルの制限事項

次に、同期モデルの制限事項をいくつか示します。

- **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。「スクリプトバージョン」『Mobile Link サーバー管理』を参照してください。

- **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。「Mobile Link システムデータベース」『Mobile Link サーバー管理』を参照してください。

- **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。「パブリケーション」『Mobile Link クライアント管理』を参照してください。

- **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ー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ー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する。

○リモートスキーマから計算カラムを除外する。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。

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 ドライバー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、ドライバー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、ドライバー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

を追加する必要があります。**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ーを使用して、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。 「sa_dependent_views システムプロシージャー」『SQL Anywhere サーバー SQL リファレンス』を参照してください。

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。

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。この項で説明すること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。

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、*%SQLANYSAMP12%¥MobiLink¥CustDB¥project.mlp* ディレクトリにあります。このプロジェクトを Sybase 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。同期論理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る。
- **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る。
- **ローのサブセット**　ULOrder テーブルから、リモートユーザーごとに異なるローセットを取得する。

  ローのサブセットの詳細については、「リモートデータベース間でのローの分割」『Mobile Link サーバー管理』を参照してください。
- **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。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る。

  「タイムスタンプベースのダウンロードの実装」『Mobile Link サーバー管理』を参照してください。
- **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。ULProduct テーブルは、この方法で同期される。

  「スナップショットを使った同期」『Mobile Link サーバー管理』を参照してください。
- **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る。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 つである。

  「プライマリキープール」『Mobile Link サーバー管理』を参照してください。

  プライマリキーをユニークにする他の方法については、「ユニークなプライマリキー」『Mobile Link サーバー管理』を参照してください。

参照

- 「チュートリアル: Ultra Light CustDB サンプルアプリケーションの構築」『Ultra Light データベース管理とリファレンス』
- 「チュートリアル: Ultra Light CustDB サンプルアプリケーションの構築」『Ultra Light データベース管理とリファレンス』
- 「CustDB サンプルデータベースアプリケーション」『SQL Anywhere 12 紹介』
- 「CustDB 統合データベースの設定」 53 ページ

CustDB の設定

この項では、CustDB サンプルアプリケーションとサンプルデータベースを作成するコードを部分的に説明します。これらのコードを以下に示します。

●サンプル SQL スクリプト。*%SQLANYSAMP12%¥MobiLink¥CustDB* にあります。

●アプリケーションのコード。*%SQLANYSAMP12%¥UltraLite¥CustDB* にあります。

●プラットフォーム固有のユーザーインターフェイスのコード。*%SQLANYSAMP12%¥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。

CustDB 統合データベースの設定

MobileLink 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 12 CustDB

SQL Anywhere 12 CustDB 統合データベースは、*%SQLANYSAMP12%¥UltraLite¥CustDB¥custdb.db* にあります。SQL Anywhere 12 CustDB という DNS はインストール環境に含まれています。

このデータベースは、*%SQLANYSAMP12%¥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、*%SQLANYSAMP12%¥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、*%SQLANYSAMP12%¥MobiLink¥CustDB* にあります。

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

統合データベースとして使用するデータベースを準備する方法の詳細については、「統合データベースの設定」『Mobile Link サーバー管理』を参照してください。

♦ 統合データベースの設定 (Adaptive Server Enterprise、MySQL、Oracle、SQL Server の場合)

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

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 Anywhere 12 インストール環境の *MobiLink¥setup* サブフォルダーにあります。

   ●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。

   ●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、サンプルユーザーテーブル、ストアドプロシージャー、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、*%SQLANYSAMP12%¥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。

   ●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ー上で、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 Anywhere 12]** » **[管理ツール]** » **[ODBC データソースアドミニストレーター]** 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ーを選択します。

      [完了] 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 **[ユーザー ID]** と **[パスワード]** を入力します。

♦ 統合データベースの設定 (DB2 LUW の場合)

1. DB2 LUW サーバー上に **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. 1 つ以上のバッファープールに 8 KB ページがあることを確認します。ない場合は、8 KB ページのバッファープールを作成してください。

b. 8 KB ページのある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。

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. 以下のように、*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 のユーザー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. サーバーまたはクライアントコンピューターで、DB2 LUW コマンドウィンドウを開きます。次のコマンドを入力して *syncdb2.sql* を実行します。

```
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ー、Mobile Link 同期スクリプトを CustDB データベースに追加します。

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、ユーザー名とパスワードを追加できます。*userid* と *password* を、使用するユーザー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. サーバーまたはクライアントコンピューターで、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。

```
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ーを起動します。

[スタート] » [プログラム] » [SQL Anywhere 12] » [管理ツール] » [ODBC データソースアドミニストレーター] を選択します。

[ODBC データソースアドミニストレーター] が表示されます。

b. **[ユーザー 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 ドライバーを選択します。たとえば、IBM DB2 UDB ODBC ドライバーを選択します。**[完了]** をクリックします。

ODBC ドライバー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
- http://www.ianywhere.jp/tech/odbc_mobilink.html

参照

- 「IBM DB2 LUW 統合データベース」『Mobile Link サーバー管理』

Ultra Light リモートデータベースの設定

以下の手順では、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 *%SQLANYSAMP12%¥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。

- **Embedded SQL の論理**　ファイル *custdb.sqc* には、Ultra Light データベースの情報を問い合わせて修正するための SQL 文と、統合データベースとの同期を開始するために必要な呼び出しが格納されています。
- **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。
- **ユーザーインターフェイス機能**　この機能は、*Samples¥UltraLite¥CustDB* のプラットフォーム固有のサブフォルダーに、別々に格納されています。

♦ サンプルアプリケーションのリモートデバイスへのインストールと同期

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 サーバーを起動します。
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ー ID とパスワードです。

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

2. Mobile Link サーバーを起動します。

   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。

```
mlsrv12 -c "DSN=CustDB" -zp
```

3. CustDB サンプルアプリケーションを起動します。

   a. **[スタート] » [プログラム] » [SQL Anywhere 12] » [Mobile Link] » [同期サーバーのサンプル]** をクリックします。

   b. 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。

   [File] » [Synchronize] 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。

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、*%SQLANYSAMP12%¥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。

CustDB テーブルの ER (実体関連) 図については、「CustDB サンプルデータベースアプリケーション」『SQL Anywhere 12 紹介』を参照してください。

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、統合データベースにのみ存在します。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、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、同期ユーザーに属するローを 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 人のユーザー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ーが同時に保持することがあります。

@@spid の詳細については、「変数」『SQL Anywhere サーバー SQL リファレンス』を参照してください。

CustDB サンプル内のユーザー

CustDB サンプルのユーザーには、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、ユーザー 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、ユーザー 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

同期論理のソースコード

Sybase 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_add_table_script(
'CustDB 12.0',
```



```
'ULProduct', 'download_cursor',
'SELECT prod_id, price, prod_name FROM ULProduct' )
go
```

CustDB サンプルの注文の同期

ビジネスルール

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る。

●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る。

●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る。

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ーは、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 番目のパラメーターは従業員 ID です。この ID は、ダウンロードするローを判断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 プロシージャーを次に示します。

  ```
  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 プロシージャーを次に示します。

```
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
```


CustDB サンプルの顧客の同期

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る。

●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る。

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、ユーザー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}
```

CustDB サンプルの製品の同期

ビジネスルール

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。

「スナップショットを使った同期」『Mobile Link サーバー管理』を参照してください。

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。

●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る。

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
```

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。

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

- **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_insert**　CustDB の upload_insert スクリプトを次に示します。

```
INSERT INTO ULCustomerIDPool ( pool_cust_id )
  VALUES( {ml r.pool_cust_id} )
```

- **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}
```

- **end_upload**　次の end_upload スクリプトは、各アップロードの完了後に 20 個の顧客 ID が顧客 ID プールに残るように処理を行います。

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

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

```
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
```

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

- **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

- **end_upload**　次の end_upload スクリプトは、各アップロードの完了後に 20 個の注文 ID が注文 ID プールに残るように処理を行います。

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

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

```
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
```

- **upload_insert**　CustDB の upload_insert スクリプトを次に示します。

```
INSERT INTO ULOrderIDPool ( pool_order_id )
  VALUES( {ml r.pool_order_id}
```

- **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}
```

CustDB データベースのリストア

サンプルをリストアするには、*%SQLANYSAMP12%¥UltraLite¥CustDB* ディレクトリから次のコマンドを実行します。

```
makedbs
```

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。この項で説明すること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 *%SQLANYSAMP12%¥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る。
- **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る。「リモートデータベース間でのローの分割」『Mobile Link サーバー管理』を参照してください。
- **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る。「タイムスタンプベースのダウンロードの実装」『Mobile Link サーバー管理』を参照してください。

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。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、一部のデータ、同期スクリプト、Mobile Link ユーザー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、サブフォルダー *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 ユーザー名、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。

♦ Contact サンプルの構築

1. コマンドプロンプトで、*%SQLANYSAMP12%¥MobiLink¥Contact* に移動します。
2. *build.bat* (Windows) または *build.sh* (UNIX) を実行します。

Contact サンプルの実行

♦ 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 サーバーと dbmlsync のコマンドラインが例示されています。*%SQLANYSAMP12%¥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ーで確認できます。

- *step1.bat*
- *step2.bat*
- *step3.bat*

1. Mobile Link サーバーを起動します。
   a. コマンドプロンプトで、*%SQLANYSAMP12%¥MobiLink¥Contact* に移動します。
   b. 次のコマンドを実行します。

      ```
      step1
      ```

      このコマンドは、Mobile Link サーバー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、*%SQLANYSAMP12%¥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 サーバーを停止します。
   a. コマンドプロンプトで、*%SQLANYSAMP12%¥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 サーバーを停止させるバッチファイルです。

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。

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、次の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ーにあります。

- *MobiLink¥Contact¥build_consol.sql*
- *MobiLink¥Contact¥build_remote.sql*

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 ユーザー名を保持する 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、ユーザー 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、*%SQLANYSAMP12%¥MobiLink ¥Contact¥mlmaint.sql* ファイルにあります。

Contact サンプル内のユーザー

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

データベースユーザー 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 人のユーザーはどちらも、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 ユーザー 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、ユーザー ID **sync_user** (パスワードは **sync_user**) もあります。このユーザー ID は、dbmlsync コマンドラインでのみ使用します。**sync_user** は REMOTE DBA 権限を持っているので、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** というユーザーが存在します。このユーザーは 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** ユーザーは接続権限を持っていません。テーブルを個々のユーザー 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、Sybase 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 ユーザー名

Mobile Link ユーザー名は、データベースユーザー 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ー ID の他に、Mobile Link ユーザー名があります。Samuel Singer の Mobile Link ユーザー名は SSinger です。Pamela Savarino の Mobile Link ユーザー名は PSavarino です。Mobile Link ユーザー名は、次のロケーションで格納または使用されています。

- リモートデータベース。Mobile Link ユーザー名が、CREATE SYNCHRONIZATION USER 文を使用して追加されています。

●統合データベース。Mobile Link ユーザー名とパスワードが、mluser ユーティリティを使用して追加されています。

●*MobiLink¥Contact¥step2.bat* 内の dbmlsync コマンドライン。同期時に、接続ユーザーの Mobile Link パスワードが指定されます。

●Mobile Link サーバー。同期時、Mobile Link ユーザー名がパラメーター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 ユーザー名パラメーターを比較します。

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。

「スナップショットを使った同期」『Mobile Link サーバー管理』を参照してください。

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ない。

●営業担当者の Mobile Link ユーザー名と rep_id 値を変更しない。

●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 ユーザー名に対応するローが 1 つだけ存在する。

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ー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ーは、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、パラメーターを使用するために指定されたダミー式です。2 番目のパラメーターは Mobile Link ユーザー名です。

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。

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。

次の項を参照してください。

- 「タイムスタンプベースのダウンロードの実装」『Mobile Link サーバー管理』
- 「リモートデータベース間でのローの分割」『Mobile Link サーバー管理』

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る。
- 営業担当者間で、顧客の再割り当てを定期的に変更できる。このプロセスは、一般に領域の再編成と呼ばれる。
- 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る。

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 = ?
```

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ーもあります。

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る。

●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る。

●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する。

トリガー

Customer テーブルのトリガーは、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。トリガーは、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、トリガー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 = ?
  ```

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ーです。

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 に変化します。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 に変化します。

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'
```


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ー mlmaint となっています。

Mobile Link チュートリアル

この項には、Mobile Link テクノロジの設定方法と使用方法を示すチュートリアルがあります。チュートリアルは、新しいユーザー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。http://www.sybase.com/detail?id=1081144 を参照してください。

注意
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。図やプロシージャーのいくつかは SQL Anywhere 12.0.1 とは異なります。

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 モニター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。Sybase 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。

必要なソフトウェア

- SQL Anywhere 12

前提知識と経験

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

概要

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
- Sybase 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 サーバーの起動
- ログファイルと Mobile Link モニターを使用した同期のモニター

参照

- 「Interactive SQL」『SQL Anywhere サーバー データベース管理』
- 「Sybase Central」『SQL Anywhere サーバー データベース管理』
- 「Mobile Link 同期」1 ページ
- 「同期スクリプトの作成」『Mobile Link サーバー管理』
- 「同期の方法」『Mobile Link サーバー管理』
- 「同期イベント」『Mobile Link サーバー管理』
- 「Mobile Link モニター」『Mobile Link サーバー管理』
- 「競合の解決」『Mobile Link サーバー管理』
- http://www.sybase.com/detail?id=1058600#319 (このページを表示するには、Sybase.com アカウントが必要です。)
- sybase.public.sqlanywhere.mobilink

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。

♦ 統合データベースの設定

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Sybase Central]** をクリックします。
2. **[ツール]** » **[SQL Anywhere 12]** » **[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. **[このコンピューター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:¥MLintro¥MLconsolidated.db* と入力します。**[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。**[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 データベースが作成されます。
8. **[データベースの作成]** ウィンドウで **[閉じる]** をクリックします。
9. **[ツール]** » **[SQL Anywhere 12]** » **[ODBC アドミニストレーターを開く]** をクリックします。
10. **[ユーザー 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 Anywhere 12]** 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 **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、デフォルトの **[データベース]** のままにして、ユーザー ID とパスワードを使用して接続します。
    e. **[ユーザー 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 **sql** と入力します。
    g. **[アクション]** ドロップダウンリストで、デフォルトの **[このコンピューターで稼働しているデータベースに接続]** のままにします。
    h. **[サーバー名]** フィールドに **MLconsolidated** と入力します。
    i. **[OK]** をクリックします。
13. **[OK]** をクリックして **[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. 「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」81 ページに進みます。

参照

- 「Mobile Link 統合データベース」『Mobile Link サーバー管理』
- 「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバー データベース管理』

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

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」80 ページを参照してください。

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。**Product** テーブルには次のカラムがあります。

カラム	説明
name	製品の名前です。
quantity	品目の販売数です。
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ーする一般的な方法です。

♦ Product テーブルの作成と移植

1. Interactive SQL で統合データベースに接続します。

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

- Sybase 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。
- 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=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,
   last_modified   TIMESTAMP DEFAULT TIMESTAMP
);
```

Interactive SQL によって、統合データベースに **Product** テーブルが作成されます。

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

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かどうかを確認します。

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. 「レッスン 3：Mobile Link プロジェクトと同期モデルの作成」82 ページに進みます。

参照

- 「Interactive SQL」『SQL Anywhere サーバー データベース管理』
- 「CREATE TABLE 文」『SQL Anywhere サーバー SQL リファレンス』

レッスン 3：Mobile Link プロジェクトと同期モデルの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」80 ページを参照してください。

このレッスンでは、**[同期モデル作成ウィザード]** を使用して新しい同期モデルを作成します。

モデルを作成する前に、**[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する必要があります。**[プロジェクト作成ウィザード]** から **[同期モデル作成ウィザード]** にアクセスできます。

♦ 新しい Mobile Link プロジェクトと同期モデルの作成

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Sybase Central]** をクリックします。
2. **[ツール]** » **[Mobile Link 12]** » **[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 **mlintro_project** と入力します。
4. **[ロケーション]** フィールドに *C:¥MLintro* と入力し、**[次へ]** をクリックします。
5. **[統合データベースをプロジェクトに追加]** オプションを選択します。
6. **[データベースの表示名]** フィールドに **mlintro_consdb** と入力します。
7. **[編集]** をクリックします。
8. **[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. **[ユーザー 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 **mlintro_consdb** を選択します。
   d. **[OK]** をクリックし、**[保存]** をクリックします。
9. **[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。
10. **[新しいモデルを作成する]** を選択して **[次へ]** をクリックします。
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. **[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 **My Application 1.0** と入力し、**[完了]** をクリックします。
13. **[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、**[OK]** をクリックします。

    [同期モデル作成ウィザード] が表示されます。
14. **[新しい同期モデルの名前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。
15. リストから **mlintro_consdb** 統合データベースを選択し、**[次へ]** をクリックします。

16. **[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]** をクリックし、**[次へ]** をクリックします。

17. **[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** テーブルだけが選択されていることを確認して、**[次へ]** をクリックします。

18. **[タイムスタンプベースのダウンロード]** をクリックし、**[次へ]** をクリックします。

    タイムスタンプベースのダウンロードでは、前回のダウンロード以降に更新されたデータのみが転送されるため、データ量を最小限に抑えることができます。

19. **[タイムスタンプダウンロードのオプション]** ページで、**[シャドウテーブルを使用してタイムスタンプカラムを保持する]** をクリックし、**[次へ]** をクリックします。

    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。

20. **[削除のダウンロード]** ページで、次のタスクを実行します。
    a. **[統合データベース上で削除されたデータを、リモートデータベース上で削除しますか ?]** オプションに **[はい]** をクリックします。
    b. **[シャドウテーブルを使用して削除を記録する]** をクリックします。
       シャドウテーブルが統合データベースに作成され、同期可能な削除が実装されます。
    c. **[次へ]** をクリックします。

21. **[はい、各リモートデータベースに同じデータをダウンロードします]** をクリックし、**[次へ]** をクリックします。

22. **[競合検出を実行しない]** をクリックし、**[次へ]** をクリックします。

23. **[パブリケーション、スクリプトバージョン、説明]** ページで、次のタスクを実行します。
    a. **[パブリケーションの名前を指定してください。]** フィールドに **sync_mlintro_publication** と入力します。
    b. **[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 **sync_mlintro_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 サーバー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 サーバー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. **[完了]** をクリックします。

24. 「レッスン 4：同期モデルの展開」 85 ページに進みます。

参照

- 「同期モデル」27 ページ
- 「統合データベースの設定」『Mobile Link サーバー管理』
- 「Mobile Link サーバーのシステムテーブル」『Mobile Link サーバー管理』
- 「Mobile Link サーバーシステムプロシージャー」『Mobile Link サーバー管理』
- 「download_delete_cursor スクリプト」『Mobile Link サーバー管理』
- 「競合の解決」『Mobile Link サーバー管理』
- 「競合解決」『Mobile Link サーバー管理』
- 「パブリケーション」『Mobile Link クライアント管理』
- 「同期モデルタスク」30 ページ
- 「ダウンロードタイプの変更」33 ページ
- 「競合の検出と解決の変更」38 ページ
- 「テーブルマッピングとカラムマッピングの変更」30 ページ

レッスン 4：同期モデルの展開

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」80 ページを参照してください。

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、2 つのリモートデータベースを作成して展開します。

♦ 2 つのリモートデバイスへの同期モデルの展開

1. Sybase Central の左ウィンドウ枠の **[Mobile Link 12]** で、**mlintro_project**、**[同期モデル]**、**sync_mlintro** の順に展開します。
2. **[ファイル]** » **[展開]** をクリックします。
3. **[次の 1 つまたは複数の項目の展開の詳細を指定する]** オプションのデフォルト設定を使用して、**[次へ]** をクリックします。
4. **[統合データベースの展開先]** ページで、次のタスクを実行します。
   a. **[次の SQL ファイルに変更を保存する]** を選択し、ファイルのデフォルトロケーションをそのまま使用します。

      Mobile Link により *.sql* ファイルが生成され、同期設定が可能になるように統合データベースが変更されます。*.sql* ファイルを後で確認して、独自に変更を行うこともできます。この場合、*.sql* ファイルを実行する必要があります。
   b. 統合データベースに変更をすぐに適用します。

      [統合データベースに接続して変更を直接適用する] を選択します。
   c. リストから **mlintro_consdb** 統合データベースを選択します。
   d. **[次へ]** をクリックします。

   consolidated ディレクトリを作成するかどうかを確認するプロンプトが表示されます。**[はい]** をクリックします。

5. **[新しい SQL Anywhere データベース]** をクリックし、**[次へ]** をクリックします。
6. **[新しい SQL Anywhere リモートデータベース]** ページで、次のタスクを実行します。
   a. **[データベースの作成コマンドが含まれたコマンドファイルと SQL ファイルを作成する]** オプションを選択します。

      このオプションを選択すると、別の *.sql* ファイルが生成されます。このファイルに含まれるコマンドにより、リモートデータベースにすべてのスキーマと同期情報が設定されます。
   b. **[SQL ファイル]** フィールドで、デフォルトロケーションをそのまま使用します。
   c. **[リモートの SQL Anywhere データベースを作成する]** オプションを選択します。
   d. **[SQL Anywhere データベースファイル]** フィールドで、デフォルトロケーションをそのまま使用します。
   e. **[次へ]** をクリックします。

   remote ディレクトリを作成するかどうかを確認するプロンプトが表示されます。**[はい]** をクリックします。
7. **[完了]** をクリックします。

   mlsrv ディレクトリを作成するかどうかを確認するメッセージが表示されます。**[はい]** をクリックします。
8. **[展開しています]** ウィンドウで **[閉じる]** をクリックします。

   統合データベースには、多くのリモートクライアントとの同期のための設定がすべて行われました。リモートデータベースが作成され、展開されました。
9. Sybase Central の左ウィンドウ枠の **[Mobile Link 12]** で、**mlintro_project**、**[同期モデル]**、**sync_mlintro** の順に展開します。
10. **[ファイル]** » **[展開]** をクリックします。
11. **[次の 1 つまたは複数の項目の展開の詳細を指定する]** をクリックし、**[リモートデータベースと同期クライアント]** と **[リモートデータベースを中央管理する設定で初期化する]** のオプションだけが選択されていることを確認します。**[次へ]** をクリックします。
12. **[新しい SQL Anywhere データベース]** をクリックし、**[次へ]** をクリックします。
13. **[新しい SQL Anywhere リモートデータベース]** ページで、次のタスクを実行します。
   a. **[データベースの作成コマンドが含まれたコマンドファイルと SQL ファイルを作成する]** オプションを選択します。

      このオプションを選択すると、別の *.sql* ファイルが生成されます。このファイルに含まれるコマンドにより、リモートデータベースにすべてのスキーマと同期情報が設定されます。
   b. **[SQL ファイル]** フィールドのファイルロケーションを *C:¥MLintro¥mlintro_project¥sync_mlintro¥remote¥sync_mlintro_remote2.sql* に変更します。

c. **[リモートの SQL Anywhere データベースを作成する]** オプションを選択します。

d. **[SQL Anywhere データベースファイル]** フィールドのファイルロケーションを *C:¥MLintro¥mlintro_project¥sync_mlintro¥remote¥sync_mlintro_remote2.db* に変更します。

e. **[次へ]** をクリックします。

14. **[Mobile Link ユーザーおよび同期プロファイル]** ページで **[完了]** をクリックします。

15. **[展開しています]** ウィンドウで **[閉じる]** をクリックします。

2 番目のリモートデータベースを正常に作成し、展開しました。

16. 「レッスン 5：Mobile Link サーバーの起動」 87 ページに進みます。

参照

- 「同期モデルの展開」 43 ページ
- 「リモートデータベースの作成」『Mobile Link クライアント管理』

レッスン 5：Mobile Link サーバーの起動

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」 80 ページを参照してください。

このレッスンでは、mlsrv12 -c オプションを使って Mobile Link サーバーを起動し、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 サーバーの動作を設定します。

