SYBASE iAnywhere

Ultra Light
AppForge プログラミング

改訂 2007 年 3 月

版権と商標

目次

はじめに

このマニュアルの内容

このマニュアルでは、Ultra Light for AppForge について説明します。Ultra Light for AppForge を使用すると、Palm OS、Symbian OS、または Windows CE を搭載しているハンドヘルド、モバイル、または埋め込みデバイスに対してデータベース・アプリケーションを開発、配備できます。

対象読者

このマニュアルは、Ultra Light リレーショナル・データベースのパフォーマンス、リソース効率、堅牢性、セキュリティを利用してデータを格納、同期することを目的とする AppForge アプリケーション開発者を対象にしています。

SQL Anywhere 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 のマニュアル

SQL Anywhere の完全なマニュアルは、各マニュアルをまとめたオンライン形式とマニュアル別の PDF ファイルで提供されます。いずれの形式のマニュアルも、同じ情報が含まれ、次のマニュアルから構成されます。

- **『SQL Anywhere 10 - 紹介』** このマニュアルでは、データの管理および交換機能を提供する包括的なパッケージである SQL Anywhere 10 について説明します。SQL Anywhere を使用すると、サーバ環境、デスクトップ環境、モバイル環境、リモート・オフィス環境に適したデータベース・ベースのアプリケーションを迅速に開発できるようになります。
- **『SQL Anywhere 10 - 変更点とアップグレード』** このマニュアルでは、SQL Anywhere 10 とそれ以前のバージョンに含まれる新機能について説明します。
- **『SQL Anywhere サーバ - データベース管理』** このマニュアルでは、SQL Anywhere データベースの実行、管理、設定について説明します。管理ユーティリティとオプションのほか、データベース接続、データベース・サーバ、データベース・ファイル、バックアップ・プロシージャ、セキュリティ、高可用性、Replication Server を使用したレプリケーションについて説明します。
- **『SQL Anywhere サーバ - SQL の使用法』** 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。
- **『SQL Anywhere サーバ - SQL リファレンス』** このマニュアルは、SQL Anywhere で使用する SQL 言語の完全なリファレンスです。また、SQL Anywhere のシステム・ビューとシステム・プロシージャについても説明しています。
- **『SQL Anywhere サーバ - プログラミング』** このマニュアルでは、C、C++、Java プログラミング言語、Visual Studio .NET 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。
- **『SQL Anywhere 10 - エラー・メッセージ』** このマニュアルでは、SQL 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。
- **『Mobile Link - クイック・スタート』** このマニュアルでは、セッションベースのリレーショナル・データベース同期システムである Mobile Link について説明します。Mobile Link テクノロジは、双方向レプリケーションを可能にし、モバイル・コンピューティング環境に非常に適しています。
- **『Mobile Link - サーバ管理』** このマニュアルでは、Mobile Link アプリケーションを設定して管理する方法について説明します。

- **『Mobile Link - クライアント管理』** このマニュアルでは、Mobile Link クライアントを設定、構成、同期する方法について説明します。Mobile Link クライアントには、SQL Anywhere または Ultra Light のいずれかのデータベースを使用できます。
- **『Mobile Link - サーバ起動同期』** 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またはその他のリモート・アクションの開始を可能にする Mobile Link の機能です。
- **『QAnywhere』** このマニュアルでは QAnywhere について説明します。QAnywhere は、従来のデスクトップ・クライアントやラップトップ・クライアント用のメッセージング・プラットフォームであるほか、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・プラットフォームでもあります。
- **『SQL Remote』** このマニュアルでは、モバイル・コンピューティング用の SQL Remote データ・レプリケーション・システムについて説明します。このシステムによって、SQL Anywhere の統合データベースと複数の SQL Anywhere リモート・データベースの間で、電子メールやファイル転送などの間接的リンクを使用したデータ共有が可能になります。
- **『SQL Anywhere 10 - コンテキスト別ヘルプ』** このマニュアルには、[接続] ダイアログ、クエリ・エディタ、Mobile Link モニタ、SQL Anywhere コンソール・ユーティリティ、インデックス・コンサルタント、Interactive SQL のコンテキスト別のヘルプが収録されています。
- **『Ultra Light - データベース管理とリファレンス』** このマニュアルでは、小型デバイス用 Ultra Light データベース・システムの概要を説明します。
- **『Ultra Light - AppForge プログラミング』** このマニュアルでは、Ultra Light for AppForge について説明します。Ultra Light for AppForge を使用すると、Palm OS、Symbian OS、または Windows CE を搭載しているハンドヘルド、モバイル、または埋め込みデバイスに対してデータベース・アプリケーションを開発、配備できます。
- **『Ultra Light - .NET プログラミング』** このマニュアルでは、Ultra Light.NET について説明します。Ultra Light.NET を使用すると、PC、ハンドヘルド、モバイル、埋め込みデバイスのデータベース・アプリケーションを開発し、これらのデバイスに配備できます。
- **『Ultra Light - M-Business Anywhere プログラミング』** このマニュアルは、Ultra Light for M-Business Anywhere について説明します。Ultra Light for M-Business Anywhere を使用すると、Palm OS、Windows CE、または Windows XP を搭載しているハンドヘルド、モバイル、または埋め込みデバイスに対して Web ベースのデータベース・アプリケーションを開発、配備できます。
- **『Ultra Light - C/C++ プログラミング』** このマニュアルでは、Ultra Light C および Ultra Light C++ のプログラミング・インタフェースについて説明します。Ultra Light を使用すると、ハンドヘルド、モバイル、埋め込みデバイスに対してデータベース・アプリケーションを開発、配備できます。

マニュアルの形式

SQL Anywhere 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 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] − [プログラム] − [SQL Anywhere 10] − [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・ディレクトリまたはインストール CD に保存されている HTML マニュアルを参照してください。

- **PDF ファイル** SQL Anywhere の完全なマニュアル・セットは、Adobe Reader で表示できる Adobe Portable Document Format (pdf) 形式のファイルとして提供されています。

  Windows では、PDF 形式のマニュアルはオンライン・マニュアルの各ページ上部にある PDF のリンクから、または Windows の [スタート] メニュー ([スタート] − [プログラム] − [SQL Anywhere 10] − [オンライン・マニュアル - PDF フォーマット]) からアクセスできます。

  UNIX では、PDF 形式のマニュアルはインストール CD にあります。

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 つだけ選択する場合や、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、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。

オペレーティング・システムの表記規則

- **Windows**　デスクトップおよびラップトップ・コンピュータ用の Microsoft Windows オペレーティング・システムのファミリのことです。Windows ファミリには Windows Vista や Windows XP も含まれます。
- **Windows CE**　Microsoft Windows CE モジュラ・オペレーティング・システムに基づいて構築されたプラットフォームです。Windows Mobile や Windows Embedded CE などのプラットフォームが含まれます。

  Windows Mobile は Windows CE 上に構築されています。これにより、Windows のユーザ・インタフェースや、Word や Excel といったアプリケーションの小規模バージョンなどの追加機能が実現されています。Windows Mobile は、モバイル・デバイスで最も広く使用されています。

  SQL Anywhere の制限事項や相違点は、基盤となっているオペレーティング・システム (Windows CE) に由来しており、使用しているプラットフォーム (Windows Mobile など) に依存していることはほとんどありません。
- **UNIX**　特に記述がないかぎり、UNIX は Linux プラットフォームと UNIX プラットフォームの両方のことです。

ファイルの命名規則

マニュアルでは、パス名やファイル名などのオペレーティング・システムに依存するタスクと機能を表すときは、通常 Windows の表記規則が使用されます。ほとんどの場合、他のオペレーティング・システムで使用される構文に簡単に変換できます。

- **ディレクトリ名とパス名**　マニュアルでは、ドライブを示すコロンや、ディレクトリの区切り文字として使用する円記号など、Windows の表記規則を使用して、ディレクトリ・パスのリストを示します。次に例を示します。

  ```
  MobiLink¥redirector
  ```

  UNIX、Linux、Mac OS X では、代わりにスラッシュを使用してください。次に例を示します。

  ```
  MobiLink/redirector
  ```

  SQL Anywhere がマルチプラットフォーム環境で使用されている場合、プラットフォーム間でのパス名の違いに注意する必要があります。
- **実行ファイル**　マニュアルでは、実行ファイルの名前は、Windows の表記規則が使用され、拡張子 *.exe* が付きます。UNIX、Linux、Mac OS X では、実行ファイルの名前には拡張子は付きません。NetWare では、実行ファイルの名前には、拡張子 *.nlm* が付きます。

  たとえば、Windows では、ネットワーク・データベース・サーバは *dbsrv10.exe* です。UNIX、Linux、Mac OS X では、*dbsrv10* になります。NetWare では、*dbsrv10.nlm* になります。
- **install-dir**　インストール・プロセスでは、SQL Anywhere をインストールするロケーションを選択できます。マニュアルでは、このロケーションは *install-dir* という表記で示されます。

インストールが完了すると、環境変数 SQLANY10 によって SQL Anywhere コンポーネントがあるインストール・ディレクトリのロケーション (*install-dir*) が指定されます。SQLANYSH10 は、SQL Anywhere が他の Sybase アプリケーションと共有しているコンポーネントがあるディレクトリのロケーションを指定します。

オペレーティング・システム別の *install-dir* のデフォルト・ロケーションの詳細については、「SQLANY10 環境変数」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **samples-dir** インストール・プロセスでは、SQL Anywhere に含まれるサンプルをインストールするロケーションを選択できます。マニュアルでは、このロケーションは *samples-dir* という表記で示されます。

  インストールが完了すると、環境変数 SQLANYSAMP10 によってサンプルがあるディレクトリのロケーション (*samples-dir*) が指定されます。Windows の [スタート] メニューから、[プログラム] – [SQL Anywhere 10] – [サンプル・アプリケーションおよびプロジェクト] を選択すると、このディレクトリで [Windows エクスプローラ] ウィンドウが表示されます。

  オペレーティング・システム別の *samples-dir* のデフォルト・ロケーションの詳細については、「サンプル・ディレクトリ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **環境変数** マニュアルでは、環境変数設定が引用されます。Windows では、環境変数を参照するのに、構文 *%envvar%* が使用されます。UNIX、Linux、Mac OS X では、環境変数を参照するのに、構文 *$envvar* または *${envvar}* が使用されます。

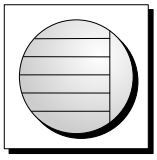
  UNIX、Linux、Mac OS X 環境変数は、*.cshrc* や *.tcshrc* などのシェルとログイン・スタートアップ・ファイルに格納されます。

グラフィック・アイコン

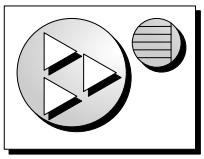
このマニュアルでは、次のアイコンを使用します。

- クライアント・アプリケーション

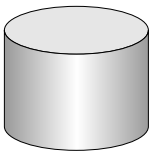
- SQL Anywhere などのデータベース・サーバ



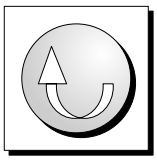
- Ultra Light アプリケーション



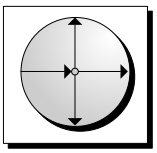
- データベース。高度な図では、データベースとデータベースを管理するデータ・サーバの両方をこのアイコンで表します。



- 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 サーバ、SQL Remote Message Agent などが挙げられます。



- Sybase Replication Server



- 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 Sybase iAnywhere 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 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 **dbeng10 -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
- sybase.public.sqlanywhere.ultralite
- ianywhere.public.sqlanywhere.qanywhere

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 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

Ultra Light for AppForge の概要

目次

Ultra Light for AppForge の特徴

Ultra Light for AppForge は、モバイル・デバイスのためのリレーショナル・データ管理システムです。ビジネス・アプリケーションに必要なパフォーマンス、リソース効率、堅牢性、セキュリティを備えています。Ultra Light では、エンタープライズ・データ・ストアとの同期も提供されます。

システムの稼働条件とサポートされるプラットフォーム

開発プラットフォーム

Ultra Light for AppForge を使用してアプリケーションを開発するには、以下が必要です。

- Microsoft .NET (Visual Basic .NET または C#) または Visual Basic 6

  お使いの AppForge MobileVB または AppForge Crossfire のバージョンの要件に合うサービス・パックをインストールしてください。詳細については、AppForge の Web サイトを参照してください。Visual Basic 6 を使用している場合は、少なくともサービス・パック 5 をインストールすることをおすすめします。

  AppForge Client
  Ultra Light for AppForge を使用してアプリケーションを配備するには、ターゲット・デバイスに適した AppForge Client が必要です。AppForge Client の詳細については、AppForge の Web サイトを参照してください。

- AppForge MobileVB または AppForge Crossfire

ターゲット・プラットフォーム

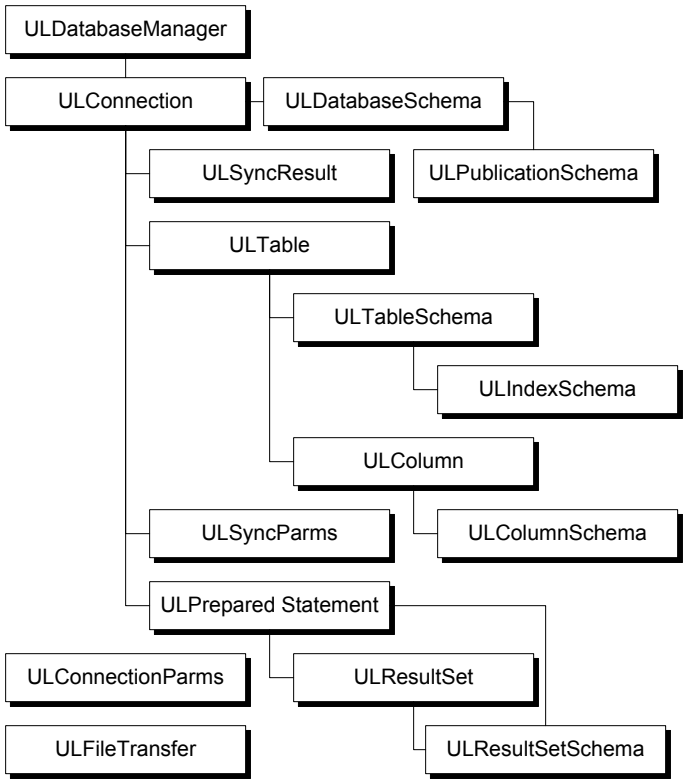
Ultra Light for AppForge は、次のターゲット・プラットフォームをサポートしています。

- ARM プロセッサの Pocket PC に搭載した Windows CE 3.0 以降 (Windows Mobile 5.0 を含む)
- ARMI を使用した Sony Ericsson UIQ 2.08 (2.1 と上位互換性あり)
- ARMI を使用した Nokia Series 60 および Series 80
- ARMI を使用した Motorola 1000
- Palm OS バージョン 4 以降

配備の詳細については、「SQL Anywhere 用 Ultra Light 展開オプション」を参照してください。

Ultra Light for AppForge のアーキテクチャ

Ultra Light プログラミング・インタフェースは、Ultra Light データベースを使用したデータ操作のためのオブジェクト・セットを公開しています。次の図は、オブジェクト階層を示します。



次のリストは、よく使用される高度なオブジェクトの一部を示します。

- **ULDatabaseManager** Ultra Light データベースへの接続を管理します。

  「ULDatabaseManager クラス」 92 ページを参照してください。

- **ULConnectionParms** 接続パラメータのセットを格納します。

  Connection Parameters コントロールを使用し、Visual Basic プロパティ・シートに接続パラメータを指定できます。

  「ULConnectionParms クラス」 90 ページを参照してください。

- **ULFileTransfer** Mobile Link サーバを使用したファイル転送を管理します。

  「ULFileTransfer クラス」 99 ページを参照してください。

- **ULConnection** データベース接続を表し、トランザクションを管理します。

「ULConnection クラス」 81 ページを参照してください。

- **ULPreparedStatement、ULResultSet、ULResultSetSchema** SQL を使用してデータベース要求とその結果を管理します。

  次の項を参照してください。

  - 「ULPreparedStatement クラス」 105 ページ
  - 「ULResultSet クラス」 112 ページ
  - 「ULResultSetSchema クラス」 125 ページ

- **ULTable と ULColumn** テーブル・ベースの API を使用してデータを管理します。

  次の項を参照してください。

  - 「ULTable クラス」 152 ページ
  - 「ULColumn クラス」 73 ページ

- **ULSyncParms と ULSyncResult** Mobile Link サーバを介して同期を管理します。

  Mobile Link との同期の詳細については、「Ultra Light クライアント」 『Mobile Link - クライアント管理』を参照してください。

第 2 章

AppForge を使用した Ultra Light 開発の概要

目次

Ultra Light for AppForge を使用するための準備

以下の手順では、Ultra Light for AppForge を使用してアプリケーションを構築する前に実行する必要のある操作を示します。

MobileVB 設計環境への Ultra Light の追加

MobileVB または Crossfire プロジェクトから Ultra Light コントロールにアクセスするには、Ultra Light for MobileVB を設計環境に追加します。

♦ **Ultra Light 接続パラメータ・コントロールを追加するには、次の手順に従います。**

1. Visual Basic メニューから、[プロジェクト] – [コンポーネント] を選択します。
2. [コントロール] タブをクリックします。
3. リストを下にスクロールして [UltraLite Connection Parameters 10.0] を選択します。[OK] をクリックします。

   使用可能なコントロールのリストにこの項目が表示されない場合は、次の手順を実行します。

   - Visual Basic を閉じ、プロジェクトを保存します。
   - コマンド・プロンプトを開き、次のコマンドを実行します。

     ```
     ulafreg -r
     ```

     「Ultra Light AppForge レジストリ・ユーティリティ (ulafreg)」『Ultra Light - データベース管理とリファレンス』を参照してください。
   - Visual Basic を再起動し、プロジェクトを開きます。
   - [プロジェクト] – [コンポーネント] を選択します。
   - [UltraLite Connection Parameters 10.0] を選択します。

   データベース・アイコンがツールバーに追加されます。このアイコンをダブルクリックして、ULConnectionParms オブジェクトをフォームに追加します。

Ultra Light for MobileVB への参照の追加

SQL Anywhere がインストールされている場合、Ultra Light for MobileVB は新しい MobileVB プロジェクトに自動的に追加されます。このため、通常は Ultra Light for MobileVB への参照をプロジェクトに手動で追加する必要はありません。次の手順は、SQL Anywhere のインストール後に MobileVB をインストールする場合など、参照を手動で追加する必要がある特別なケースに使用します。

♦ **Ultra Light for MobileVB への参照を追加するには、次の手順に従います。**

1. Visual Basic メニューから、[プロジェクト] – [参照設定] を選択します。
2. [iAnywhere Solutions, Ultra Light for MobileVB 10.0] コントロールが利用可能な参照のリストに含まれている場合、それを選択して [OK] をクリックします。

   [iAnywhere Solutions, Ultra Light for MobileVB 10.0] コントロールが利用可能な参照のリストにない場合は、次の手順に従います。

   ♦ コマンド・プロンプトを開き、次のコマンドを実行します。

   ```
   ulafreg -r
   ```

   「Ultra Light AppForge レジストリ・ユーティリティ (ulafreg)」『Ultra Light - データベース管理とリファレンス』を参照してください。

   ♦ [iAnywhere Solutions, UltraLite for MobileVB 10.0] を選択し、[OK] をクリックします。

Crossfire 設計環境への Ultra Light の追加

SQL Anywhere のインストーラでは、Ultra Light が Crossfire 設計環境に自動的に追加されますが、Ultra Light を環境に手動で追加することが必要な場合があります。たとえば、SQL Anywhere のインストール後に Crossfire をインストールした場合は、この手順を実行することが必要な場合があります。

Ultra Light を Crossfire に追加する必要があるかどうかを調べるには、新しい Crossfire プロジェクトに iAnywhere.UltraLiteForAppForge への参照が含まれているかどうかをチェックします。含まれていない場合は、Ultra Light を環境に追加する必要があります。また、ULConnectionParms クラスがツールボックスの AppForge パネルに表示されるかどうかをチェックします。表示されていない場合は、Ultra Light を環境に追加する必要があります。

♦ **Crossfire プロジェクトに Ultra Light の参照とコントロールを追加するには、次の手順に従います。**

1. Ultra Light for MobileVB を Crossfire に登録します。

   a. Crossfire が閉じていることを確認します。

   b. コマンド・プロンプトを開き、次のコマンドを実行します。

   ```
   ulafreg -r
   ```

   「Ultra Light AppForge レジストリ・ユーティリティ (ulafreg)」『Ultra Light - データベース管理とリファレンス』を参照してください。

   c. MobileVB プロジェクトをアップグレードした場合は、Visual Basic.NET ソリューション・エクスプローラから UltraLiteAFLib への参照を削除します。

   d. 参照を *iAnywhere.UltraLiteForAppForge.dll* に追加します。

i. [Microsoft Development Environment] メニューから、[プロジェクト] – [参照の追加] を選択し、SQL Anywhere インストール環境の *install-dir¥ultralite ¥UltraLiteForAppForge¥win32* サブディレクトリを参照します。

ii. *iAnywhere.UltraLiteForAppForge.dll* 選択し、[開く] をクリックします。

iii. [OK] をクリックして参照を追加します。

2. AppForge ツールボックスに ULConnectionParms コントロールを追加します。

a. Microsoft Development Environment で、AppForge ツールボックスを右クリックし、[Add/Remove Items] を選択します。ダイアログが表示されます。

b. [COM Components] タブをクリックします。

c. [ULConnectionParms Class] という名前のエントリにスクロールします。このコンポーネントの横のボックスをオンにして、[OK] をクリックします。

d. ULConnectionParms コントロールがツールボックスに追加されます。

Ultra Light データベースの作成

Ultra Light データベースの作成には、Sybase Central または ulcreate ユーティリティを使用します。

- **Sybase Central での Ultra Light** [データベース作成] ウィザードを使用して、Ultra Light データベースを作成します。

  「Sybase Central からの Ultra Light データベースの作成」『Ultra Light - データベース管理とリファレンス』を参照してください。

- **ulcreate ユーティリティ** ulcreate ユーティリティを使用して、空の Ultra Light データベースを作成できます。

  「Ultra Light データベース作成ユーティリティ (ulcreate)」『Ultra Light - データベース管理とリファレンス』を参照してください。

アプリケーションでは、Ultra Light の CreateDatabase 関数を使用して、Ultra Light データベースを動的に作成できます。アプリケーションの配備環境では、アプリケーションを使用した追加ファイルの配備がサポートされているので、ほとんどのアプリケーションはプログラム・コードとともに初期データベースを配布することで単純化できます。Ultra Light データベースは 1 つのファイルで構成されています。

Ultra Light データベースへの接続

データベースのデータを操作するには、Ultra Light アプリケーションをデータベースに接続する必要があります。この項では、Ultra Light データベースに接続する方法について説明します。

ULConnection オブジェクトの使用

ULConnection オブジェクトの次のプロパティは、アプリケーションのグローバルな動作を管理します。

「ULConnection クラス」 81 ページを参照してください。

- **コミット動作** デフォルトでは、Ultra Light アプリケーションは AutoCommit モードに設定されています。insert 文、update 文、delete 文はすべて、すぐにデータベースにコミットされます。ULConnection.AutoCommit を false に設定し、アプリケーションにトランザクションを構築します。AutoCommit を off に設定しコミットを実行すると、アプリケーションのパフォーマンスを直接向上できます。

  「Commit メソッド」 83 ページを参照してください。

- **ユーザ認証** GrantConnectTo メソッドと RevokeConnectFrom メソッドを使用すると、アプリケーションのユーザ ID とパスワードをデフォルト値の DBA と sql から別の値に変更できます。

  「ユーザの認証」 29 ページを参照してください。

- **同期** 同期を管理するオブジェクトのセットは、ULConnection オブジェクトからアクセスできます。

  「データの同期」 30 ページを参照してください。

- **テーブル** ULConnection.GetTable メソッドを使用して、Ultra Light テーブルにアクセスします。

  「GetTable メソッド」 84 ページを参照してください。

データベースへの接続

ULConnectionParms オブジェクトまたは接続文字列を使用して、データベースに接続できます。ULConnectionParms オブジェクトを使用して、異なるターゲット・デバイス・プラットフォームの複数の接続パラメータを操作します。接続文字列を使用したメソッドでは、1 つの長い文字列に異なるターゲット・プラットフォームの文字列を指定する必要があります。

接続パラメータの詳細については、「Ultra Light の接続文字列パラメータのリファレンス」『Ultra Light - データベース管理とリファレンス』を参照してください。

次に、Ultra Light データベースへの接続手順を説明します。

- **Ultra Light データベースに接続します。**

1. ULDatabaseManager オブジェクトを作成します。

DatabaseManager オブジェクトは、1 つのアプリケーションに 1 つだけ作成してください。このオブジェクトは、オブジェクト階層のルートにあります。そのため、DatabaseManager オブジェクトは、アプリケーションに対してグローバルに、またはクラスレベル変数として宣言するのが最も効果的です。

```
'MobileVB using VB6
Public DatabaseMgr As ULDatabaseManager
Set DatabaseMgr = New ULDatabaseManager

'Crossfire using vb.net
Public DatabaseMgr As New UltraLiteAFLib.ULDatabaseManager
```

2. ULConnection オブジェクトを宣言します。

   ほとんどのアプリケーションは、Ultra Light データベースへの単一接続を使用し、接続を常時開いています。そのため、ULConnection オブジェクトは、アプリケーションに対してグローバルに宣言するのが最も効果的です。

   ```
   'MobileVB using VB6
   Public Connection As New ULConnection

   'Crossfire using vb.net
   Public Connection As UltraLiteAFLib.ULDatabaseManager
   ```

3. ULConnectionParms オブジェクトを作成します。

   MobileVB ツール・パレットの ULConnectionParms オブジェクトをダブルクリックします。ULConnectinParms オブジェクトがフォームに表示されます。

4. ULConnectionParms オブジェクトの必須プロパティを設定します。

   ULConnectionParms プロパティ・ウィンドウで、データベースのロケーション、ユーザ名、パスワードなどのプロパティを指定します。

   次のプロパティを使用して、OpenConnection で使用するデータベース・ファイルを指定する必要があります。

   追加プロパティの詳細については、「プロパティ」 90 ページを参照してください。

キーワード	説明
DatabaseOnCE	Windows CE 上の Ultra Light データベースのパスとファイル名。
DatabaseOnDesktop	デスクトップ・コンピュータ上の Ultra Light データベースのパスとファイル名。

5. データベースへの接続を開きます。

   OpenConnection は、開いた接続を ULConnection オブジェクトとして返します。このメソッドは、1 つの ULConnectionParms オブジェクトを引数として取ります。

   次のコードは、既存のデータベースに接続しようとします。データベースが存在しない場合に、OpenConnection メソッドはエラーを返します。


```
'MobileVB using VB6
On Error Resume Next
Set Connection = DatabaseMgr.OpenConnection(ULConnectionParms1.ToString())

'Crossfire using vb.net
Try
  Connection = _
    DatabaseMgr.OpenConnection(ULConnectionParms1.ToString())
Catch
  If Err.Number = _
    UltraLiteAFLib.ULSQLCode.ulSQLE_ULTRALITE_DATABASE_NOT_FOUND _
  Then
    ...
End Try


// Crossfire using C#
using UltraLiteAFLib;
...
  public  UltraLiteAFLib.ULConnection Connection;
  public  UltraLiteAFLib.ULDatabaseManager DatabaseMgr;
  private UltraLiteAFLib._ULConnectionParms_ingotClass parms;
          // dropped onto design form
  parms.DatabaseOnCE = AppForge.System.AppPath + "¥¥mydb.udb";
  try {
       Connection = DatabaseMgr.OpenConnection( parms.ToString );
  } catch ( Exception ex ) {
    Debug.WriteLn("Connect failed: " + ex.Message );
  }
```

暗号化と難読化

Ultra Light データベースは、難読化または暗号化のいずれかのデータ・セキュリティを使用して作成できます。デフォルトでは、Ultra Light データベースは、データベースのデータを隠すための特定の手法は使用されずに作成されます。Ultra Light データベースを含むファイルを検証し、未加工のディスク・データを表示できるユーティリティを使用すると、データベースに保存されている文字データが公開されてしまう恐れがあります。実際のデータベース・ファイルのフォーマットは独自のものですが、内容は表示可能です。

難読化と暗号化は、作成時に設定できるオプションです。実際の暗号化キーは変更できますが、データベースのデータを難読化するか暗号化するかの選択は、データベースのアンロード、新しいデータベースの作成、データの再ロードを実行しないかぎり変更できません。

暗号化

暗号化したデータベースを作成するには、データベースの作成時に暗号化キーを指定する必要があります。

暗号化されたデータベースへの接続を開くには、ULConnectionParms.EncryptionKey プロパティを使用して、データベースの作成時に使用した暗号化キー文字列を指定します。

EncryptionKey プロパティの詳細については、「Ultra Light DBKEY 接続パラメータ」 『Ultra Light - データベース管理とリファレンス』を参照してください。

