iAnywhere SOLUTIONS™
A SYBASE COMPANY

Mobile Link チュートリアル

パート番号：DC00206-01-0902-01
改訂：2005 年 3 月

目次

はじめに

このマニュアルの内容 このマニュアルは、Mobile Link 同期テクノロジの使用方法について説明したチュートリアルです。

始める前に Mobile Link の詳細については、『Mobile Link 管理ガイド』> 「Mobile Link 同期について」を参照してください。

SQL Anywhere Studio のマニュアル

このマニュアルは、SQL Anywhere のマニュアル・セットの一部です。この項では、マニュアル・セットに含まれる各マニュアルと使用法について説明します。

SQL Anywhere Studio のマニュアル

SQL Anywhere Studio のマニュアルは、各マニュアルを 1 つの大きなヘルプ・ファイルにまとめたオンライン形式、マニュアル別の PDF ファイル、および有料の製本版マニュアルで提供されます。SQL Anywhere Studio のマニュアルは、次の分冊マニュアルで構成されています。

- **『SQL Anywhere Studio の紹介』** このマニュアルでは、SQL Anywhere Studio のデータベース管理と同期テクノロジの概要について説明します。また、SQL Anywhere Studio を構成する各部分について説明するチュートリアルも含まれています。
- **『SQL Anywhere Studio 新機能ガイド』** このマニュアルは、SQL Anywhere Studio のこれまでのリリースのユーザを対象としています。ここでは、製品の今回のリリースと以前のリリースで導入された新機能をリストし、アップグレード手順を説明しています。
- **『Adaptive Server Anywhere データベース管理ガイド』** このマニュアルでは、データベースおよびデータベース・サーバの実行、管理、設定について説明しています。
- **『Adaptive Server Anywhere SQL ユーザーズ・ガイド』** このマニュアルでは、データベースの設計と作成の方法、データのインポート・エクスポート・変更の方法、データの検索方法、ストアド・プロシージャとトリガの構築方法について説明します。
- **『Adaptive Server Anywhere SQL リファレンス・マニュアル』** このマニュアルは、Adaptive Server Anywhere で使用する SQL 言語の完全なリファレンスです。また、Adaptive Server Anywhere のシステム・テーブルとシステム・プロシージャについても説明しています。
- **『Adaptive Server Anywhere プログラミング・ガイド』** このマニュアルでは、C、C++、Java プログラミング言語を使用してデータベース・アプリケーションを構築、配備する方法につい

て説明します。Visual Basic や PowerBuilder などのツールのユーザは、それらのツールのプログラミング・インタフェースを使用できます。また、Adaptive Server Anywhere ADO.NET データ・プロバイダについても説明します。

- **『Adaptive Server Anywhere エラー・メッセージ』** このマニュアルでは、Adaptive Server Anywhere エラー・メッセージの完全なリストを、その診断情報とともに説明します。
- **『SQL Anywhere Studio セキュリティ・ガイド』** このマニュアルでは、Adaptive Server Anywhere データベースのセキュリティ機能について説明します。Adaptive Server Anywhere 7.0 は、米国政府から TCSEC (Trusted Computer System Evaluation Criteria) の C2 セキュリティ評価を授与されています。このマニュアルには、Adaptive Server Anywhere の現在のバージョンを、C2 基準を満たした環境と同等の方法で実行することを望んでいるユーザにとって役に立つ情報が含まれています。
- **『Mobile Link 管理ガイド』** このマニュアルでは、モバイル・コンピューティング用の Mobile Link データ同期システムについてあらゆる角度から説明します。このシステムによって、Oracle、Sybase、Microsoft、IBM の単一データベースと、Adaptive Server Anywhere や Ultra Light の複数データベースの間でのデータ共有が可能になります。
- **『Mobile Link クライアント』** このマニュアルでは、Adaptive Server Anywhere リモート・データベースと Ultra Light リモート・データベースの設定を行い、これらを同期させる方法について説明します。
- **『Mobile Link サーバ起動同期ユーザーズ・ガイド』** このマニュアルでは、Mobile Link のサーバによって開始される同期について説明します。サーバによって開始される同期とは、統合データベースから同期の開始を可能にする Mobile Link の機能です。
- **『QAnywhere ユーザーズ・ガイド』** このマニュアルでは、Mobile Link QAnywhere について説明します。Mobile Link QAnywhere は、従来のデスクトップ・クライアントやラップ

トップ・クライアントだけでなく、モバイル・クライアントや無線クライアント用のメッセージング・アプリケーションの開発と展開を可能にするメッセージング・プラットフォームです。

- **『Mobile Link およびリモート・データ・アクセスの ODBC ドライバ』** このマニュアルでは、Mobile Link 同期サーバから、または Adaptive Server Anywhere リモート・データ・アクセスによって、Adaptive Server Anywhere 以外の統合データベースにアクセスするための ODBC ドライバの設定方法について説明します。

- **『SQL Remote ユーザーズ・ガイド』** このマニュアルでは、モバイル・コンピューティング用の SQL Remote データ・レプリケーション・システムについて、あらゆる角度から説明します。このシステムによって、Adaptive Server Anywhere または Adaptive Server Enterprise の単一データベースと Adaptive Server Anywhere の複数データベースの間で、電子メールやファイル転送などの間接的リンクを使用したデータ共有が可能になります。

- **『SQL Anywhere Studio ヘルプ』** このマニュアルには、Sybase Central や Interactive SQL、その他のグラフィカル・ツールに関するコンテキスト別のヘルプが含まれています。これは、製本版マニュアル・セットには含まれていません。

- **『Ultra Light データベース・ユーザーズ・ガイド』** このマニュアルは、Ultra Light 開発者を対象としています。ここでは、Ultra Light データベース・システムの概要について説明します。また、すべての Ultra Light プログラミング・インタフェースに共通する情報を提供します。

- **Ultra Light のインタフェースに関するマニュアル** 各 Ultra Light プログラミング・インタフェースには、それぞれに対応するマニュアルを用意しています。これらのインタフェースは、RAD (ラピッド・アプリケーション開発) 用の Ultra Light コンポーネントとして提供されているものと、C、C++、Java 開発用の静的インタフェースとして提供されているものがあります。

このマニュアル・セットの他に、PowerDesigner と InfoMaker には、独自のオンライン・マニュアル (英語版) がそれぞれ用意されています。

マニュアルの形式

SQL Anywhere Studio のマニュアルは、次の形式で提供されています。

- **オンライン・マニュアル**　オンライン・マニュアルには、SQL Anywhere Studio の完全なマニュアルがあり、SQL Anywhere ツールに関する印刷マニュアルとコンテキスト別のヘルプの両方が含まれています。オンライン・マニュアルは、製品のメンテナンス・リリースごとに更新されます。これは、最新の情報を含む最も完全なマニュアルです。

  Windows オペレーティング・システムでオンライン・マニュアルにアクセスするには、[スタート] ー [プログラム] ー [SQL Anywhere 9] ー [オンライン・マニュアル] を選択します。オンライン・マニュアルをナビゲートするには、左ウィンドウ枠で HTML ヘルプの目次、索引、検索機能を使用し、右ウィンドウ枠でリンク情報とメニューを使用します。

  UNIX オペレーティング・システムでオンライン・マニュアルにアクセスするには、SQL Anywhere のインストール・ディレクトリに保存されている HTML マニュアルを参照してください。

- **PDF 版マニュアル**　SQL Anywhere の各マニュアルは、Adobe Acrobat Reader で表示できる PDF ファイルで提供されています。

  PDF 版マニュアルは、オンライン・マニュアルまたは Windows の [スタート] メニューから利用できます。

- **製本版マニュアル**　製本版マニュアルをご希望の方は、ご購入いただいた販売代理店または弊社営業担当までご連絡ください。

表記の規則

この項では、このマニュアルで使用されている書体およびグラフィック表現の規則について説明します。

SQL 構文の表記規則

SQL 構文の表記には、次の規則が適用されます。

- **キーワード** SQL キーワードはすべて次の例に示す ALTER TABLE のように大文字で表記します。

  ALTER TABLE [*owner*.]*table-name*

- **プレースホルダ** 適切な識別子または式で置き換えられる項目は、次の例に示す *owner* や *table-name* のように表記します。

  ALTER TABLE [*owner*.]*table-name*

- **繰り返し項目** 繰り返し項目のリストは、次の例に示す *column-constraint* のように、リストの要素の後ろに省略記号 (ピリオド 3 つ …) を付けて表します。

  ADD *column-definition* [*column-constraint*, …]

  複数の要素を指定できます。複数の要素を指定する場合は、各要素間をカンマで区切る必要があります。

- **オプション部分** 文のオプション部分は角カッコで囲みます。

  RELEASE SAVEPOINT [*savepoint-name*]

  この例では、角カッコで囲まれた *savepoint-name* がオプション部分です。角カッコは入力しないでください。

- **オプション** 項目リストから 1 つだけ選択するか、何も選択しなくてもよい場合は、項目間を縦線で区切り、リスト全体を角カッコで囲みます。

  [**ASC** | **DESC**]

  この例では、ASC と DESC のどちらか 1 つを選択しても、どちらも選択しなくてもかまいません。角カッコは入力しないでください。

- **選択肢**　オプションの中の 1 つを必ず選択しなければならない場合は、選択肢を中カッコで囲み、縦棒で区切ります。

  [**QUOTES** { **ON** | **OFF** }]

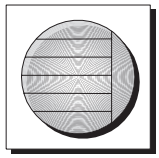
  QUOTES オプションを使用する場合は、ON または OFF のどちらかを選択する必要があります。角カッコと中カッコは入力しないでください。

グラフィック・アイコン

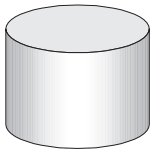
このマニュアルでは、次のアイコンを使用します。

- クライアント・アプリケーション

- Sybase Adaptive Server Anywhere などのデータベース・サーバ



- データベース。高度な図では、データベースとデータベースを管理するデータ・サーバの両方をこのアイコンで表します。



- レプリケーションまたは同期のミドルウェア。ソフトウェアのこれらの部分は、データベース間のデータ共有を支援します。たとえば、Mobile Link 同期サーバ、SQL Remote Message Agent などがあげられます。

- プログラミング・インタフェース

詳細情報の検索／フィードバックの提供

詳細情報の検索

詳しい情報やリソース (コード交換など) については、iAnywhere Developer Network (http://www.ianywhere.com/developer/) を参照してください。

ご質問がある場合や支援が必要な場合は、次に示す iAnywhere Solutions ニュースグループのいずれかにメッセージをお寄せください。

ニュースグループにメッセージをお送りいただく際には、ご使用の SQL Anywhere Studio バージョンのビルド番号を明記し、現在発生している問題について詳しくお知らせくださいますようお願いいたします。バージョン情報は、コマンド・プロンプトで **dbeng9 -v** と入力して確認できます。

ニュースグループは、ニュース・サーバ *forums.sybase.com* にあります (ニュースグループにおけるサービスは英語でのみの提供となります)。以下のニュースグループがあります。

- sybase.public.sqlanywhere.general
- sybase.public.sqlanywhere.linux
- sybase.public.sqlanywhere.mobilink
- sybase.public.sqlanywhere.product_futures_discussion
- sybase.public.sqlanywhere.replication
- ianywhere.public.sqlanywhere.ultralite

ニュースグループに関するお断り

iAnywhere Solutions は、ニュースグループ上に解決策、情報、または意見を提供する義務を負うものではありません。また、システム・オペレータ以外のスタッフにこのサービスを監視させて、操作状況や可用性を保証する義務もありません。

iAnywhere Solutions のテクニカル・アドバイザとその他のスタッフは、時間のある場合にかぎりニュースグループでの支援を行います。こうした支援は基本的にボランティアで行われるため、解決策や情報を定期的に提供できるとはかぎりません。支援できるかどうかは、スタッフの仕事量に左右されます。

フィードバック

このマニュアルに関するご意見、ご提案、フィードバックをお寄せください。

マニュアルに関するご意見、ご提案は、SQL Anywhere ドキュメンテーション・チームの iasdoc@ianywhere.com 宛てに電子メールでお寄せください。このアドレスに送信された電子メールに返信はいたしませんが、お寄せいただいたご意見、ご提案は必ず読ませていただきます。

マニュアルまたはソフトウェアについてのフィードバックは、上記のニュースグループを通してお寄せいただいてもかまいません。

第 1 章

チュートリアル：Mobile Link の概要

この章の内容

この章は、Adaptive Server Anywhere データベースを使用する同期システムの設定プロセスについて理解していただくための、入門用チュートリアルです。

概要

このチュートリアルでは、Adaptive Server Anywhere を使用して統合データベースとリモート・データベースを作成します。その後、Mobile Link 同期テクノロジを使用して、これらのデータベースを同期させます。

所要時間

チュートリアルはおよそ 30 分で終了します。

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。

- 統合システムとしての Mobile Link 同期サーバとクライアント
- Mobile Link 同期サーバとクライアントのコマンド・ラインとオプション
- Adaptive Server Anywhere 用の ODBC 接続を作成し、そのプロパティを設定する方法
- データベースを初期化する方法

主要な概念

Mobile Link 同期サーバは、ODBC を使用して統合データベースに接続します。Mobile Link 同期クライアントは、リモート・データベースに接続します。Mobile Link 同期サーバとクライアントはグループになって機能し、1 つのデータベースから別のデータベースへのデータのアップロードとダウンロードを管理します。

関連項目

Mobile Link のアーキテクチャの詳細については、『Mobile Link 管理ガイド』>「同期の基本」を参照してください。

SQL コマンドを実行できる Adaptive Server Anywhere ユーティリティの詳細については、『SQL Anywhere Studio の紹介』>「Interactive SQL の使用」を参照してください。

レッスン 1：データベースの作成と移植

Mobile Link 同期を使用するには、統合データベース、1 つ以上のリモート・データベース、各データベースの ODBC データ・ソースが存在していることが必要です。

データベース・ファイルの作成

最初の手順として、データベース・ファイルを作成します。この手順では、コマンド・ラインから dbinit ユーティリティを使用して、統合データベースとリモート・データベースを構築します。

dbinit ユーティリティは、ユーザ・テーブルもプロシージャもないデータベース・ファイルを作成します。新しく初期化したファイル内でユーザ定義テーブルとプロシージャを定義するときに、データベース・スキーマを作成します。

❖ データベースを作成するには、次の手順に従います。

1. コマンド・プロンプトを開き、SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥AutoScripting* サブディレクトリに移動します。
2. このチュートリアル用の統合データベースを作成します。次のコマンド・ラインを実行します。

   ```
   dbinit consol.db
   ```

   このチュートリアルを以前にコンピュータで実行した場合は、*consol.db* と *consol.log* が存在している可能性があります。この場合は、*dbinit* が失敗します。これらのファイルを削除してから *dbinit* を実行してください。
3. このチュートリアル用のリモート・データベースを作成します。次のコマンド・ラインを実行します。

   ```
   dbinit remote.db
   ```

   このチュートリアルを以前にコンピュータで実行した場合は、*remote.db* と *remote.log* が存在している可能性があります。この場合は、*dbinit* が失敗します。これらのファイルを削除してから *dbinit* を実行してください。

4 ディレクトリの内容一覧を表示して、データベース・ファイルが正常に作成されたかどうかを確認します。一覧に *consol.db* と *remote.db* が含まれていることを確認してください。

ODBC データ・ソースの作成

これで ODBC データ・ソースを構築する準備が整いました。この ODBC データ・ソースから、Adaptive Server Anywhere データベースに接続できます。

❖ ODBC データ・ソースを作成するには、次の手順に従います。

1 コマンド・プロンプトを開き、SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥AutoScripting* サブディレクトリに移動します。

2 次のコマンド・ラインを実行して、統合データベース用の ODBC データ・ソースを作成します。

```
dbdsn -w test_consol -y -c
"uid=DBA;pwd=SQL;dbf=consol.db;eng=Consol"
```

このコマンド・ラインは以下に示す dbdsn のオプションを指定します。

- **-w** データ・ソース定義を作成します。
- **-y** 確認メッセージを表示せずにデータ・ソースを削除または上書きします。
- **-c** 接続パラメータを接続文字列として指定します。

詳細については、『ASA データベース管理ガイド』> 「データ・ソース・ユーティリティ オプション」を参照してください。

3 次のコマンド・ラインを実行して、リモート・データベース用の ODBC データ・ソースを作成します。

```
dbdsn -w test_remote -y -c
"uid=DBA;pwd=SQL;dbf=remote.db;eng=Remote"
```

スキーマの作成

次の手順では、Interactive SQL ユーティリティを使用して SQL 文を実行することによって、テーブルを作成し、それを統合データベースに移植します。また、テーブルを作成し、同期サブスクリプションと同期パブリケーションをリモート・データベースに挿入します。