♦ Mobile Link サーバーの起動

1. コマンドプロンプトで、*c:¥MLintro¥* 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。

```
mlsrv12 -c "DSN=mlintro_consdb" -o mlsrv.mls -v+ -dl -zf -zu+ -x tcpip
```

Mobile Link サーバー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 サーバー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。 Mobile Link サーバーオプションの完全なリストについては、「Mobile Link サーバーオプション」『Mobile Link サーバー管理』を参照してください。

オプション	説明
-c	統合データベースに接続する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。

オプション	説明
-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。
-zf	各同期の始めに Mobile Link サーバーがスクリプトの変更をチェックします。
-dl	メッセージウィンドウ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ーを追加します。
-x	どのポートで受信するか、どのネットワークプロトコルで Mobile Link クライアントからの通信を受けるかを示すために、Mobile Link サーバーの通信プロトコルを設定します。TCP/IP のデフォルトポートは 2439 です。

注意
-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 サーバー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 ユーザーを同期環境に自動的に追加します。

3. 「レッスン 6：Mobile Link クライアントの起動」88 ページに進みます。

レッスン 6：Mobile Link クライアントの起動

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」80 ページを参照してください。

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 サーバーが同一のコンピューター上に存在することを前提としています。

dbeng12 コマンドラインユーティリティを使って、リモートデータベースを起動します。

♦ Mobile Link クライアントデータベースの起動

1. コマンドプロンプトで、*c:¥MLintro¥mlintro_project¥sync_mlintro¥remote* 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。

   ```
   dbeng12 sync_mlintro_remote
   ```

3. 次のコマンドを実行して、**sync_mlintro_remote2** データベースを起動します。

   ```
   dbeng12 sync_mlintro_remote2
   ```


4. 「レッスン 7：Mobile Link モニターの起動」89 ページに進みます。

レッスン 7：Mobile Link モニターの起動

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」80 ページを参照してください。

このレッスンでは、同期の発生を監視するために、Mobile Link モニターを開始して設定します。

Mobile Link モニターは、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 モニター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 モニター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モニターを使用することをおすすめします。

♦ 更新の競合を検出するための Mobile Link モニターの設定

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Mobile Link モニター]** をクリックします。

   [Mobile Link サーバーへの接続] ウィンドウが表示されます。

2. Mobile Link サーバーに Mobile Link モニターを接続します。

   [ユーザー] フィールドに **monitor_user** と入力し、**[OK]** をクリックします。

   前のセッションで Mobile Link サーバーを -zu+ オプション付きで起動したので、このユーザーは自動的に追加されています。

3. 「レッスン 8：同期」89 ページに進みます。

参照

- 「Mobile Link モニター」『Mobile Link サーバー管理』

レッスン 8：同期

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」80 ページを参照してください。

このレッスンでは、Mobile Link クライアント上で 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させて、Mobile Link サーバーに接続します。

♦ 同期クライアントの起動

1. コマンドプロンプトで、*c:¥MLintro¥mlintro_project¥sync_mlintro¥remote* 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。

```
dbmlsync -c "SERVER=sync_mlintro_remote;UID=DBA;PWD=sql" -o rem1.dbs -v+
```

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 *rem1.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。

次の表は、dbmlsync ユーティリティの各オプションを説明しています。

オプション	説明
-c	リモートデータベースに接続する文字列を指定します。
-o	メッセージログファイルを指定します。
-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。

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. 次のコマンドを実行して、**sync_mlintro_remote2** データベースを同期します。

   ```
   dbmlsync -c "SERVER=sync_mlintro_remote2;UID=DBA;PWD=sql" -o rem2.dbs -v+
   ```

   sync_mlintro_remote2 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 *rem2.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。

5. 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。

6. チャートスクロールを一時的に停止します。

   [モニター] » **[チャートスクロールの一時停止]** をクリックします。

7. Mobile Link モニターで下部の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。

8. 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。

9. 「レッスン 9：Mobile Link サーバーログファイルビューアーを使用したエラーと警告の確認」91 ページに進みます。

参照

- 「Mobile Link SQL Anywhere クライアントユーティリティ (dbmlsync)」『Mobile Link クライアント管理』

レッスン 9：Mobile Link サーバーログファイルビューアーを使用したエラーと警告の確認

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」80 ページを参照してください。

テーブルの同期が完了したら、コマンドラインで指定して生成したメッセージログファイル、つまり *mlsrv.mls*、*rem1.dbs*、*rem2.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。

♦ Mobile Link サーバーログファイルと同期クライアントログファイルでのエラーの検出

1. Sybase Central で、**[ツール]** » **[Mobile Link 12]** » **[Mobile Link サーバーログファイルビューアー]** をクリックします。
2. ログファイルをテキストエディターで開きます。

   c:¥MLintro¥mlsrv.mls を検出して、**[開く]** をクリックします。

   [Mobile Link サーバーログファイルビューアー] 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。

   [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 **[同期]** ウィンドウ枠に表示されます。個別のエラーと警告は、**[詳細]** ウィンドウ枠に表示されます。
4. **[メッセージ]** タブをクリックすると、Mobile Link サーバー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 サーバーは現在、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、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. テキストエディターで、*rem1.mls* または *rem2.mls* などのクライアントログファイルを開きます。
7. このファイル内で文字列「ROLLBACK」を検索します。トランザクションがロールバックされている場合、エラーのためトランザクションが完了していません。
8. 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。
9. 「レッスン 10：競合の検出と解決のためのテーブルの作成」92 ページに進みます。

参照

- 「Mobile Link サーバーの動作のロギング」『Mobile Link サーバー管理』

例

次の例は、Mobile Link サーバーログを確認するときに発生する可能性のあるサンプルエラーメッセージを示します。

```
E. 2010-04-12 14:49:14. Mobile Link サーバーからのエラーコード : -10355
E. 2010-04-12 14:49:14. サーバーエラー : メッセージ : テーブル 'Product' には upload_update スクリプトがありませんテーブル名 : Product
```

このメッセージは、データをアップロードする前にエラーが発生したことを示します。統合データベースの **Product** テーブルに、必要な **upload_update** スクリプトが存在しないことを示しています。

レッスン 10：競合の検出と解決のためのテーブルの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」80 ページを参照してください。

競合は、統合データベースにローをアップロードしているときに発生する可能性があります。異なるリモートデータベースで 2 人のユーザーが同じローを修正した場合、Mobile Link サーバーに 2 つめのローが到着したときに競合が検出されます。同期スクリプトを使用すると、競合を検出して解決できます。

例 (在庫管理)

2 つのクライアントが 10 個までの在庫を同時に販売するとします。最初のクライアントは 3 個の在庫を販売し、クライアントデータベース **sync_mlintro_remote** で在庫を更新します。これにより、7 個の在庫が残っていることが示されます。このデータベースは、統合データベース **mlintro_consdb** と同期されます。2 番目のクライアントは 4 個の在庫を販売し、リモートデータベース **sync_mlintro_remote2** で在庫を更新します。これにより、6 個の在庫が残っていることが示されます。各クライアントが **mlintro_consdb** と同期しようとすると、在庫の値が変更されているため、競合が検出されます。

この競合をプログラムによって解決するには、次のロー値を取得する必要があります。

- 統合データベースにある現在の値。

  統合データベースの値は、**sync_mlintro_remote** の同期後には 7 になります。

- 新しいロー値が **sync_mlintro_remote2** データベースによってアップロードされます。

- **sync_mlintro_remote2** データベースが直前の同期の間に取得した古いローの値。

次のビジネス論理を使用することで、新しい在庫数を計算して競合を解決できます。

```
現在の統合 - (古いリモート - 新しいリモート)
```

この在庫のサンプルでは、次の値がビジネス論理に置き換えられます。

```
7 - (10-6) = 3
```


この式で、リモートデータベースの古い値からリモートデータベースの新しい値を引いて求めている値は、在庫数ではなく、2 番目のクライアントが販売した個数です。

♦ 競合解決テーブルの作成

競合を解決するために、統合データベースに追加のテーブルを作成します。これらのテーブルには、**Product_old** と **Product_new** という名前を付けます。その目的は、競合の発生時に問題のある値を一時的に格納することです。

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。

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

   - Sybase 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。
   - 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=mlintro_consdb"
     ```

2. 次の SQL 文を実行します。

   ```
   CREATE TABLE Product_old (
      name          VARCHAR(128) NOT NULL PRIMARY KEY,
      quantity      INTEGER,
      last_modified   TIMESTAMP DEFAULT TIMESTAMP
   );

   CREATE TABLE Product_new (
      name          VARCHAR(128) NOT NULL PRIMARY KEY,
      quantity      INTEGER,
      last_modified   TIMESTAMP DEFAULT TIMESTAMP
   );
   ```

3. 「レッスン 11：競合の検出と解決のためのスクリプトの作成」93 ページに進みます。

参照

- 「競合検出」『Mobile Link サーバー管理』
- 「競合の解決」『Mobile Link サーバー管理』

レッスン 11：競合の検出と解決のためのスクリプトの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：Mobile Link 統合データベースの設定」80 ページを参照してください。

このレッスンでは、次のスクリプトを追加して、競合の検出と解決を行います。

- **upload_fetch**　このスクリプトを使用して、統合データベースのテーブルからローをフェッチして、競合を検出します。
- **upload_update**　このスクリプトを使用して、リモートデータベースに挿入されたデータを統合データベースに適用する方法を確認します。更新の競合の検出には、upload_update の拡

張プロトタイプも使用できます。 「upload_update テーブルイベント」『Mobile Link サーバー管理』を参照してください。

- **upload_old_row_insert** このスクリプトは、リモートデータベースがその直前の同期で取得した古いロー値を処理するために使用できます。 「upload_old_row_insert テーブルイベント」『Mobile Link サーバー管理』を参照してください。
- **upload_new_row_insert** このスクリプトは、新しいロー値 (リモートデータベースで更新された値) を処理するために使用できます。「upload_new_row_insert テーブルイベント」『Mobile Link サーバー管理』を参照してください。
- **upload_delete** このスクリプトを使用して、リモートデータベースから削除されたローを処理します。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 サーバーを設定します。
- **resolve_conflict** 競合解決スクリプトは、競合の解決のためにビジネス論理を適用します。「resolve_conflict テーブルイベント」『Mobile Link サーバー管理』を参照してください。

♦ 競合の検出と解決のための同期スクリプトのインストール

1. 競合の検出と解決のためのスクリプトを実装します。

   次の SQL 文を実行します。

```
/* upload_fetch */
CALL ml_add_table_script( 'sync_mlintro_scriptversion', 'Product',
   'upload_fetch',
   'SELECT name, quantity FROM Product WHERE name = {ml r.name}' );

/* upload_update */
CALL ml_add_table_script( 'sync_mlintro_scriptversion', 'Product',
   'upload_update',
   'UPDATE Product
      SET quantity = {ml r.quantity}, last_modified = now()
      WHERE name = {ml r.name}' );

/* upload_old_row_insert */
CALL ml_add_table_script( 'sync_mlintro_scriptversion', 'Product',
   'upload_old_row_insert',
   'INSERT INTO Product_old (name,quantity,last_modified)
      VALUES ({ml r.name}, {ml r.quantity}, now())');

/* upload_new_row_insert */
CALL ml_add_table_script( 'sync_mlintro_scriptversion', 'Product',
   'upload_new_row_insert',
   'INSERT INTO Product_new (name,quantity,last_modified)
      VALUES ({ml r.name}, {ml r.quantity}, now())');

/* upload_delete */
CALL ml_add_table_script( 'sync_mlintro_scriptversion', 'Product',
   'upload_delete', '--{ml_ignore}');

/* resolve_conflict */
CALL ml_add_table_script( 'sync_mlintro_scriptversion', 'Product',
   'resolve_conflict',
   'DECLARE @product_name VARCHAR(128);
    DECLARE @old_rem_val INTEGER;
    DECLARE @new_rem_val INTEGER;
```



```
        DECLARE @curr_cons_val INTEGER;
        DECLARE @resolved_value INTEGER;

        // obtain the product name
        SELECT name INTO @product_name FROM Product_old;

        // obtain the old remote value
        SELECT quantity INTO @old_rem_val FROM Product_old;

        //obtain the new remote value
        SELECT quantity INTO @new_rem_val FROM Product_new;

        // obtain the current value in cons
        SELECT quantity INTO @curr_cons_val FROM Product WHERE name = @product_name;

        // determine the resolved value
        SET @resolved_value = @curr_cons_val- (@old_rem_val - @new_rem_val);

        // update cons with the resolved value
        UPDATE Product
           SET quantity = @resolved_value
              WHERE name = @product_name;

       // clear the old and new row tables
       DELETE FROM Product_new;
       DELETE FROM Product_old');

    COMMIT;
```

注意
このチュートリアルでは、-zf オプションを指定して Mobile Link サーバーを実行します。これにより、同期時に統合データベースに追加された新しいスクリプトを、サーバーで検出できるようになります。このオプションを指定しない場合は、新しいスクリプトを統合データベースに追加する前に Mobile Link サーバーを停止し、追加後にサーバーを再起動する必要があります。

2. 「レッスン 12：Mobile Link モニターを使用した競合スクリプトの検証」95 ページに進みます。

参照

- 「競合検出」『Mobile Link サーバー管理』
- 「競合の解決」『Mobile Link サーバー管理』

レッスン 12：Mobile Link モニターを使用した競合スクリプトの検証

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」80 ページを参照してください。

このレッスンでは、クライアントデータベース上の同じローを更新することで競合を生成し、前のレッスンで開始した Mobile Link モニターを使用して競合を検出します。

このシナリオでは、両方のクライアントは、在庫数 10 個の Screwmaster Drill から開始します。1 番目のクライアントは 3 個を販売し、リモートデータベースを更新してから、統合データベースと同期します。2 番目のクライアントは 4 個を販売し、リモートデータベースを更新してから、統合データベースと同期します。2 番目のクライアントは、統合データベースに 6 の値が入っていると想定しているので、1 番目の販売によって競合が発生します。この競合が発生したときに、Mobile Link サーバーはこの競合を解決します。

♦ 更新の競合の生成と Mobile Link モニターでの検出

1. チャートスクロールの一時的停止を解除します。

   [モニター] » **[チャートスクロールの一時停止]** をクリックします。

2. **sync_mlintro_remote** 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=sync_mlintro_remote;UID=DBA;PWD=sql"
   ```

3. Screwmaster Drill の在庫数を 7 個に更新します。

   次の SQL 文を実行します。

   ```
   UPDATE Product SET quantity = 7
      WHERE name ='Screwmaster Drill';

   COMMIT;
   ```

4. **sync_mlintro_remote** データベースと統合データベースを同期します。

   次のコマンドを実行します。

   ```
   dbmlsync -c "SERVER=sync_mlintro_remote;UID=DBA;PWD=sql" -v+
   ```

   統合データベースで、Screwmaster Drill の在庫数が 7 個に更新されます。

5. 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。

6. **sync_mlintro_remote2** 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=sync_mlintro_remote2;UID=DBA;PWD=sql"
   ```

7. Screwmaster Drill の在庫数を 6 個に更新します。

   ```
   UPDATE Product SET quantity = 6
      WHERE name ='Screwmaster Drill';

   COMMIT;
   ```

8. **sync_mlintro_remote2** データベースと統合データベースを同期します。

次のコマンドを実行します。

```
dbmlsync -c "SERVER=sync_mlintro_remote2;UID=DBA;PWD=sql" -v+
```

統合データベースで、Screwmaster Drill の在庫数が 3 個に更新されます。

9. 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。

10. Mobile Link モニターに切り替えて、同期の結果を確認します。

11. チャートスクロールを一時的に停止します。

    [モニター] » **[チャートスクロールの一時停止]** をクリックします。

12. Mobile Link モニターの **[概要]** ウィンドウ枠 (一番下のウィンドウ枠)、**[チャート]** ウィンドウ枠、**[使用率グラフ]** ウィンドウ枠、**[詳細テーブル]** を使用して、同期についての統計情報を確認します。

    a. Mobile Link モニターの **[概要]** ウィンドウ枠で同期を見つけます。更新の競合を発生させた **sync_mlintro_remote2** の同期は、赤で表示されます。

    b. **[チャート]** ウィンドウ枠で **sync_mlintro_remote2** の同期を表示するには、**[概要]** ウィンドウ枠内の同期オブジェクトをクリックしてドラッグします。

       conflict_detected ウォッチ用に指定したパターンが適用された状態で、同期オブジェクトが表示されます。

    c. 同期の詳細を確認するには、ズームツールを使用します。

       [表示] » **[ズームイン]** を選択します。

    d. 同期オブジェクトまたは **[詳細テーブル]** ウィンドウ枠の対応するローをダブルクリックして同期プロパティを表示してから、**[アップロード]** タブをクリックして競合更新の数を確認します。

13. 「レッスン 13：SQL Anywhere モニターでの Mobile Link リソースのモニター」97 ページに進みます。

参照

- 「競合の解決」『Mobile Link サーバー管理』
- 「Mobile Link の統計のプロパティ」『Mobile Link サーバー管理』

レッスン 13：SQL Anywhere モニターでの Mobile Link リソースのモニター

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」80 ページを参照してください。

このレッスンを使用して、Mobile Link サーバーと Mobile Link サーバー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、モニター Developer Edition を使用しています。

♦ Mobile Link リソースのモニター

1. モニターを起動します。次の手順では、モニター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。

   モニター 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 Anywhere 12]** » **[管理ツール]** » **[SQL Anywhere モニター]** をクリックします。

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

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

   注意
   ネットワークを経由してモニター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、モニターが実行されているコンピューターの名前です。

2. デフォルトの管理者ユーザーとしてモニターにログインします。

   [ユーザー名] 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。

   注意
   次の手順を実行するには、モニター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ーとオペレーターユーザーにはすべてのタスクを実行する権限がありません。

   ♦ モニターユーザータイプの確認

   1. モニターにログインします。
   2. **[ツール]** » **[ユーザー設定]** を選択し、**[ユーザーのタイプ]** 設定を確認します。

   「モニターのユーザー」『SQL Anywhere サーバー データベース管理』を参照してください。

3. Mobile Link サーバーリソースをモニターに追加するには、次の手順に従います。
   a. Mobile Link 同期サーバーのサンプルを起動します。**[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[Mobile Link]** » **[同期サーバーのサンプル]** をクリックします。
   b. モニターに管理者としてログインします。
   c. 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。
   d. **[リソース]** をクリックして、**[追加]** をクリックします。
   e. **[Mobile Link サーバー]** をクリックして **[次へ]** をクリックします。
   f. **[名前]** フィールドに **MobiLinkServerSample** と入力し、**[次へ]** をクリックします。

g. **[ホスト]** フィールドに **localhost** と入力し、**[次へ]** をクリックします。

h. 必要な認証情報が要求されたら、**[ユーザー ID]** フィールドに **monitor_user** などのユーザー名を入力し、**[パスワード]** フィールドに **sql** 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 サーバーにユーザーを作成するために使用されます。モニターでは、このユーザー ID とパスワードを保存し、Mobile Link サーバーへの接続とモニタリングに使用します。

i. **[作成]** をクリックします。

j. 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。

k. **[閉じる]** をクリックします。

l. **[閉じる]** をクリックします。

m. **[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 サーバーをモニタリングするための Mobile Link サーバー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 サーバー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 サーバーリソースを追加します。

i. 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 サーバーを起動します。

```
"C:¥Program Files¥SQL Anywhere 12¥Bin32¥mlsrv12.exe" -vcrs -zu+ -c  "DSN=SQL Anywhere 12 CustDB;UID=ml_server;PWD=sql" -ot ml_tcpip.txt -zs ml_tcpip -x tcpip{port=8039}
```

ii. モニターに管理者としてログインします。

iii. **[ツール]** » **[管理]** をクリックします。

iv. **[リソース]** をクリックして、**[追加]** をクリックします。

v. **[Mobile Link サーバー]** をクリックして **[次へ]** をクリックします。

vi. **[名前]** フィールドに **ml_tcpip** と入力し、**[次へ]** をクリックします。

vii. **[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。

viii. 必要な認証情報が要求されたら、**[ユーザー ID]** フィールドに **monitor_user** などのユーザー名を入力し、**[パスワード]** フィールドに **sql** 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 サーバーにユーザーを作成するために使用されます。モニターでは、このユーザー ID とパスワードを保存し、Mobile Link サーバーへの接続とモニタリングに使用します。

ix. **[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。

x. **[閉じる]** をクリックします。

xi. **[閉じる]** をクリックします。

b. Mobile Link サーバーファームリソースを追加します。 Mobile Link サーバーファームの詳細については、「Mobile Link サーバーのサーバーファームでの実行」『Mobile Link サーバー管理』を参照してください。

i. **[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。

ii. **[リソース]** をクリックして、**[追加]** をクリックします。

iii. **[Mobile Link サーバーのファーム]** をクリックして **[次へ]** をクリックします。

iv. **[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。

v. **MobiLinkServerSample** と **ml_tcpip** をクリックして、**[作成]** をクリックします。

vi. **[閉じる]** をクリックします。

vii. **[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 サーバーリソースは **[リソースリスト]** に残ることに注意してください。

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 サーバー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]**、**[サーバー情報]** 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ー機能については、「チュートリアル : モニターによるリソースのモニタリング」『SQL Anywhere サーバー データベース管理』を参照してください。

6. 「クリーンアップ」 100 ページに進みます。

クリーンアップ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1 : Mobile Link 統合データベースの設定」 80 ページを参照してください。

♦ コンピューターからのチュートリアルの削除

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 モニター
- Mobile Link 用 SQL Anywhere モニター
- Sybase 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ーを起動します。
   b. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[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. **[ユーザー 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。

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。

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ある。
- 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 データベースがある。
- 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る。

必要なソフトウェア

- SQL Anywhere 12

概要

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する。
- 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する。
- **[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する。
- Sybase Central を使用して同期設定をカスタマイズする。

- **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する。
- リモートクライアントを統合データベースと同期する。

参照

- 「Mobile Link 同期」1 ページ

レッスン 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ー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ー。
SalesOrders	適切な地域にいる顧客の顧客 ID によるフィルター。
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 によるフィルター。

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ーされます。

注意
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する必要があります。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できるように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。

「レッスン 2：統合データベースの準備」103 ページに進みます。

レッスン 2：統合データベースの準備

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」102 ページを参照してください。

このレッスンでは、SQL Anywhere 統合データベースを設定する手順を説明します。

1. 統合データベースに接続します。
2. 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。

♦ 統合データベースの準備

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Sybase Central]** をクリックします。
2. **[接続]** » **[SQL Anywhere 12 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. **[ODBC データソース名]** フィールドに **SQL Anywhere 12 Demo** と入力します。
c. **[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 12 Demo"
   ```

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);
   ```

ユニークなプライマリキーの追加

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。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。

レッスンの後半では、統合スキーマからリモートスキーマを作成します。このため、リモートスキーマのプライマリキーは統合スキーマと同じものになります。

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。

「レッスン 3：同期モデルの作成」105 ページに進みます。

参照

- 「Mobile Link 統合データベース」『Mobile Link サーバー管理』
- 「ユニークなプライマリキー」『Mobile Link サーバー管理』

レッスン 3：同期モデルの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」102 ページを参照してください。

このレッスンでは、**[同期モデル作成ウィザード]** を使用して新しい同期モデルを作成します。

モデルを作成する前に、**[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する必要があります。**[プロジェクト作成ウィザード]** から **[同期モデル作成ウィザード]** にアクセスできます。

♦ 新しい Mobile Link プロジェクトと同期モデルの作成

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Sybase Central]** をクリックします。
2. **[ツール]** » **[Mobile Link 12]** » **[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、**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. **[統合データベースをプロジェクトに追加]** オプションを選択します。
6. **[データベースの表示名]** フィールドに **demo** と入力します。
7. **[編集]** をクリックします。
8. **[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. **[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 12 Demo** を選択します。
   b. **[OK]** をクリックし、**[保存]** をクリックします。
9. **[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。
10. **[新しいモデルを作成する]** を選択して **[次へ]** をクリックします。
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. **[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** と入力します。