暗号化キーを変更するには、新しい暗号化キーを Connection オブジェクトで指定します。アプリケーションでは、既存の暗号化キーを使用して接続してから、新しい暗号化キーを指定する必要があります。次の例では "apricot" が新しい暗号化キーです。

```
Connection.ChangeEncryptionKey("apricot")
```

「ChangeEncryptionKey メソッド」 82 ページを参照してください。

データベースが暗号化された後は、データベースへのすべての接続で正しい暗号化キーを指定する必要があります。そうしないと、接続は失敗します。暗号化キーが不明な場合は、*データベース内のデータを取り出すことはできません。*

難読化

データベースを難読化するには、データベースの作成時に難読化オプションを設定します。難読化とは、データベースの内容を簡単にマスキングする処理で、ユーティリティ・プログラムによってデータベース・ファイルの未加工の内容が公開されてしまうのを防ぎます。難読化されて作成されたデータベースの処理は、ユーザとアプリケーション・プログラムには透過的に行われます。プログラミングに関する特別な考慮事項はありません。

データベース暗号化の詳細については、「Ultra Light obfuscate プロパティ」 『Ultra Light - データベース管理とリファレンス』と「Ultra Light でのセキュリティの考慮事項」 『Ultra Light - データベース管理とリファレンス』を参照してください。

動的 SQL を使用したデータの使用

Ultra Light アプリケーションは、動的 SQL またはテーブル API を使用して、テーブル・データにアクセスできます。この項では、動的 SQL を使用したデータ・アクセスについて説明します。

テーブル API の詳細については、「テーブル API を使用したデータの使用」 20 ページを参照してください。

この項では、動的 SQL を使用して次の操作を行う方法について説明します。

- テーブルのローのスクロール
- 現在のローの値へのアクセス
- テーブルのローの検索
- ローの挿入、削除、更新

この項では、SQL 言語そのものについては説明しません。SQL 機能の詳細については、「SQL 言語の要素」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

動的 SQL に必要な操作手順は、SQL の操作と変わりません。概要については、「SQL 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

データ操作：INSERT、UPDATE、DELETE

Ultra Light では、SQL データ操作言語の操作を実行できます。これらの操作は、ULPreparedStatement クラスのメンバである ExecuteStatement メソッドを使用して実行します。

「ULPreparedStatement クラス」 105 ページを参照してください。

アプリケーションでは、準備文の使用後に Close メソッドを呼び出してリソースを解放することが重要です。

準備文でのパラメータの使用
パラメータのプレースホルダは、? 文字を使用して特定します。INSERT、UPDATE、DELETE では必ず、準備文での並び順に従ってそれぞれの ? が参照されます。たとえば、最初の ? はパラメータ 1、2 番目の ? はパラメータ 2、のようになります。

- **ローを挿入するには、次の手順に従います。**

1. ULPreparedStatement オブジェクトを宣言します。

```
'MobileVB using VB6
Dim PrepStmt As ULPreparedStatement

'Crossfire using vb.net
Dim PrepStmt As UltraLiteAFLib.ULPreparedStatement
```

```
// Crossfire using C#
 ULPreparedStatement PrepStmt = null;
```

2. INSERT 文を準備文オブジェクトに割り当てます。次のコードでは、TableName と ColumnName がテーブルとカラムの名前です。

```
'MobileVB using VB6
Set PrepStmt = Connection.PrepareStatement( _
   "INSERT INTO TableName(ColumnName) VALUES ( ? )")

'Crossfire using vb.net
PrepStmt = Connection.PrepareStatement( _
   "INSERT INTO TableName(ColumnName) VALUES( ? )")

// CrossFire using C#
try {
      PrepStmt = Connection.PrepareStatement("INSERT INTO ...", null);
} catch ( Exception ){
}
if (PrepStmt == null) // failed
```

3. その文にパラメータ値を割り当てます。

```
PrepStmt.SetStringParameter (1, "Bob")
```

4. 文を実行し、コマンドの完了後にリソースを解放します。

```
PrepStmt.ExecuteStatement
PrepStmt.Close()
```

♦ **ローを更新するには、次の手順に従います。**

1. ULPreparedStatement オブジェクトを宣言します。

```
Dim PrepStmt As ULPreparedStatement
```

2. UPDATE 文を準備文オブジェクトに割り当てます。次のコードでは、TableName と ColumnName がテーブルとカラムの名前です。

```
Set PrepStmt = Connection.PrepareStatement( _
   "UPDATE TableName SET ColumnName = ? WHERE ID = ?")
```

3. その文にパラメータ値を割り当てます。

```
PrepStmt.SetStringParameter (1, "newvalue")
PrepStmt.SetStringParameter (2, "oldvalue")
```

4. 文を実行し、コマンドの完了後にリソースを解放します。

```
PrepStmt.ExecuteStatement
PrepStmt.Close()
```

♦ **ローを削除するには、次の手順に従います。**

1. ULPreparedStatement オブジェクトを宣言します。

```
'MobileVB using VB6
Dim PrepStmt As ULPreparedStatement
```



```
'Crossfire using vb.net
Dim PrepStmt As UltraLiteAFLib.ULPreparedStatement
```

2. DELETE 文を準備文オブジェクトに割り当てます。

```
'MobileVB using VB6
Set PrepStmt = Connection.PrepareStatement( _
   "DELETE FROM customer WHERE ID = ?")

'Crossfire using vb.net
PrepStmt = Connection.PrepareStatement( _
   "DELETE FROM customer WHERE ID = ?")
```

3. その文にパラメータ値を割り当てます。

```
PrepStmt.SetStringParameter (1, "oldvalue")
```

4. 文を実行し、コマンドの完了後にリソースを解放します。

```
PrepStmt.ExecuteStatement
PrepStmt.Close()
```

データ検索：SELECT

SELECT 文を実行すると、ULPreparedStatement.ExecuteQuery メソッドは ULResultSet オブジェクトを返します。

ULResultSet クラスには、結果セット内をナビゲーションするためのメソッドが含まれています。値には、ULResultSet クラスのメソッドを使用してアクセスします。

「ULResultSet クラス」 112 ページを参照してください。

例

次のコードでは、SELECT クエリの結果に ULResultSet を使用してアクセスします。最初に割り当てられたとき、ULResultSet は最初のローの前に配置されます。次に ULResultSet.MoveFirst メソッドが呼び出され、結果セットの最初のレコードをナビゲーションします。

結果セットのナビゲーションの詳細については、「動的 SQL を使用したナビゲーション」 18 ページを参照してください。

```
'MobileVB using VB6
Dim MyResultSet As ULResultSet
Dim PrepStmt As ULPreparedStatement
PrepStmt = Connection.PrepareStatement( _
   "SELECT ID, Name FROM customer")
MyResultSet = PrepStmt.ExecuteQuery
MyResultSet.MoveFirst

'Crossfire using vb.net
Dim MyResultSet As UltraLiteAFLib.ULResultSet
Dim PrepStmt As UltraLiteAFLib.ULPreparedStatement
PrepStmt = Connection.PrepareStatement( _
   "SELECT ID, Name FROM customer")
MyResultSet = PrepStmt.ExecuteQuery
MyResultSet.MoveFirst
```


Ultra Light for AppForge では、Ultra Light データベースから結果セットへ特定の型のデータを取得するためのメソッドが提供されています。MobileVB は、Variant データ型の使用をサポートしません。このため、Ultra Light for MobileVB にはすべてのデータ型を処理する関数が備わっています。これらの各メソッドは、次のテンプレートを使用して呼び出されます。ここで、Index は SELECT 文でのカラム名の並び順です。

```
MyResultSetName.MethodName( Index )
```

例

次のコードは、GetString メソッドを使用して、現在のローのカラム値を取得する方法を説明します。

GetString メソッドは次の構文を使用します。ここで、Index は SELECT 文でのカラム名の並び順です。

```
MyResultSetName.GetString(Index)
```

データ修正の結果が反映されるように、MoveRelative(0) メソッドが呼び出され、結果セットの現在のバッファの内容をリフレッシュします。

```
If MyResultSet.RowCount = 0 Then
  lblID.Caption = ""
  txtName.Text = ""
Else
  lblID.Caption = MyResultSet.GetString(1)
  txtName.Text = MyResultSet.GetString(2)
  MyResultSet.MoveRelative(0)
End If
```

次の手順では、SELECT 文を使用して、データベースから情報を取り出します。クエリの結果は、ULResultSet オブジェクトに割り当てられます。

♦ **SELECT 文を実行するには、次の手順に従います。**

1. ULPreparedStatement オブジェクトを宣言します。

   ```
   'MobileVB using VB6
   Dim PrepStmt As ULPreparedStatement

   'Crossfire using vb.net
   Dim PrepStmt As UltraLiteAFLib.ULPreparedStatement
   ```

2. 準備文を ULPreparedStatement オブジェクトに割り当てます。次のコードでは、TableName と ColumnName がテーブルとカラムの名前です。

   ```
   Set PrepStmt = Connection.PrepareStatement( _
      "SELECT ColumnName FROM TableName")
   ```

3. クエリを実行します。

   下のコードでは、AFListBox が SELECT クエリの結果を取得します。

   ```
   Dim MyResultSet As ULResultSet
   Set MyResultSet = PrepStmt.ExecuteQuery
   While MyResultSet.MoveNext
   ```



```
    aflistbox.AddItem MyResultSet.GetString(1)
Wend
```

4. クエリの処理後、結果セットを閉じることでリソースを解放します。

```
MyResultSet.Close()
```

動的 SQL を使用したナビゲーション

Ultra Light for MobileVB は、幅広いナビゲーション作業を行うため、結果セットをナビゲーションする方法を多数提供します。

次の ULResultSet オブジェクトのメソッドを使うと、結果セット内をナビゲーションできます。

- **MoveAfterLast** 最後のローの後に移動します。
- **MoveBeforeFirst** 最初のローの前に移動します。
- **MoveFirst** 最初のローに移動します。
- **MoveLast** 最後のローに移動します。
- **MoveNext** 次のローに移動します。
- **MovePrevious** 前のローに移動します。
- **MoveRelative** いくつかのローを、現在のローを基準にして相対的に移動します。正のインデックス値は結果セット内を前に移動し、負のインデックス値は結果セット内を後ろに移動し、0 はカーソルを移動しません。ロー・バッファを再移植する場合は、0 が便利です。

例

次のコードは、MoveFirst メソッドを使用して、結果セット内をナビゲーションする方法を説明します。

```
'MobileVB using VB6
Set PrepStmt = Connection.PrepareStatement( _
    "SELECT ID, Name FROM customer")
Set MyResultSet = PrepStmt.ExecuteQuery
MyResultSet.MoveFirst

'Crossfire using vb.net
PrepStmt = Connection.PrepareStatement( _
    "SELECT ID, Name FROM customer")
MyResultSet = PrepStmt.ExecuteQuery
MyResultSet.MoveFirst
```

すべての Move メソッドで同じ方法を使用できます。

これらのナビゲーション・メソッドの詳細については、「ULResultSet クラス」 112 ページを参照してください。

ULResultSet schema プロパティ

ULResultSet.Schema プロパティを使うと、クエリのカラムに関する情報を取り出すことができます。ULResultSet.Schema オブジェクトのプロパティは、ColumnName、ColumnCount、ColumnPrecision、ColumnScale、ColumnSize、ColumnSQLType です。

例

次の例は、ULResultSet.Schema を使用して、MobileVB グリッドのスキーマ情報を表示する方法を示しています。この例では、MyResultSet という名前の ULResultSet と grdSchema という名前の MobileVB グリッドが存在すると仮定しています。

```
'MobileVB using VB6
Dim i As Integer
For i = 1 To MyResultSet.Schema.ColumnCount
  grdSchema.AddItem (MyResultSet.Schema.ColumnName(i) _
  & Chr(9) & CStr(MyResultSet.Schema.ColumnSQLType(i))), 0
Next i
grdSchema.AddItem _
  ("Column Name" & Chr(9) & "Column Type"), 0
```


テーブル API を使用したデータの使用

Ultra Light アプリケーションは、動的 SQL またはテーブル API を使用して、テーブル・データにアクセスできます。この項では、テーブル API を使用したデータ・アクセスについて説明します。

動的 SQL の詳細については、「動的 SQL を使用したデータの使用」 14 ページを参照してください。

この項では、テーブル API を使用して次の操作を行う方法について説明します。

- テーブルのローのスクロール
- 現在のローの値へのアクセス
- find メソッドと lookup メソッドを使用したテーブルのローの検索
- ローの挿入、削除、更新

テーブル API を使用したナビゲーション

Ultra Light for MobileVB は、幅広いナビゲーション作業を行うため、テーブルをナビゲーションする方法を多数提供します。

次の ULTable オブジェクトのメソッドを使うと、結果セット内をナビゲーションできます。

- **MoveAfterLast**　最後のローの後に移動します。
- **MoveBeforeFirst**　最初のローの前に移動します。
- **MoveFirst**　最初のローに移動します。
- **MoveLast**　最後のローに移動します。
- **MoveNext**　次のローに移動します。
- **MovePrevious**　前のローに移動します。
- **MoveRelative**　いくつかのローを、現在のローを基準にして相対的に移動します。正のインデックス値はテーブル内を前に移動し、負のインデックス値はテーブル内を後ろに移動し、0 はカーソルを移動しません。ロー・バッファを再移植する場合は、0 が便利です。

例

次のコードは、customer テーブルを開き、そのローをスクロールします。次に、各顧客の姓を示すメッセージ・ボックスを表示します。

```
'MobileVB using VB6
Dim TCustomer as ULTable
Set TCustomer = Conn.GetTable("customer")
TCustomer.Open
While TCustomer.MoveNext
```

```
  MsgBox TCustomer.Column( "lname" ).StringValue
Wend

'Crossfire using vb.net
Dim TCustomer as UltraLiteAFLib.ULTable
Set TCustomer = Conn.GetTable("Customer")
TCustomer.Open
While TCustomer.MoveNext
  MsgBox TCustomer.Column("LName").StringValue
Wend
```

インデックスの指定

テーブル・オブジェクトを開くと、テーブルのローがアプリケーションに公開されます。デフォルトでは、ローはプライマリ・キー値の順に公開されますが、インデックスを指定すると特定の順序でローにアクセスできます。

例

次のコードは、ix_name インデックスで順序付けられた Customer テーブルの最初のローに移動します。

```
'MobileVB using VB6
Set TCustomer = Conn.GetTable("customer")
TCustomer.Open "ix_name"
TCustomer.MoveFirst

'Crossfire using vb.net
TCustomer = Conn.GetTable("customer")
TCustomer.Open "ix_name"
TCustomer.MoveFirst
```

現在のローの値へのアクセス

ULTable オブジェクトは、次のいずれかの位置に常に置かれています。

- テーブルの最初のローの前
- テーブルのいずれかのローの上
- テーブルの最後のローの後ろ

ULTable オブジェクトがローの上に置かれている場合は、Column メソッドを適切なプロパティと一緒に使用して、現在のローのそのカラムの値を取得できます。

例

次のコードは、tCustomer ULTable オブジェクトから 3 つのカラムの値を取り出して、テキスト・ボックスに表示します。

```
Dim colID, colFirstName, colLastName As ULColumn
Set colID = tCustomer.Column("ID")
Set colFirstName = tCustomer.Column("fname")
Set colLastName = tCustomer.Column("lname")
txtID.Text = colID.IntegerValue
txtFirstName.Text = colFirstName.StringValue
txtLastName.Text = colLastName.StringValue
```


ULColumn のプロパティを使用して、値を設定することもできます。

```
colLastName.StringValue = "Kaminski"
```

これらのプロパティへの値の割り当てによって、データベース内のデータの値が変更されることはありません。

位置がテーブルの最初のローの前または最後のローの後ろにある場合でも、プロパティに値を割り当てることができます。しかし、カラムから値を取得することはできません。たとえば、次のコードはエラーを生成します。

```
' This code is incorrect
TCustomer.MoveBeforeFirst
id = TCustomer.Column( "ID" ).IntegerValue
```

バイナリ・データを操作するには、プロパティではなく GetByteChunk メソッドを使用します。

「GetByteChunk メソッド」 75 ページを参照してください。

値のキャスト

選択する ULColumn プロパティは、割り当てる Visual Basic データ型に一致させてください。Ultra Light は自動的に互換性のないデータ型をキャストするため、StringValue メソッドを使用して整数値を文字列変数にフェッチしたりできます。「データ型の明示的な変換」『Ultra Light - データベース管理とリファレンス』を参照してください。

現在のローの値へのアクセスの詳細については、「ULColumn クラス」 73 ページを参照してください。

find と lookup を使用したローの検索

Ultra Light には、データを操作するための操作モードがいくつかあります。これらのモードのうちの 2 つ (検索モードとルックアップ・モード) は、検索に使用されます。ULTable オブジェクトには、テーブル内の特定のローを検索するために、これらのモードに対応するメソッドがあります。

注意
find メソッドや lookup メソッドを使用して検索されるカラムは、テーブルを開くのに使用されたインデックスにあることが必要です。

- **find メソッド** ULTable オブジェクトを開いたときに指定したソート順に基づいて、指定された検索値と正確に一致する最初のローに移動します。

  「FindBegin メソッド」 154 ページを参照してください。

- **lookup メソッド** ULTable オブジェクトを開いたときに指定したソート順に基づいて、指定された検索値と一致するか、それより大きい値の最初のローに移動します。

  「LookupBackward メソッド」 158 ページを参照してください。

♦ **ローを検索するには、次の手順に従います。**

1. 検索モードまたはルックアップ・モードを開始します。

   FindBegin メソッドまたは LookupBegin メソッドを呼び出します。たとえば、次のコードは ULTable.FindBegin を呼び出します。

   ```
   tCustomer.FindBegin
   ```

2. 検索値を設定します。

   検索値は、現在のローの値を設定することで設定します。これらの値を設定すると、バッファに影響しますが、データベースには影響しません。たとえば、次のコードは、バッファの姓のカラムを Kaminski に設定します。

   ```
   tCustomer.Column("lname").StringValue = "Kaminski"
   ```

   マルチカラム・インデックスの場合は、最初のカラムの値が必要ですが、他のカラムは省略できます。

3. ローを検索します。

   適切なメソッドを使用して検索を実行します。たとえば、次の指示は、現在のインデックスで指定された値と正確に一致する最初のローを検索します。

   ```
   tCustomer.FindFirst
   ```

ローの挿入、更新、削除

Ultra Light は、テーブルのローを一度に 1 つずつアプリケーションに公開します。ULTable オブジェクトにはカレント・ポジションがあります。カレント・ポジションは、テーブルのローの上、最初のローの前、または最後のローの後ろになります。

アプリケーションがロケーションを変更すると、Ultra Light はバッファにそのローのコピーを作成します。値を取得または設定する操作はすべて、このバッファにあるデータのコピーにのみ影響します。データベースのデータには影響しません。

例

次の文は、バッファの ID カラムの値を 3 に変更します。

```
colID.IntegerValue = 3
```

Ultra Light のモードの使用

Ultra Light モードは、バッファ内の値の使用目的を指定します。Ultra Light には、デフォルト・モードに加えて、次の 4 つの操作モードがあります。

- **挿入モード**　ULTable.Insert メソッドを呼び出すと、バッファ内のデータが新しいローとしてテーブルに追加されます。
- **更新モード**　ULTable.Update メソッドを呼び出すと、現在のローがバッファ内のデータに置き換えられます。

- **検索モード** ULTable.Find メソッドの 1 つが呼び出されたときに、値がバッファ内のデータに正確に一致するローの検索に使用されます。
- **ルックアップ・モード** いずれかの ULTable.Lookup メソッドが呼び出されたときに、バッファ内のデータと一致するか、それより大きい値のローを検索します。

♦ ローを更新するには、次の手順に従います。

1. 更新するローに移動します。

   テーブルをスクロールするか、find メソッドや lookup メソッドを使用して検索し、ローに移動できます。

2. 更新モードを開始します。

   たとえば、次の指示は、tCustomer テーブル上で更新モードを開始します。

   ```
   tCustomer.UpdateBegin
   ```

3. 更新するローの新しい値を設定します。

   たとえば、次の指示は新しい値を Elizabeth に設定します。

   ```
   ColFirstName.StringValue = "Elizabeth"
   ```

4. Update を実行します。

   ```
   tCustomer.Update
   ```

更新操作が終了すると、直前に更新したローが現在のローになります。ULTable オブジェクトを開いたときに指定したインデックスのカラム値を変更した場合は、現在の位置は不確定です。

デフォルトでは、Ultra Light は AutoCommit モードで動作するため、更新は永続的な記憶領域のローに即時適用されます。AutoCommit モードを無効にした場合は、コミット操作を実行するまで、更新は適用されません。「トランザクションの管理」 26 ページを参照してください。

> **警告**
> ローのプライマリ・キーを更新しないでください。代わりに、ローを削除して新しいローを追加してください。

ローの挿入

ローの挿入手順は、ローの更新手順とほぼ同じです。ただし、挿入操作の場合は、テーブル内の特定のローにあらかじめ指定する必要はありません。ローは、テーブルを開くときに指定されたインデックスに従って自動的に挿入されます。

♦ ローを挿入するには、次の手順に従います。

1. 挿入モードを開始します。

   たとえば、次の指示は、CustomerTable テーブル上で更新モードを開始します。

   ```
   CustomerTable.InsertBegin
   ```


2. 新しいローの値を設定します。

   カラムに値を設定しておらず、そのカラムにデフォルト値が定義されている場合は、そのデフォルト値が使用されます。カラムにデフォルト値がない場合は、NULL が使用されます。カラムが NULL を許可しない場合は、次のデフォルトが使用されます。

   - 数値カラムの場合は 0
   - 文字カラムの場合は空の文字列

   明示的に値を NULL に設定するには、setNull メソッドを使用します。

   ```
   CustomerTable.Column("FName").StringValue = fname
   CustomerTable.Column("LName").StringValue = lname
   ```

3. 挿入を実行します。

   挿入されたローは、Commit を実行したときに永続的にデータベースに保存されます。AutoCommit モードでは、Insert メソッドの一部として Commit が実行されます。

   ```
   CustomerTable.Insert
   ```

ローの削除

挿入モードや更新モードに対応する削除モードはありません。

次の手順は、ローを削除します。

♦ ローを削除するには、次の手順に従います。

1. 削除するローに移動します。
2. 削除を実行します。

   ```
   tCustomer.Delete
   ```

BLOB データの処理

GetByteChunk メソッドを使用して、BINARY または LONG BINARY と宣言された、カラムの BLOB データをフェッチできます。

「GetByteChunk メソッド」 75 ページを参照してください。

例

次のコードは、ULColumn.GetByteChunk メソッドを使用して、BLOB データを取得する方法を説明します。

```
'MobileVB using VB6
Dim table as ULTable
Dim col As ULColumn
Dim data(1 to 1024) As Byte
Dim data_fit As Boolean
Dim size As Long
Set table = Conn.GetTable("image")
table.Open
```



```
size = 1024
Set col = table.Column("img_data")
data_fit = col.GetByteChunk(VarPtr(data(1)), size)
If data_fit Then
  'No truncation
Else
  'data truncated at 1024
End if
table.Close

'Crossfire using vb.net
Dim table as ULTable
Dim col As ULColumn
Dim data(1 to 1024) As Byte
Dim data_fit As Boolean
Dim size As Long
Set table = Conn.GetTable("image")
table.Open
size = 1024
Set col = table.Column("img_data")
' The data argument must be a local variable
data_fit = col.GetByteChunk(data, size)
If data_fit Then
  'No truncation
Else
  'data truncated at 1024
End if
table.Close
```

トランザクションの管理

Ultra Light のトランザクション処理は、データベース内のデータの整合性を保証します。トランザクションは、作業の論理単位です。トランザクション全体が実行されるか、トランザクション内の文がどれも実行されないかのいずれかです。

デフォルトでは、Ultra Light は AutoCommit モードで動作します。AutoCommit モードでは、挿入、更新、削除はそれぞれ独立したトランザクションとして実行されます。操作が完了すると、データベースに変更が加えられます。

ULConnection.AutoCommit プロパティを false に設定すると、複数文のトランザクションを使用できます。たとえば、2 つの口座間で資金を移動するアプリケーションでは、振り込み元の口座からの引き落としと振り込み先口座への振り込みが、1 つのトランザクションを構成します。AutoCommit が false に設定されている場合は、ULConnection.Commit 文を実行してトランザクションを完了し、データベースへの変更を永続的なものにするか、ULConnection.Rollback 文を実行してトランザクションのすべての処理をキャンセルしてください。AutoCommit をオフにすると、パフォーマンスが向上します。

注意
AutoCommit が False に設定されていても、同期はコミットを実行します。

スキーマ情報へのアクセス

ULConnection、ULTable、ULColumn オブジェクトにはそれぞれ、スキーマ・プロパティが含まれます。これらのスキーマ・オブジェクトは、データベース内のテーブル、カラム、インデックス、パブリケーションに関する情報を提供します。

注意
API によるスキーマの変更はできません。スキーマに関する情報の取得のみが可能です。

- **ULDatabaseSchema**　データベース内のテーブルの数と名前、日付と時刻のフォーマットなどのグローバル・プロパティ。

  ULDatabaseSchema オブジェクトを取得するには、ULConnection.Schema プロパティにアクセスします。

- **ULTableSchema**　テーブル内のカラムの数と名前、テーブルの Indexes コレクション。

  ULTableSchema オブジェクトを取得するには、ULTable.Schema プロパティにアクセスします。

- **ULColumnSchema**　デフォルト値、名前、オートインクリメントかどうかなど、個々のカラムに関する情報。

  ULColumnSchema オブジェクトを取得するには、ULColumn.Schema プロパティにアクセスします。

- **ULIndexSchema**　インデックス内のカラムに関する情報。インデックスには直接対応するデータがないので、個別の ULIndex オブジェクトはなく、ULIndexSchema オブジェクトだけが存在します。

  ULIndexSchema オブジェクトは、ULTableSchema.GetIndex メソッドを使用してアクセスできます。

- **ULPublicationSchema**　パブリケーションに含まれるテーブルとカラムの数と名前。パブリケーションもスキーマのみで構成されているため、ULPublication オブジェクトではなく、ULPublicationSchema オブジェクトが存在します。

  ULPublicationSchema オブジェクトは、ULDatabaseSchema.GetPublicationSchema メソッドを使用してアクセスできます。

- **ULResultSetSchema**　結果セットのカラムの数と名前。

  ULResultSetSchema オブジェクトは ULPreparedStatement.ResultSetSchema プロパティを使用してアクセスできます。

エラー処理

通常の操作では、Ultra Light for AppForge はエラーをスローできます。エラーは SQLCODE 値として表現され、負の数字は特定の種類のエラーを示します。

Ultra Light for AppForge によってスローされるエラー・コードのリストについては、「ULSQLCode 列挙」 126 ページを参照してください。

MobileVB または Crossfire の標準エラー処理機能を使用して、エラーを処理できます。Ultra Light オブジェクトがエラーの原因である場合、Err オブジェクトには ULSQLCode 番号が割り当てられます。ULSQLCodes エラーは、特定の種類のエラーを示す負の数字です。ULSQLCode 列挙は、これらの値に関連付けられている一連の説明的な定数を提供します。

「ULSQLCode 列挙」 126 ページを参照してください。

MobileVB 環境で型の補完を使用するには、次のようなエラー処理関数を作成します。

```
'MobileVB using VB6
Public Function GetError() As ULSQLCode
    GetError = Err.Number
End Function
```

その後、GetError 関数を使用して Ultra Light エラーに簡単にアクセスできます。

ユーザの認証

Ultra Light データベースには、ユーザ ID とそれに対応するパスワードを 4 つまで定義できます。Ultra Light データベースは、初期ユーザ ID DBA とパスワード sql を使用して作成されるため、まずこの初期ユーザとして接続します。データベースに接続するとき、アプリケーションは新しいユーザ ID に対する接続権限の付与、既存のユーザ ID のパスワードの変更、または既存のユーザ ID の接続権限の取り消しを行うことができます。

ユーザ ID を直接変更することはできません。その代わり、新しいユーザ ID を追加し、既存のユーザ ID を削除します。

Ultra Light では、ユーザ ID に特定の権限は関連付けられていません。データベースに定義されたすべてのユーザ ID を、その Ultra Light データベースへの接続に使用できます。アプリケーションのコードでは、アプリケーションに提供されたユーザ情報に基づいてさまざまな機能を強制することができます。

接続権限の付与または取り消しの詳細については、「GrantConnectTo メソッド」 84 ページと「RevokeConnectFrom メソッド」 86 ページを参照してください。

♦ **ユーザを追加する、または既存のユーザのパスワードを変更するには、次の手順に従います。**

1. 既存のユーザとしてデータベースに接続します。
2. 特定のユーザに接続権限を付与し、希望のパスワードを設定します。

```
conn.GrantConnectTo("Robert", "newPassword")
```