この処理には、2 つの定義済み SQL ファイル *build_consol.sql* と *build_remote.sql* を使用します。これらのファイルをテキスト・エディタで開いて内容をよく確認することをおすすめします。

❖ スキーマを作成するには、次の手順に従います。

1. コマンド・プロンプトを開き、SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥AutoScripting* サブディレクトリに移動します。
2. 次のコマンド・ラインを実行します。

   ```
   dbisql -c "dsn=test_consol;astop=no"
   build_consol.sql
   ```

   build_consol.sql の SQL 文は emp テーブルと cust テーブルを作成して、それを統合データベースに移植します。

   この手順には、*dbisql* ユーティリティが停止したときにサーバが停止しないように指示する `astop=no` が含まれています。
3. 次のコマンド・ラインを実行します。

   ```
   dbisql -c "dsn=test_remote;astop=no"
   build_remote.sql
   ```

   build_remote.sql の SQL 文は、リモート・テーブル emp と cust を作成し、同期サブスクリプションと同期パブリケーションを挿入します。
4. 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に emp テーブルと cust テーブルが作成されたことを確認します。

- コマンド・プロンプトで *dbisql* と入力して Interactive SQL を開きます。test_consol DSN を使用して DBA として接続します。パスワードには SQL を使用します。
- 次の SQL 文を [SQL 文] ウィンドウ枠に入力し、[F9] キーを押して実行します。

  ```
  SELECT * FROM emp, cust
  ```

  統合データベース内のテーブルにはデータが移植されています。
- test_remote DSN を使用して接続し、次の SQL 文を実行します。

  ```
  SELECT * FROM emp, cust
  ```

  リモート・データベース内のテーブルは空です。

5 次のレッスン用に、統合データベースとリモート・データベースを稼働させたままにしておきます。

詳細情報

データベースの作成方法の詳細については、『ASA データベース管理ガイド』>「初期化ユーティリティ」と『ASA データベース管理ガイド』>「dbinit コマンド・ライン・ユーティリティを使用したデータベースの作成」を参照してください。

Interactive SQL の実行方法の詳細については、『ASA データベース管理ガイド』>「Interactive SQL ユーティリティ」と『SQL Anywhere Studio の紹介』>「Interactive SQL の使用」を参照してください。

SELECT 文の詳細については、『ASA SQL リファレンス・マニュアル』>「SELECT 文」を参照してください。

ODBC データ・ソースの作成方法の詳細については、『ASA データベース管理ガイド』>「データ・ソース・ユーティリティ」を参照してください。

リモート・データベースの作成方法の詳細については、『Mobile Link クライアント』>「リモート・データベースの作成」を参照してください。

サブスクリプションとパブリケーションの作成方法の詳細については、『Mobile Link クライアント』>「データのパブリッシュ」を参照してください。

レッスン 2：Mobile Link 同期サーバの実行

統合データベースを稼働状態にしてから、Mobile Link 同期サーバを起動してください。レッスン 1 の後で統合データベースを停止した場合は、統合データベースを再起動してください。

❖ Mobile Link 同期サーバを起動するには、次の手順に従います。

1 コマンド・プロンプトを開き、SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥AutoScripting* サブディレクトリに移動します。

2 次のコマンド・ラインを実行します。

```
dbmlsrv9 -c "dsn=test_consol" -o mlserver.mls -v+ -
dl -za -zu+
```

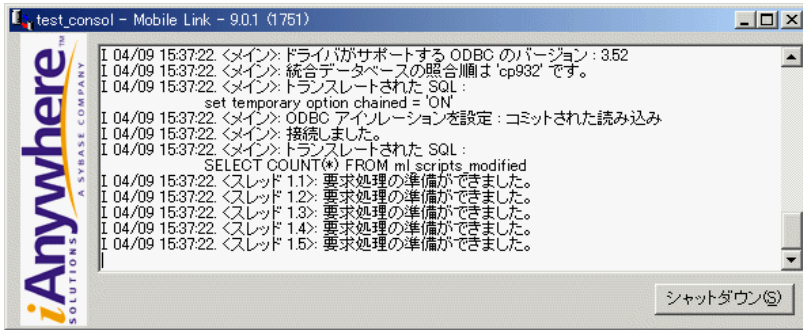
このコマンド・ラインは、以下に示す dbmlsrv9 のオプションを指定します。

- **-c** Mobile Link 同期サーバの接続文字列には統合データベースの DSN を使用します。詳細については、『Mobile Link 管理ガイド』>「-c オプション」を参照してください。
- **-o** -o オプションでは、メッセージ・ログ・ファイルを指定します。詳細については、『Mobile Link 管理ガイド』>「-o オプション」を参照してください。
- **-v+** -v+ オプションは、冗長ロギングをオンに設定します。詳細については、『Mobile Link 管理ガイド』>「-v オプション」を参照してください。
- **-dl** -dl オプションは、ログ表示機能をオンに設定します。詳細については、『Mobile Link 管理ガイド』>「-dl オプション」を参照してください。
- **-za** -za オプションは、スクリプトの自動生成をオンに設定します。詳細については、『Mobile Link 管理ガイド』>「-za オプション」を参照してください。

- **-zu+**　-zu+ オプションは、ユーザ認証処理を自動化します。詳細については、『Mobile Link 管理ガイド』> 「-zu オプション」を参照してください。

  上記のコマンドでオプション -o、-v、-dl が指定されているのは、デバッグとトラブルシューティングの情報を表示するためです。通常これらのオプションは運用環境では使用しません。

Mobile Link 同期サーバ・コマンドを実行すると、次の出力が表示されます。



dbmlsrv9 を実行したときに Mobile Link 同期サーバがすでに稼働していると、エラー・メッセージが表示されます。Mobile Link の現在のインスタンスを停止して、再びコマンドを実行してください。

詳細情報

Mobile Link 同期サーバの詳細については、『Mobile Link 管理ガイド』> 「Mobile Link 同期サーバ」を参照してください。

dbmlsrv9 のオプションの完全なリストについては、『Mobile Link 管理ガイド』> 「Mobile Link 同期サーバのオプション」を参照してください。

レッスン 3：Mobile Link 同期クライアントの実行

Adaptive Server Anywhere クライアントは dbmlsync ユーティリティを使用して Mobile Link 同期を開始します。

❖ Mobile Link 同期クライアントを起動するには、次の手順に従います。

1 コマンド・プロンプトを開き、SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥AutoScripting* サブディレクトリに移動します。

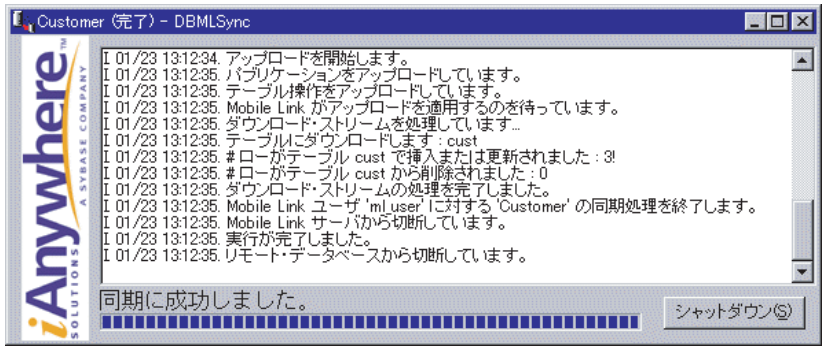
2 次のコマンド・ラインを実行します。

```
dbmlsync -c "dsn=test_remote" -o dbmlsync.out -v -e
"SendColumnNames=ON"
```

このコマンド・ラインは以下のオプションを指定します。

- **-c** データベース接続パラメータを指定します。詳細については、『Mobile Link クライアント』> 「-c オプション」を参照してください。
- **-o** メッセージ・ログ・ファイルを指定します。詳細については、『Mobile Link クライアント』> 「-o オプション」を参照してください。
- **-v** 冗長オペレーション。詳細については、『Mobile Link クライアント』> 「-v オプション」を参照してください。
- **-e** 拡張オプション。"SendColumnNames=ON" を指定すると、カラム名が Mobile Link に送信されます。dbmlsrv9 コマンド・ラインで -za を使用するときに、このオプションが必要です。詳細については、『Mobile Link クライアント』> 「SendColumnNames (scn) 拡張オプション」を参照してください。

Mobile Link 同期クライアント・コマンドを実行すると、同期の成功を示す下記の出力が表示されます。同期の後で、リモート・データベースに統合データベースのデータが移植されます。



詳細情報

dbmlsync のオプションの詳細については、『Mobile Link クライアント』>「Mobile Link 同期クライアント」を参照してください。

リモート・クライアントの詳細については、『Mobile Link 管理ガイド』>「Mobile Link クライアント」を参照してください。

dbmlsync の詳細については、『Mobile Link クライアント』>「同期の開始」を参照してください。

クリーンアップ

このチュートリアルで作成したデータベースと ODBC データ・ソースは削除してください。

❖ チュートリアルをコンピュータから削除するには、次の手順に従います。

1. コマンド・プロンプトを開き、SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥AutoScripting* サブディレクトリに移動します。
2. ファイル *clean.bat* を実行します。

まとめ

このチュートリアルでは、以下の作業を行いました。

- 新しい Adaptive Server Anywhere 統合データベースとリモート・データベースの作成と移植
- Mobile Link 同期サーバの起動
- Mobile Link 同期クライアントの起動と、リモート・データベースと統合データベースの同期

学習の成果

このチュートリアルでは、次の項目を学習しました。

- 統合システムとしての Mobile Link 同期サーバとクライアントの知識の習得
- Mobile Link 同期サーバとクライアントのコマンドの実行方法の習得
- Mobile Link 同期サーバとクライアントのコマンド・ラインとオプションの知識の習得

詳細情報

さらに詳細な情報については、まず次の役立つ情報を参照してください。

Adaptive Server Anywhere リモート・データベースの詳細については、『Mobile Link クライアント』> 「Adaptive Server Anywhere クライアント」を参照してください。

Mobile Link 同期サーバの実行方法の詳細については、『Mobile Link 管理ガイド』> 「同期の基本」を参照してください。

同期スクリプトの作成方法の詳細については、『Mobile Link 管理ガイド』> 「同期スクリプトの作成」を参照してください。

スクリプトの作成方法と追加方法を説明している上級向けのチュートリアルについては、「チュートリアル：Mobile Link スクリプトの作成と同期のモニタリング」15 ページを参照してください。

第 2 章

チュートリアル：Mobile Link スクリプトの作成と同期のモニタリング

この章の内容

この章は、競合検出スクリプトや競合解決スクリプトなどの同期スクリプトを作成するプロセスについて理解していただくためのチュートリアルです。この章では同期をモニタリングする方法についても説明します。

概要

所要時間

チュートリアルはおよそ 120 分で終了します。

目的

このチュートリアルの目的は、次のタスクに関する知識と経験を得ることです。

- 統合データベースのスキーマをリモート・データベースに移行する。
- Sybase Central または Interactive SQL を使用して同期に必要な基本スクリプトを作成し、統合データベースに保存する。
- 競合の検出と解決のためのスクリプトを作成する。
- ログ・ファイルと Mobile Link モニタを使用して同期をモニタリングする。

関連項目

Mobile Link のアーキテクチャの詳細については、『Mobile Link 管理ガイド』>「同期の基本」を参照してください。

同期スクリプトの詳細については、『Mobile Link 管理ガイド』>「同期スクリプトの概要」を参照してください。

Interactive SQL の詳細については、『SQL Anywhere Studio の紹介』>「Interactive SQL の使用」を参照してください。

Sybase Central の詳細については、『SQL Anywhere Studio の紹介』>「Sybase Central を使用したデータベースの管理」を参照してください。

同期の概要については、「チュートリアル：Mobile Link の概要」1 ページを参照してください。

Mobile Link での競合解決の詳細については、『Mobile Link 管理ガイド』>「競合の解決」を参照してください。

Mobile Link モニタの詳細については、『Mobile Link 管理ガイド』>「Mobile Link モニタ」を参照してください。

レッスン 1：Adaptive Server Anywhere 統合データベースの設定

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

1. 統合データベースとスキーマを作成します。
2. 統合データベース用の ODBC データ・ソースを定義します。

統合データベースの作成

次の手順では、Sybase Central の [データベース作成] ウィザードを使用して統合データベースを作成します。

❖ Adaptive Server Anywhere RDBMS を作成するには、次の手順に従います。

1 Sybase Central を起動します。

[スタート] － [プログラム] － [SQL Anywhere 9] － [Sybase Central] を選択します。

Sybase Central が表示されます。

2 Sybase Central から、[ツール] － [Adaptive Server Anywhere 9] － [データベースの作成] を選択します。

[データベース作成] ウィザードが表示されます。[次へ] をクリックします。

3 [このコンピュータにデータベースを作成] をデフォルトのままにします。[次へ] をクリックします。

4 データベースのファイル名とパスを入力します。たとえば、次のように入力します。

C:¥temp¥cons.db

5 ウィザードの指示に従って残りの操作を進めます。次の設定を除き、デフォルト値をそのまま使用してください。

- ページ・サイズは 4096 バイトを選択します。

  多くの環境では、ページ・サイズを 4K にするとパフォーマンスが高まります。

6 [完了] をクリックし、Sybase Central 内で新しいデータベースに接続します。

統合データベース・スキーマの生成

統合データベース・スキーマに、ハードウェア製品の名前と数量を格納する Product テーブルを追加します。

❖ 統合データベース・スキーマに Product テーブルを追加する場合は、次の手順に従います。

1 Interactive SQL を起動します。

- Sybase Central の左側のウィンドウ枠で、**cons - DBA** データベースを選択します。[ファイル] ー [Interactive SQL を開く] を選択します。

  Interactive SQL が表示されます。

2 Product テーブルを作成します。

- Interactive SQL で次のコマンドを実行します。

```
/* the Product table */
create table Product (
  name        varchar(128) not null primary key,
  quantity        integer,
  last_modified    timestamp default timestamp
)
go




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);

go
```

3　競合解決に使用するテンポラリ・テーブルを作成します。

「レッスン 6：競合検出と競合解決のためのスクリプトの作成」35 ページでは、競合が発生したときにこれらのテーブルに値を挿入するためのスクリプトを作成します。

```
/* the Product_old table */
create table Product_old (
   name         varchar(128) not null primary key,
   quantity         integer,
   last_modified    timestamp default timestamp
 )
go


/* the Product_new table */
create table Product_new (
   name         varchar(128) not null primary key,
   quantity         integer,
   last_modified    timestamp default timestamp
)
go
```

4　各テーブルの作成が成功したことを検証します。

- たとえば、Product テーブルの内容を検証するには、Interactive SQL で次のコマンドを実行します。

```
select * from Product
```

統合データベース用の ODBC データ・ソースの定義

Adaptive Server Anywhere 9.0 ドライバを使用して、cons データベース用の ODBC データ・ソースを定義します。

❖ 統合データベース用の ODBC データ・ソースを定義するには、次の手順に従います。

1　ODBC アドミニストレータを起動します。

[スタート] ー [プログラム] ー [SQL Anywhere 9] ー [Adaptive Server Anywhere] ー [ODBC アドミニストレータ] を選択します。

[ODBC データ・ソース・アドミニストレータ] が表示されます。

2 [ユーザ DSN] タブで [追加] をクリックします。

[データ・ソースの新規作成] ダイアログが表示されます。

3 [Adaptive Server Anywhere 9.0] を選択して [完了] をクリックします。

[Adaptive Server Anywhere 9 の ODBC 設定] ダイアログが表示されます。

4 [ODBC] タブで、データ・ソース名 **asa_cons** を入力します。[ログイン] タブで、ユーザ ID として **DBA**、パスワードとして **SQL** を入力します。[データベース] タブで、サーバ名として **cons** を入力します。

5 [OK] をクリックして、データ・ソースを追加します。

6 [OK] をクリックして [ODBC アドミニストレータ] を閉じます。

詳細情報

統合データベースの作成には、コマンド・ライン・ユーティリティ dbinit も使用できます。詳細については、『ASA データベース管理ガイド』>「dbinit コマンド・ライン・ユーティリティを使用したデータベースの作成」または「レッスン 2：Adaptive Server Anywhere リモート・データベースの設定」22 ページを参照してください。

統合データベースの詳細については、ASA 以外の RDBMS が統合データベースの場合も含め、『Mobile Link 管理ガイド』>「統合データベース」を参照してください。

Interactive SQL の詳細については、『ASA データベース管理ガイド』>「Interactive SQL ユーティリティ」を参照してください。

ODBC データ・ソースの作成方法の詳細については、『ASA データベース管理ガイド』>「データ・ソース・ユーティリティ」を参照してください。

レッスン 2：Adaptive Server Anywhere リモート・データベースの設定

Mobile Link は、統合データベース・サーバと多数のモバイル・データベースとの間で同期を実行できるように設計されています。この項では、リモート・データベースを 2 つ作成します。データベースごとに次の作業を行う必要があります。