13. **[このリモートスキーマ名を適用するデータベースのタイプを指定してください。]** オプションから **[Ultra Light]** を選択し、**[完了]** をクリックします。

14. 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。

15. **[OK]** をクリックします。

    [同期モデル作成ウィザード] が表示されます。

16. **[新しい同期モデルの名前を指定してください。]** フィールドに **sync_mlsqla** と入力し、**[次へ]** をクリックします。

17. **[プライマリキー要件]** ページで、3 つのチェックボックスをすべて選択します (レッスンの後半でユニークなプライマリキーを設定します)。**[次へ]** をクリックします。

18. リストから **demo** 統合データベースを選択し、**[次へ]** をクリックします。

19. **[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]** をクリックし、**[次へ]** をクリックします。

20. **[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。

    - Customers
    - CustomerProducts
    - Products
    - SalesOrders

21. **[次へ]** をクリックします。

22. **[タイムスタンプベースのダウンロード]** をクリックし、**[次へ]** をクリックします。

    タイムスタンプベースのダウンロードでは、前回のダウンロード以降に更新されたデータのみが転送されるため、データ量を最小限に抑えることができます。

23. **[タイムスタンプダウンロードのオプション]** ページで、**[シャドウテーブルを使用してタイムスタンプカラムを保持する]** をクリックし、**[次へ]** をクリックします。

    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。

24. **[削除のダウンロード]** ページで、次のタスクを実行します。

    a. **[統合データベース上で削除されたデータを、リモートデータベース上で削除しますか ?]** オプションに **[はい]** をクリックします。

    b. **[シャドウテーブルを使用して削除を記録する]** をクリックします。

       シャドウテーブルが統合データベースに作成され、同期が必要な削除が実装されます。

c. **[次へ]** をクリックします。

25. **[はい、各リモートデータベースに同じデータをダウンロードします]** をクリックします。

26. **[次へ]** をクリックします。

27. **[競合検出を実行しない]** をクリックし、**[次へ]** をクリックします。

28. **[パブリケーション、スクリプトバージョン、説明]** ページで、次のタスクを実行します。

    a. **[パブリケーションの名前を指定してください。]** フィールドに **sync_mlsqla_publication** と入力します。

    b. **[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 **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 サーバー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 サーバー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. **[完了]** をクリックします。

29. Sybase Central で、**[ビュー]** » **[フォルダー]** をクリックします。

30. **[Mobile Link 12]** の左ウィンドウ枠で、**mlsqla_project**、**[同期モデル]**、**sync_mlsqla** の順に展開します。

31. Sybase 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. 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"
INNER JOIN "GROUPO"."Customers_mod" ON "GROUPO"."Customers"."ID" =
   "GROUPO"."Customers_mod"."ID"
WHERE Region = 'Eastern';
```

    c. CustomerProducts 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顧客製品だけをダウンロードするようにします。

CustomerProductss 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';
```

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");
```

32. 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。

33. 「レッスン 4：同期モデルの展開」 108 ページに進みます。

参照

- 「同期モデル」 27 ページ
- 「統合データベースの設定」『Mobile Link サーバー管理』
- 「Mobile Link サーバーのシステムテーブル」『Mobile Link サーバー管理』
- 「Mobile Link サーバーシステムプロシージャー」『Mobile Link サーバー管理』
- 「download_delete_cursor スクリプト」『Mobile Link サーバー管理』
- 「パブリケーション」『Mobile Link クライアント管理』
- 「同期モデルタスク」 30 ページ
- 「ダウンロードタイプの変更」 33 ページ
- 「テーブルマッピングとカラムマッピングの変更」 30 ページ

レッスン 4：同期モデルの展開

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」 102 ページを参照してください。

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。

♦ 同期モデルの展開

1. Sybase Central の左ウィンドウ枠の **[Mobile Link 12]** 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。

3. **[次の 1 つまたは複数の項目の展開の詳細を指定する]** をクリックし、**[統合データベース]**、**[リモートデータベースと同期クライアント]**、**[Mobile Link サーバー]** を選択します。**[次へ]** をクリックします。

4. **[統合データベースの展開先]** ページで、次のタスクを実行します。

   a. **[次の SQL ファイルに変更を保存する]** を選択し、ファイルのデフォルトロケーションをそのまま使用します。

      Mobile Link により *.sql* ファイルが生成され、同期設定が可能になるように統合データベースが変更されます。*.sql* ファイルを後で確認して、独自に変更を行うこともできます。この場合、*.sql* ファイルを実行する必要があります。

   b. 統合データベースに変更をすぐに適用します。

      [統合データベースに接続して変更を直接適用する] を選択します。

   c. リストから **demo** 統合データベースを選択します。

   d. **[次へ]** をクリックします。

   consolidated ディレクトリを作成するかどうかを確認するプロンプトが表示されます。**[はい]** をクリックします。

5. **[新しい Ultra Light データベース]** をクリックし、**[次へ]** をクリックします。

6. **[新しい Ultra Light リモートデータベース]** ページで、次のタスクを実行します。

   a. **[データベースの作成コマンドが含まれたコマンドファイルと SQL ファイルを作成する]** オプションを選択します。

      このオプションを選択すると、別の *.sql* ファイルが生成されます。このファイルに含まれるコマンドにより、リモートデータベースにすべてのスキーマと同期情報が設定されます。

   b. **[SQL ファイル]** フィールドで、デフォルトロケーションをそのまま使用します。

   c. **[リモートの Ultra Light データベースを作成する]** オプションを選択します。

      新しいリモートデータベースを生成して、このデータベースに対して *.sql* ファイルを実行する必要があります。このオプションを適用すると、*.sql* ファイルを後で確認し、独自の変更を加えることができます。

d. **[Ultra Light データベースファイル]** フィールドで、デフォルトロケーションをそのまま使用します。

e. **[次へ]** をクリックします。

remote ディレクトリを作成するかどうかを確認するプロンプトが表示されます。**[はい]** をクリックします。

7. **[Mobile Link ユーザーおよび同期プロファイル]** ページで、次のタスクを実行します。

   a. **[Mobile Link サーバーに接続するのに使用するユーザー名を指定してください。]** フィールドに **mlsqla_remote** と入力します。

   b. **[使用するパスワードを指定してください。]** フィールドに **mlsqla_pass** と入力します。

   c. 同期プロファイル名を **mlsqla_remote_syncprofile** に変更します。

   d. **[次へ]** をクリックします。

8. **[TCP/IP]** をクリックし、**[ポート]** フィールドに **2439** と入力します。**[次へ]** をクリックします。

9. **[ホスト]** フィールドに **localhost** と入力します。**[次へ]** をクリックします。

   または、コンピューターの名前または IP アドレス、使用する別のネットワークサーバーの名前または IP アドレス、その他のクライアントストリームオプションを指定することもできます。

10. **[クライアントストリームパラメーター]**、**[Mobile Link サーバーストリームパラメーター]**、**[Mobile Link サーバーの冗長性]** の各ページで **[次へ]** をクリックして、デフォルトの設定をすべてそのまま使用します。

11. **[Mobile Link サーバーの名前を指定してください。]** フィールドに **mlsqla_mlsrv** と入力します。**[次へ]** をクリックします。

    mlsrv ディレクトリを作成するかどうかを確認するメッセージが表示されます。**[はい]** をクリックします。

12. **[完了]** をクリックします。

13. **[閉じる]** をクリックします。

    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 ユーザーを作成して統合データベースと Mobile Link サーバー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。

14. 「レッスン 5：Mobile Link サーバーの起動」111 ページに進みます。

参照

- 「同期モデルの展開」 43 ページ
- 「リモートデータベースの作成」『Mobile Link クライアント管理』
- 「Mobile Link ユーザー」『Mobile Link クライアント管理』

レッスン 5：Mobile Link サーバーの起動

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」 102 ページを参照してください。

このレッスンでは、mlsrv12 -c オプションを使って Mobile Link サーバーを起動し、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 サーバーの動作を設定できます。

♦ Mobile Link サーバーの起動

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。

   ```
   mlsrv12 -c "DSN=SQL Anywhere 12 Demo" -o mlsrv.mls -v+ -dl -zf -zu+ -x tcpip
   ```

   Mobile Link サーバー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 サーバー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。 Mobile Link サーバーオプションの完全なリストについては、「Mobile Link サーバーオプション」『Mobile Link サーバー管理』を参照してください。

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 サーバーがスクリプトの変更をチェックします。

オプション	説明
-zu+	自動的に新しいユーザーを追加します。
-x	Mobile Link クライアントの通信プロトコルとパラメーターを設定します。

注意
-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ー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 ユーザーを同期環境に自動的に追加します。

3. 「レッスン 6：同期」 112 ページに進みます。

レッスン 6：同期

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」 102 ページを参照してください。

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 サーバーと同期します。

♦ 同期クライアントの起動

1. 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=c:¥mlsqla¥mlsqla_project¥sync_mlsqla¥remote¥sync_mlsqla_remote.udb"
"Publications=sync_mlsqla_publication;MobiLinkUid=mlsqla_remote;MobiLinkPwd=mlsqla_pass;Scri
ptVersion=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。)
   - **MobiLinkUid**　Mobile Link サーバーによる認証に使用するユーザー名。
   - **MobiLinkPwd**　Mobile Link サーバー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 サーバー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 ユーザー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 サーバーが実行中であることを確認します。また、同期ログ (サーバー、クライアントとも) の内容を確認することもできます。

> **注意**
> 別のコンピューターにある ulsync アプリケーションを Mobile Link サーバーから実行している場合、Mobile Link サーバー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 サーバー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 Anywhere 12 プラグインを使用すると、Sybase 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. Sybase 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 12]** を右クリックし、**[接続]** をクリックします。
   b. **[ユーザー ID]** に **DBA** と入力し、**[パスワード]** に **sql** 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¥remote¥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 12]**、**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ー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。
6. 「クリーンアップ」 113 ページに進みます。

参照

- 「Ultra Light 同期ユーティリティ (ulsync)」『Ultra Light データベース管理とリファレンス』

クリーンアップ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」 102 ページを参照してください。

♦ コンピューターからのチュートリアルの削除

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。

- Sybase Central
- Interactive SQL

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. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[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. **[ユーザーデータソース]** リストで **mlsqla_db** をクリックし、**[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 *C:¥mlsqla* ディレクトリを削除します。
4. 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 "%SQLANYSAMP12%¥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。

チュートリアル：Oracle Database 10g 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 10g 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。

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 10g 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、サンプルスキーマを手動でインストールしたり、Oracle Database Configuration Assistant を使用してインストールすることもできます。これらのサンプルスキーマのインストールの詳細については、http://www.oracle.com/technology/obe/obe1013jdev/common/OBEConnection.htm を参照してください。

このチュートリアルでは、SYSDBA 権限がある SYS ユーザーとして 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 ユーザーのパスワードは Oracle データベースのインストール時に設定されます。

必要なソフトウェア

- SQL Anywhere 12
- Oracle Database 10g Release 2 以降

概要

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する。
- 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する。
- Mobile Link を Oracle Database 10g に接続する ODBC データソースを作成する。
- **[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する。
- Sybase Central を使用して同期モデルをカスタマイズする。
- **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する。
- リモートクライアントを統合データベースと同期する。

参照

- 「Mobile Link 同期」1 ページ
- 「Sybase Central の Mobile Link プラグイン」22 ページ

レッスン 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。ただし、各販売担当者に関する情報を取得する場合は、HR スキーマの EMPLOYEES テーブルを参照する必要があります。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。

テーブル	説明
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。
INVENTORIES	リモートでは使用しない。
ORDER_ITEMS	sales_rep_id を基準にフィルター処理。
ORDERS	すべての行を抽出。
PRODUCT_DESCRIPTIONS	リモートでは使用しない。
PRODUCT_INFORMATION	すべての行を抽出。

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ーされます。

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できる必要があります。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。

テーブル	同期
PRODUCT_INFORMATION	リモートデータベースへのダウンロードのみ。

「レッスン 2：統合データベースの準備」117 ページに進みます。

レッスン 2：統合データベースの準備

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」115 ページを参照してください。

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 のマニュアルか、http://www.oracle.com/technology/obe/obe1013jdev/common/OBEConnection.htm を参照してください。

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。ユーザー定義型として作成されたカラムは削除されます。これらのユーザー定義型を SQL Anywhere が認識できる型に変換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。次に、Mobile Link では OE のクレデンシャルを使用してトリガーを作成する必要があるため、トリガーを作成するためのパーミッションを OE ユーザーに付与する必要があります。

♦ 統合データベースの準備

1. SYSDBA 権限を持つ SYS ユーザーとして、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. ユーザー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 ユーザーのロックを解除し、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 ユーザーがトリガー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);
   ```

6. 「レッスン 3：ユニークなキーの追加」 118 ページに進みます。

参照

- 「Mobile Link 統合データベース」『Mobile Link サーバー管理』
- 「Oracle 統合データベース」『Mobile Link サーバー管理』
- 「ユニークなプライマリキー」『Mobile Link サーバー管理』

レッスン 3：ユニークなキーの追加

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」 115 ページを参照してください。

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。

♦ ユニークなプライマリキーの統合データベースへの追加

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);
```

これらの文を実行した後、Mobile Link サーバー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。

レッスンの後半では、統合スキーマからリモートスキーマを作成します。このため、リモートスキーマのプライマリキーは統合スキーマと同じものになります。

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。

「レッスン 4：Mobile Link の接続」119 ページに進みます。

レッスン 4：Mobile Link の接続

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：スキーマの設計」115 ページを参照してください。

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。

♦ 統合データベースへの Mobile Link の接続

1. ODBC データソースを作成します。

   SQL Anywhere 12 に付属の iAnywhere Solutions 12 - Oracle ODBC ドライバーを使用します。次の設定を使用してください。

ODBC タブのフィールド	値
[データソース名]	oracle_cons
[ユーザー ID]	OE
[パスワード]	sql
[TNS サービス名]	orcl
[プロシージャーは結果を返す]	選択
[配列サイズ]	60000

このチュートリアルでは、Oracle Database 10g 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。
3. 「レッスン 5：Mobile Link プロジェクトの作成」120 ページに進みます。

参照

- http://www.ianywhere.jp/tech/odbc_mobilink.html
- 「iAnywhere Solutions 12 - Oracle ODBC ドライバー」『Mobile Link サーバー管理』

レッスン 5：Mobile Link プロジェクトの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：スキーマの設計」115 ページを参照してください。

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 Mobile Link プロジェクトの作成

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Sybase Central]** をクリックします。
2. **[ツール]** » **[Mobile Link 12]** » **[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. **[ロケーション]** フィールドに *C:¥mlora* と入力し、**[次へ]** をクリックします。
5. **[統合データベースをプロジェクトに追加]** オプションを選択します。
6. **[データベースの表示名]** フィールドに **oracle_cons** と入力します。
7. **[編集]** をクリックします。
8. **[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. **[ユーザー 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 **sql** 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。

9. **[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。
10. **[新しいモデルを作成する]** オプションを選択し、**[次へ]** をクリックします。
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、**[完了]** をクリックします。
13. 「レッスン 6：同期モデルの作成と変更」121 ページに進みます。

レッスン 6：同期モデルの作成と変更

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」115 ページを参照してください。

このレッスンでは、統合データベースの同期モデルを作成します。**[同期モデル作成ウィザード]** 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の間の同期を順を追って設定できます。

♦ 同期モデルの作成と変更

1. **[ようこそ]** ページで、**[新しい同期モデルの名前を指定してください。]** フィールドに **sync_oracle** と入力し、**[次へ]** をクリックします。
2. **[プライマリキー要件]** ページで、3 つのチェックボックスをオンにします。**[次へ]** をクリックします。
3. リストから **oracle_cons** 統合データベースを選択し、**[次へ]** をクリックします。
4. **[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]** をクリックし、**[次へ]** をクリックします。
5. **[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。
   - CUSTOMERS
   - ORDERS
   - ORDER_ITEMS
   - PRODUCT_INFORMATION
6. **[次へ]** をクリックします。
7. **[タイムスタンプベースのダウンロード]** をクリックし、**[次へ]** をクリックします。

   タイムスタンプベースのダウンロードでは、前回のダウンロード以降に更新されたデータのみが転送されるため、データ量を最小限に抑えることができます。
8. **[タイムスタンプダウンロードのオプション]** ページで、**[シャドウテーブルを使用してタイムスタンプカラムを保持する]** をクリックし、**[次へ]** をクリックします。

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。

9. **[削除のダウンロード]** ページで、次のタスクを実行します。
   a. **[統合データベース上で削除されたデータを、リモートデータベース上で削除しますか ?]** オプションで、**[はい]** をクリックします。
   b. **[シャドウテーブルを使用して削除を記録する]** をクリックします。
      シャドウテーブルが統合データベースに作成され、同期が必要な削除が実装されます。
   c. **[次へ]** をクリックします。
10. **[はい、各リモートデータベースに同じデータをダウンロードします]** をクリックし、**[次へ]** をクリックします。

    同期モデルの編集時にカスタム論理を使用して、特定のデータをリモートデータベースにダウンロードする方法を指定します。
11. **[競合検出を実行しない]** をクリックし、**[次へ]** をクリックします。

    このチュートリアルでは競合検出を実行しないよう指定していますが、多くのアプリケーションでは競合検出が必要になります。
12. **[パブリケーション、スクリプトバージョン、説明]** ページで、次のタスクを実行します。
    a. **[パブリケーションの名前を指定してください。]** フィールドに **sync_oracle_publication** と入力します。
    b. **[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 **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 サーバー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 サーバー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. **[完了]** をクリックします。
13. Sybase Central で、**[ビュー]** » **[フォルダー]** をクリックします。
14. Sybase Central の左ウィンドウ枠の **[Mobile Link 12]** で、**oracle_project**、**[同期モデル]**、**sync_oracle** の順に展開します。
15. 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、**[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。

16. 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** ユーザー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。

17. 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ーします。

    a. **ORDERS** テーブルを含むローでは、**[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。

    b. 右ウィンドウ枠下部の **[サブセットのダウンロード]** タブをクリックします。

    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ーします。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. **[削除サブセット]** カラムを **[同じ]** から **[なし]** に変更します。

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。

19. 「レッスン 7：同期モデルの展開」 124 ページに進みます。

参照

- 「統合データベースの設定」『Mobile Link サーバー管理』
- 「Mobile Link サーバーのシステムテーブル」『Mobile Link サーバー管理』
- 「Mobile Link サーバーシステムプロシージャー」『Mobile Link サーバー管理』
- 「同期スクリプトの作成」『Mobile Link サーバー管理』
- 「download_delete_cursor スクリプト」『Mobile Link サーバー管理』
- 「競合の解決」『Mobile Link サーバー管理』
- 「競合解決」『Mobile Link サーバー管理』
- 「パブリケーション」『Mobile Link クライアント管理』
- 「同期イベント」『Mobile Link サーバー管理』
- 「同期モデルタスク」 30 ページ
- 「同期の方法」『Mobile Link サーバー管理』
- 「ダウンロードタイプの変更」 33 ページ
- 「競合の検出と解決の変更」 38 ページ
- 「テーブルマッピングとカラムマッピングの変更」 30 ページ

レッスン 7：同期モデルの展開

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：スキーマの設計」115 ページを参照してください。

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。

♦ 同期モデルの展開

1. Sybase Central の左ウィンドウ枠の **[Mobile Link 12]** 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. **[次の 1 つまたは複数の項目の展開の詳細を指定する]** をクリックし、**[統合データベース]**、**[リモートデータベースと同期クライアント]**、**[Mobile Link サーバー]** を選択します。**[次へ]** をクリックします。
4. **[統合データベースの展開先]** ページで、次のタスクを実行します。
   a. **[次の SQL ファイルに変更を保存する]** をクリックし、ファイルのデフォルトロケーションをそのまま使用します。

      Mobile Link により *.sql* ファイルが生成され、同期設定が可能になるように統合データベースが変更されます。*.sql* ファイルを後で確認して、独自に変更を行うこともできます。この場合、*.sql* ファイルを実行する必要があります。
   b. 統合データベースに変更をすぐに適用します。

      [統合データベースに接続して変更を直接適用する] をクリックします。
   c. リストから **oracle_cons** 統合データベースを選択します。
   d. **[次へ]** をクリックします。

   consolidated ディレクトリを作成するかどうかを確認するプロンプトが表示されます。**[はい]** をクリックします。
5. **[新しい SQL Anywhere データベース]** をクリックし、**[次へ]** をクリックします。
6. **[新しい SQL Anywhere リモートデータベース]** ページで、次のタスクを実行します。
   a. **[データベースの作成コマンドが含まれたコマンドファイルと SQL ファイルを作成する]** オプションを選択します。

      このオプションを選択すると、別の *.sql* ファイルが生成されます。このファイルに含まれるコマンドにより、リモートデータベースにすべてのスキーマと同期情報が設定されます。
   b. **[SQL ファイル]** フィールドで、デフォルトロケーションをそのまま使用します。
   c. **[リモートの SQL Anywhere データベースを作成する]** オプションをクリックします。

新しいリモートデータベースを生成して、このデータベースに対して *.sql* ファイルを実行する必要があります。このオプションを適用すると、*.sql* ファイルを後で確認し、独自の変更を加えることができます。

d. **[SQL Anywhere データベースファイル]** フィールドで、デフォルトロケーションをそのまま使用します。

e. **[次へ]** をクリックします。

remote ディレクトリを作成するかどうかを確認するプロンプトが表示されます。**[はい]** をクリックします。

7. **[Mobile Link ユーザーと同期プロファイル]** ページで、次のタスクを実行します。

   a. **[Mobile Link サーバーに接続するのに使用するユーザー名を指定してください。]** フィールドに **oracle_remote** と入力します。

   b. **[使用するパスワードを指定してください。]** フィールドに **oracle_pass** と入力します。

   c. **[次へ]** をクリックします。

8. **[TCP/IP]** を選択し、**[ポート]** フィールドに **2439** と入力します。**[次へ]** をクリックします。

9. **[ホスト]** フィールドに **localhost** と入力します。**[次へ]** をクリックします。

   または、コンピューターの名前または IP アドレス、使用する別のネットワークサーバーの名前または IP アドレス、その他のクライアントストリームオプションを指定することもできます。

10. **[クライアントストリームパラメーター]**、**[Mobile Link サーバーストリームパラメーター]**、**[Mobile Link サーバーの冗長性]** の各ページで **[次へ]** をクリックして、デフォルトの設定をすべてそのまま使用します。

11. **[Mobile Link サーバーの名前を指定してください。]** フィールドに **oracle_mlsrv** と入力します。**[次へ]** をクリックします。

    mlsrv ディレクトリを作成するかどうかを確認するメッセージが表示されます。**[はい]** をクリックします。

12. リモート同期クライアントの冗長性を選択し、リモートデータベースログファイルのデフォルトのファイル名を使用します。**[次へ]** をクリックします。

13. **[完了]** をクリックします。

14. **[閉じる]** をクリックします。

    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 ユーザーを作成して統合データベースと Mobile Link サーバーの展開を終了します。統合データベースと Mobile Link サーバー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。

15. 「レッスン 8：サーバーとクライアントの起動」126 ページに進みます。

参照

- 「同期モデルの展開」 43 ページ
- 「リモートデータベースの作成」『Mobile Link クライアント管理』
- 「Mobile Link ユーザー」『Mobile Link クライアント管理』

レッスン 8：サーバーとクライアントの起動

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」 115 ページを参照してください。

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。

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 サーバー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 サーバーが使用する 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。

♦ Mobile Link サーバーとクライアントの起動

1. SYSDBA 権限を持つ SYS ユーザーとして、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 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;
   ```

4. 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。(このフォルダーは、**[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。)

   デフォルトのディレクトリ名を使用している場合、ルートディレクトリには *sync_oracle¥mlsrv* ディレクトリがあります。

5. *mlsrv* ディレクトリから次のコマンドを実行します。

   ```
   sync_oracle_mlsrv.bat "DSN=oracle_cons;UID=OE;PWD=sql"
   ```

   - ***sync_oracle_mlsrv.bat*** Mobile Link サーバー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ー名。
- **PWD**　統合データベースへの接続に使用するパスワード。