♦ **既存のユーザを削除するには、次の手順に従います。**

1. 既存のユーザとしてデータベースに接続します。
2. 次のように、特定のユーザの接続権限を取り消します。

```
conn.RevokeConnectFrom("Robert")
```


データの同期

Ultra Light アプリケーションは、統合データベースと同期できます。同期には、Mobile Link サーバと適切なライセンスが必要です。

この項では、同期の概要について簡単に説明し、Ultra Light for AppForge を使用するうえで重要となるいくつかの機能について説明します。同期の詳細については、「Ultra Light クライアント」『Mobile Link - クライアント管理』を参照してください。

また、CustDB サンプル・アプリケーションには、同期の実例もあります。「チュートリアル：AppForge MobileVB のサンプル・アプリケーション」 57 ページを参照してください。

Ultra Light AppForge コンポーネントでは、Palm OS の ECC または FIPS 同期暗号化はサポートされていません。Symbian OS では、同期暗号化は使用できません。

Ultra Light for AppForge がサポートしているのは、TCP/IP、HTTP、HTTPS 同期です。同期は、Ultra Light アプリケーションによって開始されます。いずれの場合でも、ULConnection オブジェクトのメソッドとプロパティを使用して同期を制御します。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」『SQL Anywhere 10 - 紹介』を参照してください。

♦ **TCP/IP または HTTP で同期するには、次の手順に従います。**

1. 同期情報を準備します。

   ULConnection.SyncParms オブジェクトの必須プロパティに値を割り当てます。

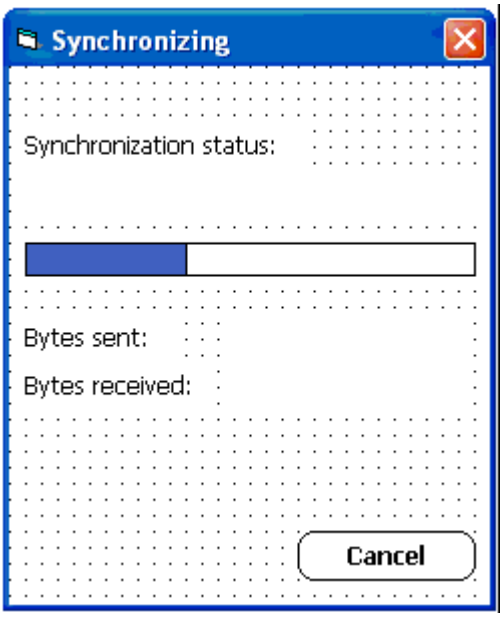
   設定するプロパティと値の詳細については、「Ultra Light クライアント」『Mobile Link - クライアント管理』を参照してください。

2. 同期を実行します。

   ULConnection.Synchronize メソッドを呼び出します。

同期テンプレートの追加

Ultra Light for MobileVB には、同期セッションのステータスをモニタするのに使用できる、テンプレート・フォームがあります。このフォームは、Palm OS 用と Windows CE 用の両方に用意されています。アプリケーションにあるこれらのテンプレートを使用したり、カスタマイズしたり、Ultra Light の同期イベントが動作する様子を学ぶのに、テンプレートを調べたりできます。



このテンプレートをアプリケーションに追加する方法は、MobileVB と Crossfire のどちらを使用しているかによって決まります。

♦ **アプリケーションに同期テンプレートを追加するには、次の手順に従います (MobileVB)。**

1. [プロジェクト] メニューから [Add Form] を選択します。
2. [Ultra Light for MobileVB Sync Form (Windows CE)] または [Ultra Light for MobileVB Sync Form (Palm)] を選択します。
3. [開く] をクリックします。フォームのコピーがアプリケーションに追加されます。

♦ **アプリケーション (Crossfire) に同期テンプレートを追加するには、次の手順に従います。**

1. [プロジェクト] メニューで [新しい項目の追加] を選択します。
2. [ローカル プロジェクト アイテム] – [Ultralite_Crossfire Forms] で、アプリケーションに応じて [UltraLite Crossfire 10 Sync Form for CE]、[UltraLite Crossfire 9 Sync Form for Palm]、または [UltraLite Crossfire 9 Sync Form for PalmHires] を選択します。
3. [開く] をクリックします。フォームのコピーがアプリケーションに追加されます。

同期フォームを使用するコードの記述

InitSyncForm 関数を呼び出し、ULConnection オブジェクトに渡します。これは、各同期の前に行います。

例

次のコードは、Form_Sync という名前の同期ステータス・フォーム、Connection という名前の ULConnection オブジェクトを使用します。

```
Form_Sync.InitSyncForm Connection
Connection.Synchronize
```

アプリケーションが同期するたびに、同期ステータス・フォームが表示されます。同期が進行する様子を、エンド・ユーザは進行状況を示すバーとバイト数で確認できます。同期が完了したら、フォームが閉じます。[キャンセル] ボタンをクリックすると、Ultra Light は現在の同期をキャンセルします。

「チュートリアル：AppForge MobileVB のサンプル・アプリケーション」 57 ページを参照してください。

Ultra Light アプリケーションの配備

アプリケーションが完了したら、またはアプリケーションをテストしたい場合、アプリケーションをデバイスに配備する必要があります。この項では、デバイスに Ultra Light アプリケーションを配備するための手順について説明します。

Windows CE への Ultra Light for MobileVB アプリケーションの配備

Ultra Light アプリケーションを Windows CE に配備するには、次の手順を実行する必要があります。

1. アプリケーションと Ultra Light コンポーネントを配備します。
   a. アプリケーションを設定します。
      - [MobileVB] メニューから、[MobileVB 設定] を選択します。ダイアログが表示されます。
      - 左ウィンドウ枠で、[依存性] を選択し、[ユーザ依存] タブをクリックします。
      - [追加] ボタンをクリックし、*c:¥tutorial¥mvb¥tutCustomer.udb* を選択します。これは、このファイルが配備に組み込まれるように MobileVB に指定します。
      - 左ウィンドウ枠で [Pocket PC 設定] 項目を選択します。
      - [デバイスのインストール・パス] に *¥tutorial¥mvb* と入力します。
      - [OK] をクリックしてダイアログを閉じます。
   b. [MobileVB] − [Deploy to Device] を選択し、[PocketPC] デバイスを選択します。プロジェクトを保存するかどうかを確認するダイアログが表示されたら、[はい] をクリックします。
2. Ultra Light データベースの初期コピーを配備します。

   このため、同期を使用してデータの初期コピーをロードできます。Palm Desktop Organizer Software に付属の Install Tool を使用して、標準的な方法で *.prc* ファイルを配備できます。

   データベース・ファイルは、アプリケーションが特定できるロケーションに配備する必要があります。このロケーションは、DatabaseOnCE 接続パラメータによって定義されます。

   「Ultra Light CE_FILE 接続パラメータ」『Ultra Light - データベース管理とリファレンス』を参照してください。
3. ActiveSync プロバイダを配備します。

   この手順が必要なのは、ActiveSync を使用してアプリケーションを同期する場合のみです。「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」にある Ultra Light の表の「ActiveSync」を参照してください。

参照

- 「Ultra Light クライアント用 ActiveSync と HotSync の配備」 『Mobile Link - クライアント管理』

Palm OS への Ultra Light for MobileVB の配備

Ultra Light アプリケーションを Palm OS デバイスに配備するには、次の手順を実行する必要があります。

- アプリケーションと Ultra Light コンポーネントを配備します。

  1. アプリケーションを設定します。
     - [MobileVB] メニューから、[MobileVB 設定] を選択します。ダイアログが表示されます。
     - 左ウィンドウ枠で、[依存性] を選択し、[ユーザ依存] タブをクリックします。
     - [追加] ボタンをクリックし、*c:¥tutorial¥mvb¥tutCustomer.pdb* を選択します。これは、このファイルが配備に組み込まれるように MobileVB に指定します。
     - 左ウィンドウ枠で [Palm OS 設定] 項目を選択し、アプリケーションの作成者 ID を入力します。有効な HotSync 名を選択します。[OK] をクリックしてダイアログを閉じます。
  2. [MobileVB] − [Deploy to Device] を選択し、[Palm OS] デバイスを選択します。プロジェクトを保存するかどうかを確認するダイアログが表示されたら、[はい] をクリックします。

- 初期 Ultra Light データベースを配備します。

  多くの場合、Ultra Light データベース・ファイルを配備すれば十分です。配備したら、同期を使用してデータをロードできます。

  Palm OS に配備するための *.pdb* ファイルは、ulcreate、ulload、ulinit などの Ultra Light ユーティリティを使用して作成できます。

  データベースは、アプリケーションがロケーションを特定できるように正しい作成者 ID を使用して指定する必要があります。DatabaseOnPalm 接続パラメータは、作成者 ID を使用してデータベースを検索します。

  「Ultra Light PALM_FILE 接続パラメータ」 『Ultra Light - データベース管理とリファレンス』を参照してください。

- HotSync 用 Mobile Link 同期コンジットを配備します。

  この手順が必要なのは、HotSync を使用してアプリケーションを同期する場合のみです。

Ultra Light Palm アプリケーションのステータスの管理

Palm OS デバイスは、一度に 1 つのアプリケーション上でのみ動作します。ただし、一般的には、ユーザが別のアプリケーションに切り替えてから元のアプリケーションに戻ってきた場合、このユーザが他のアプリケーションで作業を行っていた間中、元のアプリケーションがそのまま保留されていたかのように表示します。このような錯覚を実現するには、ユーザが別のアプリケーションに切り替えたときにアプリケーションは内部ステータスを保存する必要があります。アプリケーションは、再起動されたときに内部ステータスを復帰する必要があります。

アプリケーションは結果セットを再度開いてこれらの結果セット内のアプリケーションを再配置する必要があるため、データベース・アプリケーションのステータスを保存して復帰する作業は容易ではありません。Ultra Light には、アプリケーションのステータスを保存して復帰する機能が用意されています。

結果セット上のカーソルのステータスは、自動的に管理されます。テーブル・ベースの API を使用する Mobile VB アプリケーションは、ULTable オブジェクトの Open メソッドの永続的な名前のパラメータに値を指定します。

ステータスの管理方法の知識

Ultra Light は、ステータス情報が保存してあるテーブルの名前とテーブルをそのステータスにリストアするのに十分な情報を保管しています。テーブルと対応する名前は、テーブルの名前であってもなくてもかまいません。これは、「**永続的な名前**」と呼ばれます。

Ultra Light アプリケーションは、1 つのテーブル内の複数のインスタンスを同時に開くことができます。この場合、テーブル名はユニークではありません。たとえば、次のコード (MobileVB を使用) は、同じテーブルを 2 回開きます。

```
' MobileVB
Set table1 = Connection.GetTable( "ULCustomer" )
table1.Open "", "customer1"
' operations here
Set table2 = Connection.GetTable( "ULCustomer" )
table2.Open "", "customer2"
```

テーブルを開く場合、アプリケーションは必要に応じて永続的な名前をパラメータとして指定できます。永続的な名前のステータスが保存されている場合、そのテーブルを開いて適切なローに配置します。対応するテーブルがなければ、そのテーブルを開いて最初のローの前に配置します。

アプリケーションが終了する場合、アプリケーションは、開いているテーブルと接続を明示的に閉じることも閉じないこともあります。開いているテーブルを閉じない場合、Ultra Light は、永続的な名前が提供されていて開いている各テーブルの現在のローを記録します。永続的な名前を持たないテーブルは閉じます。

Connection オブジェクトが ULConnection 型で、ULCustomer という名前のテーブルがデータベースに存在するとします。

```
'MobileVB using VB6
Dim table As ULTable
```



```
Set table = Connection.GetTable( "ULCustomer" )
table.Open "", "customer"
```

コードの 2 番目の行は、ULCustomer テーブルを表すテーブル・オブジェクトを取得します。テーブルは、まだ読み書きするために開かれていません。

Open 呼び出し (コードの 3 行目) では、最初のパラメータは空の文字列で、データがプライマリ・キーによって順序付けられることを示します。2 番目のパラメータは、テーブルに割り当てられている永続的な名前です。このテーブルがまだ開いている間にアプリケーションが終了した場合、ステータス PDB は *customer* を ULCustomer テーブルと対応させ、現在の位置を保存します。

永続的な名前に関する注意

- 永続的な名前が空である場合、Ultra Light は、このテーブルのステータス情報を保存しません。また、このテーブルを開けるときにステータス情報を検索しません。

  テーブルのステータスを保存する必要がない場合、永続的な名前は空にしておきます。そうすると、ステータス・データベースでステータスが検索されません。

- HotSync 同期は、開いているテーブルのステータスに影響しません。ユーザがデバイスの HotSync ボタンを押すと、オペレーティング・システムは、他のアプリケーションが起動するときと同様に、アプリケーションを閉じます。その結果、開いているテーブルのステータスはステータス PDB に記録され、ユーザがアプリケーションに戻ってテーブルが再び開いている場合、ユーザは予期したローに配置されます。そのローが同期の一部として削除された場合、ユーザは次のローに (または、そのローが最後のローならば、最後のローの後に) 配置されます。

- オートコミットをオフにしたアプリケーションは、コミットされていないトランザクションで終了することがあります。Ultra Light は、アプリケーションが再起動したときに失われないように、これらのトランザクションを管理します。

- Ultra Light は、指定した永続的な名前に一致するテーブルをステータス PDB で見つけると、そのテーブルとインデックスが、位置情報が記録されたときに使用されたテーブルとインデックスと同じであるか確認します。同じでない場合、Open の呼び出しは失敗します。

- 永続的な名前の使用は、Palm OS に固有です。Ultra Light for MobileVB アプリケーションを Windows CE 用に作成する場合、永続的な名前は使用しません。Windows CE 上のアプリケーションは、デスクトップ・コンピュータと同様に動作するからです。

例：永続的な名前を使用したステータス情報の管理

PersistentName のプログラム例は、比較的簡単なプログラムで、管理されたステータス情報の使用方法を示します。このプログラムは http://www.sybase.com/detail?id=1022734 から入手できます。その例の重要な部分を次に示します。

```
'MobileVB using VB6
CustomerTable.Open
AddRow "John", "Doe", "Atlanta"
AddRow "Mary", "Smith", "Toronto"
AddRow "Jane", "Anderson", "New York"
AddRow "Margaret", "Billington", "Vancouver"
```

```
AddRow "Fred", "Jones", "London"
AddRow "Jack", "Frost", "Dublin"
AddRow "David", "Reiser", "Berlin"
AddRow "Kathy", "Stevens", "Waterloo"
AddRow "Rebecca", "Gable", "Paris"
AddRow "George", "Jenkins", "Madrid"
CustomerTable.Close
```

このコードは、ULCustomer テーブルに 10 のローを追加します。テーブルの Open を呼び出しますが、この場合、テーブルの位置の管理は不要なので、永続的な名前は提供されていません。このコードはデータを挿入するだけなので、テーブルの位置は関係ありません。

次の行は、ULCustomer テーブルを開き、ローをプライマリ・キーによって順序付け、customer の永続的な名前を割り当てます。

```
CustomerTable.Open "" , "customer"
```

アプリケーションを以前に実行した場合、customer のステータス・データベースで検索が行われます。そうでない場合、customer はこのテーブルと対応します。

customer テーブルは、アプリケーションが動作している間、開いたままです。ユーザが別のアプリケーションに切り替えた場合、Ultra Light によって、ユーザが離れたテーブルの位置が記録されます。アプリケーションが再び起動したときに、テーブルが開かれ、永続的な名前 customer を持つテーブルの位置情報が既知であることが Ultra Light によって判断されるため、ユーザはそのローに位置設定されます。

ユーザが [終了] ボタンをクリックすると、customer テーブルと接続は、アプリケーションが消える前に閉じられます。これは、customer テーブルのステータス情報を捨てるという影響があり、アプリケーションが再起動したとき、ユーザは最初のローに配置されます。

AppForge for Symbian OS についての注意

この項では、Symbian OS での AppForge を使用した Ultra Light 開発に関する注意事項について説明します。

サポートされるプラットフォームとデバイス

Ultra Light for AppForge は、AppForge Crossfire でサポートされているすべての Symbian OS 7 および Symbian OS 8 デバイスをサポートしています。たとえば、Sony Ericsson P800 および P900 シリーズなどの UIQ デバイス、Nokia N-90 などの Nokia S60 デバイス、Nokia 9300 Communicator などの Nokia S-80 デバイスなどがあります。

サポートされているデバイスは、使用している AppForge Crossfire のバージョンによって異なります。たとえば、Nokia 9300 Communicator をサポートするには、Crossfire 5.6.1 以降が必要です。

サンプル・アプリケーション

Crossfire CustDB アプリケーションには、Nokia 9300 Communicator デバイスと Sony Ericsson P800/P900 デバイス用のプロジェクトが両方用意されています。アプリケーションは *samples-dir¥UltraLiteForAppForge¥CF_CustDB¥* にあります。

♦ CustDB サンプル・アプリケーションを使用するには、次の手順に従います。

1. コマンド・プロンプトで、makedbs.bat ファイルを実行して Ultra Light データベースを作成します。
2. Visual Studio .NET 2003 でソリューション・ファイルを開きます。
3. 同期するには、[プログラム] − [SQL Anywhere 10] − [Mobile Link] − [Mobile Link サーバのサンプル] を選択して、Mobile Link サーバを起動します。

AppForge 開発者への Symbian 固有の注意

Symbian OS のファイル・システムでは、Windows に似た命名法が採用されています。dbf 接続パラメータを使用することで、Ultra Light データベースのロケーションを指定できます。

他の AppForge プロジェクトの場合と同様、iAnywhere.UltraLiteForAppForge への参照をプロジェクトに追加する必要があります。

♦ ターゲット・デバイス用のプロジェクトを設定するには、次の手順に従います。

1. 設定するプロジェクトを選択します。
2. [AppForge] − [Crossfire Settings] を選択します。
3. リストから [Dependencies] を選択し、Ultra Light データベース・ファイルを [User Dependencies] リストに追加します。
4. 適切なデバイスの種類をリストから選択し、設定ダイアログを使用してアプリケーション UID などのアプリケーション・プロパティを指定します。

クロスプラットフォーム開発の場合は、AppForge.Platforms.DeviceType 列挙を使用して、アプリケーションを実行するプラットフォームを指定できます。列挙の Symbian OS メンバは、次のとおりです。

定数	ターゲット
SymbianOSCrystal	Nokia 9300/9500
SymbianOSPearl	Nokia S60
SymbianOSQuartz	Sony Ericsson P800/P900/P910 または Motorola A1000

Visual Basic .Net で記述されたプラットフォームに依存しない接続コードの簡単な例を次に示します。

```
deviceType = sysInfo.DeviceType
path = AppForge.MobileVB.Compatibility.Device.AppPath
If deviceType = AppForge.Platforms.DeviceType.SymbianOSCrystal Or _
 deviceType = AppForge.Platforms.DeviceType.SymbianOSPearl Or _
 deviceType = AppForge.Platforms.DeviceType.SymbianOSQuartz Then
   connString = "dbf=" & path & "¥ul_custdb.udb;"
Else
   connString = "dbf=" & path & "¥..¥ul_custdb.udb;"
End If
connString += "con=custdbConn"
Try
   Connection = DBManager.OpenConnection(connString)
Catch ex As Exception
   MsgBox("Error when connecting to database: " & ex.Message)
   End
End Try
```

AppForge プロジェクトのデバイスへの配備

AppForge プロジェクトを Symbian OS デバイスに配備するには、ケーブルまたは Bluetooth 接続を使用してデバイスをコンピュータに接続しておく必要があります。接続に必要なソフトウェア、ドライバ、および手順書は、デバイスに付属しています。

アプリケーションは .sis ファイルとして Symbian OS デバイスに配備されます。デバイスが開発コンピュータに正常に接続されたら、[AppForge] − [Deploy Application To Device] を選択し、デバイスの種類をリストから選択することで、AppForge プロジェクトを Symbian OS デバイスに配備できます。

また、SIS ファイルを開発コンピュータで作成し、別の操作で SIS ファイルを配備するという方法もあります。これを行うには、[AppForge] − [Build Installation File] を選択します。

アプリケーションの同期

TCP/IP ベースまたは HTTP ベースのプロトコルを使用してアプリケーションを同期する場合は、ホスト・アドレスの指定に、SyncStream.StreamParms プロパティで指定したネットワーク・プロトコル・オプションのホスト名ではなく、IP アドレスを使用することをおすすめします。

第 3 章

チュートリアル：AppForge Crossfire のサンプル・アプリケーション

目次

Crossfire 開発チュートリアルの概要

このチュートリアルでは、AppForge Crossfire を使用して、Windows CE または Palm OS 用の Ultra Light アプリケーションを構築する方法について説明します。チュートリアルを終了すると、アプリケーションと小規模なデータベースが Windows CE デバイス上に構築されます。これを、中央の統合データベースと同期します。

テーブル API の詳細については、「Ultra Light for AppForge API リファレンス」 71 ページを参照してください。

所要時間

このチュートリアルは、コードをコピーして貼り付ける場合、約 30 分で終了します。自分でコードを入力する場合、これより長い時間が必要です。このヘルプ情報からコードをコピーして貼り付けると、大なり、小なり、アンパサンドの各特殊文字がコード・ウィンドウに正しく貼り付けられず、手作業での修正が必要になることがあります。

前提条件

このチュートリアルは、コンピュータに AppForge Crossfire がすでにインストールされていることを前提としています。また、Crossfire 開発についての基本的な知識があることも前提としています。

チュートリアルでは、コマンド・ライン・ユーティリティ ulcreate を使用するか、Sybase Central で Ultra Light を使用して、Ultra Light データベースを作成する方法を理解していることも前提としています。「Sybase Central からの Ultra Light データベースの作成」 『Ultra Light - データベース管理とリファレンス』を参照してください。

注意
Crossfire ユーザは、SQL Anywhere がなくても、このチュートリアルの大部分を実行できます。このチュートリアルの同期の項では、SQL Anywhere が必要です。

レッスン 1：プロジェクト・アーキテクチャの作成

最初の手順では、Ultra Light データベースを作成する方法を説明します。

♦ **Ultra Light データベースを作成するには、次の手順に従います。**

1. このチュートリアル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、保存先ディレクトリを *c:¥Tutorial¥crossfire* とします。別の名前のディレクトリを作成した場合は、チュートリアルを通じてそのディレクトリを使用してください。

2. Sybase Central で Ultra Light を使用して、データベースを作成します。

   「Sybase Central からの Ultra Light データベースの作成」『Ultra Light - データベース管理とリファレンス』を参照してください。

   ♦ **テーブル名** Customer

   ♦ **Customer のカラム**

カラム名	データ型 (サイズ)	カラムの NULL 値の許可	デフォルト値
ID	integer	いいえ	オートインクリメント
FName	varchar(15)	いいえ	なし
LName	varchar(20)	いいえ	なし
City	varchar(20)	はい	なし
Phone	varchar(12)	はい	555-1234

   同期しているアプリケーションでは、通常はプライマリ・キーにグローバル・オートインクリメントまたは UUID データ型を選択します。ここでは、チュートリアルを簡単にするためにオートインクリメント・メソッドを選択しています。

   ♦ **プライマリ・キー** 昇順 ID

3. データベースを保存します。

   Windows または Windows CE 用のアプリケーションを開発している場合は、[ファイル] – [保存] を選択し、ファイル名としてチュートリアル・ディレクトリ内の *tutcustomer.udb* を選択します。

Crossfire プロジェクトの作成

次の手順では、アプリケーションの AppForge Crossfire プロジェクトを作成し、Ultra Light コントロールへの参照を追加します。

開発環境の左側にあるツールバーに、標準の Visual Basic ツールに加えて AppForge ツールが表示されます。

♦ **Ultra Light の Crossfire プロジェクトを作成するには、次の手順に従います。**

1. Crossfire を起動します。

   a. [スタート] – [プログラム] – [AppForge] – [Crossfire] を選択します。[Crossfire Project Manager] ダイアログが表示されます。

   b. [新しいプロジェクト] を選択します。[Microsoft Development Environment 新しいプロジェクト] ダイアログが表示されます。

   c. [プロジェクトの種類] ウィンドウで、[Visual Basic プロジェクト] フォルダをクリックして開きます。

   d. [テンプレート] ウィンドウで、[Crossfire Application] をクリックします。

   e. プロジェクト名を CrossfireApp1 のままにし、チュートリアル・ディレクトリ (*c:¥tutorial ¥crossfire*) をロケーションとして入力します。

      [OK] をクリックします。

   f. 使用する配備プラットフォームを選択して [完了] をクリックすると、プロジェクトが作成されます。

   Microsoft Visual Basic .NET Development Environment に [Crossfire] フォームが表示されます。

2. ツールボックスが表示されない場合は、[表示] – [ツールボックス] を選択して開きます。[AppForge] タブを開きます。

3. AppForge コントロールのリストを下にスクロールして [ULConnectionParms] をダブルクリックし、データベース接続オブジェクトをフォームに追加します。

トラブルシューティング

Crossfire プロジェクトに iAnywhere.UltraLiteForAppForge.dll への参照が含まれていない場合、また ULConnectionParms クラスが AppForge コントロールのリストに表示されない場合は、Ultra Light を Crossfire に登録する必要があります。この状況は、SQL Anywhere をインストールしてから Crossfire をインストールした場合などに発生することがあります。

「Crossfire 設計環境への Ultra Light の追加」 7 ページを参照してください。

次の作業

これで、Ultra Light データベースと、フォームに Ultra Light コントロールのある Crossfire プロジェクトが作成されました。次の手順では、アプリケーション・インタフェースを作成します。

レッスン 2：アプリケーション・インタフェースの作成

次の手順は、フォームを使用してユーザ・インタフェースを作成します。この例では、実際のアプリケーションと同様に、ラベルを出力として、テキスト・ボックスとボタンを入力として使用します。

♦ **プロジェクトにコントロールを追加するには、次の手順に従います。**

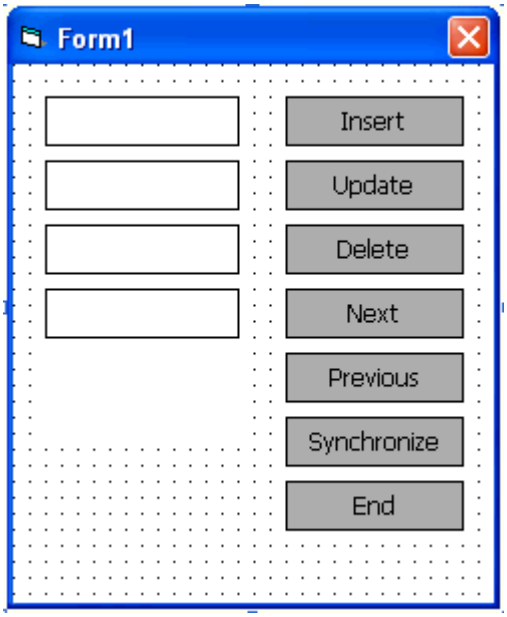
1. AppForge コントロールから、次のコントロールをフォームに追加します。

タイプ	名前	キャプションまたはテキスト
TextBox	txtFName	
TextBox	txtLName	
TextBox	txtcity	
TextBox	txtphone	
Label	lblID	
Button	btnInsert	挿入
Button	btnUpdate	更新
Button	btnDelete	削除
Button	btnNext	次へ
Button	btnPrevious	前へ
Button	btnSync	同期
Button	btnDone	終了

2. アプリケーションを確認します。

   ♦ [ビルド] – [ソリューションのビルド] を選択します。

フォームは次のように表示されます。



レッスン 3：サンプル・コードの作成

このレッスンでは、データベースへの接続、データベース内のナビゲーション、データベース内のデータの操作を行うためのコードの作成について解説します。

このレッスンには、アプリケーションを SQL Anywhere データベースと同期するための手順も含まれています。レッスンのこの部分はオプションで、SQL Anywhere を必要とします。

データベースとの接続のためのコードの作成

このアプリケーションでは、Form_Load イベントの間にデータベースに接続します。一般のモジュールを使用しても、データベースに接続できます。

この例は、*tutcustomer* データベースに接続するのに ULConnectionParms オブジェクトを使用します。これが推奨方法です。この代わりに、次のようにして、接続パラメータを直接指定してデータベース接続を確立することもできます。

```
Connection = DatabaseMgr.OpenConnection("DBF=c:¥tutorial¥crossfire¥tutCustomer.udb")
```

♦ **Ultra Light データベースに接続するためのコードを作成するには、次の手順に従います。**

1. フォームをダブルクリックして、[コード] ウィンドウを開きます。
2. 必要な Ultra Light オブジェクトを宣言します。

   Public NonInheritable Class CrossfireForm1 Inherits System.Windows.Forms.Form 行の直後に次のコードを入力します。

   ```
   Public DatabaseMgr As New UltraLiteAFLib.ULDatabaseManager
   Public Connection As UltraLiteAFLib.ULConnection
   Public CustomerTable As UltraLiteAFLib.ULTable
   ```