- 統合スキーマ内の特定部分の移行
- リモート同期パブリケーション、同期ユーザ、同期サブスクリプションの作成

Adaptive Server Anywhere データベースの作成

レッスン 1 では、Sybase Central を使用してデータベースを作成しました。ここでは、コマンド・ライン・ユーティリティを使用します。この 2 つのツールのどちらを使用しても結果は同じです。

❖ Adaptive Server Anywhere リモート・データベースを作成して起動するには、次の手順に従います。

1. コマンド・プロンプトで、リモート・データベースを作成するディレクトリに移動します。
2. 次のコマンドを入力して、データベースを作成します。

   ```
   dbinit -p 4096 remote1.db
   ```

   次のように入力して、remote2 も作成します。

   ```
   dbinit -p 4096 remote2.db
   ```

   ここでは、-p オプションを使用してページ・サイズを 4K に設定しています。このサイズに設定すると、多くの環境でパフォーマンスが向上します。

   dbinit のオプションの詳細については、『ASA データベース管理ガイド』>「dbinit コマンド・ライン・ユーティリティを使用したデータベースの作成」を参照してください。
3. 次のように入力してデータベースを起動します。

```
dbeng9 remote1.db
```

次のように入力して、remote2 も起動します。

```
dbeng9 remote2.db
```

統合データベース・スキーマのサブセットの移行

統合データベース・スキーマのサブセットを移行するには、次の作業を行う必要があります。

- リモート・データベースへの接続
- リモート・サーバと外部ログインの作成
- Sybase Central の [データベース移行] ウィザードの実行

❖ 統合データベース・スキーマのサブセットを remote1 に移行するには、次の手順に従います。

1. Sybase Central を起動します。

   [スタート] － [プログラム] － [SQL Anywhere 9] － [Sybase Central] を選択します。

2. 次の手順でリモート・データベースに接続します。

   - Sybase Central の左ウィンドウ枠で [Adaptive Server Anywhere 9] を選択します。
   - [ファイル] － [接続] を選択します。

     [接続] ダイアログが表示されます。

   - [ID] タブで、ユーザ ID として **DBA**、パスワードとして **SQL** を入力します。[データベース] タブで、サーバ名として **remote1** を入力します。
   - [OK] をクリックして接続します。

3. 次の手順でリモート・サーバを作成します。

   - [リモート・サーバ作成] ウィザードを起動します。

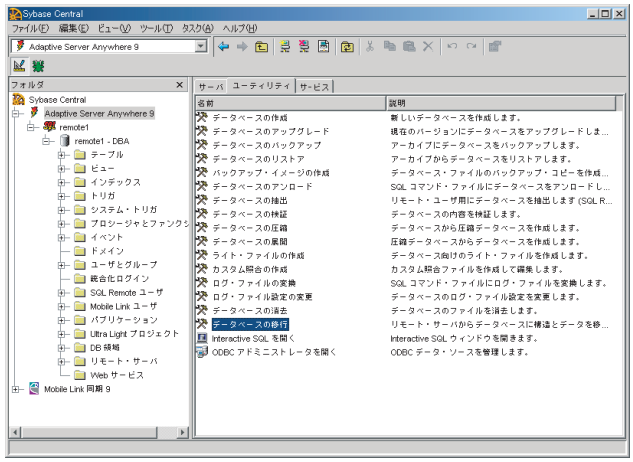
左ウィンドウ枠で [リモート・サーバ] フォルダを選択します。[ファイル] ー [新規] ー [リモート・サーバ] を選択します。

[リモート・サーバ作成] ウィザードが起動します。

- リモート・サーバに **my_asa** という名前を付けます。[次へ] をクリックして続行します。
- サーバの種類として Sybase Adaptive Server Anywhere を選択し、[次へ] をクリックします。
- ウィザードの次のページで、接続情報セクションにデータ・ソース名として **asa_cons** を入力し、[次へ] をクリックします。
- 最後に表示されるウィザードで、[現在のユーザの外部ログインを作成する] を選択します。ログイン名として **DBA**、パスワードとして **SQL** を使用します。
- [完了] をクリックして [リモート・サーバ作成] ウィザードを終了します。

4 次の手順で統合データベース・スキーマを移行します。

- 左ウィンドウ枠で Adaptive Server Anywhere 9 プラグインを選択します。右ウィンドウ枠で [ユーティリティ] タブを選択し、[データベースの移行] をダブルクリックします。



[データベース移行] ウィザードが起動します。

- 送信先データベースとして **remote1** を選択します。
- 次のページで、リモート・サーバとして **my_asa** を選択します。[次へ] をクリックして続行します。
- 移行するテーブルとして **Product** のみを選択し、[次へ] をクリックします。
- ユーザ DBA を選択して [次へ] をクリックします。
- ウィザードの最後のページで、[データの移行] オプションをオフにします。
- [完了] をクリックします。

5　remote2 データベースに対して、移行の操作 (上記の手順 1 ～ 4) を繰り返します。

同期サブスクリプションとパブリケーション

パブリケーションは、リモート・データベース上の同期対象となるテーブルとカラムを識別します。これらのテーブルとカラムを「**アーティクル**」と呼びます。同期サブスクリプションは、パブリケーションに対する Mobile Link ユーザのサブスクリプションです。

同期サブスクリプションとパブリケーションはリモート・データベースに格納されます。

パブリケーションとサブスクリプションの定義方法の詳細については、『Mobile Link クライアント』> 「データのパブリッシュ」を参照してください。

❖ リモート同期パブリケーション、同期ユーザ、同期サブスクリプションを作成するには、次の手順に従います。

1 Interactive SQL を起動します。

- Sybase Central の左側のウィンドウ枠で、**remote1 - DBA** データベースを選択します。[ファイル] ー [Interactive SQL を開く] を選択します。

2 次の手順で remote1 の同期情報を入力します。

- Interactive SQL で次のコードを実行します。

```
CREATE PUBLICATION pub_1 (TABLE Product);
CREATE SYNCHRONIZATION USER user_1;
CREATE SYNCHRONIZATION SUBSCRIPTION TO pub_1
 FOR user_1 TYPE TCPIP ADDRESS 'host=localhost'
 OPTION scriptversion='ver1';
```

3 Interactive SQL を起動し、remote2 に接続します。

4 次の手順で remote2 の同期情報を入力します。

- Interactive SQL で次のコードを実行します。

```
CREATE PUBLICATION pub_2 (TABLE Product);
CREATE SYNCHRONIZATION USER user_2;
CREATE SYNCHRONIZATION SUBSCRIPTION TO pub_2
 FOR user_2 TYPE TCPIP ADDRESS 'host=localhost'
 OPTION scriptversion='ver1';
```

これで、リモート・データベースと統合データベースの準備が完了しました。次のレッスンでは、同期スクリプトを作成します。レッスン 4 では同期を実行します。

レッスン 3：同期スクリプトの作成

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。この項では、次の同期スクリプトを作成します。

- **upload_insert** 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どのように適用するかを定義します。
- **download_cursor** 統合データベースからダウンロードする必要があるデータを定義します。

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。

❖ スクリプト・バージョンを追加するには、次の手順に従います。

1. Sybase Central の Mobile Link プラグインを使用して cons データベースに接続します。
   - Sybase Central の左ウィンドウ枠で Mobile Link Synchronization 9 プラグインを選択します。
   - [ツール] ー [接続] を選択します。

     [新しい接続] ダイアログが表示されます。
   - [ID] タブで、[ODBC データ・ソース名] オプションを選択します。データ・ソース名として **asa_cons** を入力します。
   - [OK] をクリックして接続します。

     Mobile Link プラグインに **asa_cons** データ・ソースが表示されます。

     Mobile Link プラグインの詳細については、『SQL Anywhere Studio ヘルプ』> 「Mobile Link のヘルプ」を参照してください。

2 スクリプト・バージョン ver1 を追加します。

左ウィンドウ枠の [バージョン] フォルダを選択します。右ウィンドウ枠で、[バージョンを追加] をダブルクリックします。

[新しいスクリプト・バージョンを追加] ダイアログが表示されます。

3 新しいバージョンに **ver1** という名前を付け、[完了] をクリックします。

❖ 統合データベースに同期テーブルを追加するには、次の手順に従います。

1 Sybase Central の Mobile Link 同期プラグインで、[テーブル (所有者)] フォルダを開いて [DBA] をダブルクリックします。

2 Product テーブルを右クリックして [同期テーブルに追加] を選択します。

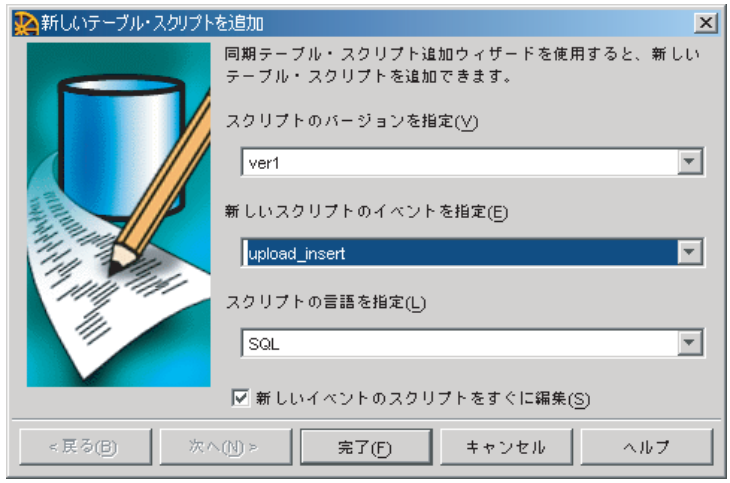
[同期テーブル] フォルダに Product テーブルが表示されます。

これで、同期テーブルの指定が完了したので、アップロード用とダウンロード用のテーブル・スクリプトを統合データベースに新しく追加できます。

❖ Product テーブルのテーブル・スクリプトを追加するには、次の手順に従います。

1 Sybase Central の Mobile Link 同期プラグインで、[同期テーブル] フォルダを開き、Product テーブルを選択します。

2 右ウィンドウ枠で、[テーブル・スクリプトを追加] をダブルクリックします。次のダイアログが表示されます。スクリプト・バージョンとして ver1 が表示されていることを確認します。



3　ドロップダウン・リストから **upload_insert** イベントを選択し、[完了] をクリックします。

[Product upload_insert] ダイアログが表示されます。

4　編集画面に次の SQL 文を入力します。

```
INSERT INTO Product( name, quantity, last_modified )
   VALUES( ?, ?, ? )
```

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 の詳細については、『Mobile Link 管理ガイド』>「upload_insert テーブル・イベント」を参照してください。

5　スクリプトを保存します。

[ファイル] ー [保存] を選択します。

6　次の SQL 文を使用して、**download_cursor** イベントに対して手順 1 ～ 5 を繰り返します。

```
SELECT name, quantity, last_modified
   FROM Product where last_modified >= ?
```

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、『Mobile Link 管理ガイド』>「download_cursor テーブル・イベント」を参照してください。

詳細情報

ここで作成したスクリプトの詳細については、『Mobile Link 管理ガイド』>「upload_insert テーブル・イベント」と『Mobile Link 管理ガイド』>「download_cursor テーブル・イベント」を参照してください。

スクリプト・バージョンの詳細については、『Mobile Link 管理ガイド』>「スクリプト・バージョン」を参照してください。

スクリプトの追加方法の詳細については、『Mobile Link 管理ガイド』>「統合データベースでのスクリプトの追加と削除」を参照してください。

テーブル・スクリプトの作成方法の詳細については、『Mobile Link 管理ガイド』>「テーブル・スクリプト」を参照してください。

同期スクリプトの作成方法の詳細については、『Mobile Link 管理ガイド』>「同期スクリプトの作成」を参照してください。

同期をカスタマイズするために設定できるイベントの完全なリストについては、『Mobile Link 管理ガイド』>「同期イベント」を参照してください。

レッスン 4：Mobile Link 同期の実行

❖ **remote1 と remote2 の同期を実行するには、次の手順に従います。**

1　Mobile Link 同期サーバを起動します。

コマンド・プロンプトで、次のコマンドを 1 行で入力します。

```
dbmlsrv9 -c "dsn=asa_cons" -o mlserver.mls -v+ -dl -
zu+ -x tcpip
```

dbmlsrv9 のオプションの完全なリストについては、『Mobile Link 管理ガイド』> 「Mobile Link 同期サーバのオプション」を参照してください。

Mobile Link サーバ・ウィンドウが表示されます。これは、Mobile Link 同期サーバで要求を処理する準備ができたことを示しています。

2　Mobile Link 同期クライアント・ユーティリティ (dbmlsync) を実行して、同期を開始します。

コマンド・プロンプトで、次のコマンドを 1 行で入力します。

```
dbmlsync -c "eng=remote1;uid=dba;pwd=sql" -o
rem1.txt -v+
```

remote2 については、次のように入力します。

```
dbmlsync -c "eng=remote2;uid=dba;pwd=sql" -o
rem2.txt -v+
```

dbmlsync のオプションの完全なリストについては、『Mobile Link クライアント』> 「Adaptive Server Anywhere クライアントの同期パラメータ」を参照してください。

Mobile Link 同期クライアントの起動が完了すると、Mobile Link の同期が成功したことを示す DBMLSync ウィンドウが表示されます。

詳細情報

Mobile Link 同期サーバの詳細については、『Mobile Link 管理ガイド』>「Mobile Link 同期サーバ」を参照してください。

dbmlsrv9 のオプションの完全なリストについては、『Mobile Link 管理ガイド』>「Mobile Link 同期サーバのオプション」を参照してください。

dbmlsync の詳細については、『Mobile Link クライアント』>「Adaptive Server Anywhere クライアント」を参照してください。

dbmlsync のオプションの完全なリストについては、『Mobile Link クライアント』>「Adaptive Server Anywhere クライアントの同期パラメータ」を参照してください。

レッスン 5：ログ・ファイルを使用した Mobile Link 同期のモニタリング

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 2 つのメッセージ・ログ・ファイル、つまり *mlserver.mls* と *rem1.txt* (または *rem2.txt*) を使用して、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のロケーションは、コマンドが実行されたディレクトリです。

❖ Mobile Link 同期ログ・ファイル内でエラーを探すには、次の手順に従います。

1　ログ・ファイルをテキスト・エディタで開きます。このチュートリアルでは、ログ・ファイルは *mlserver.mls* です。

2　このファイル内で文字列「同期サーバが起動しました。」を検索します。

3　ファイルの左側を下へスキャンします。*I.* で始まる行は情報メッセージ、*E.* で始まる行はエラー・メッセージです。次に例を示します。

```
I. 04/27 16:01:00. <Main>: ODBC isolation set to: Read Committed
I. 04/27 16:01:00. <Main>: Connected
I. 04/27 16:01:00. <Main>: Translated SQL:
                   SELECT COUNT(*) FROM ml_scripts_modified
E. 04/27 16:01:01. <Main>: Error: Unable to initialize communications stream 1: tcpip
I. 04/27 16:01:01. <Main>: Synchronization Server shutting down
I. 04/27 16:01:03. <Main>: Disconnected
I. 04/27 16:01:03. <Main>: Synchronization Server finished
```

4　この例では、*E.* の横に次のテキストがあります。

```
04/27 16:01:01. <メイン>: エラー: 通信ストリーム 1 を初期化できません : tcpip
```

このメッセージは、アップロードとダウンロードの前のエラーを示します。同期サブスクリプションまたはパブリケーションの定義にエラーが含まれる可能性があります。

5　次の形で始まる句を検索します。

```
ASA Synchronization request from:
```

この句は、同期要求が確立されたことを示します。

6 「要求を処理中」で始まる句を検索します。これは、クライアントとサーバが通信していることを示します。このメッセージは、高レベルの冗長性を指定している場合に表示されることがあります。

❖ Mobile Link 同期クライアントのログ・ファイル内でエラーを検出するには、次の手順に従います。

1 クライアント・ログ・ファイル *rem1.txt* をテキスト・エディタで開きます。

2 このファイル内で文字列「コミット」を検索します。この文字列がある場合、同期は成功しています。

3 このファイル内で文字列「ロールバック」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。

4 ファイルの左側を下へスキャンします。*E.* と表示されている場合、エラーが発生しています。エラーがない場合、同期は正常に完了しています。

詳細情報

Mobile Link 同期サーバのログ・ファイルの詳細については、『Mobile Link 管理ガイド』>「Mobile Link 同期サーバの動作のロギング」を参照してください。