このコマンドが正常に実行されると、Mobile Link サーバーメッセージウィンドウに [Mobile Link **サーバーが起動しました。**] というメッセージが表示されます。

Mobile Link サーバー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。

6. 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。

   デフォルトのディレクトリ名を使用している場合、ルートディレクトリには *sync_oracle¥remote* ディレクトリがあります。

7. 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。

   ```
   dbeng12 -n remote_eng sync_oracle_remote.db -n remote_db
   ```

   - **dbeng12**　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ー。
   - **remote_eng**　データベースサーバー名。
   - ***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。

   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ーが起動し、remote_db という名前のデータベースがロードされます。

8. 「レッスン 9：リモート ID の設定」127 ページに進みます。

参照

- 「SQL Anywhere データベースサーバーの構文」『SQL Anywhere サーバー データベース管理』
- 「展開されたモデル同期」46 ページ
- 「Mobile Link サーバーの実行」『Mobile Link サーバー管理』
- 「リモート ID」『Mobile Link クライアント管理』

レッスン 9：リモート ID の設定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：スキーマの設計」115 ページを参照してください。

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 サーバー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 は有効な販売担当者識別子の値に設定する必要があります。

リモートデバイスが最初に同期する際に、選択され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、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。

♦ リモート ID の有効な販売担当者識別子への設定

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 ユーザーとして、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=sql"
   "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ー名として remote_eng を指定。
   - **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ー名。
   - **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。

3. 「レッスン 10：同期」128 ページに進みます。

レッスン 10：同期

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：スキーマの設計」115 ページを参照してください。

このレッスンでは、リモートクライアントを初めて同期します。この作業は dbmlsync ユーティリティで行います。dbmlsync はリモートデータベースに接続して Mobile Link サーバー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。

♦ リモートクライアントの同期

1. 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ー名を指定。
- **DBN**　リモートデータベース名を指定。
- **UID**　リモートデータベースへの接続に使用するユーザー名を指定します。
- **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 サーバーによる認証に使用するユーザー名。
- **oracle_pass**　Mobile Link サーバーによる認証に使用するパスワード。

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 ユーザー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 サーバーが実行中であることを確認します。また、同期ログ (サーバー、クライアントとも) の内容を確認することもできます。

注意
別のコンピューターにある dbmlsync アプリケーションを Mobile Link サーバーから実行している場合、Mobile Link サーバーのロケーションを指定する引数を渡す必要があります。

2. 「レッスン 11：リモートデータベースのデータの表示」129 ページに進みます。

参照

- 「同期処理」14 ページ
- 「dbmlsync 構文」『Mobile Link クライアント管理』

レッスン 11：リモートデータベースのデータの表示

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：スキーマの設計」115 ページを参照してください。

Mobile Link サーバー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 Anywhere 12 プラグインを使用すると、Sybase 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。

♦ **リモートデータベースのデータの表示**

1. Sybase 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 Anywhere 12]** を右クリックして **[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、**[ユーザー ID]** に **DBA** と入力し、**[パスワード]** に **sql** と入力します。
   c. **[アクション]** ドロップダウンリストで **[このコンピューターで稼働しているデータベースに接続]** をクリックし、**[サーバー名]** 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. ORDERS テーブルをクリックして、右ウィンドウ枠の **[データ]** タブをクリックします。

   ORDERS テーブルでは、すべてのレコードが識別子 154 の販売担当者に関するものです。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ー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。
4. 「クリーンアップ」130 ページに進みます。

クリーンアップ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：スキーマの設計」115 ページを参照してください。

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ーから削除します。

♦ **チュートリアルの削除**

1. 注文エントリのデータベースを再生成します。

   oe_main.sql を実行して現在の OE スキーマを削除し、新しい OE スキーマをインストールします。*oe_main.sql* ファイルは *$ORACLE_HOME/demo/schema/order_entry* にあります。
2. 同期モデルを削除します。
   a. Sybase Central を起動します。
   b. 右ウィンドウ枠で、**[Mobile Link 12]** をダブルクリックします。
   c. **sync_oracle** モデルが右ウィンドウ枠に表示されます。
   d. **sync_oracle** を右クリックして、**[削除]** をクリックします。

e. **[削除の確認]** ウィンドウで、**[はい]** をクリックします。

3. リモートデータベースを消去します。

   dberase ユーティリティを使用します。次のコマンドを実行します。

   ```
   dberase sync_oracle¥remote¥sync_oracle_remote.db
   ```

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。

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ー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 サーバー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。また、オンライン (http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.ase_15.0.sqlug/html/sqlug/sqlug894.htm) でも参照できます。

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。Adaptive Server Enterprise での NULL のパスワード変更の詳細については、http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.ase_15.0.sag1/html/sag1/sag1615.htm を参照してください。

必要なソフトウェア

- SQL Anywhere 12
- Adaptive Server Enterprise 15.0

概要

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する。
- 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する。

- 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する。
- **[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する。
- Sybase Central を使用して同期設定をカスタマイズする。
- **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する。
- リモートクライアントを統合データベースと同期する。

参照

- 「Mobile Link 同期」 1 ページ

レッスン 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。
blurbs、publishers、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。
authors	すべての行を抽出。
discounts	stor_id を基準にフィルター処理。
sales	stor_id を基準にフィルター処理。
salesdetail	stor_id を基準にフィルター処理。
stores	stor_id を基準にフィルター処理。
titleauthor	すべての行を抽出。
titles	すべての行を抽出。
blurbs	リモートでは使用しない。
publishers	リモートでは使用しない。
roysched	リモートでは使用しない。

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ーされます。

注意
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する必要があります。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できるように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。

統合テーブル	同期
titleauthor	リモートデータベースへのダウンロードのみ。

統合テーブル	同期
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。
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。

「レッスン 2：統合データベースの準備」134 ページに進みます。

レッスン 2：統合データベースの準備

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」132 ページを参照してください。

このレッスンでは、Mobile Link 同期用の統合データベースのサイズを増やし、ユニークなプライマリキーを作成します。

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。

♦ 統合データベースの準備

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ー名を指定します。

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", 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
```

4. 「レッスン 3：ユニークなキーの追加」135 ページに進みます。

参照

- 「Adaptive Server Enterprise 統合データベース」『Mobile Link サーバー管理』

レッスン 3：ユニークなキーの追加

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」132 ページを参照してください。

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。

♦ ユニークなプライマリキーの統合データベースへの追加

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ー名を指定します。

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 サーバー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。

注意
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。

レッスンの後半では、統合スキーマからリモートスキーマを作成します。このため、リモートスキーマのプライマリキーは統合スキーマと同じものになります。

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。

5. 「レッスン 4：Mobile Link の接続」137 ページに進みます。

参照

- 「ユニークなプライマリキー」『Mobile Link サーバー管理』

レッスン 4：Mobile Link の接続

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」132 ページを参照してください。

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。

♦ 統合データベースへの Mobile Link の接続

1. ODBC データソースを作成します。

   Adaptive Server Enterprise に付属の ODBC ドライバー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**
データソース名	ase_cons
説明	
サーバー名（ASE ホスト名）	localhost
サーバーポート	5000
データベース名	pubs2
ログイン ID	sa
カーソルの使用	Off

[トランザクション] タブのフィールド	**値**
サーバー初期化トランザクション	Off

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。

   ODBC データソースを設定すると、Mobile Link 12 プラグインを使用して統合データベースに接続し、同期モデルを作成することができるようになります。

3. 「レッスン 5：Mobile Link プロジェクトの作成」138 ページに進みます。

参照

- http://www.ianywhere.jp/tech/odbc_mobilink.html

レッスン 5：Mobile Link プロジェクトの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」132 ページを参照してください。

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 Mobile Link プロジェクトの作成

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Sybase Central]** をクリックします。
2. **[ツール]** » **[Mobile Link 12]** » **[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 **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. **[統合データベースをプロジェクトに追加]** オプションを選択します。
6. **[データベースの表示名]** フィールドに **ase_cons** と入力します。
7. **[編集]** をクリックします。
8. **[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. **[ユーザー 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。
9. **[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。
10. **[新しいモデルを作成する]** オプションを選択し、**[次へ]** をクリックします。
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 **ase_remote_schema** と入力し、**[完了]** をクリックします。
13. 「レッスン 6：同期モデルの作成と変更」139 ページに進みます。

レッスン 6：同期モデルの作成と変更

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：スキーマの設計」132 ページを参照してください。

このレッスンでは、統合データベースの同期モデルを作成します。**[同期モデル作成ウィザード]** 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の間の同期を順を追って設定できます。

♦ 同期モデルの作成と変更

1. **[ようこそ]** ページで、**[新しい同期モデルの名前を指定してください。]** フィールドに **sync_ase** と入力し、**[次へ]** をクリックします。
2. **[プライマリキー要件]** ページで、3 つのチェックボックスをオンにします。**[次へ]** をクリックします。
3. リストから **ase_cons** 統合データベースを選択し、**[次へ]** をクリックします。
4. **[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]** をクリックし、**[次へ]** をクリックします。
5. **[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
6. **[次へ]** をクリックします。
7. **[タイムスタンプベースのダウンロード]** をクリックし、**[次へ]** をクリックします。

   タイムスタンプベースのダウンロードでは、前回のダウンロード以降に更新されたデータのみが転送されるため、データ量を最小限に抑えることができます。
8. **[タイムスタンプダウンロードのオプション]** ページで、**[シャドウテーブルを使用してタイムスタンプカラムを保持する]** をクリックし、**[次へ]** をクリックします。

   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。
9. **[削除のダウンロード]** ページで、次のタスクを実行します。
   a. **[統合データベース上で削除されたデータを、リモートデータベース上で削除しますか ?]** オプションで、**[はい]** をクリックします。

b. **[シャドウテーブルを使用して削除を記録する]** をクリックします。
シャドウテーブルが統合データベースに作成され、同期が必要な削除が実装されます。

c. **[次へ]** をクリックします。

10. **[はい、各リモートデータベースに同じデータをダウンロードします]** をクリックし、**[次へ]** をクリックします。

同期モデルの編集時にカスタム論理を使用して、特定のデータをリモートデータベースにダウンロードする方法を指定します。

11. **[競合検出を実行しない]** をクリックし、**[次へ]** をクリックします。

このチュートリアルでは競合検出を実行しないよう指定していますが、多くのアプリケーションでは競合検出が必要になります。

12. **[パブリケーション、スクリプトバージョン、説明]** ページで、次のタスクを実行します。

a. **[パブリケーションの名前を指定してください。]** フィールドに **sync_ase_publication** と入力します。

b. **[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 **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 サーバー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 サーバー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. **[完了]** をクリックします。

13. **[ビュー]** » **[フォルダー]** をクリックします。

14. Sybase Central の左ウィンドウ枠の **[Mobile Link 12]** で、**ase_project**、**[同期モデル]**、**sync_ase** の順に展開します。

15. 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、**[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。

16. 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ーします。

a. **stores** テーブルを含むローでは、**[サブセットのダウンロード]** を **[カスタム]** に変更します。

b. 右ウィンドウ枠下部の **[サブセットのダウンロード]** タブをクリックします。

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ーします。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。

17. **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。

> **注意**
> 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}
> ```

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。

19. 「レッスン 7：同期モデルの展開」 142 ページに進みます。

参照

- 「統合データベースの設定」『Mobile Link サーバー管理』
- 「Mobile Link サーバーのシステムテーブル」『Mobile Link サーバー管理』
- 「Mobile Link サーバーシステムプロシージャー」『Mobile Link サーバー管理』
- 「download_delete_cursor スクリプト」『Mobile Link サーバー管理』
- 「競合の解決」『Mobile Link サーバー管理』
- 「競合解決」『Mobile Link サーバー管理』
- 「パブリケーション」『Mobile Link クライアント管理』
- 「同期モデルタスク」30 ページ
- 「ダウンロードタイプの変更」33 ページ
- 「競合の検出と解決の変更」38 ページ
- 「テーブルマッピングとカラムマッピングの変更」30 ページ

レッスン 7：同期モデルの展開

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」132 ページを参照してください。

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。

♦ 同期モデルの展開

1. Sybase Central の左ウィンドウ枠の **[Mobile Link 12]** 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. **[次の 1 つまたは複数の項目の展開の詳細を指定する]** をクリックし、**[統合データベース]**、**[リモートデータベースと同期クライアント]**、**[Mobile Link サーバー]** を選択します。**[次へ]** をクリックします。
4. **[統合データベースの展開先]** ページで、次のタスクを実行します。
   a. **[次の SQL ファイルに変更を保存する]** を選択し、ファイルのデフォルトロケーションをそのまま使用します。

      Mobile Link により *.sql* ファイルが生成され、同期設定が可能になるように統合データベースが変更されます。*.sql* ファイルを後で確認して、独自に変更を行うこともできます。この場合、*.sql* ファイルを実行する必要があります。
   b. 統合データベースに変更をすぐに適用します。

      [統合データベースに接続して変更を直接適用する] を選択します。
   c. リストから **ase_cons** 統合データベースを選択します。
   d. **[次へ]** をクリックします。

   consolidated ディレクトリを作成するかどうかを確認するプロンプトが表示されます。**[はい]** をクリックします。

5. **[新しい SQL Anywhere データベース]** をクリックし、**[次へ]** をクリックします。
6. **[新しい SQL Anywhere リモートデータベース]** ページで、次のタスクを実行します。
   a. **[データベースの作成コマンドが含まれたコマンドファイルと SQL ファイルを作成する]** オプションを選択します。

      このオプションを選択すると、別の *.sql* ファイルが生成されます。このファイルに含まれるコマンドにより、リモートデータベースにすべての必要なスキーマと同期情報が設定されます。
   b. **[SQL ファイル]** フィールドで、デフォルトロケーションをそのまま使用します。
   c. **[リモートの SQL Anywhere データベースを作成する]** オプションを選択します。

      このオプションを選択しない場合は、新しいリモートデータベースを生成して、このデータベースに対して *.sql* ファイルを実行する必要があります。このオプションを適用すると、*.sql* ファイルを後で確認し、独自の変更を加えることができます。
   d. **[SQL Anywhere データベースファイル]** フィールドで、デフォルトロケーションをそのまま使用します。
   e. **[次へ]** をクリックします。

   remote ディレクトリを作成するかどうかを確認するプロンプトが表示されます。**[はい]** をクリックします。
7. **[Mobile Link ユーザーおよび同期プロファイル]** ページで、次のタスクを実行します。
   a. **[Mobile Link サーバーに接続するのに使用するユーザー名を指定してください。]** フィールドに **ase_remote** と入力します。
   b. **[使用するパスワードを指定してください。]** フィールドに **ase_pass** と入力します。
   c. **[次へ]** をクリックします。
8. **[TCP/IP]** を選択し、**[ポート]** フィールドに **2439** と入力します。**[次へ]** をクリックします。
9. **[ホスト]** フィールドに **localhost** と入力します。**[次へ]** をクリックします。

   または、コンピューターの名前または IP アドレス、使用する別のネットワークサーバーの名前または IP アドレス、その他のクライアントストリームオプションを指定することもできます。
10. **[クライアントストリームパラメーター]**、**[Mobile Link サーバーストリームパラメーター]**、**[Mobile Link サーバーの冗長性]** の各ページで **[次へ]** をクリックして、デフォルトの設定をすべてそのまま使用します。
11. **[Mobile Link サーバーの名前を指定してください。]** フィールドに **ase_mlsrv** と入力します。**[次へ]** をクリックします。

    mlsrv ディレクトリを作成するかどうかを確認するメッセージが表示されます。**[はい]** をクリックします。
12. リモート同期クライアントの冗長性を選択し、リモートデータベースログファイルのデフォルトのファイル名を使用します。**[次へ]** をクリックします。

13. **[完了]** をクリックします。

14. **[閉じる]** をクリックします。

    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 ユーザーを作成して統合データベースと Mobile Link サーバー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。

15. 「レッスン 8：サーバーとクライアントの起動」144 ページに進みます。

参照

- 「同期モデルの展開」43 ページ
- 「リモートデータベースの作成」『Mobile Link クライアント管理』
- 「Mobile Link ユーザー」『Mobile Link クライアント管理』

レッスン 8：サーバーとクライアントの起動

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」132 ページを参照してください。

このレッスンでは、Mobile Link サーバーとリモートデータベースを起動します。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。

♦ クライアントとサーバーの起動

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ーに移動します。(このフォルダーは、**[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、*sync_ase¥mlsrv* ディレクトリに移動します。

2. Mobile Link サーバーを起動するには、次のコマンドを実行します。

   ```
   sync_ase_mlsrv.bat "DSN=ase_cons;UID=sa;PWD=sa;"
   ```

   - ***sync_ase_mlsrv.bat*** Mobile Link サーバー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ー名 (Adaptive Server Enterprise の場合、デフォルトは **sa**)。
   - **pwd** **sa** として接続するためのパスワード。

   このコマンドが正常に実行されると、Mobile Link サーバーメッセージウィンドウに **[Mobile Link サーバーが起動しました。]** というメッセージが表示されます。

   Mobile Link サーバー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、*sync_ase¥remote* ディレクトリに移動します。

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。

   ```
   dbeng12 -n remote_eng sync_ase_remote.db -n remote_db
   ```

   - **dbeng12**　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ー。
   - **remote_eng**　データベースサーバー名。
   - ***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。

   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ーが起動し、remote_db という名前のデータベースがロードされます。

5. 「レッスン 9：リモート ID の設定」145 ページに進みます。

参照

- 「SQL Anywhere データベースサーバーの構文」『SQL Anywhere サーバー データベース管理』
- 「展開されたモデル同期」46 ページ
- 「Mobile Link サーバーの実行」『Mobile Link サーバー管理』

レッスン 9：リモート ID の設定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：スキーマの設計」132 ページを参照してください。

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。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 サーバー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。

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。

♦ リモート ID の有効な書店識別子への設定

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ー名を指定します。

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。
- **eng**　データベースサーバー名として 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ー名を指定します。
- **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。

3. 「レッスン 10：同期」 146 ページに進みます。

参照

- 「リモート ID」『Mobile Link クライアント管理』

レッスン 10：同期

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」 132 ページを参照してください。

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。dbmlsync は、リモートデータベースに接続して Mobile Link サーバー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。

♦ リモートクライアントの同期

1. 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=remote_eng**　リモートデータベースサーバー名を指定します。
- **DBN=remote_db**　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ー名を指定します。
- **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 サーバーによる認証に使用するユーザー名。
- **ase_pass**　Mobile Link サーバーによる認証に使用するパスワード。

[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 ユーザー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 サーバーが実行中であることを確認します。また、同期ログ (サーバー、クライアントとも) の内容を確認することもできます。

注意
別のコンピューターにある dbmlsync アプリケーションを Mobile Link サーバーから実行している場合、Mobile Link サーバーのロケーションを指定する引数も渡す必要があります。

2. 「レッスン 11：リモートデータベースのデータの表示」147 ページに進みます。

参照

- 「同期処理」14 ページ
- 「dbmlsync 構文」『Mobile Link クライアント管理』

レッスン 11：リモートデータベースのデータの表示

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」132 ページを参照してください。

Mobile Link サーバー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 12 プラグインを使用すると、Sybase Central のリモートデータベースの内容を確認できます。

♦ リモートデータベースのデータの表示

1. Sybase 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 Anywhere 12]** を右クリックして **[接続]** を選択します。
   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。
      i. **[ユーザー ID]** フィールドに **DBA** と入力します。
      ii. **[パスワード]** フィールドに **sql** と入力します。
   c. **[アクション]** ドロップダウンリストから、**[このコンピューターで稼動しているデータベースに接続]** を選択します。
   d. **[サーバー名]** 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 **[所有者フィルターの設定]** 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、store 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ー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。
5. 「クリーンアップ」148 ページに進みます。

クリーンアップ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：スキーマの設計」132 ページを参照してください。

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ーから削除します。

♦ チュートリアルの削除

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ー名を指定します。また、instpbs2 ファイルをローカルのコンピューター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. 同期モデルの削除
   a. Sybase Central を起動します。
   b. 右ウィンドウ枠で **[Mobile Link 12]** 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
   ```

チュートリアル：Java 同期論理の使用

このチュートリアルでは、Java 同期論理を使用するための基本的な手順について説明します。SQL Anywhere 統合データベースとして CustDB サンプルを使用して、Mobile Link のテーブルレベルイベント用に簡単なクラスメソッドを指定します。また、この処理では、コンパイルされた Java クラスのパスを設定するオプションを使用して、Mobile Link サーバー (mlsrv12) を実行します。

必要なソフトウェア

●SQL Anywhere 12

●Java ソフトウェア開発キット

前提知識と経験

次の知識と経験が必要です。

●Java の知識

●Mobile Link イベントスクリプトの基本的な知識

概要

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 サーバー 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

●テーブルレベルイベント用のクラスメソッドの指定

●-sl java オプションを使用した、Mobile Link サーバー (mlsrv12) 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

目的

次の項目について、知識と経験を得ることができます。

●Mobile Link テーブルレベルイベントスクリプト用の Java クラスメソッド

参照

- 「同期スクリプトの作成」『Mobile Link サーバー管理』
- 「Java 同期論理の設定」『Mobile Link サーバー管理』
- 「Java 同期の例」『Mobile Link サーバー管理』
- 「同期イベント」『Mobile Link サーバー管理』
- 「同期の方法」『Mobile Link サーバー管理』

レッスン 1: Mobile Link 参照を含む CustdbScripts Java クラスのコンパイル

このレッスンでは、CustDB サンプルデータベースに関連するクラスをコンパイルします。ULCustomer の upload_insert イベントと download_cursor イベントを処理するための論理を含む Java クラス CustdbScripts を作成します。テキストエディターに CustdbScripts コードを入力し、ファイルを *CustdbScripts.java* として保存します。

Mobile Link データベースサンプル

SQL Anywhere には、同期できるように設定された SQL Anywhere サンプルデータベース (CustDB) が付属しています。このデータベースには、同期に必要な SQL スクリプトなどが含まれています。CustDB の ULCustomer テーブルは、さまざまなテーブルレベルイベントをサポートする同期テーブルです。

CustDB は、Ultra Light クライアントと SQL Anywhere クライアントの統合データベース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 12 CustDB を使用します。

CustdbScripts クラスの作成

Java 同期論理を実行するには、Mobile Link サーバーが *mlscript.jar* のクラスにアクセスできることが必要です。この JAR ファイルには、Java メソッドで利用する Mobile Link サーバー API クラスのレポジトリが含まれています。

Mobile Link 用 Java ソースコードをコンパイルするときは、Mobile Link サーバー API を使用するための *mlscript.jar* を含める必要があります。この項では、javac ユーティリティの -classpath オプションを使用して、*mlscript.jar* を CustdbScripts クラス用に指定します。

♦ CustdbScripts クラスの作成

1. Java クラスとアセンブリ用のディレクトリを作成します。

   このチュートリアルでは、パスを *c:¥mljava* とします。

2. テキストエディターで、次のコードを作成します。

```
public class CustdbScripts {
    public static String UploadInsert() {
        return("INSERT INTO ULCustomer(cust_id,cust_name) VALUES ({ml r.cust_id}, {ml
r.cust_name})");
    }
    public String DownloadCursor(java.sql.Timestamp ts,String user ) {
        return("SELECT cust_id, cust_name FROM ULCustomer WHERE last_modified >= ' " + ts + "
' ");
    }
}
```

   注意
   独自のカスタムスクリプトを作成する場合は、クラスと関連メソッドを **public** として定義します。

3. ファイルを *CustdbScripts.java* として *c:¥mljava* に保存します。

4. ファイルをコンパイルします。

   次のコマンドを実行します。

```
javac custdbscripts.java -classpath "C:¥Program Files¥SQL Anywhere 12¥java¥mlscript.jar"
```

   C:¥Program Files¥SQL Anywhere 12 は、SQL Anywhere 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。