3. 接続パラメータを指定します。

   ♦ ULConnectionParm1 コントロールを選択します。[プロパティ] ウィンドウで、このデータベースの接続プロパティを指定します。

   Windows CE デバイスに配備する場合、次のプロパティを指定します。

プロパティ	値
DatabaseOnCE	¥tutorial¥crossfire¥tutCustomer.udb
DatabaseOnDesktop	c:¥tutorial¥crossfire¥tutCustomer.udb

   Palm デバイスに配備する場合、次のプロパティを指定します。

プロパティ	値
DatabaseOnDesktop	c:¥tutorial¥crossfire¥tutCustomer.pdb

プロパティ	値
DatabaseOnPalm	Syb3

4. Form_Load イベント CrossfireForm1_Load に、データベースに接続するためのコードを追加します。

   標準的な接続方法では、接続文字列で指定されるデータベースへの接続を開こうとします。データベースが存在しない場合は、エラー・メッセージを生成します。

```
Try
  Connection = _
    DatabaseMgr.OpenConnection(ULConnectionParms1.ToString())
Catch
  MsgBox(Err.Number)
  MsgBox(Err.Description)
End Try
```

5. 次のコードを、[終了] ボタン (btnDone) の Click イベントに追加します。

```
Connection.Close
End
```

6. アプリケーションを実行します。

   - [デバッグ] – [開始] を選択します。
   - 最初のメッセージ・ボックスの後に、フォームがロードされます。
   - アプリケーションを終了するには、[終了] をクリックします。

トラブルシューティング

これで、データベースに接続する基本コードが完成しました。

接続しようとするとデータ型変換エラーが発生する場合は、ULConnectionParms1 オブジェクトで ToString メソッドを使用していることを確認してください。

データ操作とナビゲーションのためのコードの作成

次の手順は、データの操作とナビゲーションを実装します。コードでは、テーブル API を使用します。テーブル API では、テーブルのローを一度に 1 つずつ移動して変更する方法が提供されます。さらに複雑なアプリケーションでは、Ultra Light によって SQL の実装が提供されます。

♦ テーブルを開くには、次の手順に従います。

1. テーブルを初期化して最初のローに移動するためのコードを記述します。

   このコードは、データベースの Customer テーブルを CustomerTable 変数に割り当てます。Open を呼び出すとテーブルが開き、テーブル・データの読み込みや操作ができます。このコードは、アプリケーションをテーブル内にある最初のローの前に置きます。

   次のコードを、Form1_Load イベントの End Sub 命令の直前に挿入します。


```
Try
  CustomerTable = Connection.GetTable("Customer")
  CustomerTable.Open()
Catch
  If Err.Number <> UltraLiteAFLib.ULSQLCode.ulSQLE_NOERROR _
  Then
    MsgBox(Err.Description)
  End If
End Try
```

2. 新しいプロシージャ DisplayCurrentRow を作成し、次のように実装します。

   テーブルにローが存在しない場合は、次のプロシージャによって、アプリケーションは空のコントロールを表示します。ローが存在する場合は、データベースの現在のローの各カラムに格納された値を表示します。

```
Private Sub DisplayCurrentRow()
  If CustomerTable.RowCount = 0 Then
    txtFname.Text = ""
    txtLname.Text = ""
    txtCity.Text = ""
    txtPhone.Text = ""
    lblID.Caption = ""

  Else
    lblID.Caption =
        CustomerTable.Column("ID").StringValue
    txtFname.Text =
        CustomerTable.Column("FName").StringValue
    txtLname.Text =
        CustomerTable.Column("LName").StringValue
    If CustomerTable.Column ("City").IsNull Then
      txtCity.text =""
    Else
      txtCity.Text =
      CustomerTable.Column("City").StringValue

    End If
    If CustomerTable.Column("Phone").IsNull Then
      txtphone.Text = ""
    Else
      txtphone.Text =
      CustomerTable.Column("Phone").StringValue
    End If
  End If
End Sub
```

3. フォームの Activated イベントから DisplayCurrentRow を呼び出します。この関数を呼び出すと、アプリケーションの起動時にフィールドが更新されます。

```
DisplayCurrentRow
```

♦ テーブルにローを挿入するには、次の手順に従います。

1. [挿入] ボタンを実装するためのコードを記述します。

   次のプロシージャでは、InsertBegin を呼び出すと、アプリケーションが挿入モードになり、ローのすべての値がデフォルトに設定されます。たとえば、ID カラムは、次のオートインクリメント値を受け取ります。カラム値が設定されると、新しいローが挿入されます。

   次のプロシージャを、[挿入] ボタン (btnInsert) の Click イベントに追加します。


```
Dim fname As String
Dim lname As String
Dim city As String
Dim phone As String

fname = txtFname.Text
lname = txtLname.Text
city = txtCity.Text
phone = txtPhone.Text


Try
   CustomerTable.InsertBegin
   CustomerTable.Column("FName").StringValue = _
    fname
   CustomerTable.Column("LName").StringValue = _
     lname
   If Len(city) > 0 Then
      CustomerTable.Column("City").StringValue = _
    city

   End If
   If Len(phone) > 0 Then
     CustomerTable.Column("Phone").StringValue = _
     phone
   End If
   CustomerTable.Insert
   CustomerTable.MoveLast
   DisplayCurrentRow
   Exit Sub
Catch
   MsgBox "Error:  " & CStr(Err.Description)
End Try
```

2. アプリケーションを実行します。

   最初のメッセージ・ボックスの後にフォームが表示されます。

3. 2 つのローをデータベースに挿入します。

   - 先頭のテキスト・ボックスに名前 Jane を入力し、2 つめのテキスト・ボックスに姓 Doe を入力します。[挿入] をクリックします。

     テーブルに、これらの値を持つローが追加されます。アプリケーションはテーブルの最後のローに移動し、そのローを表示します。ラベルには、Ultra Light がローに割り当てた ID カラムの自動的にインクリメントされた値が表示されます。

   - 先頭のテキスト・ボックスに名前 John を入力し、2 つめのテキスト・ボックスに姓 Smith を入力します。[挿入] をクリックします。

4. [終了] をクリックしてプログラムを終了します。

♦ テーブルのローを移動するには、次の手順に従います。

1. [次へ] と [前へ] の各ボタンを実装するためのコードを記述します。

   次のコードを、[次へ] ボタン (btnNext) の Click イベントに追加します。


```
If Not CustomerTable.MoveNext Then
   CustomerTable.MoveLast
End If
DisplayCurrentRow
```

次のコードを、[前へ] ボタン (btnPrevious) の Click イベントに追加します。

```
If Not CustomerTable.MovePrevious Then
   CustomerTable.MoveFirst
End If
DisplayCurrentRow
```

2. アプリケーションを実行します。

   最初にフォームが表示されると、現在位置が最初のローの前にあるため、コントロールは空です。

   フォームが表示されたら、[Next] と [Prev] をクリックして、テーブルのローの間を移動します。

この段階で、データの入力とテーブルのローのスクロールを行うことができます。

♦ テーブルのローを更新、削除するには、次の手順に従います。

1. [更新] ボタンを実装するためのコードを記述します。

   下のコードでは、UpdateBegin を呼び出すと、アプリケーションが更新モードに設定されます。Update が呼び出されると、カラム値が更新された後にロー自体が更新されます。

   次のコードを、[更新] ボタン (btnUpdate) の Click イベントに追加します。

```
Dim fname As String
Dim lname As String
Dim city As String
Dim phone As String

fname = txtFname.Text
lname = txtLname.Text
city = txtCity.Text
phone = txtPhone.Text

Try
   CustomerTable.UpdateBegin
   CustomerTable.Column("FName").StringValue = fname
   CustomerTable.Column("LName").StringValue = lname
   If Len(city) > 0 Then
      CustomerTable.Column("City").StringValue = city
   Else
      CustomerTable.Column("City").SetNull
   End If

   If Len(phone) > 0 Then
     CustomerTable.Column("Phone").StringValue = phone
   End If
   CustomerTable.Update
   DisplayCurrentRow
   Exit Sub
Catch
```



```
    MsgBox "Error:  " & CStr(Err.Description)
End Try
```

2. [削除] ボタンを実装するためのコードを記述します。

   次のコードでは、Delete を呼び出すことで現在の行が削除されます (アプリケーションは現在の位置のロー・データを表示します)。

   次のコードを、[削除] ボタン (btnDelete) の Click イベントに追加します。

```
If CustomerTable.RowCount = 0 Then
    Exit Sub
End If
CustomerTable.Delete
CustomerTable.MoveRelative 0
DisplayCurrentRow
```

3. アプリケーションを実行します。

同期のためのコードの記述

次の手順は、同期を実装します。同期には、SQL Anywhere が必要です。

♦ **[同期] ボタンのコードを記述するには、次の手順に従います。**

· [同期] ボタンを実装するためのコードを記述します。

  下のコードでは、ULSyncParms オブジェクトに同期パラメータが含まれています。たとえば、ULSyncParms.UserName プロパティは、Mobile Link が起動したら新しいユーザを追加することを指定します。

  次のコードを、[同期] ボタン (btnSync) の Click イベントに追加します。

```
With Connection.SyncParms
    .UserName = "CrossfireSample"
    .Stream = UltraLiteAFLib.ULStreamType.ulTCPIP
    .Version = "ul_default"
End With
Connection.Synchronize
DisplayCurrentRow
```

アプリケーションの同期

SQL Anywhere サンプル・データベースの Customers テーブルは、作成した Ultra Light データベースの **Customer** テーブルとカラムが一致しています。次の手順では、データベースを SQL Anywhere サンプル・データベースと同期します。

♦ **アプリケーションを同期するには、次の手順に従います。**

1. コマンド・プロンプトから、次のコマンドを実行して、Mobile Link サーバを起動します。

```
mlsrv10 -c "dsn=SQL Anywhere 10 Demo" -v+ -zu+
```

-zu+ コマンド・ライン・オプションによって、ユーザの追加と同期スクリプトの生成が自動的に行われます。これらのオプションの詳細については、「Mobile Link サーバのオプション」『Mobile Link - サーバ管理』を参照してください。

Mobile Link サーバが起動され、サーバの状態を示すウィンドウが表示されることを確認します。

2. Ultra Light Crossfire アプリケーションを起動します。
3. [削除] を繰り返しクリックして、テーブルのすべてのローを削除します。テーブルに残っているローがあると、SQL Anywhere サンプル・データベースの Customers テーブルに、ローがアップロードされます。
4. アプリケーションを同期します。

   [同期] をクリックします。

   Mobile Link サーバのウィンドウでは、同期の進行状況が表示されます。
5. 同期が完了したら、[次へ] および [前へ] ボタンをクリックしてテーブルのロー間を移動し、SQL Anywhere サンプル・データベースから取得されたデータを表示します。

レッスン 4：デバイスへの配備

次の手順は、アプリケーションを Palm OS または Windows CE OS ベースのデバイスに配備します。

♦ **Windows CE OS ベースのデバイスに配備するには、次の手順に従います。**

1. アプリケーションを設定します。
   - ♦ [AppForge] メニューから、[Crossfire Settings] を選択します。ダイアログが表示されます。
   - ♦ 左ウィンドウ枠で、[Dependencies] を選択し、[User Dependencies] タブをクリックします。
   - ♦ [追加] ボタンをクリックし、*c:¥tutorial¥crossfire¥tutCustomer.udb* を選択します。これは、このデータベース・ファイルが配備に組み込まれるよう Crossfire に指定します。
   - ♦ 左ウィンドウ枠で [Windows Mobile Settings] 項目を選択し、右ウィンドウ枠で [Packaging] タブを選択します。
   - ♦ [Custom Device Installation Path] に *%CE1%¥tutorial¥crossfire* と入力します。
   - ♦ [OK] をクリックしてダイアログを閉じます。
2. [AppForge] − [Deploy Application to Device] を選択し、[Windows Mobile-based Pocket PC] を選択します。プロジェクトを保存するかどうかを確認するダイアログが表示されたら、[はい] を選択します。
3. デバイスで、[スタート] − [プログラム] をクリックします。
4. [UltraLiteTutorial] をクリックして、アプリケーションを実行します。

♦ **Palm デバイスに配備するには、次の手順に従います。**

1. アプリケーションを設定します。
   - ♦ [AppForge] メニューから、[Crossfire Settings] を選択します。
   - ♦ 表示されるダイアログの左ウィンドウ枠で [Dependencies] を選択し、[User Dependencies] タブをクリックします。
   - ♦ [追加] ボタンをクリックし、*c:¥tutorial¥mvb¥tutCustomer.pdb* を選択します。これは、このファイルが配備に組み込まれるように Crossfire に指定します。
   - ♦ 左ウィンドウ枠で [Palm OS Settings] 項目を選択し、作成者 ID として Syb3 と入力します。有効な HotSync 名を選択します。
   - ♦ [OK] をクリックしてダイアログを閉じます。
2. [AppForge] − [Deploy Application to Device] を選択し、[Palm OS] デバイスを選択します。プロジェクトを保存するかどうかを確認するダイアログが表示されたら、[はい] を選択します。

3. デバイスに対して HotSync を実行し、アプリケーションがデバイスに送信されることを確認します。HotSync 処理の完了後に、アプリケーション・ファイルはデバイスに抽出されます。
4. デバイスで [ホーム] をクリックし、[UltraLightTutorial] を選択してアプリケーションを実行します。

まとめ

学習の成果

このチュートリアルでは、以下の作業を行いました。

- テーブルが 1 つ定義されているデータベース・ファイルの作成
- Crossfire 用の Ultra Light アプリケーションの作成
- Ultra Light リモート・データベースと SQL Anywhere 統合データベースの同期

その他のサンプル

この他のサンプル・アプリケーションおよびユーティリティは、iAnywhere CodeXchange から入手できます。

第 4 章

チュートリアル：AppForge MobileVB のサンプル・アプリケーション

目次

MobileVB 開発チュートリアルの概要

このチュートリアルでは、Ultra Light テーブル API を使用して Ultra Light for MobileVB アプリケーションを構築するプロセスを解説します。チュートリアルを終了すると、アプリケーションと小規模なデータベースが Windows CE デバイス上に構築されます。これを、中央のデータベースと同期します。アプリケーションのターゲット・プラットフォームは Windows CE または Palm OS です。

テーブル API の詳細については、「Ultra Light for AppForge API リファレンス」 71 ページを参照してください。

所要時間

このチュートリアルは、コードをコピーして貼り付ける場合、約 30 分で終了します。自分でコードを入力する場合、これより長い時間が必要です。

このヘルプ・ファイルからコードをそのままコピーする場合は、一部の文字が正しくコピーされない可能性があることに注意してください。アンパサンド、小なり、大なりの各記号は html マークアップ・コードとしてコピーされる場合があり、Visual Basic のコード編集ウィンドウで手作業での修正が必要になることがあります。

前提知識と経験

このチュートリアルは、次のことを前提にしています。

- MobileVB と Microsoft Visual Basic 6 がコンピュータにインストールされている
- MobileVB アプリケーションの作成、テスト、トラブルシューティングができる
- Sybase Central の Ultra Light プラグイン、または ulcreate ユーティリティを使用して、Ultra Light データベースを作成する方法を知っている
- Crossfire Client がターゲット・デバイスにインストールされている

  Crossfire Client の詳細については、AppForge の Web サイトを参照してください。

> **注意**
> SQL Anywhere がなくても、このチュートリアルの大部分を実行できます。このチュートリアルの同期の項では、SQL Anywhere が必要です。

目的

このチュートリアルの目的は、Ultra Light アプリケーションの開発プロセスについて、知識と経験を得ることです。

レッスン 1：プロジェクト・アーキテクチャの作成

最初のレッスンでは、プロジェクトにおけるファイルのロケーションと、プロジェクトで使用する Ultra Light データベースの仕様を確定します。

♦ **Ultra Light データベースを作成するには、次の手順に従います。**

1. このチュートリアル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、保存先ディレクトリを *c:¥Tutorial¥mvb* とします。名前またはロケーションが異なるディレクトリを作成した場合は、チュートリアルを通じて、*c:¥Tutorial¥mvb* の代わりにそのディレクトリを使用してください。

2. Sybase Central で Ultra Light を使用して、データベースを作成します。

   「Sybase Central からの Ultra Light データベースの作成」 『Ultra Light - データベース管理とリファレンス』を参照してください。

   ♦ **データベース・ファイル名** *c:¥Tutorial¥mvb¥tutcustomer.udb*

   ♦ **テーブル名** ULCustomer

   ♦ **ULCustomer テーブルのカラム**

カラム名	データ型 (サイズ)	カラムの null 値の許可	カラムのユニークな値	デフォルト値
cust_id	integer	いいえ	なし	オートインクリメント
cust_name	varchar (30)	いいえ	いいえ	なし

   ♦ **ULCustomer テーブルのプライマリ・キー** 昇順 cust_id

MobileVB プロジェクトの作成

次の手順では、アプリケーションの MobileVB プロジェクトを作成し、Ultra Light for MobileVB コントロールへの参照を追加します。

Visual Basic 環境の左側にある Visual Basic ツールバーに、標準の Visual Basic ツールに加えて MobileVB ツールが表示されます。Ultra Light 接続パラメータ・コントロールがない場合は、「MobileVB 設計環境への Ultra Light の追加」 6 ページを参照してください。

♦ **Ultra Light for MobileVB コントロールへの参照を追加するには、次の手順に従います。**

1. MobileVB を起動します。

   ♦ [スタート] – [プログラム] – [AppForge MobileVB] – [Start MobileVB] を選択します。

   MobileVB プロジェクト・マネージャが表示されます。

2. 新規プロジェクトを作成します。

   [新規プロジェクト] をクリックします。設計ターゲットを求められたら、適切なターゲットを選択します。このチュートリアルでは、Windows CE OS ベースのデバイスの場合の手順を説明します。

3. Ultra Light for MobileVB への参照を作成します。

   - [プロジェクト] – [参照] を選択します。
   - [iAnywhere Solutions, UltraLite Connection Parameters 10.0] を選択し、[OK] をクリックします。

4. プロジェクトに名前を付けます。

   - [プロジェクト] – [MobileVBProject1 プロパティ] をクリックします。
   - [プロジェクト名] に **UltraLiteTutorial** と入力します。これは、アプリケーションがデバイスに表示されるときの名前になります。
   - [OK] をクリックします。

5. プロジェクトを保存します。

   - [ファイル] – [プロジェクトの保存] を選択します。
   - フォーム・ファイルを *c:¥tutorial¥mvb¥Tutorial.frm* として保存します。
   - プロジェクトを *c:¥tutorial¥mvb¥Tutorial.vbp* として保存します。

レッスン 2：フォームの作成

「レッスン 1：プロジェクト・アーキテクチャの作成」 59 ページの手順を完了すると、プロジェクトにフォームが 1 つ表示されます。

次の手順は、フォームを使用してユーザ・インタフェースを作成します。この例では、実際のアプリケーションと同様に、ラベルを出力として、テキスト・ボックスとボタンを入力として使用します。

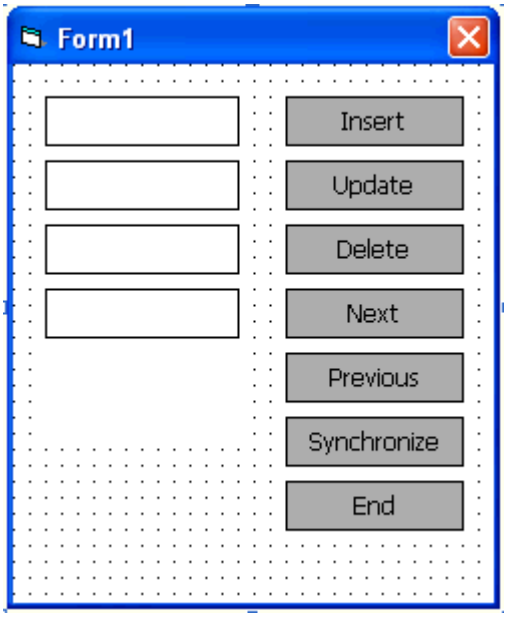
♦ **プロジェクトにコントロールを追加するには、次の手順に従います。**

1. 次の表のコントロールとプロパティをフォームに追加します。

タイプ	名前	キャプションまたはテキスト
AFTextBox	txtname	
AFLabel	lblID	
AFButton	btnInsert	挿入
AFButton	btnUpdate	更新
AFButton	btnDelete	削除
AFButton	btnNext	次へ
AFButton	btnPrevious	前へ
AFButton	btnSync	同期
AFButton	btnDone	終了

2. アプリケーションを確認します。
   - [MobileVB] – [コンパイルと検証] を選択します。

フォームは、次の図のようになります。



レッスン 3：サンプル・コードの作成

この章では、データベースに接続し、データベース内をナビゲーションし、データベース内のデータを操作するための、Visual Basic コードを作成するプロセスを解説します。

このレッスンには、アプリケーションを SQL Anywhere データベースと同期するための手順も含まれています。レッスンの同期部分はオプションで、SQL Anywhere を必要とします。

データベースとの接続のためのコードの作成

このアプリケーションでは、Form_Load イベントの間にデータベースに接続します。一般のモジュールを使用しても、データベースに接続できます。

この例は、データベースに接続するのに ULConnectionParms オブジェクトを使用します。また、アプリケーション・コードで接続文字列を使用する方法もあります。

「ULConnectionParms クラス」 90 ページを参照してください。

♦ **Ultra Light データベースに接続するためのコードを作成するには、次の手順に従います。**

1. フォームをダブルクリックして、[コード] ウィンドウを開きます。
2. 必要な Ultra Light オブジェクトを宣言します。

   フォームの宣言領域に、次のコードを入力します。

```
Public DatabaseMgr As New ULDatabaseManager
Public Connection As ULConnection
Public CustomerTable As ULTable
```

3. 接続パラメータを指定します。

   - ULConnectionParms1 という名前のフォームに ULConnectionParms オブジェクトを追加します。ULConnectionParms コントロールは、ツールボックスにあるデータベース・アイコンです。
   - [プロパティ] ウィンドウで、ULConnectionParms のプロパティを指定します。

     Windows CE デバイスに配備する場合、次のプロパティを指定します。

プロパティ	値
DatabaseOnCE	¥tutorial¥mvb¥tutcustomer.udb
DatabaseOnDesktop	c:¥tutorial¥mvb¥tutcustomer.udb

4. データベースに接続するためにのコードを Form_Load イベントに追加します。

   データベース・マネージャは、ULConnectionParms1 オブジェクトで指定されるデータベースへの接続を開きます。


```
Private Sub Form_Load()
' enable error handling
On Error Resume Next

  Set Connection = DatabaseMgr.OpenConnection(ULConnectionParms1.ToString())

  If Err.Number = ULSQLCode.ulSQLE_NOERROR Then
    MsgBox "Connected to an existing database"
  Else
    MsgBox Err.Description
    Exit Sub
  End If
End Sub
```

接続コードが動作したら、接続が確立したことを通知する Msgbox を発行する行は削除してかまいません。

ULConnectionParms オブジェクトではなく接続文字列を使用する場合は、次の構文を使用するように上記のコードを変更します。

```
Set Connection = DatabaseMgr.OpenConnection _
  ("dbf=C:¥tutorial¥mvb¥tutcustomer.udb;" & _
   "ce_file=¥tutorial¥mvb¥tutcustomer.udb")
```

ここでは、使用する可能性があるターゲット・プラットフォーム用のデータベース・ファイル名指定が含まれています (デスクトップ環境用には dbf=、Windows CE デバイス環境用には ce_file=)。

5. [終了] ボタンをクリックされたときに、アプリケーションを終了して接続を閉じるコードを追加します。

```
Sub btnDone_Click()
   Connection.Close
   End
End Sub
```

6. アプリケーションを実行します。

   - [実行] – [実行] を選択します。
   - 最初のメッセージ・ボックスの後に、フォームがロードされます。
   - アプリケーションを終了するには、[終了] ボタンをクリックします。

データ操作とナビゲーションのためのコードの作成

次の手順は、データの操作とナビゲーションを実装します。

- **テーブルを開くには、次の手順に従います。**

1. テーブルを初期化して最初のローに移動するためのコードを記述します。

このコードは、データベースの customer テーブルを CustomerTable 変数に割り当てます。Open を呼び出すとテーブルが開き、テーブル・データの読み込みや操作ができます。このコードは、アプリケーションをテーブル内にある最初のローの前に置きます。

次のコードを、Form_Load イベントの End Sub 命令の直前に挿入します。

```
Set CustomerTable = Connection.GetTable("ULCustomer")
CustomerTable.Open
If Err.Number <> ULSQLCode.ulSQLE_NOERROR Then
   MsgBox Err.Description
End If
CustomerTable.MoveFirst
```

2. 新しいプロシージャ DisplayCurrentRow を作成し、次のように実装します。

   テーブルにローが存在しない場合は、次のプロシージャによって、アプリケーションは空のコントロールを表示します。ローが存在する場合は、データベースの現在のローの各カラムに格納された値を表示します。

```
Private Sub DisplayCurrentRow()
  If CustomerTable.RowCount = 0 Then
    txtname.Text = ""
    lblID.Caption = ""
  Else
    txtname.Text = CustomerTable.Column("cust_name").StringValue
    lblID.Caption = CustomerTable.Column("cust_id").StringValue
  End If
End Sub
```

3. Form_Activate オブジェクトから DisplayCurrentRow を呼び出します。この関数を呼び出すと、アプリケーションの起動時にフィールドが更新されます。

```
Private Sub Form_Activate()
   DisplayCurrentRow
End Sub
```

♦ テーブルにローを挿入するには、次の手順に従います。

1. [挿入] ボタンを実装するためのコードを記述します。

   次のプロシージャでは、InsertBegin を呼び出すと、アプリケーションが挿入モードになり、ローのすべての値がデフォルトに設定されます。たとえば、ID カラムは、次のオートインクリメント値を受け取ります。カラム値が設定されると、新しいローが挿入されます。

   次のプロシージャをフォームに追加します。

```
Private Sub btnInsert_Click()

   On Error GoTo InsertError
   CustomerTable.InsertBegin
   CustomerTable.Column("cust_name").StringValue = txtname.Text

   CustomerTable.Insert
   CustomerTable.MoveLast
   DisplayCurrentRow
```



```
    Exit Sub

InsertError:
    MsgBox "Error:  " & CStr(Err.Description)
End Sub
```

2. アプリケーションを実行します。

   最初のメッセージ・ボックスの後にフォームが表示されます。

3. 2 つのローをデータベースに挿入します。

   - 先頭のテキスト・ボックスに名前 Jane を入力し、2 つめのテキスト・ボックスに姓 Doe を入力します。[挿入] をクリックします。

     テーブルに、これらの値を持つローが追加されます。アプリケーションはテーブルの最後のローに移動し、そのローを表示します。ラベルには、Ultra Light がローに割り当てた ID カラムの自動的にインクリメントされた値が表示されます。

   - 先頭のテキスト・ボックスに名前 John を入力し、2 つめのテキスト・ボックスに姓 Smith を入力します。[挿入] をクリックします。

4. [終了] をクリックしてプログラムを終了します。

♦ テーブルのローを移動するには、次の手順に従います。

1. [次へ] と [前へ] の各ボタンを実装するためのコードを記述します。

   次のプロシージャをフォームに追加します。

```
Private Sub btnNext_Click()
    If Not CustomerTable.MoveNext Then
        CustomerTable.MoveLast
    End If
    DisplayCurrentRow
End Sub


Private Sub btnPrevious_Click()
    If Not CustomerTable.MovePrevious Then
        CustomerTable.MoveFirst
    End If
    DisplayCurrentRow
End Sub
```

2. アプリケーションを実行します。

   最初にフォームが表示されると、現在位置が最初のローの前にあるため、コントロールは空です。

   フォームが表示されたら、[Next] と [Prev] をクリックして、テーブルのローの間を移動します。

♦ テーブルのローを更新、削除するには、次の手順に従います。

1. [更新] ボタンを実装するためのコードを記述します。

下のコードでは、UpdateBegin を呼び出すと、アプリケーションが更新モードに設定されます。Update が呼び出されると、カラム値が更新された後にロー自体が更新されます。

次のプロシージャをフォームに追加します。

```
Private Sub btnUpdate_Click()

   On Error GoTo UpdateError

   CustomerTable.UpdateBegin
   CustomerTable.Column("cust_name").StringValue = txtname.Text
   CustomerTable.Update
   DisplayCurrentRow
   Exit Sub

UpdateError:
   MsgBox "Error:  " & CStr(Err.Description)
End Sub
```

2. [削除] ボタンを実装するためのコードを記述します。

   下のコードでは、Delete を呼び出すと、アプリケーションの現在のローが削除されます。

   次のプロシージャをフォームに追加します。

```
Private Sub btnDelete_Click()
   If CustomerTable.RowCount = 0 Then
      Exit Sub
   End If
   CustomerTable.Delete
   CustomerTable.MoveRelative 0
   txtname.Text = ""
   lblID.Caption = ""
   DisplayCurrentRow
End Sub
```