レッスン 6：競合検出と競合解決のためのスクリプトの作成

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します。異なるリモート・データベースで 2 人のユーザが同じローを修正した場合、Mobile Link 同期サーバに 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。

Mobile Link での競合解決の詳細については、『Mobile Link 管理ガイド』>「競合の解決」を参照してください。

例 (在庫管理)

現場に販売担当者が 2 人いるとします。最初の在庫値は 10 個で、販売担当者 1 はそのうちの 3 個を販売しました。この担当者は、リモート・データベース remote1 に記録されている在庫値を 7 に更新します。販売担当者 2 は 4 個を販売し、remote2 に記録されている在庫値を 6 に更新します。

Mobile Link 同期クライアント・ユーティリティを使用して remote1 の同期が実行されると、統合データベース内の在庫値は 7 に更新されます。remote2 の同期が実行されると、競合が検出されます。これは、統合データベース内の在庫値が変更されているためです。

この競合をプログラムで解決するには、次のような 3 つのロー値が必要となります。

1. 統合データベースにある現在の値

   remote1 の同期が実行された後の統合データベース内の値は 7 です。

2. remote2 がアップロードした新しいローの値

3. remote2 が直前の同期の間に取得した古いローの値

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

統合データベースの現在の値 –（リモート・データベースの古い値 – リモート・データベースの新しい値）
すなわち、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、在庫値ではなく、販売担当者 2 が販売した個数です。

競合の検出と解決のための同期スクリプト

競合を検出して解決するには、次のスクリプトを追加します。

- **upload_update** upload_update 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。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 を使用して競合を検出する方法の詳細については、『Mobile Link 管理ガイド』>「競合の検出」を参照してください。

  upload_update の詳細については、『Mobile Link 管理ガイド』>「upload_update テーブル・イベント」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 の詳細については、『Mobile Link 管理ガイド』>「upload_old_row_insert テーブル・イベント」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 の詳細については、『Mobile Link 管理ガイド』>「upload_new_row_insert テーブル・イベント」を参照してください。

- **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 の詳細については、『Mobile Link 管理ガイド』 > 「resolve_conflict テーブル・イベント」を参照してください。

❖ 競合の検出と解決のためのスクリプトを実装するには、次の手順に従います。

1　Interactive SQL を起動します。

- コマンド・プロンプトで次のコマンドを入力します。

  ```
  dbisql
  ```

  [接続] ダイアログが表示されます。

- [ID] タブで、ユーザ ID として **DBA**、パスワードとして **SQL** を入力します。[データベース] タブで、サーバ名として **cons** を入力します。

2　競合の検出と解決のためのスクリプトを実装します。

Interactive SQL で次のコードを実行します。

```
/* upload_update */
call ml_add_table_script( 'ver1', 'Product',
  'upload_update',
  'UPDATE Product
   SET quantity = ?, last_modified = ?
     WHERE name = ?
     AND quantity=? AND last_modified=?' )
go



 /* upload_old_row_insert */
 call ml_add_table_script( 'ver1', 'Product',
  'upload_old_row_insert',
  'INSERT INTO Product_old
(name,quantity,last_modified)
    values (?,?,?)')
go
```

```
 /* upload_new_row_insert */
 call ml_add_table_script( 'ver1', 'Product',
  'upload_new_row_insert',
  'INSERT INTO Product_new
(name,quantity,last_modified)
    values (?,?,?)')
go


 /* resolve_conflict */
 call ml_add_table_script( 'ver1', '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
    ')
```

Adaptive Server Anywhere 統合データベースの設定はこれで完了です。

詳細情報

Mobile Link での競合の検出と解決の詳細については、『Mobile Link 管理ガイド』>「競合の解決」を参照してください。

Mobile Link 統合データベースの詳細については、『Mobile Link 管理ガイド』>「Mobile Link 統合データベース」を参照してください。

レッスン 7：Mobile Link モニタによる更新競合の検出

Mobile Link モニタを使用すると、同期が行われたときにその統計情報を収集できます。このモニタのグラフィック・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。

このモニタを使用することによって、エラーを引き起こす同期を迅速に突き止めたり、特定の条件を満たしたりできます。Mobile Link モニタ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、開発時、運用時ともこのモニタを使用することをお勧めします。

この項では、次のことを学習します。

- Mobile Link モニタを起動し、更新の競合が発生している同期をはっきりと識別できるように設定する。
- remote1 と remote2 上の同じローを更新して競合を発生させる。
- Mobile Link モニタを使用して競合を検出する。

❖ 更新の競合を検出できるように Mobile Link モニタを設定するには、次の手順に従います。

1 Mobile Link モニタを起動します。

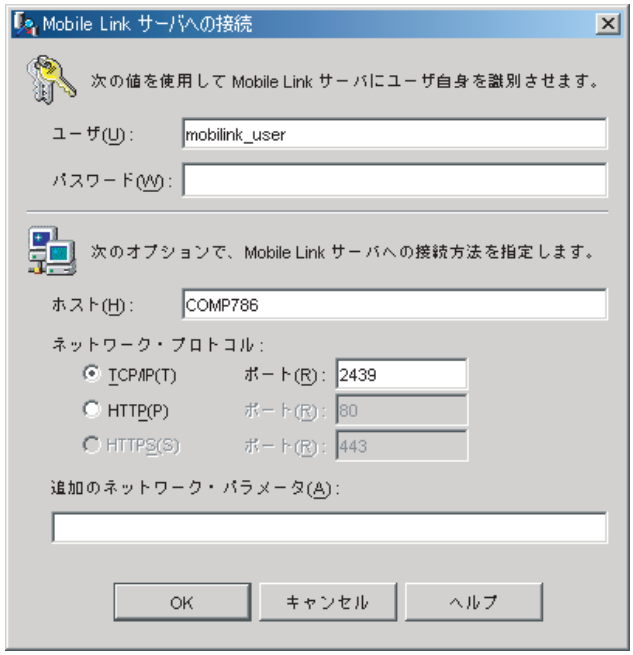
- [スタート] ー [プログラム] - [SQL Anywhere 9] ー [Mobile Link] ー [Mobile Link モニタ] を選択します。または、コマンド・プロンプトで次のように入力します。

  ```
  dbmlmon
  ```

  [Mobile Link サーバへの接続] ダイアログが表示されます。

- ユーザ ID として **monitor_user** を入力します。-zu+ オプションを指定して Mobile Link 同期サーバを起動したので、このユーザは自動的に Mobile Link ユーザとして追

加されます。このユーザにパスワードを必要としない場合は、パスワード用のフィールドをブランクのままにしてください。



- [OK] をクリックして接続します。

Mobile Link モニタが Mobile Link 同期サーバ (dbmlsrv9) に接続します。

2　Mobile Link モニタのウォッチ・マネージャを起動します。

Mobile Link モニタで、[ファイル] ー [ツール] ー [ウォッチ・マネージャ] を選択します。

[ウォッチ・マネージャ] ダイアログが表示されます。

3　更新の競合をモニタリングする新しいウォッチを追加します。

- [新規] をクリックします。

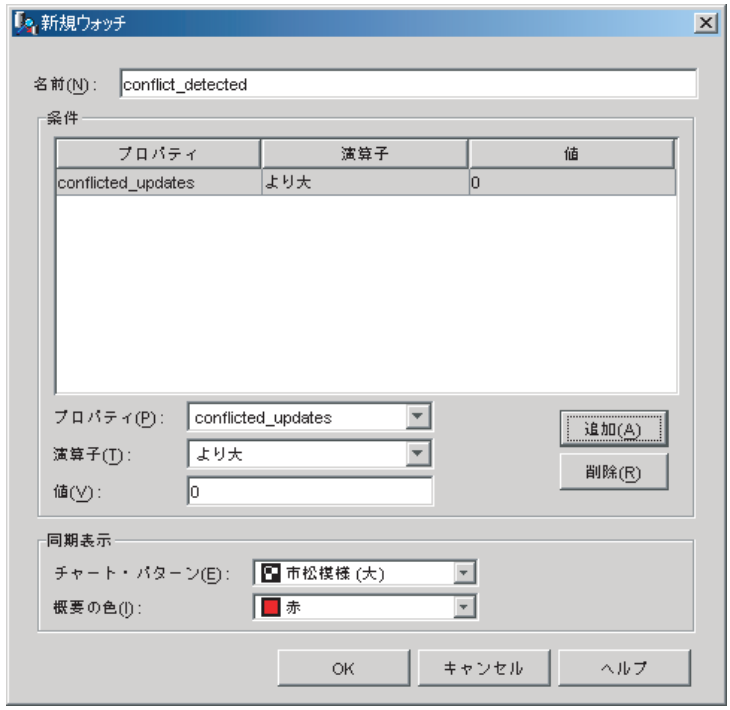
[新規ウォッチ] ダイアログが表示されます。

- ウォッチに conflict_detected という名前を付けます。
- [プロパティ] フィールドで **conflicted_updates** を選択します。

  統計のプロパティ conflicted_updates は、アップロードされた更新のうち、競合が検出された更新の数を示します。

  Mobile Link モニタの統計のプロパティの詳細については、『Mobile Link 管理ガイド』> 「Mobile Link の統計のプロパティ」を参照してください。
- 更新の競合が 1 つ以上発生したケースを検出するようにウォッチを設定します。

  [演算子] フィールドを [**より大**] に設定します。[値] フィールドを **0** に設定します。
- [追加] をクリックして設定を保存します。
- [チャート] ウィンドウ枠のウォッチのパターンを選択します。([チャート] ウィンドウ枠は、Mobile Link モニタの中央のウィンドウ枠です)。
- [概要] ウィンドウ枠のウォッチの色を選択します ([概要] ウィンドウ枠は、Mobile Link モニタの下部のウィンドウ枠です)。



4　[OK] をクリックしてウォッチを追加します。

❖ 更新の競合を生成するには、次の手順に従います。

1　remote1 の在庫値を更新します。

Screwmaster Drill の最初の在庫値は 10 個で、販売担当者 1 がそのうちの 3 個を販売しました。この担当者は、リモート・データベース remote1 に記録されている在庫値を 7 に更新します。この更新は次の手順で行います。

- Interactive SQL を起動します。remote1 にまだ接続していない場合は接続します。

  コマンド・プロンプトで次のコマンドを入力します。

  ```
  dbisql
  ```

[接続] ダイアログが表示されます。

[ID] タブで、ユーザ ID として **DBA**、パスワードとして **SQL** を入力します。[データベース] タブで、サーバ名として **remote1** を入力します。

- Screwmaster Drill の在庫値を 7 個に更新します。

  Interactive SQL で次のコードを実行します。

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

2 remote1 の同期を実行します。

コマンド・プロンプトで次のように入力して、Mobile Link 同期クライアントを起動します。

```
dbmlsync -c "eng=remote1;uid=dba;pwd=sql" -v+
```

同期が完了すると、統合データベース内の Screwmaster Drill の在庫値は 7 個に更新されます。

3 remote2 の在庫値を更新します。

販売担当者 2 は 4 個を販売し、remote2 に記録されている在庫値を 6 に更新します。remote2 の同期が実行されると、競合が検出されます。これは、統合データベース内の在庫値が変更されているためです。この更新は次の手順で行います。

- Interactive SQL を起動し、remote2 に接続します。

  コマンド・プロンプトで次のコマンドを入力します。

```
dbisql
```

[接続] ダイアログが表示されます。

[ID] タブで、ユーザ ID として **DBA**、パスワードとして **SQL** を入力します。[データベース] タブで、サーバ名として **remote2** を入力します。

- Screwmaster Drill の在庫値を 6 個に更新します。

  Interactive SQL で次のコードを実行します。

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

4　remote2 の同期を実行します。

コマンド・プロンプトで次のように入力して、Mobile Link 同期クライアントを起動します。

```
dbmlsync -c "eng=remote2;uid=dba;pwd=sql" -v+
```

以上の操作が終わると、Mobile Link モニタに切り替えて同期の結果を確認できます。

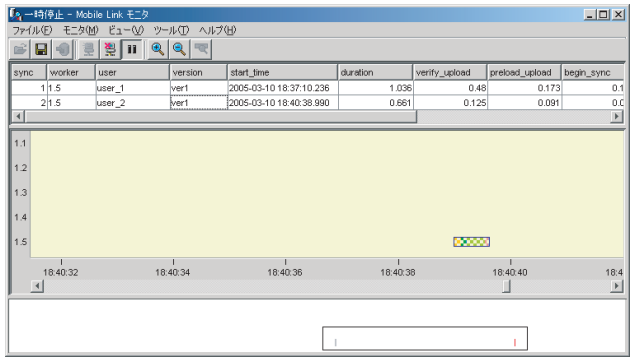
❖ Mobile Link モニタを使用して更新の競合を検出するには、次の手順に従います。

1　チャート・スクロールを一時的に停止します。

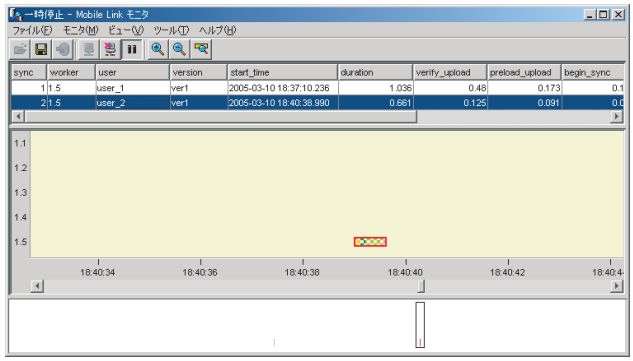
[ファイル] ー [モニタ] ー [チャート・スクロールの一時停止] を選択します。

2　Mobile Link モニタの [概要] ウィンドウ枠、[チャート] 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。

- モニタの [概要] ウィンドウ枠 (Mobile Link モニタの下部のウィンドウ枠) で同期を見つけます。更新の競合を発生させた remote2 の同期は、赤で表示されます。



- [チャート] ウィンドウ枠で 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。

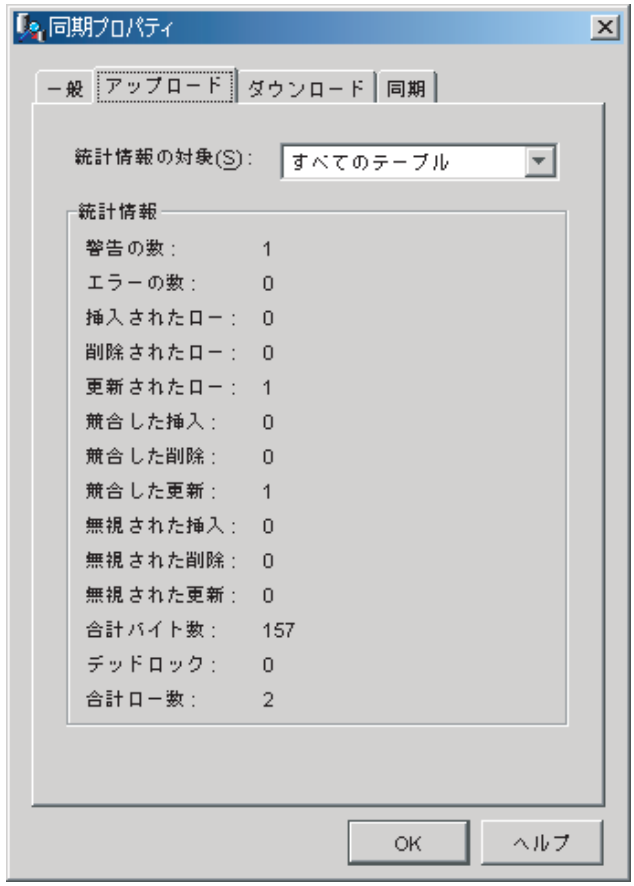


conflict_detected ウォッチ用に指定したパターンが適用された状態で、同期オブジェクトが表示されます。

- 同期の詳細を確認するには、ズーム・ツールを使用します。

  [ファイル] ー [ビュー] ー [ズーム・イン] を選択します。

- 同期プロパティを表示するには、同期オブジェクトか、詳細テーブル内の対応するローをダブルクリックします。競合が起きている更新の数を確認するには、[アップロード] タブを選択します。



詳細情報

Mobile Link での競合解決の詳細については、『Mobile Link 管理ガイド』>「競合の解決」を参照してください。

Mobile Link モニタの詳細については、『Mobile Link 管理ガイド』>「Mobile Link モニタ」を参照してください。

Mobile Link モニタの統計のプロパティの詳細については、『Mobile Link 管理ガイド』>「Mobile Link の統計のプロパティ」を参照してください。

チュートリアルのクリーンアップ

コンピュータからチュートリアルを削除してください。

❖ チュートリアルをコンピュータから削除するには、次の手順に従います。

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
   - Sybase Central
   - Interactive SQL
2. タスクバー上で、Adaptive Server Anywhere、Mobile Link、同期クライアントの各項目を右クリックし、[閉じる] を選択して閉じます。
3. 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。
   - ODBC アドミニストレータを起動します。