   CustdbScripts.class ファイルが生成されます。

5. 「レッスン 2：Mobile Link プロジェクトの作成」151 ページに進みます。

参照

- 「Mobile Link サーバー Java API リファレンス」『Mobile Link サーバー管理』
- 「メソッド」『Mobile Link サーバー管理』

レッスン 2：Mobile Link プロジェクトの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 参照を含む CustdbScripts Java クラスのコンパイル」150 ページを参照してください。

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 Mobile Link プロジェクトの作成**

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Sybase Central]** をクリックします。
2. **[ツール]** » **[Mobile Link 12]** » **[新しいプロジェクト]** をクリックします。
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 **mljava_project** と入力します。
4. **[ロケーション]** フィールドに *C:¥mljava* と入力し、**[次へ]** をクリックします。
5. **[統合データベースをプロジェクトに追加]** オプションをオンにします。
6. **[データベースの表示名]** フィールドに **mljava_db** と入力します。
7. **[編集]** をクリックします。
8. **[汎用 ODBC データベースに接続]** で次のタスクを実行します。
   a. **[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 12 CustDB** を選択します。
   b. **[OK]** をクリックし、**[保存]** をクリックします。
   c. **[完了]** をクリックします。
9. **[OK]** をクリックします。
10. 「レッスン 3：upload_insert イベントへのスクリプトのサブスクライブ」 152 ページに進みます。

レッスン 3：upload_insert イベントへのスクリプトのサブスクライブ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 参照を含む CustdbScripts Java クラスのコンパイル」 150 ページを参照してください。

このレッスンでは、Sybase Central を使用して、Java メソッドを ULCustomer の upload_insert イベント用のスクリプトとして指定します。Sybase Central を使用して CustDB データベースに接続し、upload_insert SQL スクリプトを Java スクリプトに置き換えてから、イベントを処理する CustdbScripts.UploadInsert を指定します。

別の方法として、ml_add_java_connection_script ストアドプロシージャーや ml_add_java_table_script ストアドプロシージャーも使用できます。これらのストアドプロシージャーは、特に同期イベントを処理するのに多数のメソッドが必要な場合に使用すると、より効率的です。 「ml_add_java_connection_script システムプロシージャー」『Mobile Link サーバー管理』と「ml_add_java_table_script システムプロシージャー」『Mobile Link サーバー管理』を参照してください。

♦ ULCustomer テーブル用の upload_insert イベントへの CustdbScripts.UploadInsert のサブスクライブ

1. Sybase Central で、**[ビュー]** » **[フォルダー]** をクリックします。
2. Sybase Central の左ウィンドウ枠の **[Mobile Link 12]** で、**mljava_project**、**[統合データベース]**、**mljava_db**、**[同期テーブル]**、**ULCustomer** の順に展開します。
3. 右ウィンドウ枠で、custdb 12.0 の upload_insert テーブルスクリプトを選択します。**[編集]** » **[削除]** をクリックします。
4. 新しい **upload_insert** テーブルスクリプトを作成します。
   a. **[ファイル]** » **[新規]** » **[テーブルスクリプト]** をクリックします。
   b. **[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 **custdb 12.0** をクリックします。
   c. **[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。
   d. **[新しいスクリプト定義を作成する]** を選択し、**[Java]** を選択します。
   e. **[完了]** をクリックします。
5. ロードする custdb 12.0 の upload_insert テーブルスクリプトの Java メソッド名を入力します。

   Sybase Central の右ウィンドウ枠で、**upload_insert** イベント用の次の Java スクリプトを使用します。

   ```
   CustdbScripts.UploadInsert
   ```
6. **[ファイル]** » **[保存]** をクリックしてスクリプトを保存します。
7. Sybase Central を閉じます。
8. 「レッスン 4：イベントを処理するクラスメソッドの指定」153 ページに進みます。

レッスン 4：イベントを処理するクラスメソッドの指定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 参照を含む CustdbScripts Java クラスのコンパイル」150 ページを参照してください。

前のレッスンで作成されたクラス *CustdbScripts.class* は、メソッド UploadInsert と DownloadCursor をカプセル化します。これらのメソッドには、それぞれ ULCustomer の upload_insert イベントと download_cursor イベントの実装が含まれています。

このレッスンでは、Interactive SQL を使用して CustDB データベースに接続し、ml_add_java_table_script を実行して、download_cursor イベントを処理する CustdbScripts.DownloadCursor を指定します。

♦ ULCustomer の download_cursor イベントを処理する CustdbScripts.DownloadCursor の指定

1. Interactive SQL からサンプルデータベースに接続します。
   a. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Interactive SQL]** をクリックするか、次のコマンドを実行します。

      ```
      dbisql
      ```

   b. **[ODBC データソース名]** をクリックし、**SQL Anywhere 12 CustDB** と入力します。
   c. **[接続]** をクリックします。
2. Interactive SQL で次の SQL 文を実行します。

   ```
   CALL ml_add_java_table_script(
      'custdb 12.0',
      'ULCustomer',
      'download_cursor',
      'CustdbScripts.DownloadCursor');
   COMMIT;
   ```

   次に、各パラメーターの説明を示します。

パラメーター	説明
custdb 12.0	スクリプトバージョン
ULCustomer	同期テーブル
download_cursor	イベント名
CustdbScripts.DownloadCursor	完全に修飾された Java メソッド

注意
更新された download_cursor スクリプトが Sybase Central に表示されないことがあります。Mobile Link プロジェクトに加えた最新の変更を Sybase Central で表示するには、**[表示]** » **[すべて再表示]** を選択します。

3. Interactive SQL を閉じます。
4. 「レッスン 5：-sl java を使用した、Mobile Link サーバーの実行」 155 ページに進みます。

参照

- 「スクリプトの追加と削除」『Mobile Link サーバー管理』
- 「ml_add_java_connection_script システムプロシージャー」『Mobile Link サーバー管理』
- 「ml_add_java_table_script システムプロシージャー」『Mobile Link サーバー管理』

レッスン 5：-sl java を使用した、Mobile Link サーバーの実行

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：Mobile Link 参照を含む CustdbScripts Java クラスのコンパイル」150 ページを参照してください。

このレッスンでは、-sl java -cp オプションを使用して Mobile Link サーバーを実行し、クラス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て、Java VM をサーバー起動時にロードします。

♦ Mobile Link サーバー (mlsrv12) の起動と Java アセンブリのロード

1. -sl java オプションを使用して、Mobile Link サーバーを起動します。

   次のコマンドを実行します。

   ```
   mlsrv12 -c "DSN=SQL Anywhere 12 CustDB" -sl java (-cp c:¥mljava)
   ```

   サーバーが要求を処理する準備ができたことを示すメッセージが表示されます。同期中に upload_insert イベントがトリガーされると Java メソッドが実行されます。

2. 「レッスン 6：同期のテスト」155 ページに進みます。

参照

- 「Mobile Link サーバーオプション」『Mobile Link サーバー管理』
- 「-sl java mlsrv12 オプション」『Mobile Link サーバー管理』

レッスン 6：同期のテスト

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：Mobile Link 参照を含む CustdbScripts Java クラスのコンパイル」150 ページを参照してください。

Ultra Light には、サンプル Windows クライアントが用意されています。このクライアントは、ユーザーが同期を開始したときに自動的に dbmlsync ユーティリティを起動します。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。新しい顧客名と注文情報を入力します。この情報は、今後発生する同期中に CustDB 統合データベースにアップロードされ、ULCustomer テーブルの upload_insert イベントと download_cursor イベントがトリガーされます。

♦ サンプルアプリケーションの起動と同期

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 Anywhere 12] » [Ultra Light] » [Windows サンプルアプリケーション] をクリックします。

2. Employee ID を入力して同期します。

   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。

3. **[Order]** » **[New]** を選択します。
4. Customer に **Frank Javac** と入力します。
5. Product を選択し、Quantity と Discount を入力します。
6. **[OK]** をクリックし、新規注文を追加します。

   これで、ローカルの Ultra Light データベースでデータが修正されました。このデータは、同期が行われるまで統合データベースとは共有されません。
7. **[ファイル]** » **[同期]** をクリックします。

   統合データベースに挿入が正常にアップロードされたことを示すメッセージが表示されます。
8. Interactive SQL を使用して、サンプルデータベースがサンプルアプリケーションから新しい顧客データをダウンロードしたことを確認します。
   a. Interactive SQL を使用して、サンプルデータベースに接続します。

      [スタート] » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Interactive SQL]** をクリックするか、次のコマンドを実行します。

      ```
      dbisql
      ```
   b. **[ODBC データソース名]** をクリックし、**SQL Anywhere 12 CustDB** と入力します。
   c. **[接続]** をクリックします。
9. Interactive SQL から次の SQL 文を実行します。

   ```
   SELECT * FROM ULCustomer WHERE cust_name = 'Frank Javac';
   ```

   クエリ結果は Interactive SQL の下側のウィンドウ枠に表示され、そこには、顧客 ID、顧客名、および最後に変更されたフィールドが表示されます。最後に変更されたフィールドは、顧客 **Frank Javac** が最後に更新された時期を示します。このフィールドは、サンプルアプリケーションを統合データベースに同期した日付と時刻を示します。
10. 「クリーンアップ」 156 ページに進みます。

参照

- 「Mobile Link の CustDB サンプル」 51 ページ

クリーンアップ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 参照を含む CustdbScripts Java クラスのコンパイル」 150 ページを参照してください。

♦ コンピューターからのチュートリアルの削除

1. Java ソースファイルを削除します。

   たとえば、*c:¥mljava* 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のメニューから、**[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。

   %SQLANYSAMP12%¥UltraLite¥CustDB ディレクトリから次のコマンドを実行します。

   ```
   makedbs
   ```

チュートリアル：.NET 同期論理の使用

このチュートリアルでは、.NET 同期論理を使用するための基本的な手順について説明します。SQL Anywhere 統合データベースとして CustDB サンプルを使用して、Mobile Link のテーブルレベルイベント用に簡単なクラスメソッドを指定します。また、この処理では、.NET アセンブリのパスを設定するオプションを使用して、Mobile Link サーバー (mlsrv12) を実行します。

必要なソフトウェア

- SQL Anywhere 12
- Microsoft .NET Framework SDK

前提知識と経験

次の知識と経験が必要です。

- .NET の知識
- Mobile Link イベントスクリプトの基本的な知識

概要

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 参照を含む *CustdbScripts.dll* プライベートアセンブリのコンパイル

●テーブルレベルイベント用のクラスメソッドの指定

●-sl dnet オプションを使用した、Mobile Link サーバー (mlsrv12) 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

目的

次の項目について、知識と経験を得ることができます。

●Mobile Link テーブルレベルイベントスクリプト用の .NET クラスメソッド

参照

- 「同期スクリプトの作成」『Mobile Link サーバー管理』
- 「.NET 同期論理の設定」『Mobile Link サーバー管理』
- 「.NET 同期論理のデバッグ」『Mobile Link サーバー管理』
- 「.NET 同期のサンプル」『Mobile Link サーバー管理』
- 「同期イベント」『Mobile Link サーバー管理』
- 「同期の方法」『Mobile Link サーバー管理』

レッスン 1 : Mobile Link 参照を含む *CustdbScripts.dll* アセンブリのコンパイル

このレッスンでは、CustDB サンプルデータベースに関連するクラスをコンパイルします。

Mobile Link データベースサンプル

SQL Anywhere には、同期できるように設定された SQL Anywhere サンプルデータベース (CustDB) が付属しています。このデータベースには、同期に必要な SQL スクリプトなどが含まれています。たとえば、CustDB の ULCustomer テーブルは、さまざまなテーブルレベルイベントをサポートする同期テーブルです。

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ー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 12 CustDB を使用します。

CustdbScripts アセンブリ

.NET クラスは、メソッドに同期論理をカプセル化します。この項では、ULCustomer の upload_insert イベントと download_cursor イベントを処理するための論理を含む .NET クラス CustdbScripts を作成します。

Mobile Link サーバー API

.NET 同期論理を実行するには、Mobile Link サーバーが *iAnywhere.MobiLink.Script.dll* 内のクラスにアクセスできることが必要です。*iAnywhere.MobiLink.Script.dll* には、.NET メソッドで利用する .NET クラス用 Mobile Link サーバー API のレポジトリが含まれています。

.NET 用 Mobile Link サーバー API の詳細については、「Mobile Link サーバー .NET API リファレンス」『Mobile Link サーバー管理』を参照してください。

この API を利用する場合、CustdbScripts クラスのコンパイル時にここで示したアセンブリを含める必要があります。クラスのコンパイルは、Visual Studio を使用するか、コマンドプロンプトから実行できます。

- 新しいクラスライブラリを作成し、CustdbScripts コードを入力します。*iAnywhere.MobiLink.Script.dll* をリンクし、クラスのアセンブリを構築します。
- テキストファイルに CustdbScripts コードを入力し、ファイルを *CustdbScripts.cs* (Visual Basic の場合は *CustdbScripts.vb*) として保存します。コマンドラインコンパイラーを使用して *iAnywhere.MobiLink.Script.dll* を参照し、クラスのアセンブリを構築します。

Visual Studio を使用した CustdbScripts アセンブリの作成

Visual Studio を使用すると、CustdbScripts アセンブリを作成できます。

♦ Visual Studio を使用した CustdbScripts アセンブリの作成

1. 新しい Visual C# または Visual Basic Windows クラスライブラリプロジェクトを起動します。

   プロジェクトの **[名前]** として **CustdbScripts** を使用し、適切なパスを入力します。このチュートリアルでは、パスを *c:¥mldotnet* とします。

2. Visual Studio 2010 では、トップのドロップボックスを **[.NET Framework 4]** から **[.NET Framework 3.5]** に変更します。

   注意
   v4.0 のアセンブリを使用するには、Mobile Link サーバーをロードするときに -clrVersion オプションを明示的に含める必要があります。 -clrVersion オプションの詳細については、「-sl dnet mlsrv12 オプション」『Mobile Link サーバー管理』を参照してください。

3. ソリューションエクスプローラーで、*Class1.cs* ファイルが強調表示されていることを確認します。

4. CustdbScripts コードを入力します。

   C# の場合は、既存のコードを次のコードで置き換えます。

```
namespace MLExample {
    class CustdbScripts {
        public static string UploadInsert() {
            return("INSERT INTO ULCustomer(cust_id,cust_name) values ({ml r.cust_id}, {ml
r.cust_name})");
        }

        public static string DownloadCursor(System.DateTime ts, string user ) {
            return("SELECT cust_id, cust_name FROM ULCustomer WHERE last_modified >= '" +
ts.ToString("yyyy-MM-dd hh:mm:ss.fff") +"'");
        }
    }
}
```

   Visual Basic の場合は、既存のコードを次のコードで置き換えます。

```
Namespace MLExample
    Class CustdbScripts
```

```
        Public Shared Function UploadInsert() As String
            Return("INSERT INTO ULCustomer(cust_id,cust_name) values ({ml r.cust_id}, {ml
r.cust_name})")
        End Function

        Public Shared Function DownloadCursor(ByVal ts As System.DateTime, ByVal user As String)
As String
            Return("SELECT cust_id, cust_name FROM ULCustomer " + "WHERE last_modified >= '" +
ts.ToString("yyyy-MM-dd hh:mm:ss.fff") +"'")
        End Function
    End Class
End Namespace
```

5. Visual Basic の場合は、**CustdbScripts** プロジェクトを右クリックし、テーブルの **[プロパティ]** » **[一般]** を選択します。すべてのテキストについて、**[ルート名前空間]** テキストフィールドがオフになっていることを確認してください。

6. *CustdbScripts.dll* をビルドします。

   [構築] » **[CustdbScripts の構築]** をクリックします。

   C:¥mldotnet¥CustdbScripts¥CustdbScripts¥bin¥Debug にこのファイル *CustdbScripts.dll* が作成されます。

7. 「レッスン 2：Mobile Link プロジェクトの作成」161 ページに進みます。

コマンドプロンプトでの CustdbScripts アセンブリの作成

Visual Studio IDE の代わりにコマンドライン、テキストエディター、Visual Studio コンパイラーを使用して、CustdbScripts アセンブリを作成できます。

♦ コマンドプロンプトでの CustdbScripts アセンブリの作成

1. .NET クラスとアセンブリ用のディレクトリを作成します。

   このチュートリアルでは、パスを *c:¥mldotnet* とします。

2. テキストエディターを使用して、CustdbScripts コードを入力します。

   C# の場合、次のように入力します。

```
namespace MLExample {
    class CustdbScripts {
        public static string UploadInsert() {
            return("INSERT INTO ULCustomer(cust_id,cust_name) values ({ml r.cust_id}, {ml
r.cust_name})");
        }

        public static string DownloadCursor(System.DateTime ts, string user ) {
            return("SELECT cust_id, cust_name FROM ULCustomer WHERE last_modified >= '" +
ts.ToString("yyyy-MM-dd hh:mm:ss.fff") +"'");
        }
    }
}
```

   Visual Basic の場合は、次のように入力します。


```
Namespace MLExample
    Class CustdbScripts
        Public Shared Function UploadInsert() As String
            Return("INSERT INTO ULCustomer(cust_id,cust_name) values ({ml r.cust_id}, {ml
r.cust_name})")
        End Function

        Public Shared Function DownloadCursor(ByVal ts As System.DateTime, ByVal user As String)
As String
            Return("SELECT cust_id, cust_name FROM ULCustomer " + "WHERE last_modified >= '" +
ts.ToString("yyyy-MM-dd hh:mm:ss.fff") +"'")
        End Function
    End Class
End Namespace
```

3. ファイルを *CustdbScripts.cs* (Visual Basic の場合は *CustdbScripts.vb*) として *c:¥mldotnet* に保存します。

4. ファイルをコンパイルします。

   C# の場合は、次のコマンドを実行します。

   ```
   csc /out:c:¥mldotnet¥custdbscripts.dll /target:library /reference:"C:¥Program Files¥SQL Anywhere
   12¥Assembly¥v2¥iAnywhere.MobiLink.Script.dlmldotnetmldotnet¥CustdbScripts.cs
   ```

   Visual Basic の場合は、次のコマンドを実行します。

   ```
   vbc /out:c:¥mldotnet¥custdbscripts.dll /target:library /reference:"C:¥Program Files¥SQL Anywhere
   12¥Assembly¥v2¥iAnywhere.MobiLink.Script.dll" c:¥mldotnet¥CustdbScripts.vb
   ```

   CustdbScripts.dll アセンブリが生成されます。

5. 「レッスン 2：Mobile Link プロジェクトの作成」 161 ページに進みます。

参照

- 「Mobile Link サーバー .NET API リファレンス」『Mobile Link サーバー管理』
- 「メソッド」『Mobile Link サーバー管理』

レッスン 2：Mobile Link プロジェクトの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1 : Mobile Link 参照を含む CustdbScripts.dll アセンブリのコンパイル」 158 ページを参照してください。

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 Mobile Link プロジェクトの作成

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Sybase Central]** をクリックします。

2. **[ツール]** » **[Mobile Link 12]** » **[新しいプロジェクト]** をクリックします。

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 **mldotnet_project** と入力します。
4. **[ロケーション]** フィールドに *C:¥mldotnet* と入力し、**[次へ]** をクリックします。
5. **[統合データベースをプロジェクトに追加]** オプションを選択します。
6. **[データベースの表示名]** フィールドに **mldotnet_db** と入力します。
7. **[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。
8. **[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 12 CustDB** を選択します。
9. **[OK]** をクリックし、**[保存]** をクリックします。
10. **[完了]** をクリックします。
11. **[OK]** をクリックします。
12. 「レッスン 3：upload_insert イベントへのスクリプトのサブスクライブ」 162 ページに進みます。

レッスン 3：upload_insert イベントへのスクリプトのサブスクライブ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1 : Mobile Link 参照を含む CustdbScripts.dll アセンブリのコンパイル」158 ページを参照してください。

このレッスンでは、Sybase Central を使用して、.NET メソッドを ULCustomer の upload_insert イベント用のスクリプトとして指定します。Sybase Central を使用して CustDB データベースに接続し、upload_insert SQL スクリプトを .NET スクリプトに置き換えてから、イベントを処理する MLExample.CustdbScripts.UploadInsert を指定します。

別の方法として、ml_add_dnet_connection_script ストアドプロシージャーや ml_add_dnet_table_script ストアドプロシージャーも使用できます。これらのストアドプロシージャーは、特に同期イベントを処理するのに多数の .NET メソッドが必要な場合に使用すると、より効率的です。

♦ ULCustomer テーブル用の upload_insert イベントへの CustdbScripts.uploadInsert のサブスクライブ

1. Sybase Central で、**[ビュー]** » **[フォルダー]** をクリックします。
2. Sybase Central の左ウィンドウ枠の **[Mobile Link 12]** で、**mldotnet_project**、**[統合データベース]**、**mldotnet_db**、**[同期テーブル]**、**ULCustomer** の順に展開します。
3. 右ウィンドウ枠で、custdb 12.0 の upload_insert テーブルスクリプトを選択します。**[編集]** » **[削除]** をクリックします。

4. 新しい upload_insert テーブルスクリプトを作成します。
   a. **[ファイル]** » **[新規]** » **[テーブルスクリプト]** をクリックします。
   b. **[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 **custdb 12.0** をクリックします。
   c. **[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。
   d. **[新しいスクリプト定義を作成する]** を選択し、**[.NET]** を選択します。
   e. **[完了]** をクリックします。
5. ロードする custdb 12.0 の upload_insert テーブルスクリプトの .NET メソッド名を入力します。

   Sybase Central の右ウィンドウ枠で、**upload_insert** イベント用の次の .NET スクリプトを追加します。

   ```
   MLExample.CustdbScripts.UploadInsert
   ```
6. **[ファイル]** » **[保存]** をクリックしてスクリプトを保存します。
7. Sybase Central を閉じます。
8. 「レッスン 4：イベント用のクラスメソッドの指定」163 ページに進みます。

レッスン 4：イベント用のクラスメソッドの指定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：Mobile Link 参照を含む CustdbScripts.dll アセンブリのコンパイル」158 ページを参照してください。

前のレッスンで作成されたクラス *CustdbScripts.dll* は、メソッド UploadInsert と DownloadCursor をカプセル化します。これらのメソッドには、それぞれ ULCustomer の upload_insert イベントと download_cursor イベントの実装が含まれています。

このレッスンでは、Interactive SQL を使用して CustDB データベースに接続し、ml_add_dnet_table_script を実行して、download_cursor イベント用の MLExample.CustdbScripts.DownloadCursor を指定します。

♦ ULCustomer の download_cursor イベント用の MLExample.CustdbScripts.DownloadCursor の指定

1. Interactive SQL からサンプルデータベースに接続します。
   a. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Interactive SQL]** をクリックするか、次のコマンドを実行します。

      ```
      dbisql
      ```
   b. **[ODBC データソース名]** をクリックし、**SQL Anywhere 12 CustDB** と入力します。
   c. **[接続]** をクリックします。

2. Interactive SQL で次の SQL 文を実行し、ULCustomer テーブルレベルイベントを処理するための .NET メソッドを指定します。

```
CALL ml_add_dnet_table_script(
   'custdb 12.0',
   'ULCustomer',
   'download_cursor',
   'MLExample.CustdbScripts.DownloadCursor');
COMMIT;
```

次に、各パラメーターの説明を示します。

パラメーター	説明
custdb 12.0	スクリプトバージョン
ULCustomer	同期テーブル
download_cursor	イベント名
MLExample.CustdbScripts.DownloadCursor	完全に修飾された .NET メソッド

注意
更新された download_cursor スクリプトが Sybase Central に表示されないことがあります。Mobile Link プロジェクトに加えた最新の変更を Sybase Central で表示するには、**[表示]** » **[すべて再表示]** をクリックします。

3. Interactive SQL を閉じます。

4. 「レッスン 5：-sl dnet 使用した、Mobile Link サーバーの実行」 164 ページに進みます。

参照

- 「スクリプトの追加と削除」『Mobile Link サーバー管理』
- 「ml_add_dnet_connection_script システムプロシージャー」『Mobile Link サーバー管理』
- 「ml_add_dnet_table_script システムプロシージャー」『Mobile Link サーバー管理』