3. アプリケーションを実行します。

同期のためのコードの記述

次の手順は、同期を実装します。同期には、SQL Anywhere が必要です。チュートリアルのこの部分はオプションです。

♦ [同期] ボタンのコードを記述するには、次の手順に従います。

· [同期] ボタンを実装するためのコードを記述します。

  下のコードでは、ULSyncParms オブジェクトに同期パラメータが含まれています。たとえば、UserName プロパティは、Mobile Link が起動時に指定されたユーザ名から Mobile Link スクリプトの適切なセットを特定して、同期プロセスに使用するよう指定します。このプログラムは簡単なデモ用のアプリケーションなので、ここでは DownloadOnly プロパティを true に設定して、Ultra Light データベースからデータがアップロードされないようにします。

  同期パラメータの詳細については、「Ultra Light 同期パラメータとネットワーク・プロトコル・オプション」『Mobile Link - クライアント管理』を参照してください。

次のプロシージャをフォームに追加します。

```
Private Sub btnSync_Click()
   With Connection.SyncParms
      .UserName = "50"
      .Stream = ULStreamType.ulTCPIP
      .Version = "custdb 10.0"
      .DownloadOnly = True
   End With
   Connection.Synchronize
   CustomerTable.MoveFirst
   DisplayCurrentRow
End Sub
```

アプリケーションの同期

SQL Anywhere に付属の SQL Anywhere 10 CustDB データベース・サンプルには Customer テーブルがあり、作成した Ultra Light データベースの **ULCustomer** テーブルとカラムが一致しています。次の手順では、データベースを SQL Anywhere 10 CustDB データベースと同期します。

♦ アプリケーションを同期するには、次の手順に従います。

1. コマンド・プロンプトから、次のコマンドを実行して、Mobile Link サーバを起動します。

   ```
   mlsrv10 -c "dsn=SQL Anywhere 10 CustDB" -v+ -zu+
   ```

   -v+ オプションを指定すると、冗長ロギングがオンになります (+ は "すべて表示" の意味)。アプリケーションのデバッグ時には、冗長ロギングを使用することをおすすめします。-zu + オプションを指定すると、ユーザの自動追加が行われます。「Mobile Link サーバのオプション」『Mobile Link - サーバ管理』を参照してください。

2. Ultra Light for MobileVB アプリケーションを起動します。

3. アプリケーションを同期します。

   [同期] をクリックします。

   Mobile Link サーバのウィンドウでは、同期の進行状況が表示されます。

4. 同期が完了したら、[次へ] および [前へ] をクリックしてテーブルのロー間を移動し、新しい情報が SQL Anywhere 10 CustDB データベースからダウンロードされたことを確認します。

レッスン 4：デバイスへの配備

次の手順では、アプリケーションを Windows CE OS ベースのデバイスに配備する方法を説明します。

♦ **Windows CE OS ベースのデバイスに配備するには、次の手順に従います。**

1. アプリケーションを設定します。
   - [AppForge] メニューから、[MobileVB Settings] を選択します。

     ダイアログが表示されます。
   - 左ウィンドウ枠で、[Dependencies] を選択し、[User Dependencies] タブをクリックします。
   - [追加] ボタンをクリックし、*c:¥tutorial¥mvb¥tutcustomer.udb* を選択します。これは、このファイルが配備に組み込まれるように MobileVB に指定します。
   - 左ウィンドウ枠で [Windows Mobile Settings] 項目を選択します。
   - [Device Installation Path] の [Custom path] に *¥tutorial¥mvb* と入力します。
   - [OK] をクリックしてダイアログを閉じます。
2. [AppForge] − [Deploy Application to Device] を選択し、[Windows Mobile-based Pocket PC] デバイスを選択します。プロジェクトを保存するかどうかを確認するダイアログが表示されたら、[はい] を選択します。
3. デバイスで、[スタート] − [プログラム] をクリックします。
4. [ULTutorial] をクリックして、アプリケーションを実行します。

まとめ

学習の成果

このチュートリアルでは、以下の作業を行いました。

- Ultra Light データベースの作成
- Ultra Light for MobileVB アプリケーションの作成
- Ultra Light リモート・データベースと SQL Anywhere 統合データベースの同期
- Ultra Light for MobileVB アプリケーションの開発プロセスに関する知識の取得

その他のサンプル

この他のサンプル・アプリケーションおよびユーティリティは、iAnywhere CodeXchange から入手できます。

第 5 章

Ultra Light for AppForge API リファレンス

目次

ULAuthStatusCode 列挙

ULAuthStatusCode は、ULSyncResult オブジェクトで使用される auth_status 同期パラメータです。

定数	値
ulAuthStatusUnknown	0
ulAuthStatusValid	1000
ulAuthStatusValidButExpiresSoon	2000
ulAuthStatusExpired	3000
ulAuthStatusInvalid	4000
ulAuthStatusInUse	5000

ULColumn クラス

ULColumn オブジェクトでは、データベース内のテーブルの値を取得、設定できます。各カラム・オブジェクトは、テーブル内の特定の値を表します。ローは、ULTable オブジェクトによって決定されます。

基本となるカラムが NULL の場合に、Get メソッドはエラーをスローします。アプリケーションはプロパティまたはメソッドの NULL 値をチェックしてから、取得を試みます。

Ultra Light データベースの型を Visual Basic の型に変換する場合の注意
Ultra Light では、データベースのカラムのデータ型を Visual Basic のデータ型に変換しようとします。変換が成功しなかった場合は、ULSQLE_CONVERSION_ERROR が発生します。

テーブル・オブジェクトの詳細については、「ULTable クラス」 152 ページを参照してください。

プロパティ

構文	説明
BooleanValue As Boolean	現在のローのこのカラムの値を Boolean として取得または設定する。
ByteValue As Byte	現在のローのこのカラムの値を Byte として取得または設定する。
DatetimeValue As Date	現在のローのこのカラムの値を Date として取得または設定する。
DoubleValue As Double	現在のローのこのカラムの値を Double として取得または設定する。
IntegerValue As Integer	現在のローのこのカラムの値を Integer として取得または設定する。
IsNull As Boolean (読み込み専用)	カラム値が NULL かどうかを示す。
LongValue As Long	現在のローのこのカラムの値を Long として取得または設定する。
RealValue As Single	現在のローのこのカラムの値を Single として取得または設定する。
Schema As ULColumnSchema (読み込み専用)	カラムのスキーマを表すオブジェクトを取得する。
StringValue As String	現在のローのこのカラムの値を String として取得または設定する。

AppendByteChunk メソッド

カラムの型が ulTypeLongBinary または TypeBinary の場合、バイトをローのカラムに追加します。

構文

AppendByteChunk(_
data As Long, _
data_len As Long _
)
Member of **UltraLiteAFLib.ULColumn**

パラメータ

data　MobileVB では、バイトの配列へのポインタ。バイトの配列へのポインタを取得するには、Visual Basic VarPtr() 関数を使用します。Crossfire では、バイトの配列であるローカル変数です。

data_len　追加する配列からのバイトの数。

エラー・セット

ulSQLE_INVALID_PARAMETER　データ長が 0 より小さい場合、エラーが発生します。

ulSQLE_CONVERSION_ERROR　カラムのデータ型が LONG BINARY でない場合、エラーが発生します。

例

次の例では、データが edata カラムに追加されます。

```
'MobileVB using VB6
Dim data (1 to 512) As Byte
' ...
table.Column("edata").AppendByteChunk( _
            VarPtr(data(1)), 232)

'Crossfire using vb.net
Dim data (1 to 512) As Byte
' ...
table.Column("edata").AppendByteChunk(data, 232)
```

AppendStringChunk メソッド

カラムの型が TypeLongString または TypeString の場合、カラムに文字列を追加します。

構文

AppendStringChunk(*chunk* As String **)**
Member of **UltraLiteAFLib.ULColumn**

パラメータ

data　テーブル内の既存の文字列に追加する文字列。

エラー・セット

ulSQLE_CONVERSION_ERROR　カラムのデータ型が CHAR でも LONG VARCHAR でもない場合、エラーが発生します。

GetByteChunk メソッド

TypeBinary または TypeLongBinary カラムからデータを取得します。

構文

```
GetByteChunk ( _
  offset As Long, _
  data As Long,  _
  data_len As Long, _
  filled_len As Long _
) As Boolean
Member of UltraLiteAFLib.ULColumn
```

パラメータ

offset　基本となるバイト配列へのオフセット。ソース・オフセットは、0 以上であることが必要です。それ以外の場合は、ulSQLE_INVALID_PARAMETER エラーが発生します。

data　バイトの配列へのポインタ。バイトの配列へのポインタを取得するには、Visual Basic VarPtr() 関数を使用します。

data_len　バッファ (配列) の長さ。data_len は 0 以上であることが必要です。

filled_len　これは OUT パラメータです。メソッドが呼び出された後で、有効なデータとともにフェッチされたバイト数を示します。BLOB データのサイズがあらかじめ知られていない場合は、BLOB データは、固定長のチャンクを使って、一度に 1 チャンクずつ、フェッチされます。最後にフェッチされるチャンクは、チャンク・サイズより小さくてもかまわないので、filled_len は、バッファ内にある有効なデータのバイト数を知らせます。

戻り値

このカラム値にデータがまだほかにも含まれる場合は **true**。

データベース内のこのカラム値に、これ以上データがない場合は **false**。

エラー・セット

ulSQLE_CONVERSION_ERROR　カラムのデータ型が BINARY でも LONG BINARY でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER　カラムのデータ型が BINARY でオフセットが 0 でも 1 でもない、またはデータ長が 0 より小さい場合、エラーが発生します。

カラムのデータ型が LONG BINARY で、オフセットが 1 より小さい場合も、エラーが発生します。

例

次の例で、edata はカラム名です。


```
'MobileVB using VB6
Dim filled As Long
Dim more_data As Boolean
Dim data (1 to 512) As Byte
more_data = table.Column("edata").GetByteChunk(0, _
VarPtr(data(1)), 512, filled)

'Crossfire using vb.net
Dim filled As Long
Dim more_data As Boolean
Dim data (1 to 512) As Byte
more_data = table.Column("edata").GetByteChunk(0, _
data, 512, filled)
```

GetStringChunk メソッド

TypeString または TypeLongString カラムからデータを取得します。

構文

GetStringChunk(_
  offset As Long, _
  data As String, _
  string_len As Long, _
  filled_len As Long _
) As Boolean
Member of **UltraLiteAFLib.ULColumn**

パラメータ

offset　基本となるデータへの文字オフセット。ここから文字列の取得を開始します。

data　文字列データを受信する変数。

string_length　返す文字列の長さ。

filled_len　フェッチされる文字列の長さ。

戻り値

データベースから取得するデータがまだある場合は **true**。

これ以上データがない場合は **false**。

エラー

ulSQLE_CONVERSION_ERROR　カラムのデータ型が CHAR でも LONG VARCHAR でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER　カラム・データ型が CHAR であり、src_offset が 64 K を超えている場合、エラーが発生します。

src_offset が 0 より小さいか、文字列の長さが 0 より小さい場合も、エラーが発生します。

SetByteChunk メソッド

TypeBinary または TypeLongBinary カラムにデータを設定します。

構文

SetByteChunk (_
data As Long, _
length As Long _
)
Member of **UltraLiteAFLib.ULColumn**

パラメータ

data MobileVB では、バイトの配列へのポインタ。バイトの配列へのポインタを取得するには、Visual Basic VarPtr() 関数を使用します。Crossfire では、バイトの配列であるローカル変数です。

length 配列の長さ。

エラー・セット

ulSQLE_CONVERSION_ERROR カラムのデータ型が BINARY でも LONG BINARY でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER データ長が 0 より小さいか 64K より大きい場合、エラーが発生します。

参照

- 「AppendByteChunk メソッド」 74 ページ

例

次の例では、edata はカラム名で、データ変数の最初の 232 バイトはデータベースに格納されています。

```
'MobileVB using VB6
Dim data (1 to 512) As Byte
' ...
table.Column("edata").SetByteChunk( VarPtr(data(1)), 232)

'Crossfire using vb.net
Dim data (1 to 512) As Byte
' ...
table.Column("edata").SetByteChunk( data, 232)
```

SetNull メソッド

カラムの値を null に設定します。

構文

SetNull( )
Member of **UltraLiteAFLib.ULColumn**

SetToDefault メソッド

現在のカラムを、データベース・スキーマで定義されているデフォルト値に設定します。たとえば、オートインクリメント・カラムの場合は、次に使用可能な値が割り当てられます。

構文

SetToDefault( )
Member of **UltraLiteAFLib.ULColumn**

ULColumnSchema クラス

ULColumnSchema オブジェクトを使用して、テーブル内のメタデータ、つまりカラムの属性を取得できます。属性は、テーブルのデータに依存しません。

プロパティ

構文	説明
AutoIncrement As Boolean (読み込み専用)	このカラムのデフォルトがオートインクリメント値かどうかを示す。オートインクリメントの場合は true です。
CurrentDate As Boolean (読み込み専用)	このカラムのデフォルトが現在の日付かどうかを示す。
CurrentTime As Boolean (読み込み専用)	このカラムのデフォルトが現在の時刻かどうかを示す。
CurrentTimestamp As Boolean (読み込み専用)	このカラムのデフォルトが現在のタイムスタンプかどうかを示す。
DefaultValue As String (読み込み専用)	ローの挿入時に値が指定されていない場合に使用される値を取得する。
GlobalAutoIncrement As Boolean (読み込み専用)	このカラムのデフォルトがグローバル・オートインクリメント値かどうかを示す。
GlobalAutoIncrementPartitionSize As Long (読み込み専用)	グローバル・オートインクリメント・カラムの分割サイズを取得する。
ID As Integer (読み込み専用)	カラムの ID を取得する。
Name As String (読み込み専用)	カラム名を取得する。
NewUUID As Boolean (読み込み専用)	このカラムのデフォルトが新しいユニバーサル・ユニーク識別子かどうかを示す。
Nullable As Boolean (読み込み専用)	カラムが NULL を許可するかどうかを示す。
OptimalIndex As ULIndexSchema (読み込み専用)	最初のカラムとしてこのカラムを持つインデックスを取得する。
Precision As Integer (読み込み専用)	型が ulTypeNumeric である場合は、カラムの精度値を取得する。
Scale As Integer (読み込み専用)	型が ulTypeNumeric である場合は、カラムの位取りの値を取得する。
Size As Integer (読み込み専用)	バイナリ、数値、char データ型のカラム・サイズを取得する。

構文	説明
SQLType As ULSQLType (読み込み専用)	作成時にカラムに割り当てられた SQL 型を取得する。

ULConnection クラス

ULConnection オブジェクトは、Ultra Light データベース接続を表します。これは、テーブルなど、同期対象となるテーブルなどのデータベース・オブジェクトを取得するメソッドを提供します。

同期の進行状況を受け取るための WithEvents の使用
同期を行う場合、ULConnection オブジェクトは進行情報も受け取ることができます。この情報を受け取る場合は、接続を WithEvents で宣言します。接続を WithEvents で宣言しなくても同期を実行できますが、接続オブジェクトは同期の進行状況の通知を受け取りません。

例

接続を **WithEvents** で宣言するには、MobileVB フォームで次の構文を使用します。

```
Public WithEvents Connection As ULConnection
```

WithEvents の追加により、同期の進行情報を受け取れます。

プロパティ

ULConnection のプロパティは次のとおりです。

構文	説明
AutoCommit As Boolean	AutoCommit 値を示す。true の場合、データを変更するとその直後にすべてのデータ変更がコミットされます。それ以外の場合、Commit を呼び出すまで、変更はデータベースにコミットされません。デフォルトでは、このプロパティは true です。
DatabaseID As Long	データベースの ID 番号。設定されていない場合は -1。
GlobalAutoIncrementUsage As Integer (読み込み専用)	利用可能なグローバル・オートインクリメントの値の使用済み比率 (%) を取得する。
LastIdentity As Long (読み込み専用)	デフォルトでオートインクリメントまたはグローバル・オートインクリメントに設定されたカラムに最後に挿入された値を取得する。
OpenParms As String (読み込み専用)	データベースへの接続を開くために使用される文字列を取得する。
Schema As ULDatabaseSchema (読み込み専用)	データベースの定義を表す ULDatabaseSchema オブジェクトを取得する。
SQLErrorOffset As Integer (読み込み専用)	PrepareStatement がエラーを発生した場合、エラーが記録された SQL 文で 1 から始まるオフセットを示す。この値が 0 以下の場合、オフセット情報はありません。

構文	説明
SyncParms As ULSyncParms (読み込み専用)	同期パラメータ・オブジェクトを取得する。
SyncResult As ULSyncResult (読み込み専用)	最後の同期の結果を取得する。

CancelSynchronize メソッド

同期処理中に呼び出された場合、このメソッドは同期をキャンセルします。ユーザがこのメソッドを呼び出せるのは、いずれかの同期イベントの実行時だけです。

これを可能にするには、ULConnection オブジェクトを「**WithEvents**」で宣言します。

構文

CancelSynchronize( )
Member of **UltraLiteAFLib.ULConnection**

ChangeEncryptionKey メソッド

指定されたキーを使用してデータベースを暗号化します。

構文

ChangeEncryptionKey(*newkey*As String **)**
Member of **UltraLiteAFLib.ULConnection**

パラメータ

newkey　データベースの新しい暗号化キーの値。

例

EncryptionKey に値を設定して CreateDatabase を呼び出すと、データベースは暗号化されて作成されます。暗号化キーを変更する別の方法は、新しい暗号化キーを ULConnection オブジェクトで指定することです。この例では "apricot" がキーです。

```
Connection.ChangeEncryptionKey( "apricot" )
```

OpenConnection のようなデータベースへの接続は、データベースが暗号化されてから、EncryptionKey プロパティにも *apricot* を指定します。そうしないと、接続は失敗します。

Close メソッド

データベースとの接続を閉じます。

構文

Close( )
Member of **UltraLiteAFLib.ULConnection**

備考

この接続で ULConnection オブジェクトなどのデータベース・オブジェクトのメソッドは、このメソッドを呼び出す前に呼び出します。接続が明示的に閉じられていない場合は、アプリケーションの終了時に暗黙的に閉じられます。

Commit メソッド

未処理の変更をデータベースにコミットします。AutoCommit が false の場合にのみ役立ちます。

構文

Commit( )
Member of **UltraLiteAFLib.ULConnection**

参照

- 「プロパティ」 81 ページ

CountUploadRows メソッド

次回の同期でアップロードする必要のあるロー数を返します。

構文

CountUploadRows(
  [*mask* As Long = 0], _
  [*threshold* As Long = -1] _
) As Long
Member of **UltraLiteAFLib.ULConnection**

パラメータ

mask　チェックするパブリケーションを示す、オプションのユニークな識別子。すべてのパブリケーションをチェックする場合は 0 を使用します。指定されていない場合は、値は 0 になります。

threshold　カウントするローの最大数を表す、オプションのパラメータ。最大数がないことを指定するには、-1 を使用します。指定されていない場合、この値は -1 です。

戻り値

次回の同期でアップロードする必要のあるロー数を返します。

GetNewUUID メソッド

新しいユニバーサル・ユニーク識別子を返します。

構文

GetNewUUID( ) As String
Member of **UltraLiteAFLib.ULConnection**

備考

値は *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* 形式の文字列であり、通常はデータ型 UNIQUEIDENTIFIER のカラムに格納されます。

戻り値

呼び出すたびに新しい UUID が返されます。

GetTable メソッド

指定されたテーブルの **ULTable** オブジェクトを返します。

構文

GetTable(*name* As String **)** As ULTable
Member of **UltraLiteAFLib.ULConnection**

パラメータ

id　調べるテーブルの名前。

備考

アプリケーションでテーブルのデータを読み込むことができるようにするには、テーブルを開く必要があります。

戻り値

ULTable オブジェクトを返します。

GrantConnectTo メソッド

指定したパスワードを使用してデータベースに接続するパーミッションを特定のユーザに付与します。

構文

```
GrantConnectTo(
  userid As String, _
  password As String _
)
```
Member of **UltraLiteAFLib.ULConnection**

パラメータ

userid　接続する権限を付与されるユーザ ID。

password　ユーザ ID が接続するのに指定するパスワード。

LastDownloadTime メソッド

パブリケーションの最終ダウンロード時間を返します。

構文

LastDownloadTime([*mask* As Long = 0] **)** As Date
Member of **UltraLiteAFLib.ULConnection**

パラメータ

mask　チェックするパブリケーションを示す、オプションのユニークな識別子。すべてのパブリケーションをチェックする場合は 0 を使用します。このパラメータが省略されている場合は、0 が使用されます。

戻り値

日付形式で表した最後のダウンロード時刻。

PrepareStatement メソッド

SQL 文の実行を準備します。

構文

PrepareStatement(
sqlStatement As String**,** _
persistent_name As String _
) As ULPreparedStatement
Member of **UltraLiteAFLib.ULConnection**

パラメータ

sqlStatement　準備する SQL 文。

persistent_name　Palm アプリケーションでは、文の持続的な名前。

戻り値

ULPreparedStatement を返します。文を準備するときに問題が起こった場合は、エラーが発生します。SQLErrorOffset プロパティを使用すると、エラーが発生した文のオフセットを特定できます。

ResetLastDownloadTime メソッド

マスクで指定されたパブリケーションの最後のダウンロードの時刻をリセットします。

構文

ResetLastDownloadTime([*mask* As Long] **)**
Member of **UltraLiteAFLib.ULConnection**

パラメータ

mask　リセットするパブリケーションのマスク。デフォルトは 0 で、すべてのパブリケーションを指定します。

RevokeConnectFrom メソッド

指定されたユーザがデータベースに接続できないようにします。

構文

RevokeConnectFrom(*userID* As String **)**
Member of **UltraLiteAFLib.ULConnection**

パラメータ

userid　接続する権限を取り消されるユーザ ID。

Rollback メソッド

未処理の変更をデータベースにロールバックします。AutoCommit が false の場合にのみ役立ちます。

構文

Rollback( )
Member of **UltraLiteAFLib.ULConnection**

RollbackPartialDownload メソッド

失敗した同期から変更をロールバックします。

構文

RollbackPartialDownload ( )
Member of **UltraLiteAFLib.ULConnection**

備考

同期のダウンロード時に通信エラーが発生した場合、Ultra Light では、ダウンロードした変更を適用し、同期が中断した時点から同期を再開することができます。ダウンロードした変更が不要な場合 (ダウンロードが中断した時点での再開を望まない場合)、RollbackPartialDownload を使用することで、失敗したダウンロード・トランザクションをロールバックします。

参照

- 「失敗したダウンロードの再開」『Mobile Link - サーバ管理』
- 「Keep Partial Download 同期パラメータ」『Mobile Link - クライアント管理』
- 「Partial Download Retained 同期パラメータ」『Mobile Link - クライアント管理』
- 「Resume Partial Download 同期パラメータ」『Mobile Link - クライアント管理』

StartSynchronizationDelete メソッド

削除操作が統合データベースで同期されるようにします。

構文

StartSynchronizationDelete( )
Member of **UltraLiteAFLib.ULConnection**

備考

削除操作が統合データベースで同期されるようにします。このメソッドは、StopSynchronizationDelete によって削除操作の同期が停止された後、再開するときに使用します。

参照

- 「StopSynchronizationDelete メソッド」 87 ページ

StopSynchronizationDelete メソッド

削除操作が統合データベースに同期されることを防ぎます。

構文

StopSynchronizationDelete( )
Member of **UltraLiteAFLib.ULConnection**

備考

このメソッドは、Ultra Light データベースから古い情報を削除して領域を節約するために使用すると便利です。ただし、統合データベース上の情報は削除されません。

参照

- 「StartSynchronizationDelete メソッド」 87 ページ

StringToUUID メソッド

形式が *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* の文字列として表されるユニバーサル・ユニーク識別子を 16 バイトのバイト配列に変換します。

構文

StringToUUID(
  s_uuid As String, _
  buffer_16_bytes As Long _
)
Member of **UltraLiteAFLib.ULConnection**

パラメータ

s_uuid　文字列として渡されるユニバーサル・ユニーク識別子。GetNewUUID を使用して新しい文字列 UUID を取得できます。

buffer_16_bytes　少なくとも 16 個の要素を持つ、バイト配列へのポインタ。VarPtr() 関数を使用して、ポインタ値を取得します。

備考

MobileVB アプリケーションでは、UUID を文字列形式で参照するのが便利な場合があります。したがって、ULColumn オブジェクトの UUIDValue プロパティは、文字列からバイナリ (16) へ、バイナリ (16) から文字列へ変換します。StringToUUID 関数は、MobileVB 文字列をバイト配列に簡単に変換します。この関数は、Ultra Light データベースをまったく参照しません。

バッファへのポインタに関する注意　バッファへのポインタは、少なくとも 16 バイトとして宣言します。Visual Basic には境界チェックが用意されていないため、バッファが小さすぎるとメモリが上書きされることがあります。MobileVB では、VarPtr() 関数を使用して、バッファへのポインタを取得します。ULColumn.UUIDValue プロパティも参照してください。

新しいデータベースでは不要　バージョン 9.0.2 より前に作成されたデータベースでは、UNIQUEIDENTIFIER データ型はユーザ定義データ型として定義され、UUID 値のバイナリ表現と文字列表現の間を変換するための関数が必要です。

バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型はネイティブ・データ型であり、Ultra Light が必要に応じて変換を実行します。したがって、StringToUUID 関数は不要です。

参照

- 「UNIQUEIDENTIFIER データ型」『SQL Anywhere サーバ - SQL リファレンス』

例

次の例は、文字列形式の UUID 0a141e28-323c-4650-5a64-6e78828c96a0 をバイナリ配列に変換します。

```
Dim buff(1 to 16) As Byte
conn.StringToUUID( "0a141e28-323c-4650-5a64-6e78828c96a0", VarPtr(buff(1)) )
```

Synchronize メソッド

Mobile Link を使用して統合データベースの同期を行います。

構文

Synchronize( )
Member of **UltraLiteAFLib.ULConnection**

備考

このメソッドは、同期が完了するまで戻りません。

イベントによる通知が可能です。接続の WithEvents を宣言してください。

参照

- 「ULConnection クラス」 81 ページ

UUIDToString メソッド

UUID をバイト配列から *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* 形式の文字列に変換します。

構文

UUIDToString(*buffer_16_bytes* As Long **)** As String
Member of **UltraLiteAFLib.ULConnection**

パラメータ

buffer_16_bytes　UUID を含む 16 バイトの配列。

戻り値

呼び出しごとに、*xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* の形式の文字列が返されます。

備考

バージョン 9.0.2 より前に作成されたデータベースでは、UNIQUEIDENTIFIER データ型はユーザ定義データ型として定義され、UUID 値のバイナリ表現と文字列表現の間を変換するための関数が必要です。

バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型はネイティブ・データ型であり、Ultra Light が必要に応じて変換を実行します。したがって、UUIDToString 関数は不要です。

参照

- 「UNIQUEIDENTIFIER データ型」『SQL Anywhere サーバ - SQL リファレンス』

ULConnectionParms クラス

ULConnectionParms オブジェクトを使用すると、ユーザ ID、パスワード、デスクトップ上のファイルなど、接続を指定する多くのパラメータの設定ができます。

プロパティ

ULConnectionParms クラスは、Ultra Light データベースへの接続を開くためのパラメータを指定します。

Ultra Light for MobileVB では、フォームに ULConnectionParms オブジェクトがあり、ConnectionParms ダイアログで接続プロパティを設定したことを確認します。ULConnectionParms オブジェクトは、**ULDatabaseManager.CreateDatabase** メソッドと **ULDatabaseManager.OpenConnection** メソッドとともに使用します。

注意
データベースは、1 人の認証済みユーザ DBA で作成されます。このユーザの最初のパスワードは sql です。ユーザ ID とパスワードが指定されなかった場合、接続はユーザ ID DBA とパスワード sql を使用して開かれます。

これらのパラメータの意味の詳細については、「Ultra Light の接続文字列パラメータのリファレンス」『Ultra Light - データベース管理とリファレンス』を参照してください。

構文	説明
CacheSize As String (読み込み／書き込み)	キャッシュのサイズ。CacheSize の値はバイト単位で指定します。キロバイトの単位を示すにはサフィックス k または K を使用し、メガバイトの単位を示すにはサフィックス M または m を使用します。デフォルトのキャッシュ・サイズは 16 ページです。デフォルトのページ・サイズは 4 KB なので、デフォルトのキャッシュ・サイズは 64 KB です。 「Ultra Light CACHE_SIZE 接続パラメータ」『Ultra Light - データベース管理とリファレンス』を参照してください。
ConnectionName As String (読み込み／書き込み)	接続の名前。接続名が必要となるのは、データベースとの接続を複数作成する場合だけです。 「Ultra Light CON 接続パラメータ」『Ultra Light - データベース管理とリファレンス』を参照してください。
DatabaseOnCE As String (読み込み／書き込み)	Windows CE に配備されるデータベースのファイル名。 「Ultra Light CE_FILE 接続パラメータ」『Ultra Light - データベース管理とリファレンス』を参照してください。