     [スタート] － [プログラム] － [SQL Anywhere 9] － [Adaptive Server Anywhere] － [ODBC アドミニストレータ] を選択します。

     ユーザ・データ・ソースのリストから asa_cons を選択します。[削除] をクリックします。
4. 統合データベースとリモート・データベースを削除します。
   - 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。
   - *cons.db*、*cons.log*、*remote1.db*、*remote1.log*、*remote2.db*、*remote2.log* を削除します。

詳細情報

さらに詳細な情報については、まず次の役立つ情報を参照してください。

Mobile Link 同期サーバの実行方法の詳細については、『Mobile Link 管理ガイド』>「Mobile Link 同期サーバ」を参照してください。

同期スクリプトの作成方法の詳細については、『Mobile Link 管理ガイド』>「同期スクリプトの作成」と『Mobile Link 管理ガイド』>「同期イベント」を参照してください。

タイムスタンプベースの同期などのその他の同期方法の概要については、『Mobile Link 管理ガイド』>「同期の方法」を参照してください。

Sybase Central でスクリプトをテストする方法については、『Mobile Link 管理ガイド』>「スクリプト構文のテスト」を参照してください。

Mobile Link モニタの詳細については、『Mobile Link 管理ガイド』>「Mobile Link モニタ」を参照してください。

Mobile Link での競合の検出と解決の詳細については、『Mobile Link 管理ガイド』>「競合の解決」を参照してください。

第 3 章

チュートリアル：Oracle 8i 統合データベースでの Mobile Link の使用

この章の内容

この章は、統合データベースが Oracle データベースで、リモート・データベースが Adaptive Server Anywhere データベースの場合の、同期システムの設定手順を解説するチュートリアルです。

このチュートリアルでは、2 つのデータベースを設定して同期する手順を説明します。

概要

このチュートリアルでは、Oracle 統合データベースと Adaptive Server Anywhere リモート・データベースを準備します。次に、Mobile Link を使用してこれら 2 つのデータベースを同期させます。

必要なソフトウェア

- SQL Anywhere Studio の完全なインストール環境
- Oracle Enterprise Edition 8i の完全なインストール環境

能力と経験

このチュートリアルの前提となる知識と経験は、次のとおりです。

- Sybase Central インタフェースと機能の使用経験
- Interactive SQL と Oracle SQL Plus の知識
- Oracle のプログラミングの知識

目的

このチュートリアルの目的は、次のとおりです。

- Oracle で使用する Mobile Link 同期サーバと関連コンポーネントに関する知識の習得
- Oracle 統合データベースに関連する Mobile Link サーバとクライアントのコマンドの実行に関する知識の習得

関連項目

SQL スクリプトの作成方法の詳細については、「チュートリアル：Mobile Link スクリプトの作成と同期のモニタリング」15 ページを参照してください。

Mobile Link 同期サーバの実行方法の詳細については、『Mobile Link 管理ガイド』> 「同期の基本」を参照してください。

レッスン 1：データベースの作成

Mobile Link 同期では、リレーショナル・データベースのデータ、各データベース用の ODBC データ・ソース、2 つの互換データベースが必要です。

SQL ファイル

Oracle データベースには、さまざまな方法でデータを入力できます。このチュートリアルでは、Oracle SQL Plus を使用します。

❖ データベースを作成するには、次の手順に従います。

1 SQL Plus を起動し、Oracle 統合データベースに接続します。次のコードを SQL Plus にコピーして実行します。これらの SQL 文は、統合データベースでテーブルの削除、作成、移植を行います。削除するテーブルがない場合は、SQL Plus の出力にエラーが表示されます。これは処理には影響しません。

```
CREATE SEQUENCE emp_sequence;
CREATE SEQUENCE cust_sequence;
DROP TABLE emp;
CREATE TABLE emp ( emp_id int primary key, emp_name
   varchar( 128 ) );
DROP TABLE cust;
CREATE TABLE cust ( cust_id int primary key, emp_id
int references emp(emp_id), cust_name varchar( 128 )
);
INSERT INTO emp ( emp_id, emp_name ) VALUES (
   emp_sequence.nextval, 'emp1' );
INSERT INTO emp ( emp_id, emp_name ) VALUES (
   emp_sequence.nextval, 'emp2' );
INSERT INTO emp ( emp_id, emp_name ) VALUES (
   emp_sequence.nextval, 'emp3' );
COMMIT;
INSERT INTO cust ( cust_id, emp_id, cust_name )
VALUES ( cust_sequence.nextval, 1, 'cust1' );
```

```
INSERT INTO cust ( cust_id, emp_id, cust_name )
VALUES ( cust_sequence.nextval, 1, 'cust2' );
INSERT INTO cust ( cust_id, emp_id, cust_name )
VALUES ( cust_sequence.nextval, 2, 'cust3' );
COMMIT;
```

2 Interactive SQL を起動します。

[スタート] ー [プログラム] ー [SQL Anywhere 9] ー [Adaptive Server Anywhere] ー [Interactive SQL] を選択します。

3 リモート・データベースに接続します。

4 次のコードを Interactive SQL にコピーして実行します。これらの SQL 文は、リモート・データベースでテーブルの削除と作成を行います。削除するテーブルがない場合は、Interactive SQL の出力にエラーが表示されます。これは処理には影響しません。Mobile Link 同期サーバ用の同期パラメータを定義できるように、同期サブスクリプションと同期パブリケーションも挿入されます。

```
CREATE TABLE emp ( emp_id int primary key ,emp_name
varchar( 128 ) );
 CREATE TABLE cust ( cust_id int primary key, emp_id
int references emp ( emp_id ), cust_name varchar(
128 ) );
 CREATE PUBLICATION emp_cust ( TABLE cust, TABLE emp
);
 CREATE SYNCHRONIZATION USER ml_user;
 CREATE SYNCHRONIZATION SUBSCRIPTION
   TO emp_cust FOR ml_user TYPE TCPIP ADDRESS
'host=localhost';
```

ODBC データ・ソース

Oracle 統合データベースと Adaptive Server Anywhere リモート・データベースに接続するための ODBC データ・ソースを作成する準備ができました。Mobile Link でデータ同期を実行するには、ODBC データ・ソースが必要です。

統合データ・ソース

インストール環境の ODBC 部分に必要なインスタンス名、サービス名、データベース名が判明していることを確認します。これらの値は Oracle のインストール時に設定されます。

Oracle 統合データベースの ODBC 設定を行う手順は、次のとおりです。Adaptive Server Anywhere リモート・データベースへの ODBC 接続は後で設定します。

❖ Oracle 用の ODBC データ・ソースを設定するには、次の手順に従います。

1. [スタート] － [プログラム] － [SQL Anywhere 9] － [Adaptive Server Anywhere] － [ODBC アドミニストレータ] を選択します。

   [ODBC データ・ソース・アドミニストレータ] が開きます。

2. [ユーザ DSN] タブで [追加] をクリックします。[データ・ソースの新規作成] ウィンドウが表示されます。

3. [iAnywhere Solutions 9 - Wire Protocol] を選択し、[完了] をクリックします。

   [ODBC Oracle ドライバ設定] ウィンドウが表示されます。

4. [General] タブをクリックし、データ・ソース名 **ora_consol** を入力します。これは、Oracle データベースに接続するための DSN です。このデータ・ソース名は後で必要になります。

5. サーバ名を入力します。この値は Oracle インストール環境に応じて異なります。サーバが自分のコンピュータで稼働している場合、このフィールドは空白でかまいません。

6. [Advanced] タブをクリックします。デフォルトのユーザ名を入力します。このチュートリアルでは、**system** を使用するか、オブジェクトの作成権限を持つ任意のユーザ名を使用できます。[OK] をクリックします。

7. [OK] をクリックして、[ODBC データ・ソース・アドミニストレータ] を閉じます。

Mobile Link システム・テーブル

Mobile Link には、スクリプト *syncora.sql* が付属しており、SQL Anywhere インストール環境の *MobiLink¥setup* サブディレクトリにあります。このスクリプトは、Mobile Link で Oracle データベースを使用できるように設定するために実行します。

Syncora.sql には、Oracle SQL で記述された SQL 文が格納されています。この文は、Oracle データベースを統合データベースとして使用する準備を整えます。この文で、Mobile Link で使用する一連のシステム・テーブル、トリガ、プロシージャを作成します。システム・テーブル名は ML_ で始まります。Mobile Link は、同期処理中にこれらのテーブルを操作します。

❖ Oracle 内に Mobile Link システム・テーブルを作成するには、次の手順に従います。

1 SQL Plus を起動します。[スタート] － [プログラム] － [Oracle - OraHome81] － [Application Development] － [SQL Plus] を選択します。

Oracle SQL Plus を使用して Oracle データベースに接続します。**system** スキーマとパスワード **manager** を使用してログオンします。

2 次のコマンドを入力して *syncora.sql* を実行します。

```
@path¥syncora.sql;
```

path は、SQL Anywhere 9 インストール環境の *MobiLink¥setup* サブディレクトリです。パスにスペースが含まれる場合は、パスとファイル名を引用符で囲んでください。

❖ システム・テーブルがインストールされたかどうかを確認するには、次の手順に従います。

1 SQL Plus を起動します。[スタート] － [プログラム] － [Oracle - OraHome81] － [Application Development] － [SQL Plus] を選択します。

2　次の SQL 文を実行して、Mobile Link システム・テーブル、プロシージャ、トリガのリストを生成します。

```
SELECT object_name
FROM all_objects
WHERE object_name
LIKE 'ML_%';
```

次の表に示すオブジェクトがすべて含まれる場合は、次の手順に進んでください。

OBJECT_NAME
ML_ADD_CONNECTION_SCRIPT
ML_ADD_DNET_CONNECTION_SCRIPT
ML_ADD_DNET_TABLE_SCRIPT
ML_ADD_JAVA_CONNECTION_SCRIPT
ML_ADD_JAVA_TABLE_SCRIPT
ML_ADD_LANG_CONNECTION_SCRIPT
ML_ADD_LANG_TABLE_SCRIPT
ML_ADD_TABLE_SCRIPT
ML_ADD_USER
ML_CONNECTION_SCRIPT
ML_CONNECTION_SCRIPT_TRIGGER
ML_SCRIPT
ML_SCRIPTS_MODIFIED
ML_SCRIPT_TRIGGER
ML_SCRIPT_VERSION
ML_SUBSCRIPTION
ML_TABLE
ML_TABLE_SCRIPT
ML_TABLE_SCRIPT_TRIGGER
ML_USER

注意

いずれかのオブジェクトがない場合は、実行したプロシージャが失敗しています。この場合は、Mobile Link エラー・メッセージで問題を確認して訂正し、Mobile Link システム・テーブルを次の手順で削除してください。ただし、上に示したテーブル以外に ML_ で始まるテーブルがある場合は、システム・テーブルを削除しないでください。

❖ Mobile Link システム・テーブルを削除するには、次の手順に従います。

1 SQL Plus で次の SQL 文を実行します。

```
select 'drop ' || object_type || ' ' || object_name
|| ';'
from all_objects
where object_name like 'ML_%';
```

削除されるテーブル、プロシージャ、トリガのリストが生成されます。

2 このリストをテキスト・ファイルにコピーし、*drop.sql* として *OracleTut* ディレクトリに保存します。DROP 文を含まない行をすべて削除します。

3 次のコマンドを実行して、*drop.sql* の SQL 文を実行します。

```
@path¥OracleTut¥drop.sql;
```

path を *OracleTut* ディレクトリがあるロケーションに置き換えてください。依存関係のために 1 回目の実行時に削除されなかったテーブルを再度、*drop.sql* を実行して削除します。

4 これで、上で説明した Oracle 内に Mobile Link システム・テーブルを作成する手順をもう一度実行できます。

リモート・データ・ソース

❖ リモート・データベースを初期化するには、次の手順に従います。

1 コマンド・プロンプトを開き、*OracleTut* ディレクトリ (*c:¥OracleTut* など) に移動します。次のコマンド・ラインを実行します。

```
dbinit remote.db
```

2 ディレクトリの内容一覧を取得して、データベースが正常に作成されたかどうかを確認します。一覧にファイル *remote.db* が表示されていることを確認してください。

❖ リモート・データベース用の ODBC データ・ソースを作成するには、次の手順に従います。

- コマンド・プロンプトを開き、*OracleTut* ディレクトリに移動します。次のコマンド・ラインを実行します。

```
dbdsn -w test_remote -y -c "uid=DBA;pwd=SQL;
    dbf=path¥OracleTut¥remote.db;eng=remote"
```

path を *OracleTut* ディレクトリがあるロケーションに置き換えてください。

データベース

この手順では、Interactive SQL コマンド・ライン・ユーティリティを使用して統合データベースを構築します。Interactive SQL ユーティリティを使用すると、データベース内で SQL コマンドを実行できます。この手順では各データベース内で SQL 文を実行します。

Interactive SQL の詳細については、『ASA データベース管理ガイド』>「Interactive SQL ユーティリティ」を参照してください。

❖ 統合データベースでテーブルを作成して移植するには、次の手順に従います。

1. SQL Plus を起動し、統合データベースに接続します。[スタート] － [プログラム] － [Oracle - OraHome81] － [Application Development] － [SQL Plus] を選択します。
2. 次のコマンドを実行して、*build_consol.sql* の SQL 文を実行します。

```
@path¥OracleTut¥build_consol.sql;
```

path を *OracleTut* ディレクトリがあるロケーションに置き換えてください。パスにスペースが含まれる場合は、パスとファイル名を二重引用符で囲んでください。

3 アプリケーション内から直接 SQL Plus を使用して、各テーブルが正常に作成されたことを確認します。次の SQL 文を実行します。

```
SELECT * FROM emp;
SELECT * FROM cust;
```

4 統合データベースを稼働したままにします。

❖ リモート・データベースでテーブルと同期情報を作成するには、次の手順に従います。

1 コマンド・プロンプトを開き、*OracleTut* ディレクトリに移動します。次のコマンド・ラインを実行します。

```
dbisql -c "dsn=test_remote" build_remote.sql
```

Interactive SQL プラグインが、リモート・データベースを起動して *build_remote.sql* の SQL 文を実行します。

2 Interactive SQL または Sybase Central を使用して、emp テーブルと cust テーブルが正常に作成されたことを確認します。

3 統合データベースとリモート・データベースを稼働したままにします。

レッスン 2：Mobile Link 同期サーバの実行

この時点で、Mobile Link 同期サーバをコマンド・プロンプトから起動できます。Mobile Link 同期サーバは統合データベースのクライアントであるため、統合データベースを起動してから Mobile Link を起動してください。レッスン 1 の後で統合データベースを停止した場合は、統合データベースを再起動してください。

❖ Mobile Link 同期サーバを起動するには、次の手順に従います。

1　統合データベースが稼働していることを確認します。

2　コマンド・プロンプトを開き、*OracleTut* ディレクトリに移動します。次のコマンド・ラインを実行します。

```
dbmlsrv9 -c "dsn=ora_consol;pwd=manager" -o
mlserver.mls -v+ -za -zu+
```

このコマンド・ラインは、以下に示す dbmlsrv9 のオプションを指定します。

- **-c**　接続パラメータを指定します。DSN にはユーザ ID が含まれているため、パスワードのみを使用することに注意してください。詳細については、『Mobile Link 管理ガイド』>「-c オプション」を参照してください。
- **-o**　メッセージ・ログ・ファイルを指定します。詳細については、『Mobile Link 管理ガイド』>「-o オプション」を参照してください。
- **-v+**　冗長ロギングをオンに設定します。詳細については、『Mobile Link 管理ガイド』>「-v オプション」を参照してください。
- **-dl**　ログ表示機能をオンに設定します。
- **-za**　スクリプトの自動生成をオンに設定します。詳細については、『Mobile Link 管理ガイド』>「-za オプション」を参照してください。

- **-zu+** ユーザ認証処理を自動化します。詳細については、『Mobile Link 管理ガイド』>「-zu オプション」を参照してください。

詳細情報

dbmlsrv9 の詳細については、『Mobile Link 管理ガイド』>「Mobile Link 同期サーバ」と『Mobile Link 管理ガイド』>「Mobile Link 同期サーバ」を参照してください。

レッスン 3：Mobile Link 同期クライアントの実行

この時点で、Mobile Link クライアントをコマンド・プロンプトから起動できます。Mobile Link 同期を開始するのはクライアントです。

-c オプションを使用すると、*dbmlsync* コマンド・ラインで接続パラメータを指定できます。これらのパラメータは、*リモート・データベース*用です。

❖ Mobile Link クライアントを起動するには、次の手順に従います。

1　Mobile Link 同期サーバが起動されていることを確認します。

2　コマンド・プロンプトを開き、*OracleTut* ディレクトリに移動します。次のコマンド・ラインを実行します。

```
dbmlsync -c "dsn=test_remote" -o dbmlsync.out -v+ -e
"SendColumnNames=ON"
```