レッスン 5：-sl dnet 使用した、Mobile Link サーバーの実行

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1 : Mobile Link 参照を含む CustdbScripts.dll アセンブリのコンパイル」 158 ページを参照してください。

このレッスンでは、-sl dnet オプションを使用して Mobile Link サーバーを実行し、.NET アセンブリのロケーションを指定して、CLR をサーバー起動時にロードします。

コンパイルに Visual Studio を使用した場合、*CustdbScripts.dll* のロケーションは *c:¥mldotnet¥CustdbScripts¥CustdbScripts¥bin¥Debug* になります。コマンドプロンプトを使用した場合、*CustdbScripts.dll* のロケーションは *c:¥mldotnet* になります。

♦ Mobile Link サーバー (mlsrv12) の起動と .NET アセンブリのロード

1. -sl dnet オプションを使用して、Mobile Link サーバーを起動します。

   Visual Studio を使用してアセンブリをコンパイルした場合は、次のコマンドを実行します。

   ```
   mlsrv12 -c "DSN=SQL Anywhere 12 CustDB" -dl -o c:¥mldotnet¥cons1.txt -v+ -sl dnet(-
   MLAutoLoadPath=c:¥mldotnet¥CustdbScripts¥CustdbScripts¥bin¥Debug)
   ```

   コマンドプロンプトでアセンブリをコンパイルした場合は、次のコマンドを実行します。

   ```
   mlsrv12 -c "DSN=SQL Anywhere 12 CustDB" -dl -o c:¥mldotnet¥cons1.txt -v+ -sl dnet(-
   MLAutoLoadPath=c:¥mldotnet)
   ```

   サーバーが要求を処理する準備ができたことを示すメッセージが表示されます。同期中に upload_insert イベントがトリガーされると .NET メソッドが実行されます。

2. 「レッスン 6：同期のテスト」 165 ページに進みます。

参照

- 「Mobile Link サーバーオプション」『Mobile Link サーバー管理』
- 「-sl dnet mlsrv12 オプション」『Mobile Link サーバー管理』

レッスン 6：同期のテスト

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1 : Mobile Link 参照を含む CustdbScripts.dll アセンブリのコンパイル」158 ページを参照してください。

Ultra Light には、サンプル Windows クライアントが用意されています。このクライアントは、ユーザーが同期を開始したときに自動的に dbmlsync ユーティリティを起動します。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。新しい顧客名と注文情報を入力します。この情報は、今後発生する同期中に CustDB 統合データベースにアップロードされ、ULCustomer テーブルの upload_insert イベントと download_cursor イベントがトリガーされます。

♦ サンプルアプリケーションの起動と認証のテスト

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 Anywhere 12]** » **[Ultra Light]** » **[Windows サンプルアプリケーション]** をクリックします。

2. Employee ID を入力して同期します。

   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。

3. **[Order]** » **[New]** を選択します。

4. Customer に **Frank DotNET** と入力します。
5. Product を選択し、Quantity と Discount を入力します。
6. **[OK]** をクリックし、新規注文を追加します。

   これで、ローカルの Ultra Light データベースでデータが修正されました。このデータは、同期が行われるまで統合データベースとは共有されません。
7. **[ファイル]** » **[同期]** をクリックします。

   統合データベースに挿入が正常にアップロードされたことを示すメッセージが表示されます。

   Interactive SQL を使用して、サンプルデータベースがサンプルアプリケーションから新しい顧客データをダウンロードできたかどうかを確認します。
8. Interactive SQL からサンプルデータベースに接続します。
   a. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Interactive SQL]** をクリックするか、次のコマンドを実行します。

      ```
      dbisql
      ```
   b. **[ODBC データソース名]** をクリックし、**SQL Anywhere 12 CustDB** と入力します。
   c. **[接続]** をクリックします。
9. Interactive SQL で次の SQL 文を実行します。

   ```
   SELECT * FROM ULCustomer WHERE cust_name = 'Frank DotNET';
   ```

   クエリ結果は Interactive SQL の下側のウィンドウ枠に表示され、そこには、顧客 ID、顧客名、および最後に変更されたフィールドが表示されます。最後に変更されたフィールドは、**Frank DotNET** の顧客が最後に更新された時期を示します。このフィールドは、サンプルアプリケーションを統合データベースに同期した日付と時刻を示します。
10. 「クリーンアップ」 166 ページに進みます。

参照

- 「Mobile Link の CustDB サンプル」 51 ページ

クリーンアップ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1 : Mobile Link 参照を含む CustdbScripts.dll アセンブリのコンパイル」 158 ページを参照してください。

♦ コンピューターからのチュートリアルの削除

1. .NET ソースファイルを削除します。

たとえば、*c:¥mldotnet* 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。

   %SQLANYSAMP12%¥UltraLite¥CustDB ディレクトリから次のコマンドを実行します。

   ```
   makedbs
   ```

チュートリアル：カスタムユーザー認証用の 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。

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。

必要なソフトウェア

- SQL Anywhere 12
- Java ソフトウェア開発キットまたは Microsoft .NET Framework

前提知識と経験

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

概要

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 サーバー 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 サーバー (mlsrv12) 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

目的

次の項目について、知識と経験を得ることができます。

●Mobile Link カスタム認証

参照

- 「ユーザー認証メカニズム」『Mobile Link クライアント管理』
- 「外部サーバーに対する認証」『Mobile Link クライアント管理』
- 「.NET での同期スクリプトの作成」『Mobile Link サーバー管理』
- 「Java による同期スクリプトの作成」『Mobile Link サーバー管理』

レッスン 1：カスタム認証用の Java または .NET クラスの作成 (サーバー側)

このレッスンでは、カスタム認証用の Java または .NET 同期論理を記述するクラスをコンパイルします。

Mobile Link サーバー Java API を使用したカスタム認証

Java 同期論理を実行するには、Mobile Link サーバーが *mlscript.jar* 内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 Java サーバー API クラスのレポジトリが入っています。Java クラスをコンパイルするときは、*mlscript.jar* を参照します。

♦ カスタム認証用の Java クラスの作成

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、ユーザーパラメーターとパスワードパラメーターを提供します。認証結果は、authentication_status inout パラメーターを使用して返します。

   注意
   authenticate_user 同期イベントの authenticateUser メソッドは、「レッスン 2：authenticate_user イベント用の Java または .NET スクリプトの登録」171 ページで登録します。

   アプリケーションサーバーに次のコードを使用します。

```
import ianywhere.ml.script.*;

public class MobiLinkAuth {
   public void authenticateUser (
```



```
            ianywhere.ml.script.InOutInteger authentication_status,
            String user,
            String pwd,
            String newPwd ) {

            if (user.substring(0,3).equals("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ー認証の例を示します。**128** で始まるユーザー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* をサーバーサイドコンポーネントの作業フォルダーとします。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 サーバー 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 12¥* を SQL Anywhere 12 ディレクトリに置き換えます。

      ```
      javac MobiLinkAuth.java -classpath "C:¥Program Files¥SQL Anywhere 12¥java¥mlscript.jar"
      ```

      MobiLinkAuth.class ファイルが生成されます。

4. 「レッスン 2：authenticate_user イベント用の Java または .NET スクリプトの登録」171 ページに進みます。

Mobile Link サーバー .NET API を使用したカスタム認証

.NET 同期論理を実行するには、Mobile Link サーバーが *iAnywhere.MobiLink.Script.dll* 内のクラスにアクセスできることが必要です。*iAnywhere.MobiLink.Script.dll* には、.NET メソッドで利用する Mobile Link .NET サーバー API クラスのレポジトリが入っています。.NET クラスをコンパイルするときは、*iAnywhere.MobiLink.Script.dll* を参照します。

♦ カスタム認証用の .NET クラスの作成

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、ユーザーパラメーターとパスワードパラメーターを提供します。認証結果は、authentication_status inout パラメーターを使用して返します。

注意
authenticate_user 同期イベントの authenticateUser メソッドは、「レッスン 2：authenticate_user イベント用の Java または .NET スクリプトの登録」171 ページで登録します。

アプリケーションサーバーに次のコードを使用します。

```
using iAnywhere.MobiLink.Script;

public class MobiLinkAuth {
   public void authenticateUser(
      ref int authentication_status,
      string user,
      string pwd,
      string newPwd ) {

      if(user.Substring(0,3)=="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ー認証の例を示します。**128** で始まるユーザー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* をサーバーサイドコンポーネントの作業フォルダーとします。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 サーバー .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 12¥* を SQL Anywhere 12 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
      12¥Assembly¥v2¥iAnywhere.MobiLink.Script.dll" MobiLinkAuth.cs
      ```

      MobiLinkAuth.dll ファイルが生成されます。

4. 「レッスン 2：authenticate_user イベント用の Java または .NET スクリプトの登録」171 ページに進みます。

参照

- 「authenticate_user 接続イベント」『Mobile Link サーバー管理』
- 「Java と .NET のユーザー認証」『Mobile Link クライアント管理』
- 「Java 同期の例」『Mobile Link サーバー管理』
- 「.NET 同期のサンプル」『Mobile Link サーバー管理』

レッスン 2：authenticate_user イベント用の Java または .NET スクリプトの登録

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：カスタム認証用の Java または .NET クラスの作成 (サーバー側)」168 ページを参照してください。

SQL Anywhere には、同期できるように設定された SQL Anywhere 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。このスクリプトは、Mobile Link サンプルデータベースである CustDB に追加します。

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ー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 12 CustDB** を使用します。

♦ authenticate_user イベント用の authenticateUser メソッドの登録

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 12 CustDB"
   ```

2. ml_add_java_connection_script または ml_add_dnet_connection_script ストアドプロシージャーを使用して、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_add_java_connection_script(
      'custdb 12.0',
      '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 12.0',
      'authenticate_user',
      'MobiLinkAuth.authenticateUser');

   COMMIT;
   ```

3. 「レッスン 3：Mobile Link サーバーの起動」172 ページに進みます。

参照

- 「スクリプトの追加と削除」『Mobile Link サーバー管理』
- 「Java による同期スクリプトの作成」『Mobile Link サーバー管理』
- 「.NET での同期スクリプトの作成」『Mobile Link サーバー管理』
- 「Java 同期の例」『Mobile Link サーバー管理』
- 「.NET 同期のサンプル」『Mobile Link サーバー管理』
- 「Java クラスのデバッグ」『Mobile Link サーバー管理』
- 「.NET 同期論理のデバッグ」『Mobile Link サーバー管理』
- 「同期スクリプトの作成」『Mobile Link サーバー管理』
- 「同期イベント」『Mobile Link サーバー管理』
- 「ml_add_java_connection_script システムプロシージャー」『Mobile Link サーバー管理』
- 「ml_add_dnet_connection_script システムプロシージャー」『Mobile Link サーバー管理』

レッスン 3：Mobile Link サーバーの起動

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：カスタム認証用の Java または .NET クラスの作成 (サーバー側)」168 ページを参照してください。

このレッスンでは、-sl オプションを指定して Mobile Link サーバーを実行し、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。

♦ Mobile Link サーバー (mlsrv12) の起動

1. CustDB サンプルデータベースに接続して、mlsrv12 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。

   ```
   mlsrv12 -c "DSN=SQL Anywhere 12 CustDB" -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。

   ```
   mlsrv12 -c "DSN=SQL Anywhere 12 CustDB" -o serverOut.txt -v+ -sl dnet(-MLAutoLoadPath=c:
   ¥MLauth)
   ```

   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。

2. 「レッスン 4：認証のテスト」173 ページに進みます。

参照

- 「Mobile Link サーバーオプション」『Mobile Link サーバー管理』
- 「-sl java mlsrv12 オプション」『Mobile Link サーバー管理』
- 「-sl dnet mlsrv12 オプション」『Mobile Link サーバー管理』

レッスン 4：認証のテスト

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：カスタム認証用の Java または .NET クラスの作成 (サーバー側)」168 ページを参照してください。

Ultra Light には、サンプル Windows クライアントが用意されています。このクライアントは、ユーザーが同期を開始したときに自動的に dbmlsync ユーティリティを起動します。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。

♦ サンプルアプリケーションの起動と認証のテスト

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 Anywhere 12]** » **[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 ユーザー名でもあります。**128** で始まらないユーザー名の場合、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。

   authenticate_user スクリプトからエラー 4000 が返されたことが、Mobile Link サーバーのメッセージウィンドウに表示されます。

   ユーザー ID またはパスワードが無効であることを示す SQLCODE -103 の同期エラーが **Ultra Light CustDB デモ** ウィンドウに表示されます。「ユーザー ID またはパスワードが無効です。」『エラーメッセージ』を参照してください。

3. 「クリーンアップ」173 ページに進みます。

参照

- 「Mobile Link の CustDB サンプル」51 ページ

クリーンアップ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：カスタム認証用の Java または .NET クラスの作成 (サーバー側)」168 ページを参照してください。

♦ コンピューターからのチュートリアルの削除

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。

   %SQLANYSAMP12%¥UltraLite¥CustDB ディレクトリから次のコマンドを実行します。

   ```
   makedbs
   ```

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。

このチュートリアルでは、Java 用または .NET 用の Mobile Link サーバー 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。

必要なソフトウェア

- SQL Anywhere 12
- Java ソフトウェア開発キットまたは Microsoft .NET Framework

前提知識と経験

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

概要

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 サーバー 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

参照

- 「Mobile Link 同期」 1 ページ
- 「同期の方法」『Mobile Link サーバー管理』
- 「ダイレクトローハンドリング」『Mobile Link サーバー管理』
- http://www.sybase.com/detail?id=1058600#319 (このページを表示するには、Sybase.com アカウントが必要です。)
- sybase.public.sqlanywhere.mobilink

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。

♦ テキストファイルデータソースの設定

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 can not 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* をサーバーサイド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。

4. 「レッスン 2：Mobile Link 統合データベースの設定」 175 ページに進みます。

レッスン 2：Mobile Link 統合データベースの設定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：テキストファイルデータソースの設定」 175 ページを参照してください。

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ーが含まれます。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 サーバー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。

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。

注意
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。

♦ Mobile Link 統合データベースの設定

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Sybase Central]** をクリックします。
2. **[ツール]** » **[SQL Anywhere 12]** » **[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 **[このコンピューターにデータベースを作成]** を受け入れ、**[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* と入力します。
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。**[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 Sybase Central に表示されます。
8. **[データベースの作成]** ウィンドウで **[閉じる]** をクリックします。
9. SQL Anywhere 12 ドライバーを使用して、MLconsolidated データベース用の ODBC データソースを定義します。

   Sybase Central で、**[ツール]** » **[SQL Anywhere 12]** » **[ODBC アドミニストレーターを開く]** をクリックします。
10. **[ユーザー 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 Anywhere 12]** 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. **[ユーザー 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 **sql** と入力します。
    f. **[サーバー名]** 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ーを閉じます。

    [ODBC データソースアドミニストレーター] ウィンドウで **[OK]** をクリックします。

14.「レッスン 3：Mobile Link 統合データベースでのテーブルの作成」177 ページに進みます。

参照

- 「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバー データベース管理』
- 「Mobile Link 統合データベース」『Mobile Link サーバー管理』

レッスン 3：Mobile Link 統合データベースでのテーブルの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：テキストファイルデータソースの設定」175 ページを参照してください。

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。このテーブルには次のカラムがあります。

カラム	説明
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ーする一般的な方法です。

♦ RemoteOrders テーブルの作成

1. Interactive SQL からデータベースに接続します。

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

   - Sybase 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ーを作成します。

   C:¥Program Files¥SQL Anywhere 12¥ は、SQL Anywhere 12 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 12¥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ーが作成されます。これらのテーブルとストアドプロシージャーは、同期処理中に Mobile Link サーバーによって使用されます。

4. 「レッスン 4：同期スクリプトの追加」178 ページに進みます。

参照

- 「CREATE TABLE 文」『SQL Anywhere サーバー SQL リファレンス』

レッスン 4：同期スクリプトの追加

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：テキストファイルデータソースの設定」175 ページを参照してください。

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。

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 サーバーを設定します。

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ーを使用して 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。

♦ 統合データベースへの 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の追加

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ーを使用して、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ーを使用して、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 サーバーが 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_download',
      'MobiLinkOrders.EndDownload' );
   ```

   .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_download',
      'MobiLinkOrders.EndDownload' );
   ```

   Interactive SQL によって、ユーザー定義の 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、ユーザー定義の 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。「必要なスクリプト」『Mobile Link サーバー管理』を参照してください。

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。

8. 「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」 181 ページに進みます。

参照

- 「Mobile Link イベントの概要」『Mobile Link サーバー管理』
- 「スクリプトの追加と削除」『Mobile Link サーバー管理』
- 「ローをアップロードするスクリプト」『Mobile Link サーバー管理』
- 「ローをダウンロードするスクリプト」『Mobile Link サーバー管理』
- 「upload_insert テーブルイベント」『Mobile Link サーバー管理』
- 「upload_update テーブルイベント」『Mobile Link サーバー管理』
- 「upload_delete テーブルイベント」『Mobile Link サーバー管理』
- 「download_cursor テーブルイベント」『Mobile Link サーバー管理』
- 「download_delete_cursor テーブルイベント」『Mobile Link サーバー管理』
- 「ダイレクトローハンドリング」『Mobile Link サーバー管理』
- 「ダイレクトアップロードの処理」『Mobile Link サーバー管理』
- 「ダイレクトダウンロードの処理」『Mobile Link サーバー管理』
- 「タイムスタンプベースのダウンロードの実装」『Mobile Link サーバー管理』
- 「リモートデータベース間でのローの分割」『Mobile Link サーバー管理』

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

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：テキストファイルデータソースの設定」175 ページを参照してください。

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。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。完全なリストについては、「MobiLinkOrders コードの全リスト (Java)」188 ページまたは「MobiLinkOrders コードの全リスト (.NET)」190 ページを参照してください。

♦ 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

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 ianywhere.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 iAnywhere.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 サーバーによって DBConnectionContext のインスタンスがクラスコンストラクターに渡されます。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ーを作成します。

   クラスコンストラクターが、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
        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() if the send
            // column names option is turned on at the remote.
            // 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();
    }
```

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");
```

注意
OrderComments テーブルは、「レッスン 7：Mobile Link クライアントデータベースの設定」193 ページでリモートデータベースに作成します。

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 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;
        }
    }
```

.NET の場合は、次のコードを入力します。

```
    public void EndDownload()
    {
        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 *c:¥MLdirect* に保存します。

   .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 *c:¥MLdirect* に保存します。

9. コードを検証する場合は、「MobiLinkOrders コードの全リスト (Java)」188 ページまたは「MobiLinkOrders コードの全リスト (.NET)」190 ページを参照してください。

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 サーバー API ライブラリを参照します。

       Java の場合、*%SQLANY12%¥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 12¥* を SQL Anywhere 12 ディレクトリに置き換えます。

       ```
       javac -classpath "C:¥Program Files¥SQL Anywhere 12¥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 12¥* を SQL Anywhere 12 ディレクトリに置き換えます。

       ```
       csc /out:MobiLinkServerCode.dll /target:library /reference:"C:¥Program Files¥SQL Anywhere 12¥Assembly¥v2¥iAnywhere.MobiLink.Script.dll" MobiLinkOrders.cs
       ```

注意
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。 「ユニークなプライマリキー」『Mobile Link サーバー管理』を参照してください。

11.「レッスン 6：Mobile Link サーバーの起動」 192 ページに進みます。

参照

- 「ダイレクトローハンドリング」『Mobile Link サーバー管理』
- 「Java による同期スクリプトの作成」『Mobile Link サーバー管理』
- 「.NET での同期スクリプトの作成」『Mobile Link サーバー管理』

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。 手順を追った説明については、「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」 181 ページを参照してください。

```
import ianywhere.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;
        }
    }
}
```

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。 手順を追った説明については、「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」 181 ページを参照してください。

```
using iAnywhere.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() if the send
            // column names option is turned on at the remote.
            // 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;
        }
    }
}
```

レッスン 6：Mobile Link サーバーの起動

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：テキストファイルデータソースの設定」175 ページを参照してください。

このレッスンでは、Mobile Link サーバーを起動します。-c オプションを使って Mobile Link サーバー (mlsrv12) 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。

♦ Mobile Link サーバー (mlsrv12) の起動

1. 統合データベースに接続し、mlsrv12 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。

   ```
   mlsrv12 -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。

   ```
   mlsrv12 -c "DSN=mldirect_db" -o serverOut.txt -v+ -dl -zu+ -x tcpip -sl dnet (-MLAutoLoadPath=c:
   ¥MLdirect)
   ```

   Mobile Link サーバー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 サーバー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ーを追加します。
-x	Mobile Link クライアントの通信プロトコルとパラメーターを設定します。
-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、またサーバー起動時に Java VM をロードします。
-sl dnet	.NET アセンブリのロケーションを指定し、またサーバー起動時に CLR をロードします。

2. 「レッスン 7：Mobile Link クライアントデータベースの設定」 193 ページに進みます。

参照

- 「Mobile Link サーバーオプション」『Mobile Link サーバー管理』
- 「-sl java mlsrv12 オプション」『Mobile Link サーバー管理』
- 「-sl dnet mlsrv12 オプション」『Mobile Link サーバー管理』

レッスン 7：Mobile Link クライアントデータベースの設定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：テキストファイルデータソースの設定」 175 ページを参照してください。

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 サーバーはすべて同じコンピューターに置きます。

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 サーバーでは、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 サーバーでは、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ー、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 ユーザーのサブスクリプションです。

♦ Mobile Link クライアントデータベースの設定

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 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. dbeng12 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。

   ```
   dbeng12 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=sql"
   ```

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 同期ユーザー、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';
   ```

   注意
   Mobile Link サーバーに接続する方法は、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。

7. 「レッスン 8：同期」 195 ページに進みます。

参照

- 「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバー データベース管理』
- 「Mobile Link クライアント」『Mobile Link クライアント管理』
- 「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』

レッスン 8：同期

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：テキストファイルデータソースの設定」 175 ページを参照してください。

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。

♦ クライアント側リモートデータの設定と同期

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=sql"
   ```

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=sql" -e scn=on -o rem1.txt -v+
   ```

次のテーブルには、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。
-e scn	SendColumnNames をオンに設定します。これは、カラムを名前で参照する場合にダイレクトローハンドリングで必要となります。
-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 Anywhere Mobile Link クライアントウィンドウをすべて閉じます。
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。この操作はサーバー側で行います。

   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=sql" -o rem1.txt -v+ -e scn=on
   ```

   Mobile Link クライアントユーティリティが表示されます。

   注意
   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、mlsrv12 -v+ オプションによっては出力されず、リモートの -v+ オプションによってリモートログに出力されます。