構文	説明
DatabaseOnDesktop As String (読み込み／書き込み)	開発中のデータベースのファイル名。 「Ultra Light DBF 接続パラメータ」『Ultra Light - データベース管理とリファレンス』を参照してください。
DatabaseOnPalm As String (読み込み／書き込み)	Palm デバイス上の Ultra Light データベースの作成者 ID。 「Ultra Light PALM_FILE 接続パラメータ」『Ultra Light - データベース管理とリファレンス』を参照してください。
DatabaseOnSymbian As String (読み込み／書き込み)	Symbian OS デバイス上のデータベースのファイル名。 「Ultra Light SYMBIAN_FILE 接続パラメータ」『Ultra Light - データベース管理とリファレンス』を参照してください。
EncryptionKey As String (読み込み／書き込み)	データベースを暗号化するためのキー。OpenConnection は、データベース作成中に指定されたキーと同じキーを使用する必要があります。キーの推奨ガイドラインは次のとおりです。 1. 任意の長い文字列を選択します。 2. キーを推測される可能性を減らすため、多様な数字、文字、特殊文字を使用した文字列を選択します。 「Ultra Light DBKEY 接続パラメータ」『Ultra Light - データベース管理とリファレンス』を参照してください。
PalmCreatorID As String (読み込み／書き込み)	登録されている 4 桁の Palm 作成者 ID。
Password As String (読み込み／書き込み)	認証ユーザのパスワード。データベースは最初、1 つの認証されたユーザ (DBA) とパスワード *sql* を使用して、作成されます。パスワードでは大文字と小文字が区別されます。 「Ultra Light PWD 接続パラメータ」『Ultra Light - データベース管理とリファレンス』を参照してください。
ToString As String	現在のプロパティ設定を基に作成された接続文字列。
UserID As String (読み込み／書き込み)	データベースで認証されたユーザ。データベースは最初、1 つの認証されたユーザ DBA を使用して、作成されます。データベースの大文字と小文字を区別しない場合は、UserID の大文字と小文字を区別せず、データベースの大文字と小文字を区別する場合は、UserID の大文字と小文字も区別します。デフォルト値は *DBA* です。 「Ultra Light UID 接続パラメータ」『Ultra Light - データベース管理とリファレンス』を参照してください。

ULDatabaseManager クラス

ULDatabaseManager クラスを使用して、接続とデータベースを管理します。アプリケーションは、このオブジェクトのインスタンスを 1 つだけ持ちます。データベースを作成し、そのデータベースへの接続を確立することは、Ultra Light の使用に必要な最初の手順です。CreateDatabase、OpenConnection、DropDatabase を使用し、正しく接続してからデータ操作言語でデータベースを操作するように、コードに検査制約を含めることをおすすめします。

プロパティ

ULDatabaseManager のプロパティは次のとおりです。

構文	説明
Version As String (読み込み専用)	Ultra Light コンポーネントのバージョン文字列を取得する。

CreateDatabase メソッド

新しい Ultra Light データベースを、指定した作成パラメータ、照合順、および接続パラメータを使用して作成します。

構文

CreateDatabase(*connparms* As string **,**
collation As Long **,**
createparms As String
) As ULConnection
Member of **UltraLiteAFLib.ULDatabaseManager**

パラメータ

connparms　データベースのアクセス・パラメータ。OpenConnection で指定するパラメータに似ています。

collation　照合を示すバイトの配列へのポインタ。照合のバイト配列は、*install-dir¥src¥ulcollations¥af.vb.net* フォルダにある、事前に定義された照合を使用して、Visual Basic .NET for AppForge で初期化されます。

createparms　データベース作成パラメータ (難読化やページ・サイズなどデータベース作成時のみ指定できるデータベース特性)。

「Ultra Light で使用する作成時のデータベース・プロパティの選択」『Ultra Light - データベース管理とリファレンス』を参照してください。

戻り値

データベースが正常に作成されると、接続を返します。

DropDatabase メソッド

データベース・ファイルを削除します。

構文

DropDatabase(*parms* As String **)**
Member of **UltraLiteAFLib.ULDatabaseManager**

パラメータ

parms　データベースのファイル名。

備考

データベースを削除すると、データベース・ファイルに含まれているすべてのデータが完全に削除されます。

このメソッドが失敗するのは次の場合です。

- 指定したデータベース・ファイルが存在しない。
- DropDatabase の実行時に開かれている接続がある。

例

次の例は、データベースを削除します。

```
Dim parms As String
parms = "PALM_DB=Syb1;NT_FILE=c:¥temp¥ul_CustDB.udb"
DropDatabase(parms)
```

OpenConnection メソッド

指定されたデータベースへの接続を開きます。

構文

OpenConnection(*connparms* As string **)** As ULConnection
Member of **UltraLiteAFLib.ULDatabaseManager**

パラメータ

connparms　データベースへの接続を確立するために使用されるパラメータ。パラメータは、セミコロンで区切られた一連の**キーワード=*値***の組み合わせを使用して指定します。ユーザ ID またはパスワードを指定しない場合は、デフォルトが使用されます。

戻り値

接続に成功した場合は、ULConnection オブジェクトが返されます。このオブジェクトは、指定された Ultra Light データベースへの接続を開きます。データベース・ファイル名は、*connparms* 接続文字列を使用して指定します。パラメータは、一連の**名前**=***値***の組み合わせを使用して指定します。ユーザ ID またはパスワードを指定しない場合は、デフォルトが使用されます。

「接続文字列を使用した Ultra Light 接続の確立」『Ultra Light - データベース管理とリファレンス』を参照してください。

備考

このメソッドは、作成済みのデータベースに接続するのに使用します。

このメソッドが失敗するのは次の場合です。

- データベースが存在しない。
- 接続パラメータが無効である。

呼び出しが失敗した理由を特定するには、エラー・オブジェクトを使用します。

例

次の例は、CustDB サンプル・アプリケーションからのデータベース接続を新しく作成します。

```
Set Connection = DatabaseMgr.OpenConnection("UID=JDoe;PWD=ULdb;
NT_FILE=c:¥test¥MyTestDB.udb;CE_FILE=¥database
¥MyCEDB.udb;PALM_FILE=MyPalmDB_MyCreatorID")
```

ULDatabaseSchema クラス

ULDatabaseSchema オブジェクトを使用すると、接続先データベースの属性を取得できます。

プロパティ

ULDatabaseSchema のプロパティは次のとおりです。

構文	説明
CollationName As String (読み込み専用)	データベースの照合順の名前を取得する。
DateFormat As String (読み込み専用)	データベースから取り出した日付のフォーマットを取得する。デフォルトは、'YYYY-MM-DD' です。取得される日付のフォーマットは、データベースの作成時に使用したフォーマットに対応しています。
DateOrder As String (読み込み専用)	日付フォーマットの解釈を示す。有効な値は 'MDY'、'YMD'、または 'DMY' です。
IsCaseSensitive As Boolean (読み込み専用)	データベースで大文字と小文字が区別されるかどうかを示す。
NearestCentury As String (読み込み専用)	文字列から日付への変換で、2 桁の年の解釈を示す。これは、ロールオーバ・ポイントとして動作する数値です。この値より小さい 2 桁の年は 20yy に変換され、この値以上の年は 19yy に変換されます。デフォルトは 50 です。
Precision As String (読み込み専用)	10 進法計算での結果の最大桁数を取得する。
PublicationCount As Integer (読み込み専用)	接続したデータベース内のパブリケーション数を取得する。
TableCount As Integer (読み込み専用)	接続したデータベース内のテーブル数を取得する。
TimeFormat As String (読み込み専用)	データベースから取り出した時刻のフォーマットを取得する。
TimestampFormat As String (読み込み専用)	データベースから取り出したタイムスタンプのフォーマットを取得する。

GetDatabaseProperty メソッド

指定したデータベース・プロパティの値を返します。

構文

GetDatabaseProperty(*property_name* As String **)** As
String Member of **UltraLiteAFLib.ULDatabaseSchema**

パラメータ

property_name *property_name* には、問い合わせされるプロパティ名を指定します。

プロパティ名の文字列の一覧を次に示します。これらのプロパティの意味の詳細については、「Ultra Light データベース設定のリファレンス」『Ultra Light - データベース管理とリファレンス』を参照してください。

- CaseSensitive
- CharSet
- ChecksumLevel
- CollationName
- ConnCount
- date_format
- date_order
- Encryption
- File
- global_database_id
- MaxHashSize
- ml_remote_id
- Name
- nearest_century
- PageSize
- precision
- scale
- time_format
- timestamp_format
- timestamp_increment

戻り値

指定したプロパティの値を返します。

GetPublicationName メソッド

指定したパブリケーションの名前を返します。

構文

GetPublicationName(*id* As Integer **)** As String
Member of **UltraLiteAFLib.ULDatabaseSchema**

パラメータ

id *id* は、名前が返されるパブリケーションの識別子。パブリケーション *ID* の範囲は、1 ー PublicationCount です。

戻り値

接続しているデータベース内のパブリケーションの名前を返します。

ULPublicationSchema オブジェクトの詳細については、「ULPublicationSchema クラス」 111 ページを参照してください。

GetPublicationSchema メソッド

パブリケーション名を使用して ULPublicationSchema オブジェクトを取得します。

構文

GetPublicationSchema(*Name* As String **)** As ULPublicationSchema
Member of **UltraLiteAFLib.ULDatabaseSchema**

パラメータ

name　パブリケーションの *name*。

戻り値

ULPublicationSchema オブジェクトを返します。

GetTableName メソッド

指定した *id* 値に対応する、接続しているデータベース内のテーブル名を返します。

構文

GetTableName(*id* As Integer **)** As String
Member of **UltraLiteAFLib.ULDatabaseSchema**

パラメータ

id　テーブルの *id*。

戻り値

指定した *id* のテーブル名を返します。

備考

TableCount プロパティは、接続しているデータベース内のテーブル数を返します。各テーブルには、1 ー TableCount 値までのユニークな番号があります。1 はデータベース内の最初のテーブルであり、2 はデータベース内の 2 番目のテーブルです。以下も同様に続きます。データベースのスキーマが変わると、テーブルの id が変わる場合があります。

SetDatabaseOption メソッド

データベース・オプションの値を設定します。

構文

SetDatabaseOption(
option_name As String
option_value As String
)

パラメータ

option_name　設定するデータベース・オプションの名前。設定できデータベース・オプションは global_database_id と ml_remote_id です。次の項を参照してください。

- 「Ultra Light global_database_id オプション」『Ultra Light - データベース管理とリファレンス』
- 「Ultra Light ml_remote_id オプション」『Ultra Light - データベース管理とリファレンス』

option_value　オプションの新しい値。

ULFileTransfer クラス

Mobile Link ファイル転送機能へのインタフェースです。

プロパティ

ULFileTransfer のプロパティは次のとおりです。

構文	説明
AuthStatus As ULAuthStatusCode	前回行われたファイル転送の認証ステータス・コード
AuthValue As Integer	前回行われたファイル転送の認証値
DestinationFile As String	DestinationPath に存在するダウンロード・ファイルの名前
DestinationPath As String	ダウンロード・ファイルのクライアント・ロケーション
DowloadedFile As Boolean	ファイルがダウンロードされると true に設定される。
FileAuthCode As Integer	前回行われたファイル転送の認証コード
FileName As String	ダウンロードするリモート・ファイル名
ForceDownload As Boolean	true の場合に、ファイルを無条件にダウンロードする。
Password As String	UserName プロパティに指定された同期ユーザ名のパスワード
ResumePartialDownload As Boolean	true に設定されると、前回保持された部分的なダウンロードを再開する。
Stream As ULStreamType	同期で使用するストリームのタイプ
StreamErrorCode As ULStreamErrorCode	ストリームによってレポートされるエラー・コード
StreamParms As String	指定されたストリームのタイプで使用する追加パラメータ
StreamErrorSystem As Long	ストリーム・エラー・システム固有のコード
UserName As String	同期中に Mobile Link に送信されるユーザ名
Version As String	実行する同期スクリプトのバージョン

参照

- 「MLFileTransfer 関数」 『Ultra Light - C/C++ プログラミング』

- 「Mobile Link ファイル転送ユーティリティ [mlfiletransfer]」『Mobile Link - クライアント管理』

AddAuthenticationParm メソッド

カスタム認証パラメータのリストに認証パラメータを追加します。

構文

AddAuthenticationParm(*newParm* As String **)**
As Boolean
Member of **UltraLiteAFLib.ULFileTransfer**

パラメータ

newParm　追加する認証パラメータ。

CancelTransfer メソッド

DownloadFile メソッドによって開始された、処理中のファイル転送をキャンセルします。

構文

CancelTransfer() Member of **UltraLiteAFLib.ULFileTransfer**

ClearAuthenticationParms メソッド

カスタム認証パラメータのリストをクリアします。

構文

ClearAuthenticationParms()
Member of **UltraLiteAFLib.ULFileTransfer**

DownloadFile メソッド

Mobile Link ファイル転送を使用して、現在のプロパティでファイルをダウンロードします。

構文

DownloadFile()
As Boolean
Member of **UltraLiteAFLib.ULFileTransfer**

戻り値

ファイルのダウンロードに成功するか、ファイルがローカルにすでに存在していた場合は true を返します。それ以外の場合は false を返します。ファイル転送が実際に行われたかどうかは、プロパティ ULFileTransfer.DownloadedFile で確認できます。

エラー・セット

ulSQLE_COMMMUNICATIONS_ERROR　ストリームが指定されなかった場合、エラーが発生します。

参照

- 「MLFileTransfer 関数」『Ultra Light - C/C++ プログラミング』
- 「Mobile Link ファイル転送ユーティリティ [mlfiletransfer]」『Mobile Link - クライアント管理』

ULFileTransferEvent クラス

ULFileTransferEvent オブジェクトは、Mobile Link ファイル転送中にイベントを通知するメカニズムを提供します。

「ULFileTransfer クラス」 99 ページを参照してください。

OnTransferProgressChanged メソッド

現在のファイル転送に転送済みのデータがあることをアプリケーションに通知します。

構文

OnTransferProgressChanged(
file_size As Long **,**
bytes_received As Long **,**
resumed_at_size As Long
)
Member of **UltraLiteAFLib.ULFileTransferEvent**

パラメータ

file_size　バイト単位でのファイル・サイズ。

bytes_received　転送されたバイト数。

resumed_at_size　転送が再開されたときのバイト数。

戻り値

転送が続行された場合は true を返します。

ULIndexSchema クラス

ULIndexSchema オブジェクトを使用すると、インデックスの属性を取得できます。インデックスは順序付きカラムのセットであり、これを使用してテーブル内のデータをソートします。インデックスはおもに、1 つ以上のカラムでテーブルのデータを並べ替えるために使用されます。

インデックスに外部キーを使用することができ、この場合、データベースの参照整合性が維持されます。

プロパティ

構文	説明
ColumnCount As Integer (読み込み専用)	インデックス内のカラム数を取得する。
ForeignKey As Boolean (読み込み専用)	外部キーかどうかを示す。
ForeignKeyCheckOnCommit As Boolean (読み込み専用)	コミットが行われたときだけ (TRUE) または即座に (FALSE)、参照整合性をチェックするかどうかを示す。
ForeignKeyNullable As Boolean (読み込み専用)	外部キーのカラムが NULL を許容するかどうかを示す。
Name As String (読み込み専用)	インデックスの名前を取得する。
PrimaryKey As Boolean (読み込み専用)	このテーブルのプライマリ・キーかどうかを取得する。
ReferencedIndexName As String (読み込み専用)	インデックスが外部キーの場合、このインデックスが参照するインデックスの名前を取得する。
ReferencedTableName As String (読み込み専用)	インデックスが外部キーの場合、このインデックスが参照するテーブルの名前を取得する。
UniqueIndex As Boolean (読み込み専用)	インデックスの値がユニークである必要があるかどうかを示す。
UniqueKey As Boolean (読み込み専用)	インデックスがテーブルの一意性制約かどうかを示す。true の場合は、インデックス内のカラムはユニークであり、NULL 値を使用できません。

GetColumnName メソッド

インデックス内のカラム名を返します。

構文

GetColumnName(*col_pos_in_index* As Integer **)** As String
Member of **UltraLiteAFLib.ULIndexSchema**

パラメータ

col_pos_in_index　インデックス内のカラム位置。*col_pos_in_index* には 1 ー ColumnCount までの値を設定します。

戻り値

インデックス内のカラム名を返します。

IsColumnDescending メソッド

インデックス内の指定したカラムが降順であるかどうかを示します。

構文

IsColumnDescending(*col_name* As String **)** As Boolean
Member of **UltraLiteAFLib.ULIndexSchema**

パラメータ

col_name　インデックス・カラム名。

戻り値

カラムが降順の場合は **true**。

カラムが昇順の場合は **false**。

ULPreparedStatement クラス

ULPreparedStatement は、実行の準備ができたコンパイル済みの SQL 文を表します。準備文を使用して、SQL クエリを実行できます。ULPreparedStatement を使用すれば、多くの入力パラメータを使用して、同じ文を複数回実行することもできます。準備文はコンパイル済みなので、最初の実行以後の追加については、わずかな処理で済みます。複数のローで比較的速いデータ操作言語が必要な場合は、ULPreparedStatement と Dynamic SQL を使用します。

プロパティ

構文	説明
HasResultSet As Boolean (読み込み専用)	準備文が結果セットを生成するかどうかを示す。 文が結果セットを持つ場合は true、持たない場合は false です。 true の場合、ExecuteStatement ではなく ExecuteQuery を呼び出します。
Plan (読み込み専用) As String	クエリを実行するのに Ultra Light が使用するアクセス・プランを取得する。このプロパティは、主に開発中の使用を目的とします。
ResultSetSchema As ULResultSetSchema (読み込み専用)	その文が結果セット用の場合、結果セットのスキーマの説明を取得する。

AppendByteChunkParameter メソッド

カラムの型が ulTypeLongBinary の場合、バイトのバッファをローのカラムに追加します。

構文

AppendByteChunkParameter (
param_id As Integer**,**
data As Long**,**
data_len As Long**)**
Member of **UltraLiteAFLib.ULPreparedStatement**

パラメータ

parameter_id　設定する 1 から始まるパラメータ番号。

data　追加するバイトの配列。

data_len　配列から追加するバイト数。

エラー・セット

ulSQLE_INVALID_PARAMETER　データ長が 0 より小さい場合、エラーが発生します。

ulSQLE_CONVERSION_ERROR　カラムのデータ型が LONG BINARY でない場合、エラーが発生します。

AppendStringChunkParameter メソッド

カラムの型が ulTypeLongString の場合、カラムに文字列を追加します。

構文

AppendStringChunkParameter(
param_id As Integer **,**
chunk As String **)**
Member of **UltraLiteAFLib.ULPreparedStatement**

パラメータ

parameter_id　設定する 1 から始まるパラメータ番号。

chunk　テーブル内の既存の文字列に追加する文字列。

エラー・セット

ulSQLE_CONVERSION_ERROR　カラムのデータ型が LONG VARCHAR でない場合、エラーが発生します。

Close メソッド

ULPreparedStatement に関連付けられているリソースを解放します。

構文

Close( )
Member of **UltraLiteAFLib.ULPreparedStatement**

ExecuteQuery メソッド

クエリを実行し結果セットを返します。

構文

ExecuteQuery( ) As ULResultSet
Member of **UltraLiteAFLib.ULPreparedStatement**

戻り値

ULResultSet オブジェクト。ULResultSet は、SELECT 文で要求したデータです。

クエリの結果に関する詳細については、「ULResultSetSchema クラス」 125 ページを参照してください。

ExecuteStatement メソッド

文を実行します。

構文

ExecuteStatement( ) As Long
Member of **UltraLiteAFLib.ULPreparedStatement**

戻り値

更新されたローの数。

SetBooleanParameter メソッド

パラメータを、渡された Boolean 値に設定します。

構文

SetBooleanParameter(
param_number As Integer
param_value As Boolean
)
Member of **UltraLiteAFLib.ULPreparedStatement**

パラメータ

param_number　設定する 1 から始まるパラメータ番号。

param_value　パラメータが受け取る値。

SetByteChunkParameter メソッド

データを binary または long binary のカラムに設定します。

構文

SetByteChunkParameter(
param_number As Integer**,**
data As Long**,**
data_len As Long
)
Member of **UltraLiteAFLib.ULPreparedStatement**

パラメータ

param_number　設定する 1 から始まるパラメータ番号。

data　バイトの配列。

data_len　設定する配列からのバイトの数。SetByteChunk は、現在の内容を上書きします。

既存の値への追加の詳細については、「AppendByteChunkParameter メソッド」 105 ページを参照してください。

SetByteParameter メソッド

指定した Byte 値に対するパラメータを設定します。

構文

SetByteParameter(
param_number As Integer
param_value As Byte
)
Member of **UltraLiteAFLib.ULPreparedStatement**

パラメータ

param_number　設定する 1 から始まるパラメータ番号。

param_value　パラメータが受け取る値。

SetDatetimeParameter メソッド

指定した Datetime 値に対するパラメータを設定します。

構文

SetDatetimeParameter(
param_number As Integer
param_value As String
)
Member of **UltraLiteAFLib.ULPreparedStatement**

パラメータ

param_number　設定する 1 から始まるパラメータ番号。

param_value　パラメータが受け取る値。

SetDoubleParameter メソッド

指定した Double 値に対するパラメータを設定します。

構文

SetDoubleParameter(
param_number As Integer
param_value As String
)
Member of **UltraLiteAFLib.ULPreparedStatement**

パラメータ

param_number　設定する 1 から始まるパラメータ番号。

param_value　パラメータが受け取る値。

SetIntegerParameter メソッド

指定した Integer 値に対するパラメータを設定します。

構文

SetIntegerParameter(
param_number As Integer
param_value As String
)
Member of **UltraLiteAFLib.ULPreparedStatement**

パラメータ

param_number　設定する 1 から始まるパラメータ番号。

param_value　パラメータが受け取る値。

SetLongParameter メソッド

指定した Long 値に対するパラメータを設定します。

構文

SetLongParameter(
param_number As Integer
param_value As String
)
Member of **UltraLiteAFLib.ULPreparedStatement**

パラメータ

param_number　設定する 1 から始まるパラメータ番号。

param_value　パラメータが受け取る値。

SetNullParameter メソッド

パラメータを NULL に設定します。

構文

SetNullParameter(*param_id* As Integer **)**
Member of **UltraLiteAFLib.ULPreparedStatement**

パラメータ

parameter_id　設定する 1 から始まるパラメータ番号。

SetRealParameter メソッド

指定した Long 値に対するパラメータを設定します。

構文

SetRealParameter(
param_number As Integer
param_value As String
)
Member of **UltraLiteAFLib.ULPreparedStatement**

パラメータ

param_number　設定する 1 から始まるパラメータ番号。

param_value　パラメータが受け取る値。

SetStringParameter メソッド

指定した文字列に対するパラメータを設定します。

構文

SetStringParameter(
param_number As Integer
param_value As String
)
Member of **UltraLiteAFLib.ULPreparedStatement**

パラメータ

param_number　設定する 1 から始まるパラメータ番号。

param_value　パラメータが受け取る値。

ULPublicationSchema クラス

ULPublicationSchema オブジェクトを使用すると、パブリケーションの属性を取得できます。

プロパティ

構文	説明
Mask As Long (読み込み専用)	パブリケーションのマスクを取得する。
Name As String (読み込み専用)	パブリケーションの名前を取得する。

ContainsTable メソッド

指定したテーブルがこのパブリケーションに含まれるかどうかを示します。

構文

ContainsTable(*name* As String **)** As Boolean
Member of **UltraLiteAFLib.ULPublicationSchema**

パラメータ

name　ターゲット・テーブル名。

戻り値

テーブルがパブリケーションに含まれる場合は **true**。

テーブルがパブリケーションに含まれない場合は **false**。

ULResultSet クラス

ULResultSet オブジェクトは、SQL クエリが返したローを移動します。ULResultSet オブジェクトには、クエリが返したデータが含まれているので、INSERT、UPDATE、または DELETE のようなデータ操作言語の操作を行った後では、クエリの結果セットをリフレッシュする必要があります。このためには、ExecuteStatement を実行してから、ExecuteQuery を実行します。

ULResultSet カラムには、結果セットの 1 から始まる相対カラム番号を表す序数を使用してアクセスします。このパラメータは、以降の説明では *index* という名前になっています。

注意
基本となるカラムが NULL の場合に、Get メソッドはエラーをスローします。アプリケーションはプロパティまたはメソッドの NULL 値をチェックしてから、取得を試みます。

プロパティ

構文	説明
BOF As Boolean (読み込み専用)	現在のローの位置が最初のローの前かどうかを示す。現在のローの位置が最初のローの前なら true を返し、そうでなければ false を返す。
EOF As Boolean (読み込み専用)	現在のローの位置が最後のローの後かどうかを示す。最後のローの後であれば、EOF は true であり、そうでなければ false となる。
RowCount As Long (読み込み専用)	結果セット内のロー数。
Schema As ULResultSetSchema (読み込み専用)	この結果セットのスキーマの説明。

AppendByteChunk メソッド

カラムの型が ulTypeLongBinary の場合、バイトのバッファをローのカラムに追加します。

構文

AppendByteChunk(
index As Integer,
data As Long,
data_len As Long**)**
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　設定する 1 から始まるパラメータ番号。

data　追加するバイトの配列。

data_len　追加する配列からのバイトの数。

エラー・セット

ulSQLE_INVALID_PARAMETER　データ長が 0 より小さい場合、エラーが発生します。

ulSQLE_CONVERSION_ERROR　カラムのデータ型が LONG BINARY でない場合、エラーが発生します。

AppendStringChunk メソッド

ローのカラムの型が ulTypeLongString の場合に文字列を追加します。

構文

AppendStringChunk(
index As Integer **,**
data As String **)**
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　設定する 1 から始まるパラメータ番号。

data　テーブル内の既存の文字列に追加する文字列。

エラー・セット

ulSQLE_CONVERSION_ERROR　カラムのデータ型が LONG VARCHAR でない場合、エラーが発生します。

Close メソッド

このオブジェクトに関連付けられているリソースを解放します。

構文

Close()
Member of **UltraLiteAFLib.ULResultSet**

Delete メソッド

テーブルの現在のローを削除します。

構文

Delete()
Member of **UltraLiteAFLib.ULResultSet**

GetBoolean メソッド

カラム値をブール値として取得します。

構文

GetBoolean(*index* As Integer **)** As Boolean
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　結果セットでの 1 から始まる序数。

戻り値

ブール値として返される値。

備考

このメソッドは、基本となるカラムが NULL の場合にエラーをスローします。アプリケーションは、プロパティまたはメソッドが NULL 値かどうかをチェックする必要があります。

GetByte メソッド

カラム値をバイトとして取得します。

構文

GetByte(*index* As Integer **)** As Byte
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　結果セットでの 1 から始まる序数。

戻り値

バイトとして返される値。

備考

このメソッドは、基本となるカラムが NULL の場合にエラーをスローします。アプリケーションは、プロパティまたはメソッドが NULL 値かどうかをチェックする必要があります。

GetByteChunk メソッド

渡されたバッファ (配列) に、カラム内のバイナリ・データを入れます。

構文

GetByteChunk (_
index As Integer, _
src_offset As Long, _
data As Long, _
data_len As Long, _

filled_len As Long _
) As Boolean
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　バイナリ・データを含むカラムの 1 から始まる序数。

offset　基本となるバイト配列へのオフセット。ソース・オフセットは、0 以上であることが必要です。それ以外の場合は、SQLE_INVALID_PARAMETER エラーが発生します。64K を超えるバッファも許されます。

data　バイトの配列へのポインタ。バイトの配列へのポインタを取得するには、Visual Basic VarPtr() 関数を使用します。

data_len　バッファ (配列) の長さ。data_len は 0 以上であることが必要です。

filled_len　フェッチされたバイト数。BLOB データの長さは事前には分からないので、BLOB データは通常、固定長のチャンクを使って、一度に 1 チャンクずつ、フェッチします。最後のチャンクは、チャンク・サイズよりも小さい場合があります。filled_len は、実際にフェッチされたバイト数をレポートします。

戻り値

読み込まれたバイト数。

備考

このメソッドは、基本となるカラムが NULL の場合にエラーをスローします。アプリケーションは、プロパティまたはメソッドが NULL 値かどうかをチェックする必要があります。

このメソッドは BLOB に適しています。

エラー・セット

ulSQLE_CONVERSION_ERROR　カラムのデータ型が BINARY でも LONG BINARY でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER　カラムのデータ型が BINARY でオフセットが 0 でも 1 でもない、またはデータ長が 0 より小さい場合、エラーが発生します。

カラムのデータ型が LONG BINARY で、オフセットが 1 より小さい場合も、エラーが発生します。

例

次の例で、edata はカラム名です。渡される *data_len* パラメータの長さが十分でない場合は、アプリケーション全体が終了します。

```
Dim data (512) As Byte
...
table.Column("edata").GetByteChunk(0,data)
```


GetDatetime メソッド

カラム値を Date として取得します。

構文

GetDatetime(*index* As Integer **)** As Date
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　結果セットで取得する 1 から始まる序数。