このコマンド・ラインは以下に示す dbmlsync のオプションを指定します。

- **-c**　データベース接続パラメータを指定します。詳細については、『Mobile Link クライアント』>「-c オプション」を参照してください。
- **-o**　メッセージ・ログ・ファイルを指定します。詳細については、『Mobile Link クライアント』>「-o オプション」を参照してください。
- **-v+**　冗長オペレーション。詳細については、『Mobile Link クライアント』>「-v オプション」を参照してください。
- **-e**　拡張オプション。"SendColumnNames=ON" を指定すると、カラム名が Mobile Link に送信されます。詳細については、『Mobile Link クライアント』>「dbmlsync 拡張オプション」を参照してください。

詳細情報

dbmlsync の詳細については、『Mobile Link クライアント』> 「Mobile Link 同期クライアント」を参照してください。

まとめ

このチュートリアルでは、以下の作業を行いました。

- リモート・データベースとして機能する新しい Adaptive Server Anywhere データベースの作成
- Oracle 統合データベースで動作する Mobile Link 同期サーバの起動
- Mobile Link 同期クライアントの起動、リモート・データベースと Oracle 統合データベースの同期処理

学習の成果

このチュートリアルでは、次の項目を学習しました。

- Mobile Link 同期サーバとクライアントの操作方法、両者による Oracle データベースの操作方法の知識
- Mobile Link サーバとクライアントのコマンドの実行方法の習得

詳細情報

さらに詳細な情報については、まず次の役立つ情報を参照してください。

Mobile Link 同期サーバの実行方法の詳細については、『Mobile Link 管理ガイド』>「Mobile Link 同期サーバの実行」を参照してください。

同期スクリプトの作成方法の詳細については、『Mobile Link 管理ガイド』>「同期スクリプトの作成」と『Mobile Link 管理ガイド』>「同期イベント」を参照してください。

タイムスタンプベースの同期などのその他の同期方法の概要については、『Mobile Link 管理ガイド』>「同期の方法」を参照してください。

第 5 章

チュートリアル：Adaptive Server Anywhere での .NET 同期論理

この章の内容

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。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 同期サーバ (dbmlsrv9) を実行します。

概要

Mobile Link の接続レベル・イベント・スクリプトとテーブル・レベル・イベント・スクリプトは、SQL、Java、または .NET で作成できます。Java と .NET は共に、クラス・メソッドにイベント論理をカプセル化します。このチュートリアルでは、Mobile Link テーブル・レベル・イベントに .NET クラス・メソッドをサブスクライブします。

必要なソフトウェア

- SQL Anywhere Studio 9.0
- Microsoft .NET Framework SDK

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

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・メソッドの利用

主要な概念

この項では、次の手順に従って、Mobile Link CustDB サンプル・データベースを使用した基本的な .NET 同期を実装します。

- Mobile Link 参照を含む *CustdbScripts.dll* プライベート・アセンブリをコンパイルします。
- テーブル・レベル・イベント用のクラス・メソッドを指定します。
- -sl dnet オプションを使用して、Mobile Link サーバ (dbmlsrv9) を実行します。
- サンプル Windows クライアント・アプリケーションを使用して、同期をテストします。

関連項目

同期スクリプトの詳細については、『Mobile Link 管理ガイド』> 「同期スクリプトの概要」を参照してください。

Sybase Central の詳細については、『SQL Anywhere Studio の紹介』>「Sybase Central を使用したデータベースの管理」を参照してください。

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

.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 Studio には、同期できるように設定された Adaptive Server 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 Adaptive Server Anywhere クライアントの両方の統合データベース・サーバとなるように設計されています。CustDB データベースには、UltraLite 9.0 Sample という DSN が含まれています。

この項では、ULCustomer の upload_insert イベントと upload_update 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。

CustdbScripts アセンブリ

Mobile Link API

.NET 同期論理を実行するには、Mobile Link 同期サーバが *iAnywhere.MobiLink.Script.dll* 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 API クラスのリポジトリが入っています。

Mobile Link .NET API の詳細については、『Mobile Link 管理ガイド』>「Mobile Link .NET API リファレンス」を参照してください。

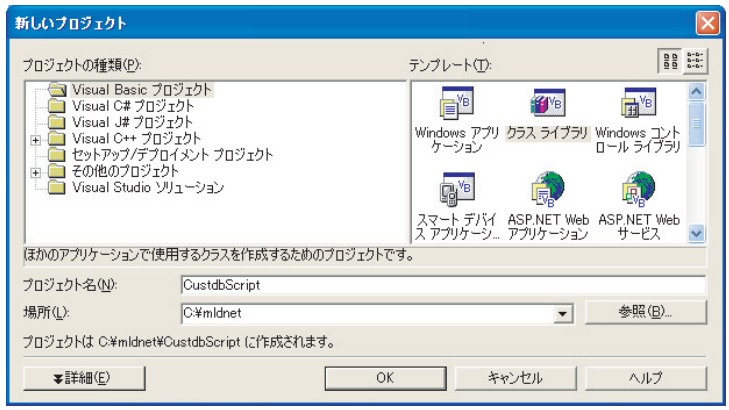
この API を利用する場合、CustdbScripts クラスのコンパイル時にここで示したアセンブリを含める必要があります。クラスのコンパイルは、Visual Studio .NET またはコマンド・ラインを使用して実行できます。

- Visual Studio .NET を使用する場合、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。
- コマンド・ラインを使用する場合、テキスト・エディタに CustdbScripts コードを入力し、ファイルを *CustdbScripts.cs* (Visual Basic .NET の場合 *CustdbScripts.vb*) として保存します。コマンド・ライン・コンパイラを使用して *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 .NET を使用して CustdbScripts アセンブリを作成するには、次の手順に従います。

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

プロジェクト名として 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:¥mldnet* とします。



2　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 (?,?)");
  }
  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 .NET の場合、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 (?,?)")
   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 Mobile Link API への参照を追加します。

- Visual Studio .NET で、[プロジェクト] ー [既存項目の追加] を選択します。

- SQL Anywhere Studio インストール環境の *win32* ディレクトリにある *iAnywhere.MobiLink.Script.dll* を選択します。[開く] ドロップダウン・メニューから [リンクファイル] を選択します。

4　*CustdbScripts.dll* を構築します。

[ビルド] ー [CustdbScripts のビルド] を選択します。

これにより、*C:¥mldnet¥CustdbScripts¥CustdbScripts¥bin¥Debug* に *CustdbScripts.dll* が作成されます。

❖ コマンド・ラインを使用して CustdbScripts アセンブリを作成するには、次の手順に従います。

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:¥mldnet* とします。

2　テキスト・エディタを使用して、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 (?,?)");
  }
  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 .NET の場合、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 (?,?)")
  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 *c:¥mldnet* に *CustdbScripts.cs* (Visual Basic .NET の場合は *CustdbScripts.vb*) として保存します。

4 次のコマンドを使用して、ファイルをコンパイルします。

C# の場合、次のように入力します。

```
csc /out:c:¥mldnet¥custdbscripts.dll /
target:library /
reference:"%asany9%¥win32¥iAnywhere.MobiLink.Script
.dll" c:¥mldnet¥CustdbScripts.cs
```

Visual Basic .NET の場合、次のように入力します。

```
vbc /out:c:¥mldnet¥custdbscripts.dll /
target:library /
reference:"%asany9%¥win32¥iAnywhere.MobiLink.Script
.dll" c:¥mldnet¥CustdbScripts.vb
```

CustdbScripts.dll アセンブリが生成されます。

詳細情報

Mobile Link API の詳細については、『Mobile Link 管理ガイド』> 「Mobile Link .NET API リファレンス」を参照してください。

.NET メソッドの詳細については、『Mobile Link 管理ガイド』> 「メソッド」を参照してください。

レッスン 2：イベント用のクラス・メソッドの指定

CustDB サンプル・データベースの詳細と代替 RDBMS サーバの使用については、「CustDB 統合データベースの設定」135 ページを参照してください。

前のレッスンで作成された *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 upload_update イベントの実装が含まれています。

この項では、次の 2 つの方法を使用して、テーブル・レベル・イベント用のクラス・メソッドを指定します。

1. Mobile Link 同期プラグインを使用する方法

   Sybase Central を使用して CustDB データベースに接続し、upload_insert スクリプトの言語を .NET に変更して、イベントを処理するための MLExample.CustdbScripts.UploadInsert を指定します。

2. ml_add_dnet_table_script ストアド・プロシージャを使用する方法

   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 テーブル用の upload_insert イベントに CustdbScripts.uploadInsert() をサブスクライブするには、次の手順に従います。

1 Mobile Link 同期プラグインを使用して、サンプル・データベースに接続します。

- Sybase Central を起動します。

  [スタート] ー [プログラム] ー [SQL Anywhere 9] ー [Sybase Central] を選択します。

- 左ウィンドウ枠で、[Mobile Link 同期 9] プラグインを右クリックし、[接続] を選択します。

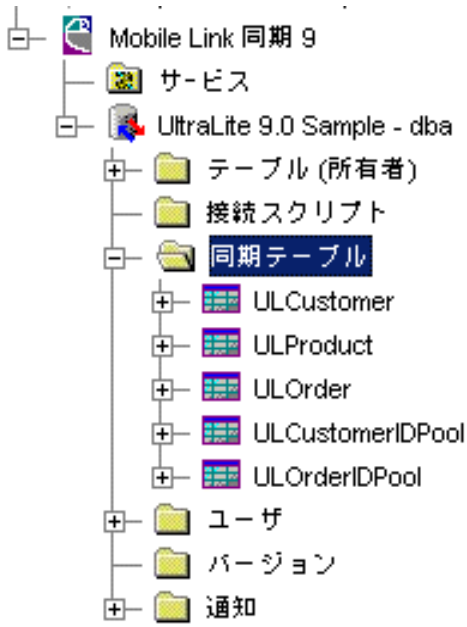
- ODBC データ・ソースとして [UltraLite 9.0 Sample] を選択します。

  [ID] タブで、[ODBC データ・ソース名] オプションを選択します。データ・ソース名として **UltraLite 9.0 Sample** を入力します。

  [データベース] タブで、ネットワーク・データベース・サーバの検索オプションが選択されていないことを確認します。

- [OK] をクリックして接続します。

- これで、Sybase Central は Mobile Link 同期 9 プラグインで CustDB データ・ソースを表示します。



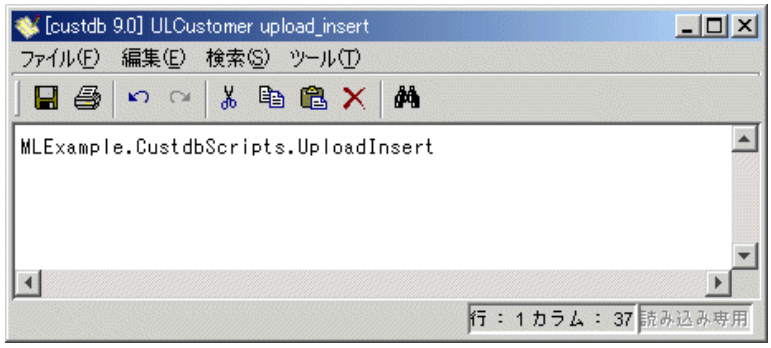
2 ULCustomer テーブルの upload_insert イベントの言語を .NET に変更します。

- 左ウィンドウ枠で、[同期テーブル] フォルダを開き、ULCustomer テーブルを選択します。テーブル・レベル・スクリプトのリストが、右ウィンドウ枠に表示されます。
- custdb 9.0 の upload_insert イベントに関連するテーブル・スクリプトを右クリックします。[ファイル] ー [スクリプト言語を .NET に設定] を選択します。

3 upload_insert スクリプト用として、完全に修飾された .NET メソッド名を入力します。

- upload_insert イベントに関連するテーブル・スクリプトをダブルクリックします。

  スクリプトの内容を示すウィンドウが表示されます。
- スクリプトの内容を、完全に修飾されたメソッド名 **MLExample.CustdbScripts.UploadInsert** に変更します。



注意：
完全に修飾されたメソッド名では、大文字と小文字が区別されます。

- [ファイル] ー [保存] を選択して、スクリプトを保存します。

4　Sybase Central を終了します。

この手順では、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した。

別の方法として、ml_add_dnet_connection_script ストアド・プロシージャや ml_add_dnet_table_script ストアド・プロシージャも使用できます。これらのストアド・プロシージャは、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。

詳細については、『Mobile Link 管理ガイド』> 「ml_add_dnet_connection_script」と『Mobile Link 管理ガイド』> 「ml_add_dnet_table_script」を参照してください。

次の項では、Interactive SQL を使用して CustDB に接続し、ml_add_dnet_table_script を実行して、download_cursor イベントに MLExample.CustdbScripts.DownloadCursor を割り当てます。

❖ ULCustomer の download_cursor イベント用の MLExample.CustdbScripts.DownloadCursor を指定するには、次の手順に従います。

1　Interactive SQL を使用して、サンプル・データベースに接続します。

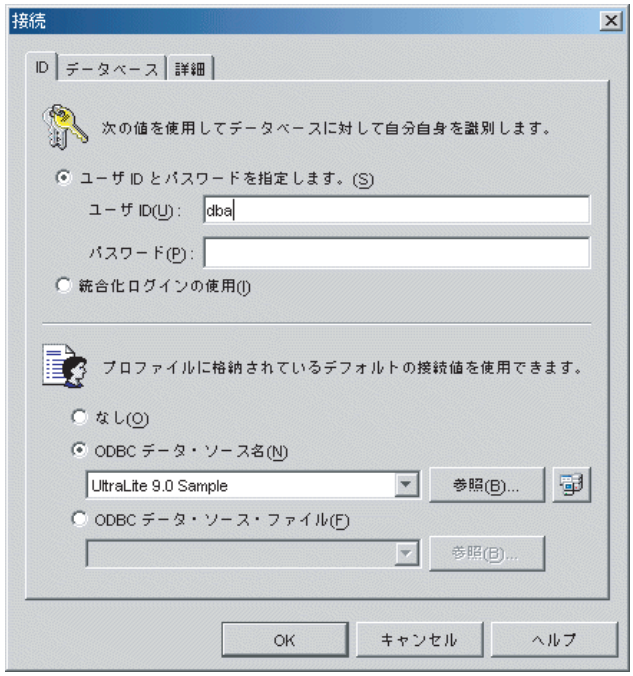
- Interactive SQL を起動します。

  [スタート] ー [プログラム] ー [SQL Anywhere 9] ー [Adaptive Server Anywhere] ー [Interactive SQL] を選択します。または、コマンド・プロンプトで次のように入力します。

  ```
  dbisql
  ```

  [接続] ダイアログが表示されます。

- [ID] タブで、ODBC データ・ソースとして [UltraLite 9.0 Sample] を選択します。



  [データベース] タブで、ネットワーク・データベース・サーバの検索オプションが選択されていないことを確認します。

2 Interactive SQL で次のコマンドを実行します。

```
call ml_add_dnet_table_script(
'custdb 9.0',
'ULCustomer',
'download_cursor',
'MLExample.CustdbScripts.DownloadCursor');
commit;
```

次に、各パラメータの説明を示します。

パラメータ	説明
custdb 9.0	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

3　Interactive SQL を終了します。

このレッスンでは、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した。次のレッスンでは、Mobile Link サーバが確実に適切なクラス・ファイルと Mobile Link API をロードするように指定します。

詳細情報

同期スクリプトの追加方法と削除方法の詳細については、『Mobile Link 管理ガイド』>「統合データベースでのスクリプトの追加と削除」を参照してください。

このレッスンで使用したスクリプトの詳細については、『Mobile Link 管理ガイド』>「ml_add_dnet_connection_script」と『Mobile Link 管理ガイド』>「ml_add_dnet_table_script」を参照してください。

レッスン 3：-sl dnet を使用した Mobile Link の実行

-sl dnet オプションを使用して Mobile Link サーバを実行すると、.NET アセンブリのロケーションを指定して、CLR をサーバ起動時にロードできます。

コンパイルに Visual Studio .NET を使用した場合、*CustdbScripts.dll* のロケーションは *c:¥mldnet¥CustdbScripts¥CustdbScripts¥bin¥Debug* になります。コマンド・ラインを使用した場合、*CustdbScripts.dll* のロケーションは *c:¥mldnet* になります。