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 OrderComments;
   ```


10.「クリーンアップ」197 ページに進みます。

参照

- 「SQL Anywhere クライアント」『Mobile Link クライアント管理』
- 「Mobile Link SQL Anywhere クライアントユーティリティ (dbmlsync)」『Mobile Link クライアント管理』

クリーンアップ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：テキストファイルデータソースの設定」175 ページを参照してください。

♦ チュートリアルの削除

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ーを起動します。
      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*、*remote1.log* を削除します。

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。

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。

必要なソフトウェア

- SQL Anywhere 12
- Java ソフトウェア開発キット

●Microsoft Office Excel 2007 以降

前提知識と経験

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

概要

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 サーバー 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

参照

- 「Mobile Link 同期」 1 ページ
- 「同期の方法」『Mobile Link サーバー管理』
- 「ダイレクトローハンドリング」『Mobile Link サーバー管理』
- http://www.sybase.com/detail?id=1058600#319 (このページを表示するには、Sybase.com アカウントが必要です。)
- sybase.public.sqlanywhere.mobilink

レッスン 1：Excel ワークシートの設定

このレッスンでは、Excel ワークシートを作成し、Microsoft Excel ドライバーを使用して ODBC データソースを定義します。Excel ワークシートには製品情報を格納します。

♦ Excel データソースの設定

1. Microsoft Excel を開き、新しいワークブックを作成します。
2. デフォルトのワークシートで、**A**、**B**、**C** の各カラムヘッダーで次の内容を追加します。

comment_id	order_id	order_comment
2	1	出荷する宣伝用資料
3	1	必要な資料の詳細

3. 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。

4. Excel ワークブックを保存します。

   このチュートリアルでは、*c:¥MLobjexcel* をサーバーサイドコンポーネントの作業ディレクトリとします。ワークブックを *order_central.xlsx* という名前でこの作業ディレクトリに保存します。

5. Microsoft Excel ドライバーを使用して ODBC データソースを作成します。
   a. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[ODBC データソースアドミニストレーター]** をクリックします。
   b. **[ユーザー 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. **[完了]** 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ー] ウィンドウで **[OK]** をクリックします。

6. 「レッスン 2：Mobile Link 統合データベースの設定」199 ページに進みます。

レッスン 2：Mobile Link 統合データベースの設定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：Excel ワークシートの設定」198 ページを参照してください。

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ーが含まれます。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 サーバー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。

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。

注意
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。

♦ 統合データベースの設定

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Sybase Central]** をクリックします。

2. **[ツール]** » **[SQL Anywhere 12]** » **[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. **[このコンピューター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* と入力します。
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。**[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 Sybase Central に表示されます。
8. **[データベースの作成]** ウィンドウで **[閉じる]** をクリックします。
9. Sybase Central で、**[ツール]** » **[SQL Anywhere 12]** » **[ODBC アドミニストレーターを開く]** をクリックします。
10. **[ユーザー DSN]** タブをクリックしてから、**[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 Anywhere 12]** 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 **mlexcel_db** と入力します。
    c. **[ログイン]** タブをクリックします。
    d. **[ユーザー 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 **sql** と入力します。
    f. **[アクション]** ドロップダウンリストから、**[このコンピューター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. **[サーバー名]** フィールドに **MLconsolidated** と入力します。
    h. **[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ーを閉じます。

    [ODBC データソースアドミニストレーター] ウィンドウで **[OK]** をクリックします。
14. 「レッスン 3：Mobile Link 統合データベースでのテーブルの作成」201 ページに進みます。

参照

- 「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバー データベース管理』
- 「CREATE TABLE 文」『SQL Anywhere サーバー SQL リファレンス』
- 「Mobile Link 統合データベース」『Mobile Link サーバー管理』

レッスン 3：Mobile Link 統合データベースでのテーブルの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Excel ワークシートの設定」198 ページを参照してください。

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。このテーブルには次のカラムがあります。

カラム	説明
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ーする一般的な方法です。

♦ RemoteOrders テーブルの作成

1. Interactive SQL からデータベースに接続します。

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

   - Sybase 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、Mobile Link のシステムテーブルとストアドプロシージャーを作成します。

   C:¥Program Files¥SQL Anywhere 12¥ は、SQL Anywhere 12 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 12¥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ーが作成されます。これらのテーブルとストアドプロシージャーは、同期処理中に Mobile Link サーバーによって使用されます。

4. 「レッスン 4：同期スクリプトの追加」202 ページに進みます。

レッスン 4：同期スクリプトの追加

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Excel ワークシートの設定」198 ページを参照してください。

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。

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 サーバーを設定します。

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。

♦ 統合データベースへの 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の追加

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ーを使用して、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ーを使用して、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。 「必要なスクリプト」『Mobile Link サーバー管理』を参照してください。

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. 「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」204 ページに進みます。

参照

- 「Mobile Link イベントの概要」『Mobile Link サーバー管理』
- 「スクリプトの追加と削除」『Mobile Link サーバー管理』
- 「ローをアップロードするスクリプト」『Mobile Link サーバー管理』
- 「ローをダウンロードするスクリプト」『Mobile Link サーバー管理』
- 「upload_insert テーブルイベント」『Mobile Link サーバー管理』
- 「upload_update テーブルイベント」『Mobile Link サーバー管理』
- 「upload_delete テーブルイベント」『Mobile Link サーバー管理』
- 「download_cursor テーブルイベント」『Mobile Link サーバー管理』
- 「download_delete_cursor テーブルイベント」『Mobile Link サーバー管理』
- 「ダイレクトローハンドリング」『Mobile Link サーバー管理』
- 「ダイレクトアップロードの処理」『Mobile Link サーバー管理』
- 「ダイレクトダウンロードの処理」『Mobile Link サーバー管理』
- 「タイムスタンプベースのダウンロードの実装」『Mobile Link サーバー管理』
- 「リモートデータベース間でのローの分割」『Mobile Link サーバー管理』

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

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Excel ワークシートの設定」198 ページを参照してください。

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。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 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、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 のクラスを作成する方法を示します。 完全なリストについては、「MobiLinkOrders コードの全リスト (Java)」 208 ページを参照してください。

♦ ダウンロード専用のダイレクトローハンドリング用の Java のクラスの作成

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 ianywhere.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 サーバーによって DBConnectionContext のインスタンスがクラスコンストラクターに渡されます。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ーを作成します。

   クラスコンストラクターが、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" );
```

   注意
   このテーブルは、「レッスン 7 : Mobile Link クライアントデータベースの設定」211 ページでリモートデータベースに作成します。

   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* という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場合は、「MobiLinkOrders コードの全リスト (Java)」208 ページを参照してください。

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 サーバー API ライブラリを参照する MobiLinkOrders をコンパイルします。

%SQLANY12%¥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 12¥* を SQL Anywhere 12 ディレクトリに置き換えます。

```
javac -classpath "C:¥Program Files¥SQL Anywhere 12¥java¥mlscript.jar" MobiLinkOrders.java
```

8. 「レッスン 6：Mobile Link サーバーの起動」209 ページに進みます。

参照

- 「ダイレクトローハンドリング」『Mobile Link サーバー管理』
- 「Java による同期スクリプトの作成」『Mobile Link サーバー管理』

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。手順を追った説明については、「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」204 ページを参照してください。

```
import ianywhere.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();
    }
    }
}
```

レッスン 6：Mobile Link サーバーの起動

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Excel ワークシートの設定」198 ページを参照してください。

このレッスンでは、Mobile Link サーバーを起動します。-c オプションを使用して Mobile Link サーバー (mlsrv12) 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 のクラスをロードします。

♦ ダイレクトローハンドリング用の Mobile Link サーバーの起動

1. 統合データベースに接続し、mlsrv12 のコマンドラインでクラスをロードします。

   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。

   ```
   mlsrv12 -c "DSN=mlexcel_db" -o serverOut.txt -v+ -dl -zu+ -x tcpip  -sl java (-cp c:¥MLobjexcel)
   ```

   Mobile Link サーバー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 サーバー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ーを追加します。
-x	Mobile Link クライアントの通信プロトコルとパラメーターを設定します。
-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、またサーバー起動時に Java VM をロードします。

2. 「レッスン 7：Mobile Link クライアントデータベースの設定」211 ページに進みます。

参照

- 「Mobile Link サーバーオプション」『Mobile Link サーバー管理』
- 「-sl java mlsrv12 オプション」『Mobile Link サーバー管理』

レッスン 7：Mobile Link クライアントデータベースの設定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：Excel ワークシートの設定」198 ページを参照してください。

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 サーバーはすべて同じコンピューターに置きます。

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 サーバーでは、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 サーバーでは、特別なイベントを使用して OrderComments テーブルが処理されます。

♦ Mobile Link クライアントデータベースの設定

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 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. dbeng12 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。

   ```
   dbeng12 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=sql"
   ```

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 同期ユーザー、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';
```

   注意
   Mobile Link サーバーに接続する方法は、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。

7. 「レッスン 8：同期」212 ページに進みます。

参照

- 「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバー データベース管理』
- 「Mobile Link クライアント」『Mobile Link クライアント管理』
- 「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』

レッスン 8：同期

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Excel ワークシートの設定」198 ページを参照してください。

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。

♦ クライアント側リモートデータの設定と同期

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=sql"
```

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=sql" -e scn=on -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。
-e scn	SendColumnNames をオンに設定します。これは、カラムを名前で参照する場合にダイレクトローハンドリングで必要となります。
-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 OrderComments;
```

注意
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、mlsrv12 -v+ オプションによっては出力されず、リモートの -v+ オプションによってリモートログに出力されます。

7. 「クリーンアップ」214 ページに進みます。

参照

- 「SQL Anywhere クライアント」『Mobile Link クライアント管理』
- 「Mobile Link SQL Anywhere クライアントユーティリティ (dbmlsync)」『Mobile Link クライアント管理』

クリーンアップ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Excel ワークシートの設定」198 ページを参照してください。

♦ コンピューターからのチュートリアルの削除

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. 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 アドミニストレーターを起動します。
      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*、*remote1.log* を削除します。

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。

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。

必要なソフトウェア

- SQL Anywhere 12
- Java ソフトウェア開発キット
- XML DOM ライブラリ

前提知識と経験

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

概要

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 サーバー 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

参照

- 「Mobile Link 同期」 1 ページ
- 「同期の方法」『Mobile Link サーバー管理』
- 「ダイレクトローハンドリング」『Mobile Link サーバー管理』
- http://www.sybase.com/detail?id=1058600#319 (このページを表示するには、Sybase.com アカウントが必要です。)
- sybase.public.sqlanywhere.mobilink

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。

♦ XML データソースの設定

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* をサーバーサイド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。

3. 「レッスン 2：Mobile Link 統合データベースの設定」216 ページに進みます。

レッスン 2：Mobile Link 統合データベースの設定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：XML データソースの設定」215 ページを参照してください。

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ーが含まれます。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 サーバー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。

このレッスンでは、次のタスクを実行します。

●データベースを作成し、ODBC データソースを定義します。

●同期するデータテーブルをリモートクライアントに追加します。

●Mobile Link のシステムテーブルとストアドプロシージャーをインストールします。

注意
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。

♦ Mobile Link 統合データベースの設定

1. Sybase Central を起動します。

   [スタート] » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Sybase Central]** をクリックします。

2. **[ツール]** » **[SQL Anywhere 12]** » **[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. **[このコンピューター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* と入力します。**[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。**[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 Sybase Central に表示されます。

8. **[データベースの作成]** ウィンドウで **[閉じる]** をクリックします。

9. **[ツール]** » **[SQL Anywhere 12]** » **[ODBC アドミニストレーターを開く]** をクリックします。

10. **[ユーザー 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 Anywhere 12]** 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. **[ユーザー 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 **sql** と入力します。
    f. **[サーバー名]** 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ーを閉じます。

    [ODBC データソースアドミニストレーター] ウィンドウで **[OK]** をクリックします。

14. 「レッスン 3：Mobile Link 統合データベースでのテーブルの作成」217 ページに進みます。

参照

- 「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバー データベース管理』
- 「Mobile Link 統合データベース」『Mobile Link サーバー管理』

レッスン 3：Mobile Link 統合データベースでのテーブルの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：XML データソースの設定」215 ページを参照してください。

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。このテーブルには次のカラムがあります。

カラム	説明
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ーする一般的な方法です。

♦ RemoteOrders テーブルの作成

1. Interactive SQL を使用してデータベースに接続します。

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

   ●Sybase 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ーを作成します。

   C:¥Program Files¥SQL Anywhere 12¥ は、SQL Anywhere 12 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 12¥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ーが作成されます。これらのテーブルとストアドプロシージャーは、同期処理中に Mobile Link サーバーによって使用されます。

4. 「レッスン 4：同期スクリプトの追加」219 ページに進みます。

参照

- 「CREATE TABLE 文」『SQL Anywhere サーバー SQL リファレンス』

レッスン 4：同期スクリプトの追加

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：XML データソースの設定」215 ページを参照してください。

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。

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 サーバーを設定します。

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。 独自の Java クラスを「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」221 ページで作成します。

♦ 統合データベースへの 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の追加

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ーを使用して、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ーを使用して、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。 「必要なスクリプト」『Mobile Link サーバー管理』を参照してください。

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。

7. 「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」221 ページに進みます。

参照

- 「Mobile Link イベントの概要」『Mobile Link サーバー管理』
- 「スクリプトの追加と削除」『Mobile Link サーバー管理』
- 「ローをアップロードするスクリプト」『Mobile Link サーバー管理』
- 「ローをダウンロードするスクリプト」『Mobile Link サーバー管理』
- 「upload_insert テーブルイベント」『Mobile Link サーバー管理』
- 「upload_update テーブルイベント」『Mobile Link サーバー管理』
- 「upload_delete テーブルイベント」『Mobile Link サーバー管理』
- 「download_cursor テーブルイベント」『Mobile Link サーバー管理』
- 「download_delete_cursor テーブルイベント」『Mobile Link サーバー管理』
- 「ダイレクトローハンドリング」『Mobile Link サーバー管理』
- 「ダイレクトアップロードの処理」『Mobile Link サーバー管理』
- 「ダイレクトダウンロードの処理」『Mobile Link サーバー管理』
- 「タイムスタンプベースのダウンロードの実装」『Mobile Link サーバー管理』
- 「リモートデータベース間でのローの分割」『Mobile Link サーバー管理』

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

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：XML データソースの設定」215 ページを参照してください。

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。

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。 完全なリストについては、「MobiLinkOrders の Java コードのリスト」225 ページを参照してください。

♦ ダウンロード専用のダイレクトローハンドリング用の Java クラスの作成

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 ianywhere.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 サーバーによって DBConnectionContext のインスタンスがクラスコンストラクターに渡されます。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ーを作成します。

   クラスコンストラクターが、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* という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場合は、「MobiLinkOrders の Java コードのリスト」 225 ページを参照してください。

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 サーバー API ライブラリを参照する MobiLinkOrders をコンパイルします。

      %SQLANY12%¥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 12¥* を SQL Anywhere 12 ディレクトリに置き換えます。

      ```
      javac -classpath "C:¥Program Files¥SQL Anywhere 12¥java¥mlscript.jar" MobiLinkOrders.java
      ```

10. 「レッスン 6：Mobile Link サーバーの起動」 228 ページに進みます。

参照

- 「ダイレクトローハンドリング」『Mobile Link サーバー管理』
- 「Java による同期スクリプトの作成」『Mobile Link サーバー管理』

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。手順を追った説明については、「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」 221 ページを参照してください。

```
import ianywhere.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();
        }
    }
}
```

レッスン 6：Mobile Link サーバーの起動

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：XML データソースの設定」215 ページを参照してください。

このレッスンでは、Mobile Link サーバーを起動します。-c オプションを使用して Mobile Link サーバー (mlsrv12) 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。

♦ Mobile Link サーバー (mlsrv12) の起動

1. 統合データベースに接続し、mlsrv12 のコマンドラインでクラスをロードします。

   c:¥MLobjxml 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。

   次のコマンドを実行します。

   ```
   mlsrv12 -c "DSN=mlxml_db" -o serverOut.txt -v+ -dl -zu+ -x tcpip  -sl java (-cp c:¥MLobjxml)
   ```

   Mobile Link サーバー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 サーバー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ーを追加します。
-x	Mobile Link クライアントの通信プロトコルとパラメーターを設定します。
-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、またサーバー起動時に Java VM をロードします。

2. 「レッスン 7：Mobile Link クライアントデータベースの設定」229 ページに進みます。

参照

- 「Mobile Link サーバーオプション」『Mobile Link サーバー管理』
- 「-sl java mlsrv12 オプション」『Mobile Link サーバー管理』

レッスン 7：Mobile Link クライアントデータベースの設定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：XML データソースの設定」215 ページを参照してください。

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 サーバーはすべて同じコンピューターに置きます。

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 サーバーでは、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 サーバーでは、特別なイベントを使用して OrderComments テーブルが処理されます。

♦ Mobile Link クライアントデータベースの設定

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 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. dbeng12 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。

   ```
   dbeng12 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=sql"
   ```

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 同期ユーザー、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';
   ```

注意
Mobile Link サーバーに接続する方法は、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。

7. 「レッスン 8：同期」231 ページに進みます。

参照

- 「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバー データベース管理』
- 「Mobile Link クライアント」『Mobile Link クライアント管理』
- 「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』

レッスン 8：同期

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：XML データソースの設定」215 ページを参照してください。

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。

♦ クライアント側リモートデータの設定と同期

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=sql"
   ```

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=sql" -e scn=on -o rem1.txt -v+
   ```

   次のテーブルには、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。
-e scn	SendColumnNames をオンに設定します。これは、カラムを名前で参照する場合にダイレクトローハンドリングで必要となります。
-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 XML ファイルに挿入されました。

6. 「クリーンアップ」232 ページに進みます。

参照

- 「SQL Anywhere クライアント」『Mobile Link クライアント管理』
- 「Mobile Link SQL Anywhere クライアントユーティリティ (dbmlsync)」『Mobile Link クライアント管理』

クリーンアップ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：XML データソースの設定」215 ページを参照してください。

♦ コンピューターからのチュートリアルの削除

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ーを起動します。
      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*、*remote1.log* を削除します。

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。

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。

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。詳細については、http://www.sybase.jp/detail?id=1081142 を参照してください。

注意
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。図やプロシージャーのいくつかは SQL Anywhere 12.0.1 とは異なります。

必要なソフトウェア

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ーに SQL Anywhere (Mobile Link と Sybase 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。

Mobile Link エージェントの展開の詳細については、「SQL Anywhere Mobile Link クライアントの配備」『Mobile Link サーバー管理』と「Ultra Light Mobile Link クライアントの配備」『Mobile Link サーバー管理』を参照してください。

概要

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。

- 「レッスン 1：統合データベースの作成」
- 「レッスン 2：Mobile Link プロジェクトの作成」
- 「レッスン 3：Mobile Link サーバーの起動」
- 「レッスン 4：リモートスキーマ名の定義」
- 「レッスン 5：Mobile Link ユーザーの定義」
- 「レッスン 6：エージェントの定義」
- 「レッスン 7：リモートデバイスでのエージェントの設定」
- 「レッスン 8：同期モデルの作成」
- 「レッスン 9：同期モデルの展開」
- 「レッスン 10：リモートタスクの作成」
- 「レッスン 11：リモートタスクの展開」
- 「レッスン 12：リモートタスクのステータスのチェック」
- 「レッスン 13：リモートデバイスでのリモートデータベースの作成」
- 「レッスン 14：同期のスケジュール」
- 「レッスン 15：スケジュールされた同期の変更」
- 「レッスン 16：即時同期の強制」
- 「レッスン 17：リモートスキーマの変更」
- 「レッスン 18：リモートデータベースの問い合わせ」
- 「レッスン 19：SIRT を使用したファイルのアップロード」

参照

- 「リモートデータベースの集中管理」『Mobile Link サーバー管理』

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：Mobile Link プロジェクトの作成」235 ページに進みます。

♦ 統合データベースの作成

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、中央のサーバー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 consol.db
   start dbeng12 consol.db
   dbdsn -w cadmin_tutorial_consol consol -y -c
   "UID=DBA;PWD=sql;DBF=consol.db;SERVER=consol"
   cd ..
   ```

3. 「レッスン 2：Mobile Link プロジェクトの作成」235 ページに進みます。

レッスン 2：Mobile Link プロジェクトの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成」234 ページを参照してください。

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。

♦ Mobile Link プロジェクトの作成

1. Sybase Central を起動するには、**[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Sybase Central]** をクリックします。
2. 左ウィンドウ枠の **[フォルダー]** ビューで **[Mobile Link 12]** を右クリックし、**[新規]** » **[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デフォルトロケーションをそのまま使用します。**[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、**[統合データベースをプロジェクトに追加]** を選択します。**[データベースの表示名]** に **Tutorial** と入力します。
5. 既存の同期システムがある場合は、**[接続文字列]** フィールドに統合データベースの接続文字列を入力します。それ以外の場合は、**[接続文字列]** に次の値を入力します。

   ```
   UID=DBA;PWD=sql;DSN=cadmin_tutorial_consol
   ```
6. **[パスワードを記憶]** を選択し、**[完了]** をクリックしてウィザードを完了します。
7. 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ーが追加されます。**[はい]** をクリックし、**[OK]** をクリックします。
8. 「レッスン 3：Mobile Link サーバーの起動」235 ページに進みます。

レッスン 3：Mobile Link サーバーの起動

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成」234 ページを参照してください。

Mobile Link サーバーは、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。このレッスンでは、次の手順を使用して Mobile Link サーバーを起動します。

既存の同期システムがある場合は、サーバーがすでに稼働しているため、この項を省略してもかまいません。ただし、サーバーコマンドラインで -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。

♦ Mobile Link サーバーを起動します。

1. 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 mlsrv12.exe -c "DSN=cadmin_tutorial_consol;UID=DBA;PWD=sql" -ftr download -ftru upload -x
tcpip(port=2439) -v+ -ot mlsrv.txt
cd ..
```

   このコマンドは、Mobile Link サーバーを起動し、リモートデバイスとの間でアップロード／ダウンロードするファイルが含まれるアップロードフォルダーとダウンロードディレクトリをそれぞれ作成します。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ーを指定します。
   - **-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 サーバーに接続する方法を定義する通信パラメーターを指定します。
   - **-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。

2. 「レッスン 4：リモートスキーマ名の定義」236 ページに進みます。

レッスン 4：リモートスキーマ名の定義

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」234 ページを参照してください。

次に、リモートスキーマ名を定義します。リモートスキーマ名によって、同じスキーマを共有するリモートアプリケーションデータベースのグループが識別されます。通常、特定のアプリケーションの同じバージョンで使用されるデータベースになります。

♦ リモートスキーマ名の定義

1. Sybase Central の **[フォルダー]** ビューに戻ります。**[Central Admin Tutorial]** で、**[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 v1.0** と入力します。

3. データベースタイプに **[SQL Anywhere]** を選択し、**[完了]** をクリックします。

4. 「レッスン 5：Mobile Link ユーザーの定義」237 ページに進みます。

レッスン 5：Mobile Link ユーザーの定義

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成」234 ページを参照してください。

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 サーバー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 ユーザー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 ユーザーとパスワードは、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。

このレッスンでは、エージェントが使用する Mobile Link ユーザーを定義します。既存の同期システムがあり、既存のいずれかの Mobile Link ユーザーを Mobile Link エージェントが使用して同期する場合は、この項を省略してもかまいません。

♦ Mobile Link ユーザーの定義

1. Sybase Central で、**[ビュー]** » **[フォルダー]** をクリックします。
2. [Mobile Link 12] で、**[Central Admin Tutorial]**、**[統合データベース]**、**Tutorial** の順に展開します。
3. **[ユーザー]** を右クリックし、**[新規]** » **[ユーザー]** をクリックします。

   ユーザー作成ウィザードが表示されます。
4. **[ようこそ]** ページで、新規ユーザーの名前に **JOHN** と入力し、**[次へ]** をクリックします。
5. **[パスワードの指定]** ページで、**[このユーザーは接続時にパスワードが必要]** 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 **sql** 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 サーバー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 ユーザーが登録されます。 「-zu mlsrv12 オプション」『Mobile Link サーバー管理』を参照してください。
6. 「レッスン 6：エージェントの定義」237 ページに進みます。

レッスン 6：エージェントの定義

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成」234 ページを参照してください。

次に、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。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。

♦ エージェントの定義

1. Sybase Central で、**[ビュー]** » **[フォルダー]** をクリックします。
2. [Mobile Link 12] で、**[Central Admin Tutorial]**、**[統合データベース]**、**Tutorial** の順に展開します。
3. **[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。
4. **[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。
5. **[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 ユーザー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。
6. **[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。
7. 既存の同期システムがある場合は、デバイスにすでに存在するリモートデータベースに接続するために Mobile Link エージェントが使用できる接続文字列を **[接続文字列]** フィールドに入力します。

   新しい同期システムの場合は、**[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=dbeng12;SERVER=tutorial_v1;DBF={db_location}¥tutorial_v1.db;UID=DBA;PWD=sql
   ```

   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。
8. **[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。
9. **[エージェント設定]** ページで、**10** と入力し、**[同期ポーリング間隔]** に **[秒]** を選択します。管理ポーリング間隔によって、サーバー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。

   注意
   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、サーバーの負荷が大きくなり、パフォーマンスが低下します。

10. **[完了]** をクリックします。

11. 「レッスン 7：リモートデバイスでのエージェントの設定」239 ページに進みます。

レッスン 7：リモートデバイスでのエージェントの設定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成」234 ページを参照してください。

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 サーバーと同じコンピューターでエージェントを実行します。

♦ リモートデバイスでのエージェントの設定

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 sql
   ```