戻り値

Date として返される値。

備考

このメソッドは、基本となるカラムが NULL の場合にエラーをスローします。アプリケーションは、プロパティまたはメソッドが NULL 値かどうかをチェックする必要があります。

GetDouble メソッド

カラム値を Double として取得します。

構文

GetDouble(*index* As Integer **)** As Double
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　結果セットで取得する 1 から始まる序数。

戻り値

Double として返される値。

備考

このメソッドは、基本となるカラムが NULL の場合にエラーをスローします。アプリケーションは、プロパティまたはメソッドが NULL 値かどうかをチェックする必要があります。

GetInteger メソッド

カラム値を Integer として取得します。

構文

GetInteger(*index* As Integer **)** As Integer
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　結果セットで取得する 1 から始まる序数。

戻り値

Integer として返される値。

備考

このメソッドは、基本となるカラムが NULL の場合にエラーをスローします。アプリケーションは、プロパティまたはメソッドが NULL 値かどうかをチェックする必要があります。

GetLong メソッド

カラム値を Long として取得します。

構文

GetLong(*index* As Integer **)** As Long
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　結果セットで取得する 1 から始まる序数。

戻り値

Long として返される値。

備考

このメソッドは、基本となるカラムが NULL の場合にエラーをスローします。アプリケーションは、プロパティまたはメソッドが NULL 値かどうかをチェックする必要があります。

GetReal メソッド

カラム値を Real として取得します。

構文

GetReal(*index* As Integer **)** As Single
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　結果セットで取得する 1 から始まる序数。

戻り値

Real として返される値。

備考

このメソッドは、基本となるカラムが NULL の場合にエラーをスローします。アプリケーションは、プロパティまたはメソッドが NULL 値かどうかをチェックする必要があります。

GetString メソッド

カラム値を String として取得します。

構文

GetString(*index* As Integer **)** As String
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　結果セットで取得する 1 から始まる序数。

戻り値

String として返される値。

備考

このメソッドは、基本となるカラムが NULL の場合にエラーをスローします。アプリケーションは、プロパティまたはメソッドが NULL 値かどうかをチェックする必要があります。

GetStringChunk メソッド

渡された文字列に、カラム内のバイナリ・データを挿入します。Long Varchar に適しています。

構文

```
GetStringChunk( _
  index As Integer, _
  offset As Long, _
  data As String, _
  string_len As Long, _
  filled_len As Long _
) As Boolean
```
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　ターゲット・カラムの 1 から始まるカラム ID。

offset　基本となるデータへの文字オフセット。ここから文字列の取得を開始します。

data　データ文字列。

string_len　返す文字列の長さ。

filled_len　挿入される文字列の長さ。

戻り値

BLOB データを binary または long binary のカラムから取得します。

備考

このメソッドは、基本となるカラムが NULL の場合にエラーをスローします。アプリケーションは、プロパティまたはメソッドが NULL 値かどうかをチェックする必要があります。

エラー・セット

ulSQLE_CONVERSION_ERROR　カラムのデータ型が CHAR でも LONG VARCHAR でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER　カラム・データ型が CHAR であり、src_offset が 64 K を超えている場合、エラーが発生します。オフセットが 0 より小さいか、文字列の長さが 0 より小さい場合も、エラーが発生します。

IsNull メソッド

このカラムに null 値が含まれているかどうかを示します。

構文

IsNull(*index* As Integer **)** As Boolean
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　カラムのインデックス値。

戻り値

値が NULL の場合は true。

MoveAfterLast メソッド

ULResultSet の最後のローの後に移動します。

構文

MoveAfterLast( )
Member of **UltraLiteAFLib.ULResultSet**

MoveBeforeFirst メソッド

最初のローの前に移動します。

構文

MoveBeforeFirst( )
Member of **UltraLiteAFLib.ULResultSet**

MoveFirst メソッド

最初のローに移動します。

構文

MoveFirst( ) As Boolean
Member of **UltraLiteAFLib.ULResultSet**

戻り値

成功の場合は **true**。

失敗の場合は **false**。たとえば、ローがない場合、メソッドは失敗します。

MoveLast メソッド

最後のローに移動します。

構文

MoveLast( ) As Boolean
Member of **UltraLiteAFLib.ULResultSet**

戻り値

成功の場合は **true**。

失敗の場合は **false**。たとえば、ローがない場合、メソッドは失敗します。

MoveNext メソッド

次のローに移動します。

構文

MoveNext( ) As Boolean
Member of **UltraLiteAFLib.ULResultSet**

戻り値

成功の場合は **true**。

失敗の場合は **false**。たとえば、ローがない場合、メソッドは失敗します。

MovePrevious メソッド

前のローに移動します。

構文

MovePrevious( ) As Boolean
Member of **UltraLiteAFLib.ULResultSet**

戻り値

成功の場合は **true**。

失敗の場合は **false**。たとえば、ローがない場合、メソッドは失敗します。

MoveRelative メソッド

いくつかのローを、現在のローを基準にして相対的に移動します。

構文

MoveRelative(*index* As Long **)** As Boolean
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　移動するローの数。値は、正、負、または 0 です。結果セットでのカーソルの現在位置を基準にして、正のインデックス値は結果セット内を前に移動し、負のインデックス値は結果セット内を後ろに移動し、0 はカーソルを移動しません。

戻り値

成功の場合は **true**。

失敗の場合は **false**。たとえば、ローがない場合、メソッドは失敗します。

SetBoolean メソッド

指定したカラムを、渡された Boolean 値に設定します。

構文

SetBoolean(
  index As Integer**, _**
  value As Boolean
)
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　設定対象の結果セット内のカラムの 1 から始まる序数。

value　新しいブール値。

SetByte メソッド

指定したカラムを、渡された Byte 値に設定します。

構文

SetByte(
index As Integer **, _**
data As Byte
)
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　結果セットでの 1 から始まる序数。

SetByteChunk メソッド

指定したカラムを、渡されたバイナリ値に設定します。

構文

```
SetByteChunk(
  index As Integer, _
  data As Long , _
  data_len As Long
)
```
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　設定対象の結果セット内のカラムの 1 から始まる序数。

data　新しいデータが格納されているバッファへのポインタ。

data_len　データ・バッファの長さ。

SetDatetime メソッド

現在のローの指定したカラムを、指定された Datetime 値に設定します。

構文

```
SetDatetime(
  index As Integer
  value As Date
)
```
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　設定対象となる現在のローのカラムの 1 から始まる序数。

value　カラムが受け取る値。

SetDouble メソッド

指定した Double 値に対するパラメータを設定します。

構文

```
SetDouble(
  index As Integer
  value As Double
```

)
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　設定対象となる現在のローのカラムの 1 から始まる序数。

value　パラメータが受け取る値。

SetInteger メソッド

指定した Integer 値に対するパラメータを設定します。

構文

SetInteger(
index As Integer
value As Integer
)
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　設定対象となる現在のローのカラムの 1 から始まる序数。

value　パラメータが受け取る値。

SetLong メソッド

指定した Long 値に対するパラメータを設定します。

構文

SetLong(
index As Integer
value As Long
)
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　設定対象となる現在のローのカラムの 1 から始まる序数。

value　パラメータが受け取る値。

SetNull メソッド

指定したカラムを NULL に設定します。

構文

SetNull(*index* As Integer **)**
Member of **UltraLiteAFLib.ULResultSet**

パラメータ

index　設定対象となる現在のローのカラムの 1 から始まる序数。

Update メソッド

テーブルの現在のローを現在のデータで更新します。

構文

Update()
Member of **UltraLiteAFLib.ULResultSet**

UpdateBegin メソッド

現在のローの内容を変更するためにテーブルを準備します。

構文

UpdateBegin()
Member of **UltraLiteAFLib.ULResultSet**

ULResultSetSchema クラス

ULResultSetSchema は、結果セットのスキーマに関する情報を提供します。

プロパティ

構文	説明
ColumnBaseName As String (読み込み専用)	結果セット内の指定されたカラムにベース・カラムがある場合、そのベース・カラム名を取得する。
ColumnBaseTableName As String (読み込み専用)	結果セット内の指定されたカラムにベース・テーブルがある場合、そのベース・テーブル名を取得する。
ColumnCount As Integer (読み込み専用)	結果セット内のカラム数を取得する。
ColumnID As Integer (読み込み専用)	結果セット内の指定されたカラムのカラム ID を取得する。
ColumnName As String (読み込み専用)	結果セットに含まれるカラムの名前を取得する。
ColumnPrecision As Integer (読み込み専用)	カラムが数値の場合は、カラムのデータ型の精度を取得する。
ColumnScale As Integer (読み込み専用)	カラムが数値の場合は、カラムのデータ型の位取りを取得する。
ColumnSize As Integer (読み込み専用)	カラムのデータのサイズを取得する。
ColumnSQLType As ULSQLType (読み込み専用)	カラムの ULSQLType を取得する。

ULSQLCode 列挙

ULSQLCode 定数は、Ultra Light によってレポートされる SQL コードを示します。

エラー・コードの詳細については、『*SQL Anywhere 10* - エラー・メッセージ』マニュアルの説明に移動するリンクをクリックしてください。

メンバ	説明
SQLE_AGGREGATES_NOT_ALLOWED	「集合関数の不正な使用です。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_ALIAS_NOT_UNIQUE	「エイリアス '%1' がユニークではありません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_ALIAS_NOT_YET_DEFINED	「エイリアス '%1' の定義は、最初の参照前に記述する必要があります。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_AMBIGUOUS_INDEX_NAME	「インデックス名 '%1' があいまいです。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_BAD_ENCRYPTION_KEY	「暗号化キーが不正であるか、見つかりません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_BAD_PARAM_INDEX	「入力パラメータ・インデックスが範囲外です。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CANNOT_ACCESS_FILESYSTEM	「デバイス上のファイルシステムにアクセスできません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CANNOT_CHANGE_USER_NAME	「前回のアップロードのステータスが不明の場合、同期の user_name は変更できません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CANNOT_CONVERT	「不正なデータ変換」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CANNOT_EXECUTE_STMT	「文を実行することができませんでした。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CANNOT_MODIFY	「テーブル '%2' のカラム '%1' を変更できません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CLIENT_OUT_OF_MEMORY	「クライアントでメモリが不足しています。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_COLUMN_AMBIGUOUS	「カラム '%1' が複数のテーブルで見つかりました。相関名が必要です。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_COLUMN_CANNOT_BE_NULL	「テーブル '%2' のカラム '%1' は NULL 値を持つことはできません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。

メンバ	説明
SQLE_COLUMN_IN_INDEX	「インデックスのカラムを変更することはできません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_COLUMN_NOT_FOUND	「カラム '%1' が見つかりません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_COLUMN_NOT_INDEXED	「カラム '%1' は、それを含んでいるテーブルのどのインデックスにも属していません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_COLUMN_NOT_STREAMABLE	「操作が失敗しました。カラム '%1' のタイプがストリーミングをサポートしていません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_COMMUNICATIONS_ERROR	「通信エラーが発生しました。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CONNECTION_ALREADY_EXISTS	「この接続はすでに存在します。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CONNECTION_NOT_FOUND	「接続が見つかりません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CONNECTION_RESTORED	「Ultra Light の接続がリストアされました。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CONSTRAINT_NOT_FOUND	「制約 '%1' が見つかりません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CONVERSION_ERROR	「値 %1 をデータ型 %2 に変換できません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_COULD_NOT_FIND_FUNCTION	「ダイナミック・ライブラリ '%2' に '%1' が見つかりませんでした。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_COULD_NOT_LOAD_LIBRARY	「ダイナミック・ライブラリ '%1' をロードできませんでした。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CURSOR_ALREADY_OPEN	「カーソルはすでに開いています。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CURSOR_NOT_DECLARED	「カーソルが宣言されていません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CURSOR_NOT_OPEN	「カーソルが開きません。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CURSOR_RESTORED	「Ultra Light のカーソル (あるいは結果セットまたはテーブル) がリストアされました。」『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_CURSOROP_NOT_ALLOWED	「不正なカーソル処理をしようとしました。」『SQL Anywhere 10 - エラー・メッセージ』を参照。

メンバ	説明
SQLE_DATABASE_ERROR	「データベースの内部エラー %1 -- トランザクションはロールバックされました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DATABASE_NAME_REQUIRED	「サーバを起動するには、データベース名が必要です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DATABASE_NOT_CREATED	「データベースの作成に失敗しました : %1」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DATATYPE_NOT_ALLOWED	「式にサポートされていないデータ型があります。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DB_INIT_NOT_CALLED	「db_init が呼び出されていないか db_init の呼び出しに失敗しました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DBLIB_ENGINE_MISMATCH	「クライアントとデータベース・サーバのバージョンが適合しません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DBSPACE_FULL	「データベース領域が最大ファイル・サイズに達しています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DESCRIBE_NONSELECT	「SELECT 文以外は記述できません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DEVICE_IO_FAILED	「'%1' のファイル I/O に失敗しました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DIV_ZERO_ERROR	「ゼロで除算しようとしました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DOUBLE_REQUEST	「アクティブなデータベース要求を 2 つ出そうとしました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DOWNLOAD_CONFLICT	「既存のローと競合しているため、ダウンロードに失敗しました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DOWNLOAD_RESTART_FAILED	「ダウンロードをリトライできません。アップロードが完了していません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DROP_DATABASE_FAILED	「データベース '%1' の削除に失敗しました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DUPLICATE_CURSOR_NAME	「カーソル名 '%1' はすでに存在します。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DUPLICATE_FOREIGN_KEY	「テーブル '%2' の外部キー '%1' は、既存の外部キーと重複しています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。

メンバ	説明
SQLE_DUPLICATE_OPTION	「オプション '%1' が複数回指定されています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_DYNAMIC_MEMORY_EXHAUSTED	「動的メモリが足りません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_ENCRYPTION_INITIALIZATION_FAILED	「暗号化 DLL を初期化できませんでした : '%1'」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_ENGINE_ALREADY_RUNNING	「データベース・サーバはすでに起動しています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_ENGINE_NOT_RUNNING	「データベース・サーバが見つかりません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_ERROR	「実行時 SQL エラーです -- %1」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_ERROR_CALLING_FUNCTION	「外部関数の呼び出しのためのリソースを割り付けられませんでした。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_ERROR_IN_ASSIGNMENT	「割り当てのエラー」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_EXPRESSION_ERROR	「'%1' 付近に無効な式があります。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_FEATURE_NOT_ENABLED	「呼び出そうとしたメソッドは、お使いのアプリケーションでは使用できません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_FILE_BAD_DB	「指定されたデータベースを開始できません。'%1' は有効なデータベース・ファイルではありません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_FILE_IN_USE	「指定されたデータベース・ファイルはすでに使用されています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_FILE_NOT_DB	「指定されたデータベースを開始できません。'%1' はデータベースではありません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_FILE_VOLUME_NOT_FOUND	「データベース '%1' に対して指定したファイルシステム・ボリュームが見つかりません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_FILE_WRONG_VERSION	「指定されたデータベースを開始できません。'%1' は異なるバージョンのソフトウェアで作成されています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_FOREIGN_KEY_NAME_NOT_FOUND	「外部キー '%1' は見つかりません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。

メンバ	説明
SQLE_IDENTIFIER_TOO_LONG	「識別子 '%1' が長すぎます。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INCORRECT_VOLUME_ID	「'%1' のボリューム ID が不正です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INDEX_NAME_NOT_UNIQUE	「インデックス名 '%1' はユニークではありません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INDEX_NOT_FOUND	「インデックス '%1' が見つかりません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INDEX_NOT_UNIQUE	「テーブル '%2' のインデックス '%1' はユニークでなければなりません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INTERRUPTED	「文の実行がユーザによって中断させられました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_CONSTRAINT_REF	「制約 '%1' への参照または操作が無効です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_DESCRIPTOR_INDEX	「記述子のインデックスが正しくありません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_DESCRIPTOR_NAME	「SQL 記述子名が正しくありません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_DISTINCT_AGGREGATE	「グループ化されたクエリに、複数の異なる集合関数が含まれています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_FOREIGN_KEY	「テーブル '%2' の外部キー '%1' に対応するプライマリ・キーの値がありません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_FOREIGN_KEY_DEF	「外部キーのカラム '%1' にプライマリー・キーと異なる定義があります。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_GROUP_SELECT	「'%1' に対する関数またはカラムの参照も GROUP BY 句に記述する必要があります。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_INDEX_TYPE	「'%1' のインデックスの型の指定は不正です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_LOGON	「ユーザ ID またはパスワードが無効です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_OPTION_SETTING	「不正なオプション '%1' を設定しています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。

メンバ	説明
SQLE_INVALID_OPTION_VALUE	「'%1' は '%2' に対して無効な値です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_ORDER	「ORDER BY 句の指定が不正です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_PARAMETER	「不正なパラメータです。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_PARSE_PARAMETER	「解析エラーです : %1」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_PUBLICATION_MASK	「指定されたパブリケーション・マスクが不正です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_SQL_IDENTIFIER	「SQL の識別子が無効です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_STATEMENT	「文が無効です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_INVALID_UNION	「UNION、INTERSECT、または EXCEPT の select リストの長さが一致していません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_KEYLESS_ENCRYPTION	「このデータベースはキー不使用の暗号化を使用するため、要求された操作を実行できません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_LOCKED	「'%2' のローは、ユーザ '%1' によってロックされています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_MEMORY_ERROR	「メモリエラー -- トランザクションはロールバックされました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_METHOD_CANNOT_BE_CALLED	「メソッド '%1' は現時点では呼び出せません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_NAME_NOT_UNIQUE	「アイテム '%1' はすでに存在しています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_NO_COLUMN_NAME	「抽出されたテーブル '%1' にはカラム %2 に対する名前がありません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_NO_CURRENT_ROW	「カーソルの現在のローがありません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_NO_INDICATOR	「NULL に対して、インジケータ変数が用意されていません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。

メンバ	説明
SQLE_NO_MATCHING_SELECT_ITEM	「派生テーブル '%1' の select リストに '%2' と一致する式がありません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_NO_PRIMARY_KEY	「テーブル '%1' にはプライマリ・キーが定義されていません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_NOERROR	SQLE_NOERROR(0) – このコードは、エラーまたは警告がなかったことを示します。
SQLE_NON_UPDATEABLE_COLUMN	「式を更新できません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_NON_UPDATEABLE_CURSOR	「FOR UPDATE が READ ONLY カーソルに誤って指定されました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_NOT_IMPLEMENTED	「'%1' の機能は実装されていません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_NOT_SUPPORTED_IN_ULTRALITE	「Ultra Light では使用できない機能です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_NOTFOUND	「ローが見つかりません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_ONLY_ONE_TABLE	「カーソルの INSERT/DELETE は、1 度に 1 つのテーブルにしかできません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_OVERFLOW_ERROR	「値 %1 は、対象先にとって大きすぎます。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_PAGE_SIZE_INVALID	「無効なデータベース・ページ・サイズです。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_PARTIAL_DOWNLOAD_NOT_FOUND	「部分ダウンロードが見つかりませんでした。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_PERMISSION_DENIED	「パーミッションがありません : %1」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_PRIMARY_KEY_NOT_UNIQUE	「テーブル '%1' のプライマリ・キーがユニークではありません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_PRIMARY_KEY_TWICE	「テーブルに 2 つのプライマリ・キーを定義することはできません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_PRIMARY_KEY_VALUE_REF	「テーブル '%1' 内のローのプライマリ・キーがテーブル '%3' 内の外部キー '%2' によって参照されています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。

メンバ	説明
SQLE_PUBLICATION_NOT_FOUND	「パブリケーション '%1' が見つかりません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_PUBLICATION_PREDICATE_IGNORED	「パブリケーションの述部は評価されませんでした。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_RESOURCE_GOVERNOR_EXCEEDED	「%1 のリソース・ガバナーが制限を超えています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_ROW_DELETED_TO_MAINTAIN_REFERENTIAL_INTEGRITY	「参照整合性を保つためにテーブル %1 からローが削除されました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_SERVER_SYNCHRONIZATION_ERROR	「サーバ %1 でエラーが発生したため、同期に失敗しました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_START_STOP_DATABASE_DENIED	「データベースの起動／停止の要求は拒否されました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_STATEMENT_ERROR	「SQL 文にエラーがあります。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_STRING_RIGHT_TRUNCATION	「文字列データの右側がトランケートされます。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_SUBQUERY_SELECT_LIST	「select リストの中にカラムが 2 つ以上指定されています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_SYNC_INFO_INVALID	「同期の情報が不完全か無効です。'%1' を確認してください。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_SYNC_INFO_REQUIRED	「同期の情報が指定されていません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_SYNC_NOT_REENTRANT	「同期処理に戻ることができませんでした。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_SYNC_STATUS_UNKNOWN	「最後の同期アップロードのステータスは不明です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_SYNTAX_ERROR	「'%1' %2 の近くに構文エラーがあります。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_TABLE_ALREADY_INCLUDED	「テーブル '%1' はすでにインクルードされています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_TABLE_IN_USE	「テーブルは使用されています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_TABLE_NOT_FOUND	「テーブル '%1' は見つかりませんでした。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。

メンバ	説明
SQLE_TOO_MANY_BLOB_REFS	「BLOB への参照が多すぎます。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_TOO_MANY_CONNECTIONS	「データベース・サーバに接続できる限界数を超えています。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_TOO_MANY_PUBLICATIONS	「パブリケーション・マスクに指定されているパブリケーションが多すぎます。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_TOO_MANY_TEMP_TABLES	「接続しているテンポラリ・テーブルが多すぎます。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_TOO_MANY_USERS	「データベースのユーザが多すぎます。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_ULTRALITE_DATABASE_NOT_FOUND	「データベース '%1' が見つかりませんでした。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_ULTRALITE_OBJ_CLOSED	「閉じられたオブジェクトに対する操作は無効です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_ULTRALITE_WRITE_ACCESS_DENIED	「書き込みアクセスが拒否されました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_UNABLE_TO_CONNECT	「データベースが起動できません -- %1」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_UNABLE_TO_CONNECT_OR_START	「サーバが見つからないため自動起動できません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_UNABLE_TO_START_DATABASE	「指定されたデータベースを起動できません : %1」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_UNABLE_TO_START_DATABASE_VER_NEWER	「指定されたデータベースを起動できません : データベース %1 を起動するにはサーバをアップグレードする必要があります。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_UNABLE_TO_START_ENGINE	「データベース・サーバを起動することができません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_UNCOMMITTED_TRANSACTIONS	「コミットされていないトランザクションの同期またはアップグレードはできません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_UNKNOWN_FUNC	「関数 '%1' はありません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_UNKNOWN_OPTION	「'%1' は認識できないオプションです。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_UNKNOWN_USERID	「'%1' というユーザ ID はありません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。

メンバ	説明
SQLE_UNRECOGNIZED_OPTION	「オプション '%1' が認識されません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_UPLOAD_FAILED_AT_SERVER	「同期サーバがアップロードのコミットに失敗しました。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_VALUE_IS_NULL	「要求されたデータ型として NULL の結果を返すことができません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_VARIABLE_INVALID	「無効なホスト変数です。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_WRONG_NUM_OF_INSERT_COLS	「INSERT コマンドへの値が正しくありません。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。
SQLE_WRONG_PARAMETER_COUNT	「関数 '%1' のパラメータ数が誤りです。」 『SQL Anywhere 10 - エラー・メッセージ』を参照。

ULSQLType 列挙

ULSQLType は、テーブル・カラムの型として使用されている、Ultra Light SQL データベースの型をリストします。

定数	Ultra Light データベースの型	値
ulTypeLong	Integer	0
ulTypeShort	UnsignedInteger	1
ulTypeUnsignedLong	SmallInt	2
ulTypeUnsignedShort	UnsignedSmallInt	3
ulTypeBig	Big	4
ulTypeUnsignedBig	UnsignedBig	5
ulTypeByte	Byte	6
ulTypeBit	Bit	7
ulTypeDateTime	Time	8
ulTypeDate	Date	9
ulTypeTime	Timestamp	10
ulTypeDouble	Double	11
ulTypeReal	Real	12
ulTypeBinary	LongBinary	13
ulTypeLongBinary	Numeric	14
ulTypeString	(Var)Char	15
ulTypeLongString	LongVarchar	16
ulTypeNumeric	(Var)Binary	17
ulTypeUUID	UniqueIdentifier	18

ULStreamErrorCode 列挙

ULStreamErrorCode 定数は、同期時の通信エラーを識別します。

これらのエラーの詳細については、「Mobile Link 通信エラー・メッセージ」 『SQL Anywhere 10 - エラー・メッセージ』を参照してください。

定数	値
ulStreamErrorCodeNone	0
ulStreamErrorCodeParameter	1
ulStreamErrorCodeParameterNotUint32	2
ulStreamErrorCodeParameterNotUint32Range	3
ulStreamErrorCodeParameterNotBoolean	4
ulStreamErrorCodeParameterNotHex	5
ulStreamErrorCodeMemoryAllocation	6
ulStreamErrorCodeParse	7
ulStreamErrorCodeRead	8
ulStreamErrorCodeWrite	9
ulStreamErrorCodeEndWrite	10
ulStreamErrorCodeEndRead	11
ulStreamErrorCodeNotImplemented	12
ulStreamErrorCodeWouldBlock	13
ulStreamErrorCodeGenerateRandom	14
ulStreamErrorCodeInitRandom	15
ulStreamErrorCodeSeedRandom	16
ulStreamErrorCodeCreateRandomObject	17
ulStreamErrorCodeShuttingDown	18
ulStreamErrorCodeDequeuingConnection	19
ulStreamErrorCodeSecureCertificateRoot	20
ulStreamErrorCodeSecureCertificateCompanyName	21
ulStreamErrorCodeSecureCertificateChainLength	22

定数	値
ulStreamErrorCodeSecureCertificateRef	23
ulStreamErrorCodeSecureCertificateNotTrusted	24
ulStreamErrorCodeSecureDuplicateContext	25
ulStreamErrorCodeSecureSetIo	26
ulStreamErrorCodeSecureSetIoSemantics	27
ulStreamErrorCodeSecureCertificateChainFunc	28
ulStreamErrorCodeSecureCertificateChainRef	29
ulStreamErrorCodeSecureEnableNonBlocking	30
ulStreamErrorCodeSecureSetCipherSuites	31
ulStreamErrorCodeSecureSetChainNumber	32
ulStreamErrorCodeSecureCertificateFileNotFound	33
ulStreamErrorCodeSecureReadCertificate	34
ulStreamErrorCodeSecureReadPrivateKey	35
ulStreamErrorCodeSecureSetPrivateKey	36
ulStreamErrorCodeSecureCertificateExpiryDate	37
ulStreamErrorCodeSecureExportCertificate	38
ulStreamErrorCodeSecureAddCertificate	39
ulStreamErrorCodeSecureTrustedCertificateFileNotFound	40
ulStreamErrorCodeSecureTrustedCertificateRead	41
ulStreamErrorCodeSecureCertificateCount	42
ulStreamErrorCodeSecureCreateCertificate	43
ulStreamErrorCodeSecureImportCertificate	44
ulStreamErrorCodeSecureSetRandomRef	45
ulStreamErrorCodeSecureSetRandomFunc	46
ulStreamErrorCodeSecureSetProtocolSide	47
ulStreamErrorCodeSecureAddTrustedCertificate	48
ulStreamErrorCodeSecureCreatePrivateKeyObject	49

定数	値
ulStreamErrorCodeSecureCertificateExpired	50
ulStreamErrorCodeSecureCertificateCompanyUnit	51
ulStreamErrorCodeSecureCertificateCommonName	52
ulStreamErrorCodeSecureHandshake	53
ulStreamErrorCodeHttpVersion	54
ulStreamErrorCodeSecureSetReadFunc	55
ulStreamErrorCodeSecureSetWriteFunc	56
ulStreamErrorCodeSocketHostNameNotFound	57
ulStreamErrorCodeSocketGetHostByAddr	58
ulStreamErrorCodeSocketLocalhostNameNotFound	59
ulStreamErrorCodeSocketCreateTcpip	60
ulStreamErrorCodeSocketCreateUdp	61
ulStreamErrorCodeSocketBind	62
ulStreamErrorCodeSocketCleanup	63
ulStreamErrorCodeSocketClose	64
ulStreamErrorCodeSocketConnect	65
ulStreamErrorCodeSocketGetName	66
ulStreamErrorCodeSocketGetOption	67
ulStreamErrorCodeSocketSetOption	68
ulStreamErrorCodeSocketListen	69
ulStreamErrorCodeSocketShutdown	70
ulStreamErrorCodeSocketSelect	71
ulStreamErrorCodeSocketStartup	72
ulStreamErrorCodeSocketPortOutOfRange	73
ulStreamErrorCodeLoadNetworkLibrary	74
ulStreamErrorCodeActsyncNoPort	75
ulStreamErrorCodeHttpExpectedPost	89

ULStreamErrorContext 列挙

ULStreamErrorContext 定数は、ULStreamErrorContext の指定に使用できる定数を識別します。ULStreamErrorContext は、ストリーム・エラーが発生したときに実行されるネットワーク・オペレーションです。