❖ Mobile Link サーバ (dbmlsrv9) を起動し、.NET アセンブリをロードするには、次の手順に従います。

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

  Visual Studio .NET を使用してアセンブリをコンパイルした場合は、次の手順に従います。

  コマンド・ラインで、次のコマンドを 1 行で入力します。

  ```
  dbmlsrv9 -c "dsn=ultralite 9.0 sample" -dl -o
  cons1.txt -v+ -sl dnet(-
  MLAutoLoadPath=c:¥mldnet¥CustdbScripts¥CustdbScript
  s¥bin¥Debug)
  ```

  コマンド・ラインを使用してアセンブリをコンパイルした場合は、次の手順に従います。

  コマンド・ラインで、次のコマンドを 1 行で入力します。

  ```
  dbmlsrv9 -c "dsn=ultralite 9.0 sample" -dl -o
  cons1.txt -v+ -sl dnet(-MLAutoLoadPath=c:¥mldnet)
  ```

  サーバが要求を処理する準備ができたことを示すメッセージ・ダイアログが表示されます。これで、同期中に upload_insert イベントがトリガされると .NET メソッドが実行されるようになりました。

詳細情報

詳細については、『Mobile Link 管理ガイド』> 「-sl dnet オプション」を参照してください。

レッスン 4：同期のテスト

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、ユーザが同期を発行したときに自動的に dbmlsync ユーティリティを起動します。これは簡単な販売状況アプリケーションで、前のレッスンで起動した CustDB 統合データベースに対して実行できます。

アプリケーションの起動 (Windows)

❖ **サンプル・アプリケーションを起動して同期するには、次の手順に従います。**

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 9] － [Ultra Light] － [Windows アプリケーションのサンプル] を選択します。

2 従業員 ID を入力します。

50 の値を入力し、[ENTER] を押します。

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。

次の項では、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 イベントがトリガされます。

注文情報の追加 (Windows)

❖ **注文情報を追加するには、次の手順に従います。**

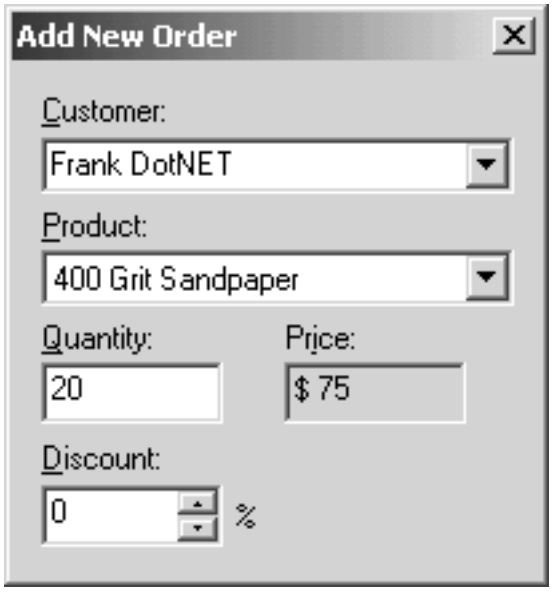
1 [Order] － [New] を選択します。

[Add New Order] 画面が表示されます。

2　新しい顧客名を入力します。

たとえば、Frank DotNET と入力します。

3　製品を選択し、数量と割引率を入力します。



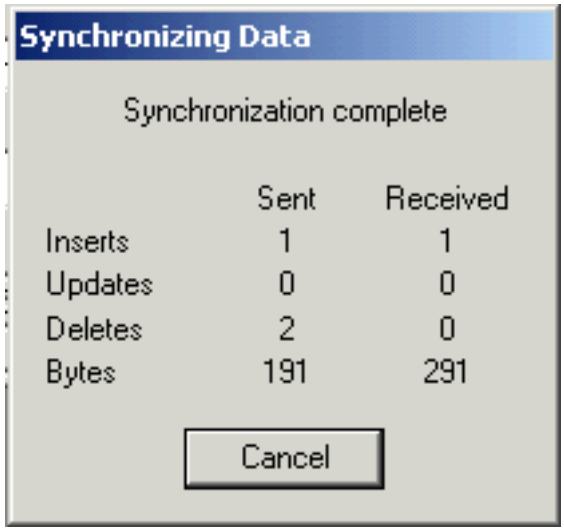
4　[Enter] を押して、新しい注文を追加します。

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。

❖ 統合データベースと同期し、upload_insert イベントをトリガするには、次の手順に従います。

- [File] － [Synchronize] を選択します。

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ウィンドウが表示されます。



詳細情報

CustDB Windows アプリケーションの詳細については、「CustDB サンプル・アプリケーション」131 ページを参照してください。

クリーンアップ

コンピュータからチュートリアルを削除してください。

❖ チュートリアルをコンピュータから削除するには、次の手順に従います。

1. ULCustomer テーブルの upload_insert スクリプトと download_cursor スクリプトを元の SQL 論理に戻します。

   - Interactive SQL を開きます。

     [スタート] ー [プログラム] ー [SQL Anywhere 9] ー [Adaptive Server Anywhere] ー [Interactive SQL] を選択します。または、コマンド・プロンプトで次のように入力します。

     ```
     dbisql
     ```

     [接続] ダイアログが表示されます。

   - [ID] タブで、ODBC データ・ソースとして [UltraLite 9.0 Sample] を選択します。

   - [OK] をクリックして接続します。

   - Interactive SQL で次のコマンドを実行します。

     ```
     call ml_add_table_script( 'custdb 9.0',
         'ULCustomer',
         'upload_insert',
        'INSERT INTO ULCustomer( cust_id, cust_name
     ) VALUES( ?, ? )' );

     call ml_add_table_script( 'custdb 9.0',
         'ULCustomer',
         'download_cursor',
         'SELECT "cust_id", "cust_name" FROM
     "ULCustomer" WHERE "last_modified" >= ?' );
     ```

2 タスクバー上で、Adaptive Server Anywhere、Mobile Link、同期クライアントの各項目を右クリックし、[閉じる] を選択して閉じます。

3 チュートリアルに関連するすべての .NET ソースを削除します。

CustdbScripts.cs ファイルと *CustdbScripts.dll* ファイルを含むフォルダ (*c:¥mldnet*) を削除します。

注意：
c:¥mldnet に他の重要なファイルが含まれていないことを確認してください。

詳細情報

さらに詳細な情報については、まず次の役立つ情報を参照してください。

.NET で Mobile Link 同期スクリプトを作成する方法の詳細については、『Mobile Link 管理ガイド』>「.NET 同期論理の設定」を参照してください。

.NET 同期論理をデバッグする方法の詳細については、『Mobile Link 管理ガイド』>「.NET 同期論理のデバッグ」を参照してください。

カスタム認証用の .NET 同期スクリプトの使用例については、『Mobile Link 管理ガイド』>「.NET 同期のサンプル」を参照してください。

同期スクリプトの作成方法の詳細については、『Mobile Link 管理ガイド』>「同期スクリプトの作成」と『Mobile Link 管理ガイド』>「同期イベント」を参照してください。

タイムスタンプベースの同期などのその他の同期方法の概要については、『Mobile Link 管理ガイド』>「同期の方法」を参照してください。

第 6 章

Contact サンプル・アプリケーション

この章の内容

この章では、Contact サンプル・アプリケーションを使用して、一般的な同期タスクに使用できるさまざまな方法について説明します。

それぞれの方法は SQL スクリプトを使用して説明されています。多くの方法は Java または .NET 同期論理を使用して実装することも可能です。

概要

この章では、Contact サンプル・アプリケーションについて説明します。このサンプルは、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 Adaptive Server Anywhere 統合データベースと 2 つの Adaptive Server Anywhere リモート・データベースで構成されています。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。この章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 Adaptive Server 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、SQL Anywhere インストール環境の *Samples¥MobiLink¥Contact* サブディレクトリにあります。概要については、*Samples¥MobiLink¥Contact¥readme.txt* を参照してください。

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 テーブルの単一のローの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。

  詳細については、『Mobile Link 管理ガイド』>「リモート・データベース間でローを分割する」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。

詳細については、『Mobile Link 管理ガイド』>「タイムスタンプベースの同期」を参照してください。

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、いくつかのデータ、同期スクリプト、Mobile Link ユーザ名がデータベースにロードされます。
- *remote.db* という名前の 2 つのリモート・データベースが、サブディレクトリ *remote_1* と *remote_2* に作成されます。両方のデータベースに共通の情報がロードされ、カスタマイズ内容が適用されます。カスタマイズ内容には、グローバル・データベース識別子、Mobile Link ユーザ名、2 つのパブリケーションに対するサブスクリプションが含まれます。

❖ Contact サンプルを構築するには、次の手順に従います。

1. コマンド・プロンプトを開き、SQL Anywhere インストール環境の *Samples¥MobiLink¥Contact* サブディレクトリに移動します。
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 同期サーバと dbmlsync のコマンド・ラインが例示されています。次のバッチ・ファイルの内容をテキスト・エディタで調べることができます。これらのファイルは、SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥Contact* サブディレクトリにあります。

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

❖ Contact サンプルを実行するには、次の手順に従います。

1　Mobile Link 同期サーバを起動します。

- コマンド・プロンプトを開き、SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥Contact* サブディレクトリに移動して、次のコマンドを実行します。

```
step1
```

このコマンドは、Mobile Link 同期サーバを冗長モードで起動するバッチ・ファイルを実行します。このモードは、開発中やトラブルシューティング中には便利ですが、パフォーマンスが低下するため、通常の運用環境では使用しません。

2　両方のリモート・データベースを同期します。

- コマンド・プロンプトを開き、SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥Contact* サブディレクトリに移動して、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 同期サーバを停止します。

- コマンド・プロンプトを開き、SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥Contact* サブディレクトリに移動して、次のコマンドを実行します。

```
step3
```

これは、Mobile Link サーバを停止させるバッチ・ファイルです。

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。

- *Samples¥MobiLink¥Contact¥build_consol.sql*
- *Samples¥MobiLink¥Contact¥build_renmote.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** 製品の識別番号を保持するプライマリ・キー・カラム。
- **name** 個々の品目の名前。
- **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**　製品が現在がアクティブであるか (1)、またはこの製品が既に製造中止になったか (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、この製品に対応するローがリモート・データベースから削除されます。

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

Contact サンプル内のユーザ

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

データベース・ユーザ ID

2 つのリモート・データベースは、営業担当者 Samuel Singer (rep_id 856) と Pamela Savarino (rep_id 949) が所持しています。

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

また、各リモート・データベースには、ユーザ ID **sync_user** (パスワードは **sync_user**) もあります。このユーザ ID は、dbmlsync コマンド・ラインでのみ使用します。これは REMOTE DBA 権限を持つユーザなので、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。したがって、このユーザ ID とパスワードは広範囲で使用できますが問題にはなりません。

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

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

- リモート・データベース。Mobile Link ユーザ名が、CREATE SYNCHRONIZATION USER 文を使用して追加されています。
- 統合データベース。Mobile Link ユーザ名とパスワードが、dbmluser ユーティリティを使用して追加されています。

- *Samples¥MobiLink¥Contact¥step2.bat* 内の dbmlsync コマンド・ライン。同期時、接続ユーザの Mobile Link パスワードが指定されます。
- Mobile Link 同期サーバ。同期時、Mobile Link ユーザ名がパラメータとして多数のスクリプトに指定されます。
- 統合データベース側の SalesRep テーブル。ml_username カラムがあります。同期スクリプトは、このカラムの値と Mobile Link ユーザ名パラメータを比較します。

同期

以下の項では、Contact サンプルの同期論理を説明します。

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。

詳細については、『Mobile Link 管理ガイド』>「スナップショットを使った同期」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ヘッドはほとんどないため、単純なスナップショットの **download_cursor** スクリプトを使用します。

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

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

アップロード

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

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。

詳細については、『Mobile Link 管理ガイド』>「タイムスタンプベースの同期」を参照してください。

詳細については、『Mobile Link 管理ガイド』>「リモート・データベース間でローを分割する」を参照してください。

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、特定の営業担当者に割り当てられた顧客のみをダウンロードします。

```
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。

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

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

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

顧客が修正されるたびにすべての窓口レコードを更新することで、トリガは顧客とその関連窓口を結合するため、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、明示的に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 = ?
```

  このテーブルに対する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、それぞれの注文が Product テーブルの quantity カラムから減算されます。たとえば、Sam 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。Pam Savarino が Sam 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。

Sam 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。Pam 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_typ )
 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 文はファイル *Samples¥MobiLink¥Contact¥mlmaint.sql* にあります。

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

第 7 章

CustDB サンプル・アプリケーション

この章の内容

この章では、CustDB サンプル・アプリケーションを使用して、一般的な同期タスクに使用できるさまざまな方法について説明します。

これらの方法は、SQL スクリプトと Java 同期論理を使用して説明します。この方法の多くは、.NET 同期論理を使用して実装することも可能です。

概要

この章では、CustDB (Customer データベース) Mobile Link サンプル・アプリケーションについて説明します。CustDB は販売管理アプリケーションです。

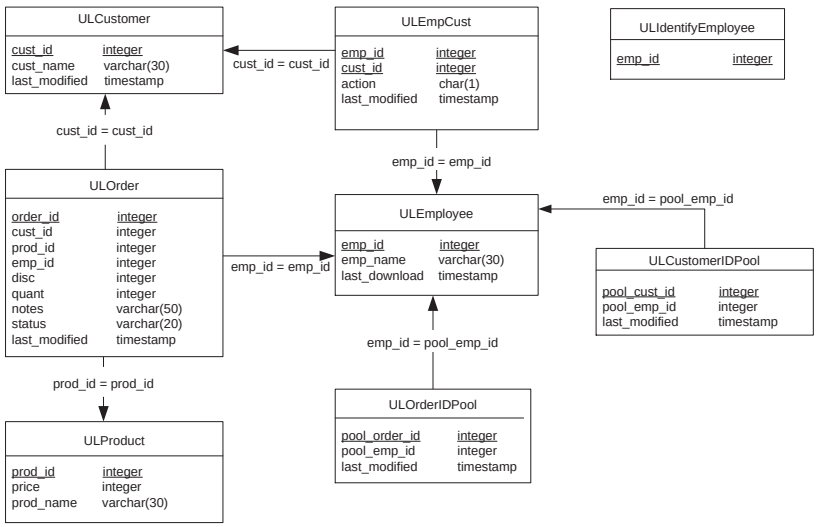
CustDB サンプルは、Mobile Link 開発者にとって貴重なリソースです。このサンプルは、Mobile Link アプリケーションの開発時に必要なさまざまな手法の実装例を示しています。

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。この章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 のロケーションと設定手順については、「CustDB 統合データベースの設定」135 ページを参照してください。

CustDB のスキーマは以下のとおりです。



シナリオ

CustDB のシナリオを以下に示します。

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
- 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、『Mobile Link 管理ガイド』>「リモート・データベース間でローを分割する」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。

詳細については、『Mobile Link 管理ガイド』>「タイムスタンプベースの同期」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。

  詳細については、『Mobile Link 管理ガイド』>「スナップショットを使った同期」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)** 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。

  詳細については、『Mobile Link 管理ガイド』>「キー・プールを使用したユニークなプライマリ・キーの管理」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、『Mobile Link 管理ガイド』>「ユニークなプライマリ・キーの管理」を参照してください。

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。

- サンプルの SQL スクリプト。SQL Anywhere インストール環境の *Samples¥MobiLink¥CustDB* サブディレクトリにあります。
- アプリケーション・コード。*Samples¥UltraLite¥CustDB* にあります。
- プラットフォーム固有のユーザ・インタフェースのコード。*Samples¥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

統合データベースとして使用できるのは、Adaptive Server Anywhere、Sybase Adaptive Server Enterprise、Microsoft SQL Server、Oracle、または IBM DB2 です。

以下の SQL スクリプトが、SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥CustDB* サブディレクトリに用意されています。これらのスクリプトは、以下のプラットフォームで統合データベースを構築するために使用します。

- Adaptive Server Anywhere データベース用のファイルは、*custdb.sql* です。
- IBM DB2 データベース用のファイルは、*custdb2.sql* です。
- Adaptive Server Enterprise データベース用のファイルは、*custase.sql* です。
- Microsoft SQL Server データベース用のファイルは、*custmss.sql* です。
- Oracle データベース用のファイルは、*custora.sql* です。

統合データベースの作成

以下の手順では、サポートされる各種の統合データベースで CustDB 用の統合データベースを作成します。

Adaptive Server Anywhere 以外のデータベースの場合は、最初にスクリプトを実行して Mobile Link システム・テーブルを追加する必要があります。このプラットフォーム固有のスクリプトは、SQL Anywhere 9 インストール環境の *MobiLink¥setup* サブディレクトリにあります。

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、『Mobile Link 管理ガイド』>「統合データベースの設定」を参照してください。