   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 サーバー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 サーバー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。Sybase 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 ユーザーを指定します。この値は、主にエージェントを認証するために Mobile Link サーバーで使用されます。
   - **-p**　-u オプションで指定された Mobile Link ユーザー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、Sybase Central に戻ります。**[フォルダー]** ビューの **[Mobile Link 12]** で、**[Central Admin Tutorial]**、**[統合データベース]**、**Tutorial**、**[エージェント]** の順に展開します。**AID_JOHN** を選択し、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。

> **注意**
>
> **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 ユーザー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。

5. 「レッスン 8：同期モデルの作成」240 ページに進みます。

レッスン 8：同期モデルの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」234 ページを参照してください。

このレッスンでは、同期モデルを作成します。集中管理を既存の同期システムに追加している場合は、「レッスン 9：同期モデルの展開」241 ページに進みます。

♦ 同期モデルの作成

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。Sybase Central の **[フォルダー]** ビューの **[Mobile Link 12]** で、**[Central Admin Tutorial]**、**[統合データベース]** の順に展開します。**Tutorial** 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. Sybase Central の **[フォルダー]** ビューで、**[Central Admin Tutorial]** を右クリックし、**[新規]** » **[同期モデル]** をクリックします。

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。

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。

10. **[ダウンロードタイプ]** ページで、**[スナップショットダウンロード]** を選択し、**[次へ]** をクリックします。

11. **[削除のダウンロード]** ページで、**[統合データベース上で削除されたデータを、リモートデータベース上で削除しますか ?]** という質問に対して **[いいえ]** と回答し、**[次へ]** をクリックします。

12. **[サブセットのダウンロード]** ページで、**[はい、各リモートデータベースに同じデータをダウンロードします]** をオンにし、**[次へ]** をクリックします。

13. **[アップロード競合の検出]** ページで、**[競合検出を実行しない]** を選択し、**[次へ]** をクリックします。

14. **[パブリケーション、スクリプトバージョン、説明]** ページでデフォルトをそのまま使用し、**[完了]** をクリックします。

    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。

15. 「レッスン 9：同期モデルの展開」241 ページに進みます。

レッスン 9：同期モデルの展開

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成」234 ページを参照してください。

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。

♦ 同期モデルの展開

1. Sybase Central の **[フォルダー]** ビューの **[Mobile Link 12]** 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、**[次の 1 つまたは複数の項目の展開の詳細を指定する]** を選択し、**[統合データベース]**、**[リモートデータベースと同期クライアント]**、および **[リモートデータベースを中央管理する設定で初期化する]** の各オプションをオンにし、**[次へ]** をクリックします。

3. **[統合データベースの展開先]** ページで、**[次の SQL ファイルに変更を保存する]** をクリアし、**[統合データベースに接続して変更を直接適用する]** をオンにします。**Tutorial** データベースを選択し、**[次へ]** をクリックします。

4. **[リモートデータベースの展開]** ページで、**[新しい SQL Anywhere データベース]** を選択し、**[次へ]** をクリックします。

5. **[新しい SQL Anywhere リモートデータベース]** ページで、**[データベースの作成コマンドが含まれたコマンドファイルと SQL ファイルを作成する]** をオンにし、**[SQL ファイル]** フィールドに **c:¥cadmin_demo¥remote_db¥tutorial_v1.sql** と入力します。**[リモートの SQL Anywhere データベースを作成する]** をクリアします。**[次へ]** をクリックして続行し、**[はい]** をクリックして新しいフォルダーとファイルの作成を確認します。

6. **[Mobile Link ユーザーおよび同期プロファイル]** ページでは、デフォルト値をそのまま使用します。

   {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 ユーザー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 ユーザー名に置き換えられ、ここでは JOHN に置き換えられます。

7. **[Mobile Link 認証用にこのユーザーを統合データベースに登録する]** をクリアして、**[次へ]** をクリックします。

8. **[同期ストリームパラメーター]** ページで、**[TCP/IP]** を選択し、**[ポート]** に **2439** と入力します。これは、Mobile Link サーバーを前に起動したときに使用したストリームパラメーターです。**[SQL Anywhere リモート同期クライアントの詳細オプション]** ページが表示されるまで **[次へ]** をクリックし、**[完了]** をクリックしてから **[はい]** をクリックしてディレクトリを作成します。

   同期モデルから移動する場合、変更内容を保存するかどうかが尋ねられます。**[はい]** をクリックします。

9. 「レッスン 10：リモートタスクの作成」243 ページに進みます。

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 *cadmin_demo¥remote_db* 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。

レッスン 10：リモートタスクの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」234 ページを参照してください。

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。

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。

♦ リモートタスクの作成

1. 新しいリモートタスクの作成 Sybase Central の **[フォルダー]** ビューの **[Mobile Link 12]** 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. **[トリガーのメカニズム]** 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. **[フォルダー]** 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ー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。

7. 「レッスン 11：リモートタスクの展開」243 ページに進みます。

レッスン 11：リモートタスクの展開

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」234 ページを参照してください。

前のレッスンでは、リモートタスクを作成しました。ここでは、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する必要があります。

♦ リモートタスクの展開

1. Sybase Central の **[フォルダー]** ビューの **[Mobile Link 12]** 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 サーバー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、**[フォルダー]** ビューのリストに **Hello World** タスクの 2 つのコピーが表示されます。展開済みコピーは、**[リモートタスク]** » **[展開済みタスク]** の **[フォルダー]** 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。

7. 「レッスン 12：リモートタスクのステータスのチェック」245 ページに進みます。

レッスン 12：リモートタスクのステータスのチェック

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」234 ページを参照してください。

次に、Sybase Central でリモートタスクのステータスをチェックします。

♦ リモートタスクのステータスのチェック

1. Sybase Central の **[フォルダー]** ビューの **[Mobile Link 12]** 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。タスクの結果は自動的に再表示されないため、[F5] キーを押してタブを再表示します。**[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。

3. 「レッスン 13：リモートデバイスでのリモートデータベースの作成」245 ページに進みます。

レッスン 13：リモートデバイスでのリモートデータベースの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」234 ページを参照してください。

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、「レッスン 14：同期のスケジュール」246 ページに進みます。

♦ リモートデバイスでのリモートデータベースの作成

1. 新しいリモートタスクの作成 Sybase Central の **[フォルダー]** ビューの **[Mobile Link 12]** 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. **[トリガーのメカニズム]** 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. **[フォルダー]** 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_demo¥remote_db¥tutorial_v1.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. **[フォルダー]** 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. Sybase Central の **[フォルダー]** ビューの **[Mobile Link 12]** で、**[Central Admin Tutorial]**、**[統合データベース]**、**[Tutorial]**、**[エージェント]** の順に展開し、**AID_JOHN** を右クリックします。
   b. **[イベント]** タブを選択し、**[Create DB]** タスクを検索します。結果を再表示するために [F5] キーを押す必要があることがあります。

8. 「レッスン 14：同期のスケジュール」246 ページに進みます。

レッスン 14：同期のスケジュール

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」234 ページを参照してください。

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。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。

♦ 同期のスケジュール

1. 新しいリモートタスクを作成します。Sybase Central の **[フォルダー]** ビューの **[Mobile Link 12]** 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. **[トリガーのメカニズム]** 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. **[フォルダー]** 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。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。

12. 「レッスン 15：スケジュールされた同期の変更」248 ページに進みます。

レッスン 15：スケジュールされた同期の変更

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」234 ページを参照してください。

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。

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。

♦ スケジュールされた同期の変更

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. Sybase Central の **[フォルダー]** ビューの **[Mobile Link 12]** 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. **[フォルダー]** 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。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。
4. 「レッスン 16：即時同期の強制」249 ページに進みます。

レッスン 16：即時同期の強制

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」234 ページを参照してください。

前のレッスンでは、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、サーバー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。

♦ SIRT を使用した即時同期の強制

1. Sybase Central の **[フォルダー]** ビューの **[Mobile Link 12]** 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。

   タスクのすべての受信者は、次にサーバー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ー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。

2. 「レッスン 17：リモートスキーマの変更」249 ページに進みます。

レッスン 17：リモートスキーマの変更

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」234 ページを参照してください。

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、常にユーザーが任意で行います。

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ー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。

このチュートリアルでは、リモートデータベースに新しいテーブルを追加することによって、リモートスキーマを変更します。

♦ リモートスキーマの変更

1. Sybase Central の **[フォルダー]** ビューに戻ります。**[Mobile Link 12]** 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** と入力します。

3. データベースタイプに **[SQL Anywhere]** を選択し、**[完了]** をクリックします。

4. 新しいリモートタスクを作成します。Sybase Central の **[フォルダー]** 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。

5. **[ようこそ]** ページで、**[名前]** フィールドに **Schema Upgrade** と入力します。

6. **[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。

7. **[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。

8. **[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タスク名は **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。

9. **[フォルダー]** ビューの **Central Admin Tutorial** » **[統合データベース]** » **[Tutorial]** » **[エージェント]** で、**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。

10. **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。
11. 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。
12. **[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。
13. 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。
14. 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。
15. 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。
16. **[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。
17. **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。
18. **[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。

    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。
19. 「レッスン 18：リモートデータベースの問い合わせ」251 ページに進みます。

レッスン 18：リモートデータベースの問い合わせ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成」234 ページを参照してください。

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、結果をサーバーに返します。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。

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、ユーザー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。

♦ リモートデータベースの問い合わせ

1. Sybase Central の **[フォルダー]** ビューの **[Mobile Link 12]** 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. **[トリガーのメカニズム]** 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 systable WHERE table_name = 'product'
   go
   SELECT * FROM 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. **[フォルダー]** 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。結果がタブに表示されない場合は、[F5] キーを押して再表示します。

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。必要に応じて、[F5] キーを押して再表示します。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。

12. 「レッスン 19：SIRT を使用したファイルのアップロード」253 ページに進みます。

レッスン 19：SIRT を使用したファイルのアップロード

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」234 ページを参照してください。

このレッスンでは、サーバー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。

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。

♦ ファイルのアップロード

1. Sybase Central の **[フォルダー]** ビューの **[Mobile Link 12]** 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. **[フォルダー]** 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. **[サーバーファイル名]** 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。コマンドエディターで省略記号 (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。

   [サーバーファイル名] 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ー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ー上の異なる場所にアップロードするため、すべての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、Sybase 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。

    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* サブフォルダーがあります。このサブフォルダーに、アップロードした *agent.txt* ファイルがあります。

クリーンアップ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」234 ページを参照してください。

♦ コンピューターからのチュートリアルの削除

1. タスクバー上で、Interactive SQL、Sybase 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ーを起動します。

   b. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[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. **[ユーザーデータ 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。

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。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。

必要なソフトウェア

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ーに 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。

概要

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。

- 「レッスン 1：統合データベースの作成と設定」
- 「レッスン 2：リモートデータベースの作成と設定」
- 「レッスン 3：リモートデータベースの同期」
- 「レッスン 4：リモートデータベースへのデータの挿入」
- 「レッスン 5：統合データベースでのスキーマ変更の実行」
- 「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」
- 「レッスン 7：リモートデータベースへのデータの挿入」
- 「レッスン 8：同期」

参照

- 「リモート MobiLink クライアントでのスキーマの変更」『Mobile Link クライアント管理』

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。

♦ 統合データベースの作成と設定

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 consol.db
dbeng12 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=sql;DBF=consol.db;SERVER=consol"
```

3. データベースを Mobile Link 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ー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" %SQLANY12%¥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 サーバーを起動します。

```
start mlsrv12 -c "DSN=dsn_consol" -v+ -ot mlsrv.txt -zu+
```

7. 「レッスン 2：リモートデータベースの作成と設定」256 ページに進みます。

レッスン 2：リモートデータベースの作成と設定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成と設定」255 ページを参照してください。

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。

♦ リモートデータベースの作成と設定

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 remote.db
dbeng12 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=sql"
```

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 ユーザー、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。

5. 「レッスン 3：リモートデータベースの同期」257 ページに進みます。

レッスン 3：リモートデータベースの同期

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成と設定」255 ページを参照してください。

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。

♦ リモートデータベースの同期

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=sql;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。

4. 「レッスン 4：リモートデータベースへのデータの挿入」258 ページに進みます。

レッスン 4：リモートデータベースへのデータの挿入

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成と設定」255 ページを参照してください。

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。

♦ リモートデータベースへのデータの挿入

1. 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;
```

2. 「レッスン 5：統合データベースでのスキーマ変更の実行」258 ページに進みます。

レッスン 5：統合データベースでのスキーマ変更の実行

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成と設定」255 ページを参照してください。

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。

♦ 統合データベースでのスキーマ変更の実行

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;
```

3. 「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」259 ページに進みます。

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

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成と設定」255 ページを参照してください。

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。

♦ リモートデータベースでのスキーマ変更の実行

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;
   ```

4. 「レッスン 7：リモートデータベースへのデータの挿入」259 ページに進みます。

レッスン 7：リモートデータベースへのデータの挿入

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成と設定」255 ページを参照してください。

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。

♦ **リモートデータベースへのデータの挿入**

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;
   ```

3. 「レッスン 8：同期」260 ページに進みます。

レッスン 8：同期

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成と設定」255 ページを参照してください。

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。

♦ **同期**

● 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=sql;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。

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。

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。

注意
可能な場合は、ScriptVersion 拡張オプションを使用しないことをおすすめします。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。

必要なソフトウェア

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ーに 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。

概要

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。

●「レッスン 1：統合データベースの作成と設定」

●「レッスン 2：リモートデータベースの作成と設定」

●「レッスン 3：リモートデータベースの同期」

●「レッスン 4：統合データベースでのスキーマ変更の実行」

●「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」

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。

♦ 統合データベースの作成と設定

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 consol.db
   dbeng12 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=sql;DBF=consol.db;SERVER=consol"
   ```

3. データベースを Mobile Link 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ー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" %SQLANY12%¥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 サーバーを起動します。

   ```
   start mlsrv12 -c "DSN=dsn_consol" -v+ -ot mlsrv.txt -zu+
   ```

7. 「レッスン 2：リモートデータベースの作成と設定」262 ページに進みます。

レッスン 2：リモートデータベースの作成と設定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成と設定」261 ページを参照してください。

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。

♦ リモートデータベースの作成と設定

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 remote.db
   dbeng12 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=sql"
   ```

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 ユーザー、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。

5. 「レッスン 3：リモートデータベースの同期」263 ページに進みます。

レッスン 3：リモートデータベースの同期

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成と設定」261 ページを参照してください。

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。

♦ リモートデータベースの同期

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=sql;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。

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」263 ページに進みます。

レッスン 4：統合データベースでのスキーマ変更の実行

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成と設定」261 ページを参照してください。

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。

♦ 統合データベースでのスキーマ変更の実行

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;
```

3. 「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」264 ページに進みます。

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

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成と設定」261 ページを参照してください。

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。この操作を行う前に、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。

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。

♦ **リモートデータベースでのスキーマ変更の実行**

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=sql;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。

   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。

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ー上でシミュレートする方法について説明します。

必要なソフトウェア

●SQL Anywhere 12

前提知識と経験

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

概要

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 統合データベースの設定

●Mobile Link サーバー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

参照

●「Mobile Link Replay ユーティリティ (mlreplay)」『Mobile Link サーバー管理』

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。

♦ Mobile Link 統合データベースの設定

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 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。

   ```
   dbeng12 cons.db
   ```

5. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[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. **[ユーザー 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 Anywhere 12]** 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. **[ユーザー 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 **sql** と入力します。
   f. **[アクション]** ドロップダウンリストから、**[このコンピューターで稼動しているデータベースに接続]** を選択します。

g. **[サーバー名]** 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ーを閉じます。

   [ODBC データソースアドミニストレーター] 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ーを作成します。*C:¥Program Files¥SQL Anywhere 12¥* は、SQL Anywhere 12 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 12¥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ーが作成されます。これらのテーブルとストアドプロシージャーは、同期処理中に Mobile Link サーバー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。

14. 「レッスン 2：Mobile Link プロジェクトの作成」267 ページに進みます。

レッスン 2：Mobile Link プロジェクトの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」266 ページを参照してください。

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 Mobile Link プロジェクトの作成**

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[管理ツール]** » **[Sybase Central]** をクリックします。
2. **[ツール]** » **[Mobile Link 12]** » **[新しいプロジェクト]** をクリックします。
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 **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. **[統合データベースをプロジェクトに追加]** オプションをオンにします。
6. **[データベースの表示名]** フィールドに **cons** と入力します。
7. **[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。
8. **[ユーザー ID]** フィールドに **DBA** と入力します。
9. **[パスワード]** フィールドに **sql** と入力します。
10. **[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。
11. **[OK]** をクリックし、**[保存]** をクリックします。
12. **[パスワードを記憶]** オプションをオンにし、**[完了]** をクリックします。
13. **[OK]** をクリックします。
14. 「レッスン 3：同期スクリプトの追加」268 ページに進みます。

レッスン 3：同期スクリプトの追加

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」266 ページを参照してください。

Sybase 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 サーバーを設定します。

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。

♦ 同期スクリプトの追加

1. **[ビュー]** » **[フォルダー]** をクリックします。
2. Sybase Central の左ウィンドウ枠の **[Mobile Link 12]** 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. Sybase Central の左ウィンドウ枠の **[Mobile Link 12]** 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. Sybase Central の左ウィンドウ枠の **[Mobile Link 12]** 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. Sybase 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。「upload_insert テーブルイベント」『Mobile Link サーバー管理』を参照してください。

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. Sybase 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。 download_cursor の詳細については、「download_cursor テーブルイベント」『Mobile Link サーバー管理』を参照してください。

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. Sybase 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。

25. 「レッスン 4：記録のための Mobile Link サーバーの起動」271 ページに進みます。

参照

- 「Mobile Link イベントの概要」『Mobile Link サーバー管理』
- 「スクリプトの追加と削除」『Mobile Link サーバー管理』
- 「ローをアップロードするスクリプト」『Mobile Link サーバー管理』
- 「ローをダウンロードするスクリプト」『Mobile Link サーバー管理』
- 「upload_insert テーブルイベント」『Mobile Link サーバー管理』
- 「upload_update テーブルイベント」『Mobile Link サーバー管理』
- 「upload_delete テーブルイベント」『Mobile Link サーバー管理』
- 「download_cursor テーブルイベント」『Mobile Link サーバー管理』
- 「download_delete_cursor テーブルイベント」『Mobile Link サーバー管理』
- 「ダイレクトローハンドリング」『Mobile Link サーバー管理』
- 「ダイレクトアップロードの処理」『Mobile Link サーバー管理』
- 「ダイレクトダウンロードの処理」『Mobile Link サーバー管理』
- 「タイムスタンプベースのダウンロードの実装」『Mobile Link サーバー管理』
- 「リモートデータベース間でのローの分割」『Mobile Link サーバー管理』

レッスン 4：記録のための Mobile Link サーバーの起動

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」266 ページを参照してください。

このレッスンでは、-c オプションを使って Mobile Link サーバー (mlsrv12) を起動し、統合データベースに接続します。

♦ 記録のための Mobile Link サーバーの起動

1. 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。

   ```
   mlsrv12 -c "DSN=cons" -zu+ -zs mlreplay_svr -x tcpip -ot mlsrv.mls -v+ -rp .
   ```

   Mobile Link サーバーメッセージウィンドウが表示されます。

   使用している Mobile Link サーバー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 サーバー名を設定します。
-zu+	自動的に新しいユーザーを追加します。

2. 「レッスン 5：Mobile Link クライアントデータベースの設定」272 ページに進みます。

参照

- 「Mobile Link サーバーオプション」『Mobile Link サーバー管理』

レッスン 5：Mobile Link クライアントデータベースの設定

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」266 ページを参照してください。

Mobile Link は、統合データベースサーバー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ます。次のタスクを実行する必要があります。

●統合データベースと同期する **T1** テーブルの作成

●同期パブリケーション、同期ユーザー、同期サブスクリプションの作成

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 サーバーはすべて同じコンピューターに置きます。

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 サーバーでは、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ー、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 ユーザーのサブスクリプションです。

♦ Mobile Link クライアントデータベースの設定

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 remote.db
   ```

2. dbeng12 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。

   ```
   dbeng12 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=sql"
   ```

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 同期ユーザー、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。

7. 「レッスン 6：同期の記録」273 ページに進みます。

参照

- 「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバー データベース管理』
- 「Mobile Link クライアント」『Mobile Link クライアント管理』
- 「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』

レッスン 6：同期の記録

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」266 ページを参照してください。

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。

♦ 統合データベースとの同期

1. スキーマが Mobile Link サーバー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=sql" -ot remote1.mls -e
   "sv=MLReplayDemo;scn=on" -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	同期するスクリプトバージョンを指定すると、そのカラム名は mlreplay で使用するためにアップロードに送信されます。
-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。

   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 で **remote** データベースに接続します。

   ```
   dbisql -c "SERVER=remote;UID=DBA;PWD=sql"
   ```

3. リプレイセッション中に Mobile Link サーバー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=sql" -ot remote2.mls -e
   "sv=MLReplayDemo;scn=on" -v+
   ```

5. 「レッスン 7：リプレイのための Mobile Link サーバーの再起動」275 ページに進みます。

参照

- 「SQL Anywhere クライアント」『Mobile Link クライアント管理』
- 「Mobile Link SQL Anywhere クライアントユーティリティ (dbmlsync)」『Mobile Link クライアント管理』

レッスン 7：リプレイのための Mobile Link サーバーの再起動

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」266 ページを参照してください。

このレッスンでは、記録を停止するために Mobile Link サーバーを停止してから、-rp オプションを指定せずにサーバーを再起動して、サーバーでリプレイの準備をします。

♦ リプレイのための Mobile Link サーバーの停止と再起動

1. 次のコマンドを実行して、Mobile Link サーバーである **mlreplay_svr** を停止します。

   ```
   mlstop -w -t 1m mlreplay_svr
   ```

   Mobile Link サーバーは、同期記録とともに停止します。

   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、サーバー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ーをシャットダウンするように指定します。

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。

   ```
   mlsrv12 -c "DSN=cons" -zu+ -zs mlreplay_svr -x tcpip -ot server_replay.mls -v+
   ```

   Mobile Link サーバーメッセージウィンドウが表示されます。

   使用している Mobile Link サーバー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 サーバー名を設定します。

オプション	説明
-zu+	自動的に新しいユーザーを追加します。

3. 「レッスン 8：同期のリプレイ」276 ページに進みます。

参照

- 「Mobile Link サーバーオプション」『Mobile Link サーバー管理』

レッスン 8：同期のリプレイ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」266 ページを参照してください。

このレッスンでは、スキーマが Mobile Link サーバー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。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。

♦ 複数のクライアント間での同期のリプレイ

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=sql" -ot remote3.mls -e
"sv=MLReplayDemo;scn=on" -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	同期するスクリプトバージョンを指定すると、そのカラム名は mlreplay で使用するためにアップロードに送信されます。
-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. 作業ディレクトリに、そのファイルを *mlreplay.csv* として保存します。

   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 サーバー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. テキストエディターでログファイルを開き、Mobile Link リプレイの結果を確認します。

6. 「クリーンアップ」277 ページに進みます。

参照

- 「Mobile Link Replay ユーティリティ (mlreplay)」『Mobile Link サーバー管理』

クリーンアップ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：Mobile Link 統合データベースの設定」266 ページを参照してください。

♦ コンピューターからのチュートリアルの削除

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ーを起動します。
   b. **[スタート]** » **[プログラム]** » **[SQL Anywhere 12]** » **[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. **[ユーザーデータソース]** 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。

索引

記号

A

C

D

E

I

J

L

M

O

P

S

U

X

あ

ち

つ

て

と

は

ふ

ま

も

り

ろ