定数	値
ulStreamErrorContextUnknown	0
ulStreamErrorContextRegister	1
ulStreamErrorContextUnregister	2
ulStreamErrorContextCreate	3
ulStreamErrorContextDestroy	4
ulStreamErrorContextOpen	5
ulStreamErrorContextClose	6
ulStreamErrorContextRead	7
ulStreamErrorContextWrite	8
ulStreamErrorContextWriteFlush	9
ulStreamErrorContextEndWrite	10
ulStreamErrorContextEndRead	11
ulStreamErrorContextYield	12
ulStreamErrorContextSoftshutdown	13

ULStreamErrorID 列挙

ULStreamErrorID は、同期が失敗したときにおそらくエラーの原因となった、ネットワーク・レイヤの列挙です。

定数	値
ulStreamErrorIDTcpip	0
ulStreamErrorIDSerial	1
ulStreamErrorIDFake	2
ulStreamErrorIDPalmConduit	3
ulStreamErrorIDPalmSs	4
ulStreamErrorIDNettech	5
ulStreamErrorIDRimbb	6
ulStreamErrorIDHttp	7
ulStreamErrorIDHttps	8
ulStreamErrorIDDhCast	9
ulStreamErrorIDSecure	10
ulStreamErrorIDCerticom	11
ulStreamErrorIDJavaCerticom	12
ulStreamErrorIDCerticomSsl	13
ulStreamErrorIDCerticomTls	14
ulStreamErrorIDWirestrm	15
ulStreamErrorIDWireless	16
ulStreamErrorIDReplay	17
ulStreamErrorIDStrm	18
ulStreamErrorIDUdp	19
ulStreamErrorIDEmail	20
ulStreamErrorIDFile	21
ulStreamErrorIDActivesync	22
ulStreamErrorIDRsaTls	23

定数	値
ulStreamErrorIDJavaRsa	24
ulStreamErrorIDOpenSslRsa	25
ulStreamErrorIDPalmSsl	26

ULStreamType 列挙

ULStreamType 定数は、ストリーム・タイプの指定に使用できる定数を識別します。これらの定数は、同期に使用できる、Mobile Link 同期ストリームのタイプを表します。

定数	値	説明
ulUnknown	0	ストリーム・タイプは設定されていない。ストリーム・タイプを設定してから同期を行ってください。
ulTCPIP	1	TCP/IP ストリーム
ulHTTP	2	HTTP ストリーム
ulHTTPS	3	HTTPS 同期
ulPalmConduit	4	HotSync 同期用
ulTLS	5	TLS (トランスポート・レイヤ・セキュリティ)

ULSyncEvent クラス

OnReceive イベント

Mobile Link を介した、統合データベースからアプリケーションへのダウンロード情報をレポートします。

構文

OnReceive(
nBytes As Long, _
nInserts As Long, _
nUpdates As Long, _
nDeletes As Long _
)
Member of **UltraLiteAFLib.ULSyncEvent**

パラメータ

nBytes リモート・アプリケーションで受信した、統合データベースからの累積バイト数。

nInserts リモート・アプリケーションで受信した、統合データベースからの挿入の累積回数。

nUpdates リモート・アプリケーションで受信した、統合データベースからの更新の累積回数。

nDeletes リモート・アプリケーションで受信した、統合データベースからの削除の累積回数。

備考

このイベントは、複数回呼び出すことができます。

例

このメソッドの例については、CustDB アプリケーションを参照してください。CustDB for AppForge の場所の詳細については、「CustDB サンプル・ファイルの場所」 『Ultra Light - データベース管理とリファレンス』を参照してください。

OnSend イベント

Mobile Link を介した、リモート・データベースから統合データベースへのアップロード情報をレポートします。

構文

OnSend(
nBytes As Long, _
nInserts As Long, _
nUpdates As Long, _
nDeletes As Long _
)
Member of **UltraLiteAFLib.ULSyncEvent**

パラメータ

nBytes　Mobile Link を介して、リモート・アプリケーションが統合データベースに送信した累積バイト数。

nInserts　Mobile Link を介して、リモート・アプリケーションが統合データベースに送信した挿入の累積回数。

nUpdates　Mobile Link を介して、リモート・アプリケーションが統合データベースに送信した更新の累積回数。

nDeletes　Mobile Link を介して、リモート・アプリケーションが統合データベースに送信した削除の累積回数。

備考

このイベントは、複数回呼び出すことができます。

例

このメソッドの例については、CustDB アプリケーションを参照してください。CustDB for AppForge の場所の詳細については、「CustDB サンプル・ファイルの場所」 『Ultra Light - データベース管理とリファレンス』を参照してください。

OnStateChange イベント

このイベントは、同期ステータスが変更されるたびに呼び出されます。

構文

OnStateChange(
  newState As ULSyncState, _
  oldState As ULSyncState _
)
Member of **UltraLiteAFLib.ULSyncEvent**

パラメータ

newState　直後に開始される同期処理のステータス。

oldState　直前に完了した同期処理のステータス。

参照

- 「ULSyncState 列挙」 151 ページ

例

このメソッドの例については、CustDB アプリケーションを参照してください。CustDB for AppForge の場所の詳細については、「CustDB サンプル・ファイルの場所」 『Ultra Light - データベース管理とリファレンス』を参照してください。

OnTableChange イベント

このイベントは、同期処理が次のテーブルの同期を開始するたびに呼び出されます。

構文

OnTableChange(
newTableIndex As Long, _
numTables As Long _
)
Member of **UltraLiteAFLib.ULSyncEvent**

パラメータ

newTableIndex　現在同期が行われているテーブルのインデックス番号。この番号はテーブル ID とは異なるため、ULDatabaseSchema.GetTableName メソッドでは使用できません。

numTables　同期が行われるテーブル数。

例

このメソッドの例については、CustDB アプリケーションを参照してください。CustDB for AppForge の場所の詳細については、「CustDB サンプル・ファイルの場所」 『Ultra Light - データベース管理とリファレンス』を参照してください。

OnWaiting イベント

このイベントは、Mobile Link の応答を同期が待機中のときに必ず呼び出されます。

構文

OnWaiting() Member of **UltraLiteAFLib.ULSyncEvent**

ULSyncParms クラス

ULSyncParms オブジェクトの属性セットは、データベースと統合データベースまたはデスクトップ・データベースとの同期方法を決定します。読み込み専用の属性は、最後の同期ステータスを反映します。

プロパティ

ULSyncParms のプロパティは次のとおりです。

構文	説明
CheckpointStore As Boolean	true の場合は、同期中にデータベースのチェックポイントを追加して、同期処理中にデータベースが大きくなりすぎないように制限する。これは、多くの更新を伴う大量のダウンロードに最適です。 「Checkpoint Store 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。
DownloadOnly As Boolean	同期はデータのダウンロードのみを行うかどうかを示す。 「Download Only 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。
KeepPartialDownload As Boolean	ダウンロード時の通信エラーが原因で同期が失敗した場合、すべての変更をロールバックしないで、正常にダウンロードされた変更を適用する。 「Keep Partial Download 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。
NewPassword As String	次の同期で、この新しいパスワードにユーザ・パスワードを変更する。 「New Password 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。
Password As String	特定のユーザ名に対応したパスワード 「Password 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。
PingOnly As Boolean	true の場合は、サーバの活性をチェックするが、データを同期しない。 「Ping 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。

構文	説明
PublicationMask As Long	同期させるパブリケーションを指定する。デフォルトでは、すべてのデータを同期する。 「publication 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。
ResumePartialDownload As Boolean	同期が失敗したときにダウンロードされる予定だった変更のみを適用し、ダウンロード時の通信エラーが原因で失敗した同期を再開する。 「Resume Partial Download 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。
SendColumnNames As Boolean	SendColumnNames が true の場合、Mobile Link サーバにカラム名を送信する。 「Send Column Names 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。
SendDownloadAck As Boolean	SendDownloadAck が true の場合、同期中にダウンロード確認を送信する。 「Send Download Acknowledgment 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。
Stream As ULStreamType constants	同期中に使用するストリームのタイプを設定する。 「Stream Type 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。
StreamParms As String	特定のストリーム・タイプのネットワーク・プロトコル・オプションを設定する。 詳細については、「Stream Parameters 同期パラメータ」『Mobile Link - クライアント管理』と「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」『Mobile Link - クライアント管理』を参照してください。
TableOrder As String	テーブルの同期順序を指定する。 「Table Order 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。
UploadOnly As Boolean	同期はデータのアップロードのみを行うかどうかを示す。 「Upload Only 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。

構文	説明
UserName As String	同期用の Mobile Link ユーザ名。 「User Name 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。
Version As String	実行する同期スクリプト・バージョン。 「Version 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。

例

次の例は、Ultra Light for MobileVB アプリケーションの同期パラメータを設定します。

```
With Connection.SyncParms
   .UserName = "afsample"
   .Stream = ULStreamType.ulTCPIP
   .Version = "ul_default"
End With
Connection.Synchronize
```

AddAuthenticationParm メソッド

authenticate_parms Mobile Link 同期スクリプトに渡すパラメータを追加します。

構文

AddAuthenticationParm(BSTR parm **)**
Member of **UltraLiteAFLib.ULSyncParms**

パラメータ

parm　追加するパラメータ。

参照

- 「Authentication Parameters 同期パラメータ」『Mobile Link - クライアント管理』
- 「authenticate_parameters 接続イベント」『Mobile Link - サーバ管理』

ClearAuthenticationParms メソッド

authenticate_parms Mobile Link 同期スクリプトに渡すパラメータをすべてクリアします。

構文

ClearAuthenticationParms( )
Member of **UltraLiteAFLib.ULSyncParms**

参照

- 「Authentication Parameters 同期パラメータ」『Mobile Link - クライアント管理』
- 「authenticate_parameters 接続イベント」『Mobile Link - サーバ管理』

ULSyncResult クラス

ULSyncResult オブジェクトの属性は、最後の同期の結果を格納します。

プロパティ

次は、ULSyncResult のプロパティです。

構文	説明
AuthStatus As ULAuthStatusCode (読み込み専用)	前回行われた同期の認証ステータス・コードを取得する。 「Authentication Status 同期パラメータ」 『Mobile Link - クライアント管理』を参照してください。
AuthValue As Long (読み込み専用)	Mobile Link 認証値を取得する。 「Authentication Value 同期パラメータ」 『Mobile Link - クライアント管理』を参照してください。
PartialDownloadRetained As Boolean (読み込み専用)	ダウンロード時に同期が失敗し、部分的にダウンロードが保持されていることを示す。 「Partial Download Retained 同期パラメータ」 『Mobile Link - クライアント管理』を参照してください。
IgnoredRows As Boolean (読み込み専用)	前回行われた同期でローが無視されたかどうかを示す。 「Ignored Rows 同期パラメータ」 『Mobile Link - クライアント管理』を参照してください。
StreamErrorCode As ULStreamErrorCode (読み込み専用)	同期ストリームによってレポートされるエラー・コードを取得する。
StreamErrorContext As ULStreamErrorContext (読み込み専用)	実行される基本的なネットワーク・オペレーションを取得する。
StreamErrorID As ULStreamErrorID (読み込み専用)	エラーをレポートするネットワーク・レイヤを取得する。
StreamErrorSystem As Long (読み込み専用)	ストリーム・エラー・システム固有のコードを取得する。
Timestamp as Date (読み込み専用)	最後の同期のタイムスタンプを取得する。
UploadOK As Boolean (読み込み専用)	前回行われた同期でデータが正常にアップロードされたかどうかを示す。 「Version 同期パラメータ」 『Mobile Link - クライアント管理』を参照してください。

ULSyncState 列挙

定数	値	説明
ulSyncStateStarting	0	同期処理はまだ行われていない。
ulSyncStateConnecting	1	同期ストリームは構築されたが、まだ開かれていない。
ulSyncStateSendingHeader	2	同期ストリームが開かれ、ヘッダが送信されようとしている。
ulSyncStateSendingTable	3	テーブルの送信中。
ulSyncStateSendingData	4	現在のテーブルのデータの送信中。
ulSyncStateFinishingUpload	5	アップロードが完了。送信された最終的なロー数が、このイベントに含まれます。
ulSyncStateReceivingUploadAck	6	アップロード完了の確認の受信中。
ulSyncStateReceivingTable	7	テーブルの受信中。
ulSyncStateReceivingData	8	現在のテーブルのデータの受信中。
ulSyncStateCommittingDownload	9	ダウンロードのコミット中。受信された最終的なロー数が、このイベントに含まれます。
ulSyncStateSendingDownloadAck	10	ダウンロード完了の確認の送信中。
ulSyncStateDisconnecting	11	同期ストリームが閉じられようとしている。
ulSyncStateDone	12	同期が正常に完了した。SyncResult オブジェクトが更新されました。
ulSyncStateError	13	同期は完了したが、エラーが発生した。SyncResult と SQLCode の詳細を確認してください。
ulSyncStateRollingBackDownload	14	ダウンロード中にエラーが発生したため、同期によってダウンロードがロールバックされている。後続の ulSyncStateError 進行状況レポートにエラーがレポートされます。
ulSyncStateCancelled	99	同期がキャンセルされた。

ULTable クラス

ULTable クラスは、テーブルのデータの格納、削除、更新、読み込みに使用します。

Open メソッドを呼び出さないと、テーブルのデータを操作できません。ULTable では次のテーブル・モードを使用してテーブルを操作します。

モード	説明
FindBegin	検索モードを開始
InsertBegin	挿入モードを開始
LookupBegin	ルックアップ・モードを開始
UpdateBegin	更新モードを開始

プロパティ

構文	説明
BOF As Boolean (読み込み専用)	現在のローの位置が最初のローの前かどうかを示す。現在のローの位置が最初のローの前なら true を返し、そうでなければ false を返す。
EOF As Boolean (読み込み専用)	現在のローの位置が最後のローの後かどうかを示す。現在のローの位置が最初のローの前なら true を返し、そうでなければ false を返す。
IsOpen As Boolean (読み込み専用)	テーブルが現在開いているかどうかを示す。
RowCount As Long (読み込み専用)	テーブルのローの数を取得する。
Schema As ULTableSchema (読み込み専用)	テーブル・スキーマに関する情報を取得する。

Close メソッド

テーブルに関連付けられているリソースを解放します。

構文

Close( )
Member of **UltraLiteAFLib.ULTable**

備考

このメソッドは、テーブルに関するすべての処理が完了した後で呼び出してください。

Palm OS では、テーブルを閉じていない場合は、現在位置で再び開くことができます。

Column メソッド

指定したカラム名のオブジェクトを返します。

構文

Column(*name* As String **)** As ULColumn
Member of **UltraLiteAFLib.ULTable**

パラメータ

name　返されるカラムの名前。

戻り値

Column のオブジェクトを返します。

参照

- 「ULColumn クラス」 73 ページ

Delete メソッド

現在のローをテーブルから削除します。

構文

Delete( )
Member of **UltraLiteAFLib.ULTable**

DeleteAllRows メソッド

テーブルのすべてのローを削除します。

構文

DeleteAllRows( )
Member of **UltraLiteAFLib.ULTable**

備考

アプリケーションによっては、テーブル内のローをすべて削除してから、新しいデータ・セットをテーブルにダウンロードする方が便利なことがあります。

ローを、Ultra Light データベースからは削除し、統合データベースには残すことができます。

- **ULConnection.StopSynchronizationDelete** を使用します。
- このメソッドの代わりに **Truncate** を呼び出します。

参照

- 「StopSynchronizationDelete メソッド」 87 ページ
- 「Truncate メソッド」 162 ページ

FindBegin メソッド

検索のためにテーブルを準備します。

構文

FindBegin( )
Member of **UltraLiteAFLib.ULTable**

FindFirst メソッド

テーブルを先頭から順方向に移動しながら、現在のインデックスの値かそのセットに完全に一致するローを検索します。

構文

FindFirst([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

パラメータ

num_columns FindFirst で使用するカラム数を参照するオプションのパラメータ。たとえば、2 が渡されると、最初の 2 つのカラムが FindFirst に使用されます。num_columns が、インデックスされたカラムの数を超えると、すべてのカラムが FindFirst で使用されます。

戻り値

成功の場合は **true**。

失敗の場合は **false**。

備考

現在のインデックスは、テーブルのソート順の指定に使用されているインデックスです。このソート順は、アプリケーションが Open メソッドを呼び出したときに指定されます。デフォルトのインデックスはプライマリ・キーです。

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値と完全に一致した最初のローで停止します。失敗すると、カーソル位置は最後のローの後ろ (EOF) になります。

> **注意**
> このメソッドを使用するには、先に FindBegin を呼び出す必要があります。

参照

- 「FindBegin メソッド」 154 ページ

FindLast メソッド

テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセットに一致する最初のローを検索します。

構文

FindLast([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

パラメータ

num_columns　FindLast で使用するカラム数を参照するオプションのパラメータ。たとえば、2 が渡されると、最初の 2 つのカラムが FindLast に使用されます。num_columns が、インデックスされたカラムの数を超えると、すべてのカラムが FindLast で使用されます。

戻り値

成功の場合は **true**。

失敗の場合は **false**。

備考

現在のインデックスは、テーブルのソート順の指定に使用されます。このソート順は、アプリケーションが Open メソッドを呼び出したときに指定されます。デフォルトのインデックスはプライマリ・キーです。

検索する値を指定するには、値を検索するインデックスの各カラムに値を設定します。カーソルは、インデックスの値と完全に一致した最後のローで停止します。失敗すると、カーソル位置は最初のローの前 (BOF) になります。

> **注意**
> このメソッドを使用するには、先に FindBegin を呼び出す必要があります。

参照

- 「Open メソッド」 161 ページ
- 「FindBegin メソッド」 154 ページ

FindNext メソッド

現在の位置からテーブルを順方向に移動しながら、現在のインデックスの値かそのセットに完全に一致する次のローを検索します。

構文

FindNext([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

パラメータ

num_columns　FindNext で使用するカラム数を参照するオプションのパラメータ。たとえば、2 が渡されると、最初の 2 つのカラムが FindNext に使用されます。num_columns が、インデックスされたカラムの数を超えると、すべてのカラムが FindNext で使用されます。

戻り値

成功の場合は **true**。

失敗の場合 (EOF) は **false**。

備考

現在のインデックスは、テーブルのソート順の指定に使用されているインデックスです。このソート順は、アプリケーションが Open メソッドを呼び出したときに指定されます。デフォルトのインデックスはプライマリ・キーです。

カーソルは、インデックスの値と完全に一致した最初のローで停止します。失敗すると、カーソル位置は最後のローの後ろ (EOF) になります。

> **注意**
> このメソッドを使用するには、先に FindFirst または FindLast を呼び出す必要があります。

参照

- 「Open メソッド」 161 ページ
- 「FindFirst メソッド」 154 ページ
- 「FindLast メソッド」 155 ページ

FindPrevious メソッド

現在の位置からテーブルを逆方向に移動しながら、現在のインデックスの値かそのセットに完全に一致する前のローを検索します。

構文

FindPrevious([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

パラメータ

num_columns　FindPrevious で使用するカラム数を参照するオプションのパラメータ。たとえば、2 が渡されると、最初の 2 つのカラムが FindPrevious に使用されます。num_columns が、インデックスされたカラムの数を超えると、すべてのカラムが FindPrevious で使用されます。

戻り値

成功の場合は **true**。

失敗の場合 (BOF) は **false**。

備考

現在のインデックスは、テーブルのソート順の指定に使用されます。このソート順は、アプリケーションが Open メソッドを呼び出したときに指定されます。デフォルトのインデックスはプライマリ・キーです。

失敗すると、カーソル位置は最初のローの前 (BOF) になります。

参照

- 「Open メソッド」 161 ページ

Insert メソッド

直前に実行された **Set** メソッドで指定された値を使って、テーブルにローを挿入します。

構文

Insert( ) As Boolean
Member of **UltraLiteAFLib.ULTable**

備考

このメソッドを使用するには、先に InsertBegin を呼び出す必要があります。

戻り値

成功の場合は **true**。

失敗の場合 (BOF) は **false**。

参照

- 「InsertBegin メソッド」 157 ページ

InsertBegin メソッド

カラム値をデフォルトに設定し、新しいローを挿入できるようにテーブルを準備します。

構文

InsertBegin( )
Member of **UltraLiteAFLib.ULTable**

例

次の例では、InsertBegin は挿入モードに設定して、CustomerTable カラムへのデータ値の代入を開始できるようにします。

```
CustomerTable.InsertBegin
CustomerTable.Column("Fname").StringValue = fname
CustomerTable.Column("Lname").StringValue = lname
CustomerTable.Insert
```


参照

- 「UpdateBegin メソッド」 162 ページ

LookupBackward メソッド

テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセットに一致する最初のローか、それより小さい値の最初のローを検索します。

構文

LookupBackward([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

パラメータ

num_columns　複合インデックスのための、ルックアップで使用するカラムの数。

戻り値

成功の場合は **true**。

失敗の場合は **false**。

備考

現在のインデックスは、テーブルのソート順の指定に使用されます。このソート順は、アプリケーションが Open メソッドを呼び出したときに指定されます。デフォルトのインデックスはプライマリ・キーです。

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値に一致するか、それより少ない値の最後のローで停止します。検索が失敗した場合 (つまり、検索する値より小さい値のローがない場合)、カーソル位置は最初のローの前 (BOF) になります。

参照

- 「Open メソッド」 161 ページ

LookupBegin メソッド

ルックアップのためにテーブルを準備します。

構文

LookupBegin( )
Member of **UltraLiteAFLib.ULTable**

LookupForward メソッド

テーブルを最初から順方向に移動しながら、現在のインデックスの値またはそのセットに一致するか、その値より大きい値を持つ最初のローを検索します。

構文

LookupForward([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

備考

現在のインデックスは、テーブルのソート順の指定に使用されます。このソート順は、アプリケーションが Open メソッドを呼び出したときに指定されます。デフォルトのインデックスはプライマリ・キーです。

検索値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値に一致するか、それより大きい値の最初のローで停止します。検索が失敗した場合 (つまり、検索する値より大きい値のローがない場合)、カーソル位置は最後のローの後ろ (EOF) になります。

パラメータ

num_columns　複合インデックスのための、ルックアップで使用するカラムの数。

戻り値

成功の場合は **true**。

失敗の場合は **false**。

参照

- 「Open メソッド」 161 ページ

MoveAfterLast メソッド

最後のローの後に移動します。

構文

MoveAfterLast( ) As Boolean
Member of **UltraLiteAFLib.ULTable**

戻り値

成功の場合は **true**。

処理が失敗した場合は **false**。

MoveBeforeFirst メソッド

最初のローの前に移動します。

構文

MoveBeforeFirst( ) As Boolean
Member of **UltraLiteAFLib.ULTable**

戻り値

成功の場合は **true**。

処理が失敗した場合は **false**。

MoveFirst メソッド

最初のローに移動します。

構文

MoveFirst( ) As Boolean
Member of **UltraLiteAFLib.ULTable**

戻り値

成功の場合は **true**。

テーブル内にデータがない場合は **false**。

MoveLast メソッド

最後のローに移動します。

構文

MoveLast( ) As Boolean
Member of **UltraLiteAFLib.ULTable**

戻り値

成功の場合は **true**。

テーブル内にデータがない場合は **false**。

MoveNext メソッド

次のローに移動します。

構文

MoveNext( ) As Boolean
Member of **UltraLiteAFLib.ULTable**

戻り値

成功の場合は **true**。

テーブル内にデータがない場合は **false**。たとえば、ローがこれ以上ない場合 MoveNext は失敗します。

MovePrevious メソッド

前のローに移動します。

構文

MovePrevious( ) As Boolean
Member of **UltraLiteAFLib.ULTable**

戻り値

成功の場合は **true**。

テーブル内にデータがない場合は **false**。たとえば、ローがこれ以上ない場合 MovePrevious は失敗します。

MoveRelative メソッド

いくつかのローを、現在のローを基準にして相対的に移動します。

構文

MoveRelative(*index* As Long **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

パラメータ

index　移動するローの数。値は、正、負、または 0 です。ロー・バッファを再移植する場合は、0 が便利です。

戻り値

成功の場合は **true**。

移動が失敗した場合は **false**。たとえば、最初または最後のローを超えてカーソルが移動する場合は失敗します。

Open メソッド

テーブルを開き、読み込みまたは操作ができるようにします。

構文

Open(
  [*index_name* As String], _
  [*persistent_name* As String] _
)
Member of **UltraLiteAFLib.ULTable**

パラメータ

index_name　インデックスの名前を参照するオプションのパラメータ。

persistent_name　Palm Computing Platform アプリケーションの場合、格納されたテーブル名を参照するオプションのパラメータ。

備考

デフォルトでは、ローはプライマリ・キーによって順序付けられます。インデックスの名前を指定すると、ローを別の方法で並べ替えることができます。

カーソルは、テーブル内にある最初のローの前に置かれます。

Truncate メソッド

このテーブルからデータをすべて削除します。

構文

Truncate( )
Member of **UltraLiteAFLib.ULTable**

備考

変更は同期されません。同期の際、リモート Ultra Light データベースのトランケートされたデータは統合データベースから削除されません。

参照

- 「StartSynchronizationDelete メソッド」 87 ページ

Update メソッド

ULColumn メソッドで指定された値を使って、テーブルのローを更新します。

構文

Update( )
Member of **UltraLiteAFLib.ULTable**

備考

このメソッドを使用するには、先に UpdateBegin を呼び出す必要があります。

参照

- 「UpdateBegin メソッド」 162 ページ

UpdateBegin メソッド

現在のローの内容を変更するためにテーブルを準備します。

構文

UpdateBegin( )
Member of **UltraLiteAFLib.ULTable**

例

```
CustomerTable.UpdateBegin
CustomerTable.Column("Fname").StringValue = fname
'...
CustomerTable.Update
```


ULTableSchema クラス

ULTableSchema オブジェクトを使用すると、テーブルの属性を取得できます。

プロパティ

ULTableSchema は、テーブルに関するメタデータを表します。ULTableSchema クラスのプロパティは次のとおりです。

構文	説明
ColumnCount As Integer (読み込み専用)	このテーブル内のカラム数
IndexCount As Integer (読み込み専用)	このテーブルのインデックス数
Name As String (読み込み専用)	このテーブルの名前
NeverSynchronized As Boolean (読み込み専用)	テーブルが同期から常に除外されるかどうかを示す。
PrimaryKey As ULIndexSchema (読み込み専用)	このテーブルのプライマリ・キー
UploadUnchangedRows As Boolean (読み込み専用)	前回の同期以降に変更されたローだけでなく、同期に対してテーブルのすべてのローをアップロードするかどうかを示す。

GetColumnName メソッド

指定した *id* 値に対応するカラムの名前を返します。

構文

GetColumnName(*id* As Integer **)** As String
Member of **UltraLiteAFLib.ULTableSchema**

パラメータ

id　カラムの ID。

戻り値

カラムの名前。

備考

ColumnCount プロパティは、テーブルのカラム数を返します。各カラムは、1 ー ColumnCount 値までのユニークな番号を持ちます。1 はテーブル内の最初のカラムであり、2 はテーブル内の 2 番目のカラムです。以下も同様に続きます。

GetIndex メソッド

指定のインデックスの ULIndexSchema オブジェクトを返します。

構文

GetIndex(*name* As String **)** As ULIndexSchema
Member of **UltraLiteAFLib.ULTableSchema**

パラメータ

name　インデックスの名前。

戻り値

テーブルの指定されたインデックスのスキーマ・オブジェクトを返します。

参照

- 「ULIndexSchema クラス」 103 ページ

GetIndexName メソッド

指定した *id* 値に対応する、テーブル内のインデックス名を返します。

構文

GetIndexName(*id* As Integer **)** As String
Member of **UltraLiteAFLib.ULTableSchema**

パラメータ

id　インデックスの ID。

戻り値

インデックスの名前を返します。

備考

IndexCount プロパティは、テーブル内のインデックス数を返します。各インデックスは、1 ー IndexCount 値までのユニークな番号を持ちます。1 はテーブル内の最初のインデックスであり、2 はテーブル内の 2 番目のインデックスです。以下も同様に続きます。

GetPublicationPredicate メソッド

指定されたパブリケーション名にパブリケーション述部があれば、それを取得します。

構文

GetPublicationPredicate(
pub_name As String
) As String
Member of **UltraLiteAFLib.ULTableSchema**

パラメータ

pub_name　パブリケーション名。

戻り値

指定されたパブリケーションのパブリケーション述部文字列または空の文字列を返します。

InPublication メソッド

このテーブルが、指定されたパブリケーションの一部であるかどうかを示します。

構文

InPublication(*publicationName* As String **)** As Boolean
Member of **UltraLiteAFLib.ULTableSchema**

パラメータ

publicationName　確認するパブリケーションの名前。

戻り値

テーブルがパブリケーションの一部である場合は **true**。

テーブルがパブリケーションの一部でない場合は **false**。

索引

A

B

C

D

E

F

G

I

K

L

M

N

O

P

R

S

T

U

V

W

あ

い

え

お

か

き

け