❖ 統合データベースを設定するには、次の手順に従います (Adaptive Server Enterprise、Oracle、または SQL Server)。

1 統合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・テーブルを追加します。これらのスクリプトは、SQL Anywhere 9 インストール環境の *MobiLink¥setup* サブディレクトリにあります。

- バージョン 12.5 よりも前の Adaptive Server Enterprise 統合データベースの場合は、*syncase.sql* を実行します。それ以外の場合は、*syncase125.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 CustDB データベースにテーブルを追加します。これらのスクリプトは、SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥CustDB* サブディレクトリにあります。

- Adaptive Server Enterprise 統合データベースの場合は、*custase.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・マシン上で、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。

- [スタート] － [プログラム] － [SQL Anywhere 9] － [Adaptive Server Anywhere] － [ODBC アドミニストレータ] を選択します。
- [追加] をクリックします。
- リストから適切なドライバを選択します。

  [完了] をクリックします。
- この ODBC データ・ソースに CustDB という名前を付けます。
- [ログイン] タブをクリックします。データベースのユーザ ID とパスワードを入力します。

❖ 統合データベースを設定するには、次の手順に従います (Adaptive Server Anywhere)。

1　次のようにして、統合データベースを作成します。

SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥CustDB* サブディレクトリに移動し、次のコマンド・ラインを実行します。

```
dbinit consol.db
```

2 SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥CustDB* サブディレクトリにある *custdb.sql* を実行して CustDB データベースにテーブルを追加します。

- [スタート] ー [プログラム] ー [SQL Anywhere 9] ー [Sybase Central] を選択します。
- Sybase Central の右ウィンドウ枠で [Adaptive Server Anywhere 9] を右クリックして、作成した統合データベースに接続します。デフォルトのユーザ ID とパスワードは、DBA と SQL です。
- 右ウィンドウ枠で統合データベースを右クリックし、ポップアップ・メニューで [Interactive SQL を開く] を選択します。
- Interactive SQL で、[ファイル] ー [スクリプトの実行] を選択します。*custdb.sql* を検索します。[開く] をクリックします。
- Interactive SQL を閉じます。

3 クライアント・マシン上で、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。

- Sybase Central で、[ツール] ー [Adaptive Server Anywhere 9] ー [ODBC アドミニストレータを開く] を選択します。
- [追加] をクリックします。
- Adaptive Server Anywhere 9.0 ドライバを選択します。[完了] をクリックします。
- この ODBC データ・ソースに CustDB という名前を付けます。
- [ログイン] タブをクリックします。データベースのユーザ ID とパスワードを入力します。デフォルト値は、DBA と SQL です。

- [データベース] タブをクリックします。データベース・ファイルのロケーションを参照します。

❖ 統合データベースを設定するには、次の手順に従います (IBM DB2)。

1 DB2 サーバ上に DB2 データベースを作成します。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。

- 1 つ以上のバッファ・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・プールを作成してください。
- 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 のマニュアルを参照してください。

2 ファイル *MobiLink¥setup¥syncdb2long.sql* を使用して、Mobile Link システム・テーブルを追加します

- *syncdb2long.sql* の接続コマンドを変更します。*DB2Database* を ODBC データ・ソースの名前に置き換えます。この例では、ODBC データ・ソースは CustDB です。以下に示すように、ユーザ名とパスワードを追加することもできます。*userid* と *password* を、使用するユーザ名とパスワードに置き換えてください。

  ```
  connect to DB2Database user userid using password ~
  ```

- サーバまたはクライアント・コンピュータで、DB2 コマンド・ウィンドウを開きます。次のコマンドを入力して *syncdb2long.sql* を実行します。

  ```
  db2 -c -ec -td~ +s -v -f syncdb2long.sql
  ```

3 *syncdb2long.sql* に定義されているストアド・プロシージャを DB2 で使用する場合は、SQL Anywhere インストール環境の *MobiLink¥setup* サブディレクトリにある *syncdb2long_version* という名前の Java ファイルとクラス・ファイルを、DB2 インストール環境の *FUNCTION* サブディレクトリにコピーします。

4 SQL Anywhere インストール環境の *Samples¥MobiLink¥CustDB* サブディレクトリにある *custdb2.class* を、DB2 サーバ・マシンの *SQLLIB¥FUNCTION* ディレクトリにコピーします。このクラスには、CustDB サンプルで使用されるプロシージャが含まれています。

5 以下のように、テーブルを CustDB に追加します。

- 必要に応じて、*custdb2.sql* の接続コマンドを変更します。たとえば、以下に示すように、ユーザ名とパスワードを追加できます。*userid* と *password* を、使用するユーザ名とパスワードに置き換えてください。

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

- サーバまたはクライアント・コンピュータで、DB2 コマンド・ウィンドウを開きます。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
  ```

- 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
  ```

6 DB2 クライアント・マシン上で、DB2 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。

- ODBC アドミニストレータを起動します。

  [スタート] ー [プログラム] ー [SQL Anywhere 9] ー [Adaptive Server Anywhere] ー [ODBC アドミニストレータ] を選択します。

[ODBC データ・ソース・アドミニストレータ] が表示されます。

- [ユーザ DSN] タブで [追加] をクリックします。

  [データ・ソースの新規作成] ダイアログが表示されます。

- [iAnywhere Solutions 9 - DB2 Wire Protocol] を選択し、[完了] をクリックします。

  [ODBC DB2 Wire Protocol Driver Setup] ダイアログが表示されます。

- [General] タブで、データ・ソース名として **CustDB** と入力し、適切な IP アドレスと TCP ポートを入力します。IBM DB2 サンプル・データベースを参照する場合、データベース名として SAMPLE を使用してください。

- [Bind] タブで、[Create Package] をクリックします。IBM DB2 インスタンスのユーザ名とパスワードを入力します。

- [OK] をクリックします。

7 以下のようにして、DB2 クライアント・マシン上で *custdb2setuplong* Java アプリケーションを実行します。このアプリケーションは、DB2 データベースの CustDB サンプルをリセットします。初期設定が終了したら、同じコマンド・ラインを入力してこのアプリケーションを実行し、DB2 CustDB データベースをいつでもリセットできます。

- データ・ソースに CustDB 以外の名前を使用する場合は、以下のように入力して *custdb2setuplong.java* の接続コードを変更し、再コンパイルする必要があります。システム変数 *%db2tempdir%* で指定するパスにスペースが含まれている場合は、パスを引用符で囲んでください。

```
javac -g -classpath
%db2tempdir%¥java¥jdk¥lib¥classes.zip;
%db2tempdir%¥java¥db2java.zip;
%db2tempdir%¥java¥runtime.zip
custdb2setuplong.java
```

- 次のように入力します。ここでは、*userid* と *password* は CustDB ODBC データ・ソースに接続するためのユーザ名とパスワードです。

  ```
  java custdb2setuplong userid password
  ```

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、SQL Anywhere 9 インストール環境の *Samples¥UltraLite¥CustDB* サブディレクトリにあります。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 *custdbapi.cpp* には、C++ API の論理が格納されています。
- **ユーザ・インタフェース機能** この機能は、*Samples¥UltraLite¥CustDB* のプラットフォーム固有のサブディレクトリに、別々に格納されています。

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

❖ サンプル・アプリケーションをリモート・デバイスにインストールするには、次の手順に従います。

1　統合データベースを起動します。

2　Mobile Link 同期サーバを起動します。

3　サンプル・アプリケーションをリモート・デバイスにインストールします。

4　リモート・デバイスでサンプル・アプリケーションを起動します。

5　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 Palm デバイスに CustDB サンプルをインストールします。

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 コマンド・ウィンドウを開き、DB2 データベースに対して次のコマンド・ラインを実行します。*userid* と *password* には、DB2 データベースに接続するためのユーザ ID とパスワードを指定します。

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

2. Mobile Link 同期サーバを起動します。

   - DB2 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。

     ```
     dbmlsrv9 -c "DSN=CustDB" -zp
     ```

3. 次のようにして、サンプル・アプリケーションを Palm デバイスにインストールします。

   - 使用中の PC で、Palm Desktop を起動します。
   - [Palm Desktop] ツールバーの [インストール] をクリックします。

- [追加] をクリックし、SQL Anywhere 9 インストール環境の *UltraLite¥palm¥68k* サブディレクトリにある *custdb.prc* を検索します。
- [開く] をクリックします。
- Palm デバイスで HotSync を実行します。

4. 次のようにして、Palm デバイスで CustDB サンプル・アプリケーションを起動します。

   - Palm デバイスをクレードルに置きます。

     サンプル・アプリケーションを初めて起動した場合は、データの初期コピーの同期とダウンロードを実行することを求めるメッセージが表示されます。この手順が必要なのは、アプリケーションを初めて起動する場合だけです。これによってダウンロードされたデータは、Ultra Light データベースに格納されます。

   - サンプル・アプリケーションを起動します。

     [Applications] ビューで、[CustDB] を選択します。

     初期ダイアログが表示され、従業員 ID を入力するプロンプトが表示されます。

   - 従業員 ID を入力します。

     チュートリアルとして実行するときは、値 50 を入力してください。このサンプル・アプリケーションでは、51、52、または 53 の値も使用できますが、これらの値を入力した場合は、動作が多少異なります。

     各ユーザ ID を使用したときの動作の詳細については、「CustDB サンプル内のユーザ」151 ページを参照してください。

     次の処理に進む前に同期を行う必要があることを示すメッセージ・ボックスが表示されます。

   - アプリケーションを同期します。

     HotSync を使用して、データの初期コピーを取得します。

- データが同期されたことを確認します。

  [Applications] ビューで、[CustDB] アプリケーションを選択します。顧客用の入力シートにデータが入力されて画面に表示されます。

5. 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。データベースを変更したときは、この手順のみを行ってください。

   - 統合データベースと Mobile Link 同期サーバが動作中であることを確認します。
   - Palm デバイスをクレードルに置きます。
   - HotSync ボタンを押して同期を行います。

クリーンアップ

このサンプルを再度起動するために、CustDB データベースのデータをリセットすることができます。CustDB Ultra Light データベースのデータを最初の状態に戻すには、次の手順を行います。

❖ サンプル・アプリケーションのデータをリセットするには、次の手順に従います。

1　次のようにして、ULUtil をデバイスにインストールします。

- Palm デバイス用に、PC で Palm Desktop を起動します。
- [Palm Desktop] ツールバーの [インストール] をクリックします。
- [追加] をクリックし、SQL Anywhere 9 インストール環境の *UltraLite¥palm¥68k* サブディレクトリにある *ulutil.prc* を検索します。
- [完了] をクリックします。
- Palm デバイスで HotSync を実行します。

2　次のように、ULUtil を使用してデータを削除します。

- Palm デバイスで [ULUtil] アイコンを選択します。

- CustDB を選択し、[データを削除します] を選択します。
- Palm デバイスで HotSync を実行します。

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、SQL Anywhere 9 インストール環境の *Samples¥MobiLink¥CustDB* サブディレクトリにあるプラットフォーム固有のファイル内に格納されています。

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

ULCustomer

ULCustomer テーブルには、顧客リストがあります。

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

- **cust_id**　顧客を識別するユニークな整数を保持するプライマリ・キー・カラム。
- **cust_name**　顧客の名前を保持する 30 バイトの文字列。

統合データベースの ULCustomer には、次のカラムもあります。

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

ULProduct

ULProduct テーブルには、製品リストがあります。

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

- **prod_id**　製品を識別するユニークな整数を保持するプライマリ・キー・カラム。
- **price**　単価を識別する整数。
- **prod_name**　製品の名前を保持する 30 バイトの文字列。

ULOrder

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

リモート・データベースの ULOrder には、次のカラムがあります。

- **order_id** 注文を識別するユニークな整数を保持するプライマリ・キー・カラム。
- **cust_id** ULCustomer を参照する外部キー・カラム。
- **prod_id** ULProduct を参照する外部キー・カラム。
- **emp_id** ULEmployee を参照する外部キー・カラム。
- **disc** 注文に適用される値引きを保持する整数。
- **quant** 注文された製品の数を保持する整数。
- **notes** 注文に関する注記を保持する 50 バイトの文字列。
- **status** 注文のステータスが記述された 20 バイトの文字列。

統合データベースの ULOrder には、次のカラムもあります。

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

ULOrderIDPool

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

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

- **pool_order_id** 注文 ID を識別するユニークな整数を保持するプライマリ・キー・カラム。

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

- **pool_emp_id** 注文 ID が割り当てられたリモート・データベースの所有者の従業員 ID を保持する整数カラム。
- **last_modified** ローが最後に変更されたときのタイムスタンプ。

ULCustomerIDPool

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

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

- **pool_cust_id**　顧客 ID を識別するユニークな整数を保持するプライマリ・キー・カラム。

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

- **pool_emp_id**　リモート・データベースで生成された新しい従業員に使用される従業員 ID を保持する整数カラム。
- **last_modified**　ローが最後に変更されたときのタイムスタンプ。

以下のテーブルは、統合データベースにのみ存在します。

ULIdentifyEmployee_nosync

ULIdentifyEmployee_nosync テーブルは、統合データベースにのみ存在します。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。Adaptive Server 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 と Oracle にはテンポラリ・テーブルがないため、同期ユーザに属するローを Mobile Link が識別できるようになっている必要があります。これらのテーブルはベース・テーブルであるため、5 人のユーザが同期している場合は、これらのテーブルのローを各ユーザが同時に保持することがあります。

@@spid の詳細については、『ASA SQL リファレンス・マニュアル』>「変数」を参照してください。

CustDB サンプル内のユーザ

CustDB サンプルのユーザには、営業担当者とモバイル管理者の 2 つのタイプがあります。相違点は次のとおりです。

- **営業担当者**　営業担当者に関連付けられたリモート・データベースは、ユーザ ID 51、52、53 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
    - 注文の承認や拒否
    - 変更内容に関する統合データベースとの同期
- **モバイル管理者**　モバイル管理者に関連付けられたリモート・データベースは、ユーザ ID 50 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

以下の項では、CustDB サンプルの同期論理を説明します。

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

CustDB サンプルでの注文の同期

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は、最終ダウンロード・タイムスタンプです。これは、最後の同期以後にリモート・データベースまたは統合データベースのいずれかで修正されたローのみをダウンロードするために使用されます。2 番目のパラメータは従業員 ID です。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( ?, ? )
```

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

```
ALTER 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, 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 ( ( o.status = 'Approved' AND o.last_modified >= ?
)
  OR ( ec.action = 'D'  )  )
  AND ec.emp_id = ?
```

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", "disc", "quant", "notes", "status" )
   VALUES ( ?, ?, ?, ?, ?, ?, ? )
```

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

```
UPDATE ULOrder SET cust_id=?, prod_id=?, emp_id=?,
disc=?, quant=?, notes=?, status=?
    WHERE 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" = ?
```

- **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 = ?
```

- **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( ?, ?, ?, ?, ?, ?, ?, ? )
```

- **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( ?, ?, ?, ?, ?, ?, ?, ? )
```

競合解決

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

```
   CALL ULResolveOrderConflict
```

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

```
ALTER 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 >= ?
```

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 ( ?, ? )
```

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

```
UPDATE ULCustomer SET cust_name = ?
WHERE "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 = ?
```

CustDB サンプルの製品の同期

ビジネス・ルール

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

```
SELECT pool_cust_id FROM ULCustomerIDPool
   WHERE last_modified >= ?
     AND pool_emp_id = ?
```

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( ? )
```

- **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 = ?
```

- **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 ULCustomerIDPool_maintain( ? )
```

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

```
ALTER 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 >= ?
     AND pool_emp_id = ?
```

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( ? )
```

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

```
ALTER PROCEDURE ULOrderIDPool_maintain ( IN
syncuser_id INTEGER )
 BEGIN
   DECLARE pool_count INTEGER;
   -- Determine how many ids to add to the pool
   SELECT COUNT(*) INTO pool_count
     FROM ULOrderIDPool
```

```
    WHERE pool_emp_id = syncuser_id;
  -- Top up the pool with new ids
  WHILE pool_count < 20 LOOP
    INSERT INTO ULOrderIDPool ( pool_emp_id )
      VALUES ( syncuser_id );
    SET pool_count = pool_count + 1;
  END LOOP;
END
```

- **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(
? )
```

- **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 = ?
```

詳細情報

さらに詳細な情報については、まず次の役立つ情報を参照してください。

スクリプトの種類の詳細については、『Mobile Link 管理ガイド』> 「スクリプトの種類」を参照してください。

各スクリプトやそのパラメータなどのリファレンス情報については、『Mobile Link 管理ガイド』> 「同期イベント」を参照してください。

索引

記号

C

D

I

J

M

O

S

U

あ

う

き

く

さ

す

ち

て

と

に

は

ひ

ふ

ま

ろ